



THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

MULTI-LEVEL VIDEO FILTERING
USING NON-TEXTUAL CONTENTS

Xingzhong Du
Master of Computer Science

*A thesis submitted for the degree of Doctor of Philosophy at
The University of Queensland in 2017
School of Information Technology & Electrical Engineering*

Abstract

Video is one of the major media human uses to store information. As the recording and storing devices become cheaper, there are numerous videos generated nowadays. The unprecedentedly large volume creates considerable new requirements on accessing the videos. Therefore, how to perform video filtering, i.e. obtaining a set of relevant video clips from the video repository becomes a challenging research topic. In previous works, video filtering required user entering some texts to filter the irrelevant video clips, which made the video filtering methods same as the document filtering methods for a long time. However, there are three limitations of the text-based video filtering: (1) it dismisses the rich contents in the videos; (2) it is inapplicable when the texts are absent, incomplete or sparse; (3) it fails to support in-video filtering. These limitations make the text-based video filtering powerless after the new requirements emerge. In recent years, there sees a tendency that computer could parse more meaningful contents from the videos. These non-textual contents are complementary to the texts in many cases. Enlightened by that, existing video filtering research gradually shifts from text-based to non-textual-based. Under this direction, we study how to improve the video filtering systematically from three levels.

Frame-level. We propose to use detected visual object to filter the videos. In previous works, the visual objects were obtained manually where human took the responsibility of identifying the visual objects and connecting them in the videos. The process of obtaining the visual objects is costly when the data keep changing. Therefore, we proposed to leverage the object detection to obtain the visual objects automatically for frame-level filtering. However, object detection itself is unable to identify and connect the visual objects like human. To achieve that, we proposed a hybrid method to identify and connect the visual objects, which is further divided into local merge, propagation and global merge. We examined the proposed method on a real-world dataset then studied two issues: (1) whether the identifications and connections were accurate, as well as (2) how the environment influenced the proposed method. The experimental results were promising and proved that using detected visual objects for frame-level filtering is feasible.

Video-level. We discover a new small content set for surveillance video filtering. Surveillance video filtering, namely surveillance event detection (SED), is important for many safety and security applications. It aims to alarm the events from the surveillance videos. Different from classical video filtering which extracts video content vectors from diverse sources, SED is only able to leverage the motion contents. And the state-of-the-art content set for surveillance is made up of STIP and MoSIFT.

In our study, we proposed a new content set by using dense trajectory (DT) and improved dense trajectory (IDT). According to our analysis, our new content set captures both the individual motions and crowd motions in the surveillance, which leads to higher filtering accuracy in our experiments. Based on the new content set, we investigated how feature transformation, codebook training, encoding process and vector normalization influence the filtering accuracy. The corresponding findings helped us win the TRECVID SED 2015 competition.

User-level. We propose to leverage rich content set to filter the videos. User-level filtering, namely video recommendation, performs personalized filtering for individuals based on user collaboration and video content vectors. Previous works combined the user collaboration with texts to filter the videos. This usually makes the video filtering inaccurate when texts are scarce. In our study, we tried to make user collaboration work with state-of-the-art non-textual content vectors to filter the videos. We used diverse non-textual content vectors to represent the videos, and reproduced existing methods over them. Through the reproduction, we found all of the existing methods have significant drawbacks that limited the filtering accuracy. To address these problems, we proposed the collaborative embedding regression (CER) method to perform more accurate user-level video filtering. Based on CER, we further studied how to combine the results from multiple contents into a unified one. The experiments revealed the high accuracy of the proposed methods in different scenarios. Additionally, the simulation experiment showed that the filtering accuracy is improved when the texts are scarce.

Declaration by Author

This thesis is composed of my original work, and contains no material previously published or written by another person except where due reference has been made in the text. I have clearly stated the contribution by others to jointly-authored works that I have included in my thesis.

I have clearly stated the contribution of others to my thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, and any other original research work used or reported in my thesis. The content of my thesis is the result of work I have carried out since the commencement of my research higher degree candidature and does not include a substantial part of work that has been submitted to qualify for the award of any other degree or diploma in any university or other tertiary institution. I have clearly stated which parts of my thesis, if any, have been submitted to qualify for another award.

I acknowledge that an electronic copy of my thesis must be lodged with the University Library and, subject to the policy and procedures of The University of Queensland, the thesis be made available for research and study in accordance with the Copyright Act 1968 unless a period of embargo has been approved by the Dean of the Graduate School.

I acknowledge that copyright of all material contained in my thesis resides with the copyright holder(s) of that material. Where appropriate I have obtained copyright permission from the copyright holder to reproduce material in this thesis.

Publications during candidature

- Xingzhong Du, Hongzhi Yin, Zi Huang, Yi Yang, Xiaofang Zhou. *Using Detected Visual Objects To Index Video Database*. Australasian Database Conference (2016): 333-345.
- Xingzhong Du, Yan Yan, Pingbo Pan, Guodong Long, Lei Zhao. *Multiple Graph Unsupervised Feature Selection*. Signal Processing, Vol. 120, (2016): 754-760.
- Shicheng Xu, Huan Li, Xiaojun Chang, Shoou-I Yu, Xingzhong Du, Xuanchong Li, Lu Jiang, Zexi Mao, Zhenzhong Lan, Susanne Burger, Alexander Hauptmann. *Incremental Multimodal Query Construction for Video Search*. International Conference on Multimedia Retrieval (2016): 675-678.
- Tieke He, Zhenyu Chen, Jia Liu, Xiaofang Zhou, Xingzhong Du, Weiqing Wang. *An Empirical Study on User-Topic Rating Based Collaborative Filtering Methods*. WWW Journal (accepted in 2016).
- Xingzhong Du, Xuanchong Li, Xiaofang Zhou, Alexander Hauptmann. *WARD-CMU @ TRECVID 2015*. Proceedings of TRECVID. (2015)
- Shoou-I Yu, Lu Jiang, Zexi Mao, Xiaojun Chang, Xingzhong Du, Chuang Gan, Zhenzhong Lan, Zhongwen Xu, Xuanchong Li, Yang Cai, Anurag Kumar, Yajie Miao, Lara Martin, Nikolas Wolfe, Shicheng Xu, Huan Li, Ming Lin, Zhigang Ma, Yi Yang, Deyu Meng, Shiguang Shan, Pinar Duygulu Sahin, Susanne Burger, Florian Metze, Rita Singh, Bhiksha Raj, Teruko Mitamura, Richard Stern, Alexander Hauptmann. *Informedia @ TRECVID 2014 MED and MER*. Proceedings of TRECVID. (2014)

Publications included in this thesis

Xingzhong Du, Hongzhi Yin, Zi Huang, Yi Yang, Xiaofang Zhou *Using Detected Visual Objects To Index Video Database*. Australasian Database Conference (2016): 333-345. -incorporated as Chapter 3.

Contributor	Statement of contribution
Xingzhong Du	Experiment design and conduction (100%) Paper writing (70%)
Hongzhi Yin	Proof reading (70%)
Zi Huang	Proof reading (20%)
Yi Yang	Proof reading (10%)
Xiaofang Zhou	Paper writing (30%)

Xingzhong Du, Xuanchong Li, Xiaofang Zhou, Alexander Hauptmann *WARD-CMU @ TRECVID 2015*. Proceedings of TRECVID. (2015) -incorporated as Chapter 4.

Contributor	Statement of contribution
Xingzhong Du	Experiment design and conduction (100%) Paper writing (70%)
Xuanchong Li	Proof Reading (60%)
Xiaofang Zhou	Paper writing (30%)
Alexander Hauptmann	Proof Reading (40%)

Shicheng Xu, Huan Li, Xiaojun Chang, Shoou-I Yu, Xingzhong Du, Xuanchong Li, Lu Jiang, Zexi Mao, Zhenzhong Lan, Susanne Burger, Alexander Hauptmann *Incremental Multimodal Query Construction for Video Search*. International Conference on Multimedia Retrieval (2016): 675-678. -incorporated as Chapter 5.

Contributor	Statement of contribution
Shicheng Xu	Experiment design and conduction (50%) Paper writing (50%)
Huan Li	Experiment design and conduction (15%) Paper writing (20%)
Xiaojun Chang	Experiment design and conduction (15%) Paper writing (10%)
Shoou-I Yu	Experiment design and conduction (10%) Paper writing (15%)
Xingzhong Du	Experiment design and conduction (10%) Paper writing (5%)
Others	Proof Reading (100%)

Contributions by others to the thesis

For all the published research work included in this thesis, Prof. Xiaofang Zhou, as my principle advisor, has provided very helpful insight into the overall as well as the technical details and research problems; guidance for problem formulation as well as constructive comments and feedback. He also assisted with both the refinement of the idea and the pre-submission edition.

Statement of parts of the thesis submitted to qualify for the award of another degree

None.

Acknowledgments

I would like to express my special thanks to my principal supervisor, Prof. Xiaofang Zhou, for his generous support, and his valuable and in-depth guidance for my PhD study and research. In the past days, I learned a lot from him. With his help, I learned how to discover fresh and intriguing research topics, how to peer-review publications, and how to write good papers. The research experience will also be of great benefit in my future career.

I would also like to thank my associative supervisor, Dr. Yi Yang. I appreciate all his contributions from multimedia domain, to make my Ph.D. experience stimulating and rewarding.

I am very thankful to Dr. Hongzhi Yin. He gave me very valuable advises on recommendation research and related technical details.

Keywords

non-textual, video filtering, visual objects, retrieval, recommendation

Australian and New Zealand Standard Research Classifications (ANZSRC)

ANZSRC code: 080109, Pattern Recognition and Data Mining, 50%

ANZSRC code: 080201, Analysis of Algorithms and Complexity, 20%

ANZSRC code: 080604, Database Management, 30%

Fields of Research (FoR) Classification

FoR code: 0806, Information Systems 50%

FoR code: 0801, Artificial Intelligence and Image Processing, 40%

FoR code: 0803, Computer Software, 10%

Contents

1	Introduction	1
1.1	Background	1
1.2	Problem Statement	3
1.2.1	Frame-level Filtering	3
1.2.2	Video-level Filtering	4
1.2.3	User-level Filtering	4
1.3	Contributions	5
1.3.1	Frame-level Filtering Using Detected Visual Objects	5
1.3.2	Video-level Filtering Using Small Non-textual Content Set	6
1.3.3	User-level Filtering Using Rich Content Set	6
1.4	Thesis Organization	7
2	Literature Review	9
2.1	Overview	9
2.2	Visual Object Detection	10
2.2.1	Classifier Based Method	10
2.2.2	Neural Network Based Method	11
2.3	Visual Object Tracking	12
2.3.1	Classifier Based Method	12
2.3.2	Neural Network Based Method	13
2.4	Video Content Vector	13
2.4.1	Textual Content Vector	13
2.4.2	Non-textual Content Vector	14
2.4.3	Vector Normalization	17

2.5	Video Filtering	18
2.5.1	Frame-level Filtering	18
2.5.2	Video-level Filtering	20
2.5.3	User-level Filtering	22
2.6	Summary	23
3	Frame-level Filtering using Detected Visual Objects	25
3.1	Introduction	25
3.2	Problem Statement	29
3.2.1	Preliminaries	29
3.2.2	Method Overview	31
3.3	Related Work	32
3.4	Proposed Method	34
3.4.1	Detection	34
3.4.2	Local Merge	35
3.4.3	Propagation	39
3.4.4	Global Merge	41
3.4.5	Complexity Analysis	42
3.4.6	Running Example	43
3.5	Experiment	44
3.5.1	Settings	44
3.5.2	Evaluation Plan	49
3.5.3	Accuracy	50
3.5.4	Efficiency	52
3.5.5	Impact of Factor	52
3.5.6	Analysis	53
3.6	Summary	55
4	Video-level Filtering Using Small Non-textual Content Set	57
4.1	Introduction	57
4.2	Problem Statement	59
4.2.1	Preliminaries	59

4.2.2	System Overview	60
4.3	Related Work	61
4.4	Proposed System	62
4.4.1	Preprocessing	62
4.4.2	Feature Extraction	63
4.4.3	Feature Encoding	64
4.4.4	Model Training	65
4.4.5	Score Fusion	67
4.5	Experiments	67
4.5.1	Dataset	67
4.5.2	Evaluation Plan	68
4.5.3	Experimental Results	69
4.6	Summary	72
5	User-level Filtering Using Rich Content Set	73
5.1	Introduction	73
5.2	Problem Statement	76
5.2.1	Preliminaries	76
5.2.2	System Overview	77
5.3	Related Work	77
5.4	Proposed System	78
5.4.1	Content Vector Generation	78
5.4.2	Video Recommendation	80
5.5	Experiments	86
5.5.1	Dataset Description	86
5.5.2	Experimental Settings	87
5.5.3	Experimental Results and Analysis	90
5.6	Summary	95
6	Conclusion and future work	97
6.1	Conclusion	97
6.2	Future work	98

List of Figures

1.1	Multi-level video filtering	2
2.1	The structure of literature review	10
2.2	The pipeline of non-textual vector generation	14
3.1	The data structure for supporting frame-level filtering.	26
3.2	Visual object is located by a minimum bounding rectangle on a video frame	30
3.3	The pipeline of the proposed method.	32
3.4	The revised data structure for supporting frame-level filtering.	33
3.5	Four factors prevent image matching correctly in videos.	35
3.6	Use continuity to improve matching results.	36
3.7	A running example to display how the frame-level filtering is performed	44
3.8	The benefits from unique object table and occurrence table.	54
3.9	Share comparison	54
4.1	The pipeline of proposed video-level filtering system	60
4.2	best aDCR for PersonRuns in recent five years' retrospective and interactive systems.	72
5.1	Implicit rating matrix for in-matrix and out-of-matrix recommendation.	76
5.2	The flowchart of exploiting rich contents to recommend videos.	79
5.3	Performance of the state-of-the-art methods in both in-matrix and out-of-matrix settings. To clearly display the methods which only support in-matrix recommendation, we shift the origin of the vertical axis to a higher position.	81
5.4	Accuracy@k of different methods under in-matrix setting	90
5.5	Accuracy@k of different methods and features in out-of-matrix setting.	91

5.6	Out-of-matrix recommendation accuracy in the text sparsity setting.	94
-----	---	----

List of Tables

3.1	Differences between the assistant methods where \checkmark denotes yes and \times denotes no . . .	27
3.2	Summary of notations	29
3.3	Server Configuration	46
3.4	The accuracy of object identifying	51
3.5	The accuracy of object connecting	51
3.6	Impact of factor	53
4.1	Differences between normal and surveillance videos	58
4.2	Differences between IDT, STIP and MoSIFT	63
4.3	The statistics of events on training videos	68
4.4	Evaluations for fusion strategy	69
4.5	Filtering accuracy with resized and original videos	70
4.6	Fusion under resized videos and original videos	70
4.7	Ground True (GT), Positive Miss (P_{miss}) and False Alarm (FP) comparison	71
4.8	Competition results in TRECVID SED 2015	71
5.1	The dimensions of the encoded non-textual content vectors.	79
5.2	The state-of-the-art recommender models and the corresponding content features in use.	80
5.3	An example of the weights generated in the late fusion method when p is set to 0.5.	86
5.4	Fusion results on different feature combinations	92
5.5	Training efficiency of WMF-based models.	94

Chapter 1

Introduction

In this chapter, we give a brief introduction of the research in this thesis, including background, problem statements, contributions, and organization of the thesis.

1.1 Background

As the video recording devices become popular, more and more videos are generated every day. The numerous amount of the videos enriches the choices of the users but also enlarges the difficulty of accessing useful information. For example, on Youtube¹, there are 300 hours of video uploaded every minute². This makes browsing all the videos then choosing the useful ones impossible. To improve this situation, many websites provide the video filtering services whose core function is to select the most relevant video clips from the huge repository.

The video filtering services are divided into three levels as shown in Figure 1.1, namely, frame-level, video-level and user-level. They support different process granularities.

- **Frame-level filtering** leverages the annotations on the frames to generate the most relevant video clips. It accepts a set of annotations given by the users, and filters off the frames which do not have the given annotations. After filtering, the remaining frames are reformed into video clips as the output;
- **Video-level filtering** leverages the video content vectors to filter the videos. It accepts a set

¹<https://www.youtube.com/>

²<http://www.statisticbrain.com/youtube-statistics/>

of exemplar videos as input, and extracts the content vectors accordingly. During the filtering, the videos whose contents are dissimilar with the exemplar videos' are filtered off. After thresholding, the remaining videos are returned as output;

- **User-level filtering** leverages the rating matrix and video content vectors to filter the videos in a personalized way. Different from the frame-level and video-level filtering which requires user explicit inputs, the user-level filtering can leverage the user implicit feedbacks as input. After filtering, the videos which the user has no feedbacks but potentially likes are returned as result.

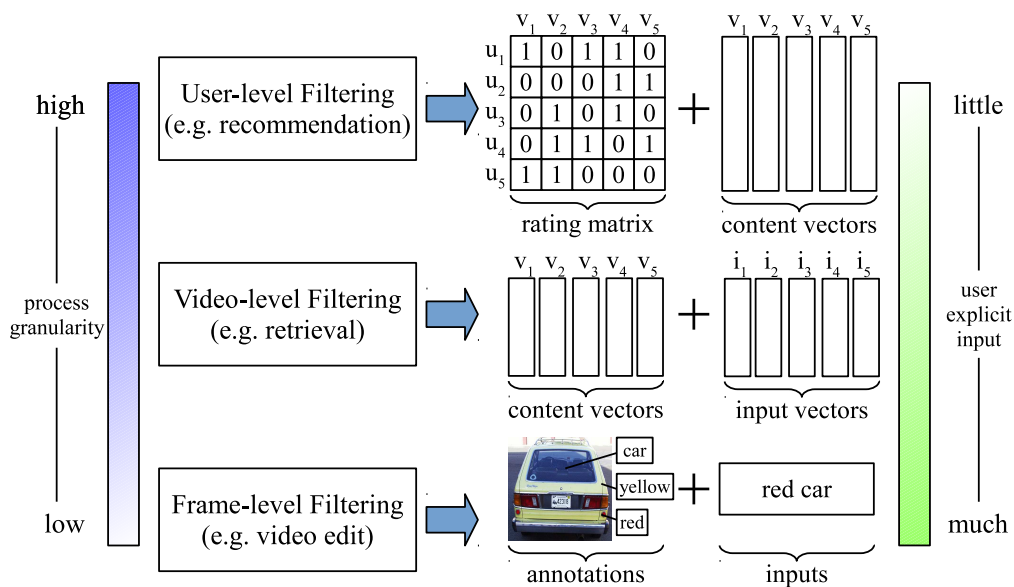


FIGURE 1.1: Multi-level video filtering

In previous works [74, 106], all the filtering methods heavily depend on the texts associated with the videos. For instance, the keyword based video search is a kind of video-level filtering methods. It requires the video providers to generate the textual descriptions for the videos, and the users to provide the key words guiding the filtering explicitly. Obviously, when the texts associated with the videos are sparse, the keyword based search is inapplicable. The same problem is shared by the other text based methods. To improve the situation, exploiting the non-textual contents in the videos has drawn considerable attention in recent years.

The widely used non-textual contents are as follows:

- **Visual Object.** The images of real-world objects projected in the video frames as well as their locations and occurrence time in the videos;

- **Video Content Vector.** The aggregated content vectors derived from the raw features extracted from the videos. The raw feature source channels usually are:
 - **Audio.** The audio changes in the sound tracks of the videos;
 - **Scene.** The texture, object and scene information in the video frames;
 - **Motion.** The changes between two adjacent frames in the videos;
- **User Collaboration.** The co-feedback behaviors shared by the users.

The video filtering methods on different levels exploit the non-textual contents in various ways:

- For frame-level filtering, the visual objects are exploited to replace the textual annotations. Existing works [63, 24, 57] employ human labors to annotate the visual objects in the videos;
- For video-level filtering, the non-textual content vectors are exploited to replace the textual content vectors. Existing works [4, 123] extract the content vectors in terms of audio, scene and motion respectively and try to make the content set as rich as possible;
- For user-level filtering, the behavior matrix and non-textual content vectors are exploited to perform the filtering. Existing works have already investigated the benefits from single content type such as MFCC [78] and CNN [42], as well as the benefits from fusion of CNN and text [124].

In summary, the research of multi-level video filtering using non-textual contents is valuable. In the next section, we will describe the problems we addressed in our study.

1.2 Problem Statement

The data sparsity problem suffers the availability of video filtering, even though the non-textual contents have been introduced recently. In the following parts, we will explain the specific problems for each level one by one.

1.2.1 Frame-level Filtering

The frame-level filtering aims at remaining the frames which meet the constraints given by the users. The applications widely exist in video editing, surveillance, retrieval and so on [57, 24, 103, 63].

When the users provide the visual objects as the constraints, the filtering should remain the frames which contain these objects simultaneously. This requires all the visual objects are annotated in advance, which is infeasible at present. Existing works [57, 24, 115] employ human to obtain the visual objects on each frame. As the growth rate of videos is unprecedentedly high, the manually annotated visual objects are usually scarce on large-scale or dynamic data. In order to improve that, there calls for a more automatic way to discover the visual objects on the frames. Accordingly, we try to introduce object detection instead of human annotation in our study [27]. However, the object detection only returns the object occurrences on each frame. It cannot make the visual objects identified and connected as human. We have reviewed some methods [69, 114] which could identify or connect the visual objects. But none of them can support identifying and connecting at the same time.

1.2.2 Video-level Filtering

Video-level video filtering has been studied for many years as video retrieval. It aims at returning the most relevant videos according to their content relevances to the exemplars. The applications widely exist in video classification, video caption, multimedia event detection and so on. Existing works [4, 123] try to make the content set as rich as possible. This is achieved by extracting content vectors from multiple sources with different methods [4, 123, 109, 116]. However, the rich content set is not always available. For surveillance scenario, as the videos are muted, untrimmed and full of noise, the applicable content set is usually small. For example, the state-of-the-art retrospective system [14] only equips with STIP [61] and MoSIFT [12] for surveillance. Obviously, the smaller content set limits the performance of the surveillance video-level filtering. Recent work in [123] introduces a more powerful content, namely improved dense trajectory (IDT) [109], into the surveillance. However, IDT fails to improve the filtering accuracy alone. In addition to that, the situation is not improved even though IDT is fused with STIP and MoSIFT.

1.2.3 User-level Filtering

User-level video filtering generates outputs based on the user implicit feedbacks [47, 87] and video content vectors [110]. It aims at generating a personalized top-k videos which has not been watched for each user. To facilitate the learning process, the user implicit feedbacks are transformed into rating

matrix. According to whether the unwatched videos are in the rating matrix, the filtering operates in two scenarios [106]. The first scenario is in-matrix where the unwatched videos are in the rating matrix but not rated by the target users. The filtering results are dominated by the rating matrix. The second scenario is out-of-matrix where the unwatched videos are not in the rating matrix. The filtering results are dominated by the video content vectors. Most of the existing methods [78, 110, 42, 124] only focus on how to leverage the content vectors to improve the in-matrix filtering where the rating matrix has a considerable amount of ratings. The corresponding findings are not very helpful for out-of-matrix filtering where the rating matrix is sparse or inapplicable [106]. Therefore, how to improve the out-of-matrix filtering with single content type and how to fuse multiple contents to achieve higher accuracy are still challenging.

1.3 Contributions

1.3.1 Frame-level Filtering Using Detected Visual Objects

Object detection discovers the visual objects on the video frames, but it fails to identify and connect them to support frame-level filtering. Some assistant methods can overcome part of the drawbacks but they cannot provide the complete support for identifying and connecting. To improve this problem, we proposed a hybrid method which consists of matching-based and tracking-based methods to assist object detection. The hybrid method has three steps, namely local merge, propagation and global merge. They have following responsibilities in our hybrid method:

- Local merge identifies the visual objects discovered by object detection locally;
- Propagation connects the visual objects in the videos locally;
- Global merge identifies and connects the visual objects from different videos globally.

Our experiments show that the proposed hybrid method has achieved higher overall accuracy than the existing assistant methods. It costs less time than object detection and discovers more object occurrences for frame-level filtering.

1.3.2 Video-level Filtering Using Small Non-textual Content Set

In this study [25, 26], we try to figure out why improved dense trajectory (IDT) performs poor in the surveillance scenario. We find that it is because IDT applies dense sampling and camera removal. These properties also make IDT fail to fuse with STIP and MoSIFT. Therefore, in our work, we choose another content type, dense trajectory (DT), to fuse with IDT. With the new content set, we carefully examine the impacts from different factors. The significant findings are: (1) the new content set is much more accurate than the old content set which is made up of STIP and MoSIFT; (2) the surveillance videos must be resized before feature extraction, otherwise the performance drops a lot; (3) whiten principle component analysis (whiten PCA) is beneficial to the filtering accuracy; (4) the event durations influence the filtering accuracy. According to above findings, we implemented a retrospective system to perform video-level filtering on the surveillance videos. The proposed system helped us obtain the first place in the TRECVID-SED competition in 2015.

1.3.3 User-level Filtering Using Rich Content Set

In this study [28], we firstly try to figure out the real limitations of the existing method [47, 87, 106, 78, 110, 42]. We reproduce them on our dataset and evaluate their accuracies in both in-matrix and out-of-matrix scenarios. Our reproduction shows that none of the existing methods perform well in both in-matrix and out-of-matrix scenarios with the non-textual content vectors. To overcome the limitations, we propose the collaborative embedding regression (CER) method in our work. CER performs well in both scenarios compared to the existing methods. In addition to that, we investigate how to fuse multiple contents to achieve higher filtering accuracy in the out-of-matrix scenario. We study both early and late fusion strategies, then propose a new late fusion strategy in our work. The experiment shows that the proposed late fusion strategy achieves the highest accuracy compared to average, learning-to-rank and early fusion strategies. Based on that, our simulation experiment indicates that our findings make the user-level filtering more accurate even though the texts are scarce or inapplicable.

1.4 Thesis Organization

The rest of this thesis is organized as follows: In Chapter 2, we review the fields of visual object detection, video content vectors and filtering methods on different levels. In Chapter 3, we analyze the drawbacks of existing works on frame-level filtering, then present our improvement in this area. In Chapter 4, we analyze the drawbacks of the existing content set and present our improvement for surveillance filtering. In Chapter 5, we describe and display the reproduction of the state-of-the-art user-level filtering methods, then figure out none of these methods can achieve the highest accuracy in both scenarios. We therefore propose the collaborative embedding regression (CER) method to overcome the limitations, and further propose a new late fusion strategy. Finally, the conclusions and the future research directions suggested by the thesis are given in Chapter 6.

Chapter 2

Literature Review

In this chapter, we review the literatures related to multi-level video filtering, which includes the non-textual content generation and existing works for the filtering.

2.1 Overview

The key components of video filtering are the non-textual content generation and the methods. For frame-level filtering, the visual objects need to be generated by detection. Therefore, we will review how the existing works detect the visual objects from the frames. In addition to that, we will review how the existing works connect the visual objects by tracking. For video-level filtering, the video content vectors need to be generated. Accordingly, we will review the literatures about how to generate the content vectors for the videos. Since our video-level study focuses on the surveillance, we will also review how to perform video-level filtering for surveillance. For user-level filtering, the rating matrix and content vectors need to be generated. Since the generation of rating matrix is simple and the generation of content vectors is overlapped with video-level filtering, we will skip this part. Alternatively, we will focus on reviewing the main user-level filtering methods which include collaborative filtering, content-based filtering and hybrid filtering. The organization and connection of the literature review is illustrated by Figure 2.1. In details, in Section 2.2, we will review the literatures about object detection and tracking; in Section 2.4, we will review the literatures about the video content vector generation; in Section 2.5, we will review the literatures about multi-level video filtering methods covered by this thesis.

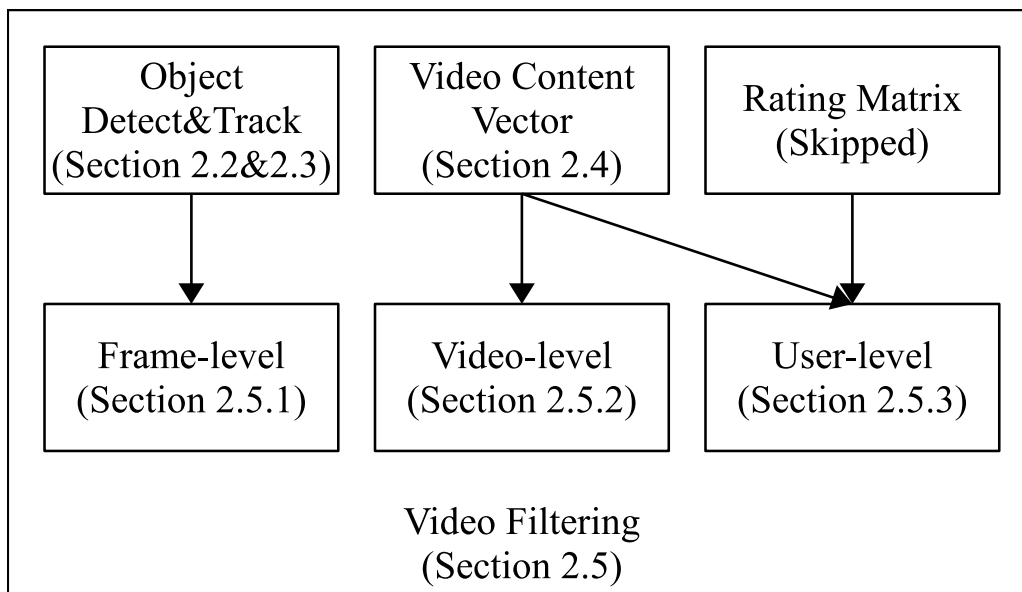


FIGURE 2.1: The structure of literature review

2.2 Visual Object Detection

Object detection is a popular research topic in computer vision area. It aims at discovering objects from images and videos by their appearances. Here, in order to differentiate the concept of object in ontology, we use visual object instead in the following statements. Recent years, as the neural networks become popular, the detection method is gradually divided into two branches. The first branch is classifier based method which obtains handcrafted feature for learning and uses linear classifier as detector. The second branch is neural network based method which integrates feature learning and detector learning into one uniform framework. In this section, we will briefly review the major progress in these two branches.

2.2.1 Classifier Based Method

The classifier based method treats the visual object detection as classification. The typical detection process has several steps [105]: firstly, the train images are spliced into many small patches; secondly, the handcrafted visual features are extracted from the small patches; thirdly, the classifier is trained on the training set where the positives are the patches having large overlap with the ground truth and the negatives are the patches having no overlap with the ground truth; finally, after the test images are sliced, extracted and classified, the positive results are suppressed to generate the final result.

[105] proposed a early system for detection. The classifier in use was AdaBoost [34] and the handcrafted features in use were simple rectangle features. [105] has several limitations. The first limitation is that the rectangle features do not have scale, transformation and rotation invariant properties. This makes detection inaccurate in many cases. The second limitation is that the classifier in use is a decision tree based method. It is not efficient with high dimension feature vector. To overcome above limitations, new handcrafted features such as color histogram, SIFT [69], SURF [7], HOG[18], ORB [91] and etc. were introduced into visual object detection [70]. [31] proposed a structural classifier, namely discriminatively trained part-based model (DPM), to achieve more accurate visual object detection. The base classifier is support vector machine (SVM) [11]. Instead of treating the image patches as one object, the authors in [31] treat the images patches as a bag of decomposed parts. Through training on separate parts and the whole image patches, the proposed DPM performs detection more accurately when the objects are occluded. [58] proposed a method to detect objects in the videos. It first trains the model on the ImageNet dataset. Then, it detects objects and tracks the high confident objects in the videos. Then, it updates the model by the newly detected and tracked objects. Compared to previous methods which only performed training on the images, the method in [58] gradually update the model to adapt the new changes from the videos. In object detection, some image patches may belong to the same objects. To suppress them into one image patch, there usually applies a non-maximum suppression (NMS) after classification. [75] leverages integral image to accelerate the NMS process efficiently.

Recent researches on visual object detection focus on how to generate accurate object proposals. Compared to image patches, the object proposals indicate there exist objects and they are usually of smaller amount. [102] proposed a selective search method. Instead of splicing images into many patches, selective search leverages low-level color features and to filter object proposals hierarchically. [15] proposed an efficient method to generate the object proposals. It transforms the image patches into HOG space and encode the features into binary code. Together with an efficient selection algorithm, the speed of generating the object proposals can reach 300 fps. [56].

2.2.2 Neural Network Based Method

The neural network based method tries to integrate different parts of detection pipeline into an end-to-end learning process. It is different from the classifier based method which divides the detection into four steps and improve the accuracy of each step independently. The early version of neural network

method was proposed in [100]. It leveraged pretrained CNN model based on ImageNet dataset [92] and append a MBR regression layer to generate MBR. This very first work shows promising result on object detection. A more accurate work was proposed in [86]. It proposes a faster region-based convolutional neural network (R-CNN). Compared to [100], R-CNN perform co-training on object recognition and detection, which shows state-of-the-art detection accuracy. Besides, some recent works try to locate objects not with MBRs. [35] uses the features learned by CNN to perform image segmentation. In this direction, the pixels which are labeled as same object classes are combined to represent the objects. The follow-up work in [125] improves the accuracy by training conditional random field as recurrent neural network.

2.3 Visual Object Tracking

Visual object tracking aims at locating given objects in continuous video frames [121]. It can perform accurate localization even though the tracked objects have large transformation. Like visual object detection, the tracking method can be also divided into two branches, namely, classifier based method and neural network based method. In this section, we will briefly review the major literatures in this area.

2.3.1 Classifier Based Method

Classifier based method treats the input image as positive exemplar and tries to separate it from the background. Nowadays, some works can achieve real-time tracking. [37] achieved real-time tracking by a novel on-line AdaBoost algorithm method. It treats the given visual object as positive exemplar and the image patches around as the negative exemplar. Together with local binary pattern features, its tracking speed is very fast. However, the method in [37] is not very accurate when the objects in the videos are moving too fast. This problem is called tracing drift. Multiple instance learning (MIL) method was proposed in [6]. MIL crops some image patches which have large overlap with the given visual objects as positive exemplars. Since the positives are enriched, MIL is better than the method in [37] on handling tracking drift. Some works try to improve the tracking by automatically providing the input. STRCUK tracker is proposed in [38], it leverages the structural outputs from multiple kernels to improve the drift problem.

The object tracking becomes more and more robust recently. However, there is still a common

problem shared by all the object tracking methods. That is, they cannot stop when the given visual objects disappear in the videos [121, 114]. Accordingly, some methods leverage object detection to provide the input for the object tracking, namely, tracking-by-detection [9]. They use a frame counter to decide whether the tracking should stop when the detection has no results for several frames. In [9], a tracking-by-detection method was proposed to track the pedestrians in the videos. This method leverages detection to provide objects for the tracking. In order to ensure the accuracy, the proposed method calculates the similarities between the detected objects and tracked objects on each frames and applied a greedy strategy to match the detected objects and the tracked objects. With some common technique tricks, the method in [9] can accurately track the pedestrians in the videos. [9] is enhanced in [51] by learning positives from the tracking results.

2.3.2 Neural Network Based Method

As the convolutional neural networks (CNN) become popular in the computer vision community, numerous CNN based tracking methods have been proposed in recent years. Deep learning tracking method is proposed in [112]. It leverages stacked de-noising auto-encoder (SDAE) to learn the object appearance. Since SDAE models the object appearances robustly based on a large mini image dataset, the tracking accuracy is improved significantly. In [66], the robust tracking is achieved by on-line convolutional neural network. Like MIL in [6], [66] crops multiple instances around the positive and use these new instances as positives to train the CNN model. A recent comprehensive comparison between different trackers could be found in [114].

2.4 Video Content Vector

2.4.1 Textual Content Vector

Traditional content-based video filtering systems [20, 110, 36] capture the video contents by texts. The textual contents often include titles, descriptions, reviews as well as meta information for the videos. Based on these texts, two kinds of textual features were extracted frequently: word features and meta features. To construct the word vector, the title, description and reviews associated with the given video are concatenated into one virtual document. After removing stop words and stemming [106, 110], the top discriminative and meaningful words are selected by TF-IDF value to

compose the word vectors. The meta vector stores the meta data about the video such as its producers, countries, languages, release dates, actors, genres and so on [2, 36, 39]. The top discriminative meta items are selected by global frequency to form the codebook. Unlike the word vector where a word may appear more than once, the meta item in the meta vector just appears once. Accordingly, the meta vector is binary and usually very sparse.

2.4.2 Non-textual Content Vector

There are three components to form the non-textual content vectors, namely, raw features, encoding method, and vector normalization. The pipeline is illustrated in Figure 2.2. In the following parts, we will describe the function of each component.

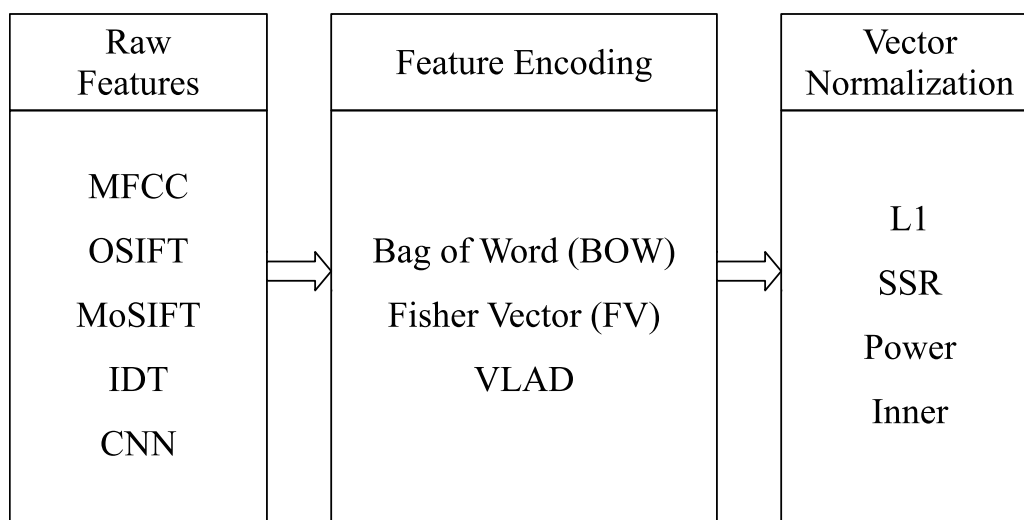


FIGURE 2.2: The pipeline of non-textual vector generation

Raw Feature

In addition to the textual features, videos themselves also contain rich content information. Yang et. al [118] and Deldjoo et. al [21] extract the normalized color histogram and aural tempos to represent the videos. However, the experimental results reported in [118, 21] show that these features are not significantly effective in improving video filtering. This is because these features fail to distinguish between videos that share similar colors but are unrelated in content. For example, given a video about the sky and another video about the sea, the normalized color histogram will result in a high

similarity between the two videos due to the common color blue. In this case, it is very likely that sky-related videos will pass the filtering to the users who like seas.

The limitations of the normalized color histogram and aural tempos do not mean that all non-textual video features are useless for video filtering. In fact, some non-textual features have been proven to be effective in recent video filtering applications [74, 104, 4, 109, 96]. The representative features are MFCC, SIFT, IDT and CNN. We will review their functions and extractions as follows.

1. **MFCC (mel-frequency cepstral coefficients)** [74] measure the audio changes in sound by computing the cosine values from multiple channels between adjacent time points. MFCC features can be extracted through the following steps [4]: 1) down-sampling the audio track of a video to 16 kHz with 16 bit resolution; 2) using a window size of 25 ms and a step size of 10 ms to set the MFCC extractor and setting the number of channels to 13; and 3) concatenating MFCC and their first and second derivatives as well as energy to form a 40 dimensional feature. Each time window will obtain a feature. Accordingly, an audio file will result in a feature array after extraction while the length of the array is proportional to the audio duration.
2. **SIFT (scale invariant feature transform)** [69] captures the texture information of images. Since SIFT features can match the same visual objects of different scales [69], it has been widely applied to scene classification [104] and image retrieval [50]. There are two kinds of SIFT variants widely used nowadays. They are OSIFT (opponent SIFT) [104] and MoSIFT (motion SIFT) [12]. OSIFT transforms the original RGB color space by light color change and shift, which provides more robust SIFT features. An OSIFT feature has 384 dimensions. MoSIFT leverages the optical flow between frames to select SIFT features so as to capture some motions in the videos. A MoSIFT feature has 256 dimensions. For both OSIFT and MoSIFT, the length of feature array after extraction is uncertain. Rich texture information and serious motions will result in long length.
3. **IDT (improved dense trajectory)** [109] captures motion information in videos. IDT currently is the state-of-the-art handcrafted motion features. Its early version, dense trajectory, was proposed in [107]. The key idea is to use dense sampling key points instead of sparse sampling key points to capture the motion information. IDT improves DT in two folds: first, it uses homography to wrap the camera motion from the optical flow; second, it uses weakly human

detectors to refine the camera motion. IDT uses 2D normalized trajectory, HOG [18], HOF [19] and MBH [107] to describe motions. This makes an IDT feature has 426 dimensions.

4. **CNN (convolutional neural network)** [56] captures the semantic information in images. Recently, CNN has shown its advantage over the other models in the object classification competition [93]. Some recent research shows that using a pre-trained CNN on ImageNet to extract features from images is beneficial for video retrieval [116]. Inspired by its superior performance in video search, we first sample frames from a video and then use the pre-trained CNN model from the VGG group [96] to extract visual features from the pool_5 layer. The original pool_5 features are tensors. We apply spatial pooling to transform tensor into vector, following [41]. Thus, each sampled frame has 49 CNN features with 512 dimensions.

Unlike MFCC, MoSIFT and IDT which take the whole audio or video file as input, OSIFT and CNN are applied to the frames sampled from the video. Following [4, 116], 5 frames should be fetched uniformly every second from the video. After that, there is usually a normalization process on the raw features. The state-of-the-art method is SSR (signed squared root), we will introduce it later in normalization part.

Feature Encoding

After feature extraction, each video obtains an array of features. These arrays are not ready because video filtering needs them to be vectorized. Feature encoding is the process to transform the array into vector by quantizing the features. Until now, there are three state-of-the-art feature encoding methods. They are bag-of-word (BOW), fisher vector (FV) and VLAD. We will introduce them by the publish date from early to late.

1. **Bag-of-Word (BOW)** [65] quantizes the features by the visual words which are usually generated by k-means clustering algorithm. BOW simulates the word vector. It calculates the Euclidean distances between features and visual words, assigns the features with the nearest cluster index, then counts the frequency in the corresponding bins. The assignment is further divided into hard assignment [65] and soft assignment [67]. The hard assignment only assigns the nearest cluster index while soft assignment assigns k nearest cluster indices. Given K centroids, the dimension of the encoded vector is K .

2. **Fisher Vector (FV)** [82] quantizes the features by Gaussian mixture model (GMM). It firstly generates the GMM by EM algorithm on the sample data, then calculates the derivatives with regard to the means and variances of GMM to concatenate a vector given the feature array. The variances of GMM for fisher vector is a symmetry matrix. In practice, this increases the computation dramatically. [94] leverages PCA to make the variances into diagonal matrix which increases the efficiency significantly. In later work [80], whiten PCA is applied to improve the recognition accuracy. Compared to BOW, FV's dimension is proportional to both the number of Gaussian components and the feature dimension. Given K Gaussian components, D dimensional features and half feature dimension reducing by PCA, the dimension of a FV vector is KD .
3. **VLAD** [50] quantizes the features by centroids which are usually generated by k-means clustering algorithm. However, instead of recording the assignment information, VLAD concatenates the differences between the features and the centroids to form a vector. Like FV, VLAD also applies PCA to reduce the feature dimension to half. Given K centroids and D dimensional features, the dimension of a VLAD vector is $\frac{KD}{2}$.

2.4.3 Vector Normalization

After feature encoding, another important step for video content vector generation is normalization. For different encoding methods, there are different ways to normalize the vectors. In this paper, we will introduce the state-of-the-art normalization methods.

1. **L1 normalization** [119] makes the sum of the values in the vector equal to one. It is usually applied on the content vector derived by BOW, because it generates histogram-based vector. The similarity between two BOW vectors is therefore measured by chi-square distance.
2. **Signed Square Root (SSR) normalization** [5] makes the values in the vector squared by the absolute value but remain the signs. SSR are usually applied on raw features and encoded content vectors. After SSR, there is usually following the power normalization.
3. **Power normalization** [82] makes the length of the vector equal to one. This is because FV and VLAD generate gradient-based vectors. The similarity between FV or VLAD vectors is therefore measured by cosine distance.

4. **Inner normalization** [80] is a variant of power normalization. Instead of making the length of whole vector equal to one, inner normalization separate the whole vector into several parts and make their length equal to one. The similarity is still measured by cosine distance.

2.5 Video Filtering

2.5.1 Frame-level Filtering

The frame-level filtering using non-textual contents has been studied for many years in video database area. The research mainly focuses on designing new filtering queries to make a better use of the video data. Until now, three query types have been proposed in this area, namely, low-level query, spatial-temporal query and semantic query [24]. The low-level query is equivalent to content-based video retrieval [83]. Given some example videos, the query leverages the low-level visual and audio features to search the similar videos [46]. In recent years, the low-level query has been improved a lot by the semantic feature [56][35], motion feature [109] and feature encoding methods [82][50]. However, the granularity of low-level query is not low. That is to say, if the user want to search some objects in the video database, the low-level query cannot provide accurate evidences. According to this, the spatial-temporal query is proposed to achieve lower granularity [53]. It is the query whose conditions include any combination of directional relations, topological relations, 3D relations, external-predicate, object-appearance, trajectory projection, and similarity-based object trajectory [24]. In the spatial-temporal query, the user is allowed to use the visual objects as the input. Together with the spatial and temporal relations, arbitrary segments of videos can be returned in response to user queries [24]. The semantic query is a high-level query type. In previous works, the semantic query is usually decomposed into some sub-queries which are made up of the low-level query and the spatial-temporal query [1][83].

Many frame-level filtering systems using non-textual contents have been proposed in the past. OVID is proposed in [77]. OVID uses video sequences as the input and designs VideoSQL for the users to retrieve other video clips. However, OVID does not support spatial-temporal query. [33] proposes a Query by Image and Video Content (QBIC) system. Essentially, QBIC only supports the low-level query for the video frames. It achieves querying videos by using the most representative frames from the video shots. However, QBIC does not support spatial-temporal query either. In

[1], the Advanced Video Information System (AVIS) is proposed. It uses the frame segment tree to organize the video sequences and a set of arrays to store the objects, events and their associations. It supports the semantic query and the spatial query but it only accepts texts as the query input. A content-based video query language (CVQL) is proposed in [57]. CVQL treats the objects as points and express the spatial relations by distance or motion. It also designs indices to accelerate the query processing. BilVideo is proposed in [103]. It has three modules, namely, fact-extractor, video-annotator and object extractor, for assisting users to extract the information. These modules extract the spatial-temporal relations between objects, the semantic data from video clips and the salient objects from video keyframes respectively. Therefore, BilVideo can support all the three query types. Besides that, BilVideo can return arbitrary segments of videos as the results compared to QBIC, AVIS and OVID. It is worth noting that all these systems depend on user annotations to extract information from the videos. BilVideo even needs the users to set the threshold to extract the saliency objects.

Recent years see a increasing tendency to leverage computer vision techniques to identify objects in the images automatically. The related techniques are object recognition and object detection. The object recognition indicates which classes are the images belonging to [56], while the object detection locates the objects in the images then recognizes them [31]. Object recognition gets a significant improvement by Deep Convolution Neural Network (DCNN) [56][96] in recent years' ImageNet competition [92]. Some researchers even claim their models can excel human on the ImageNet Dataset in terms of predicting top-5 annotations [40]. However, the success in object recognition does not improve the object detection significantly [100]. It is because the object detection not only needs to recognize the objects but also needs to locate them in the images. Some researchers divide object detection into localization and recognition. The localization only focuses on which regions in the images have objects [15]. After that, the recognition can predicts the labels based on the images from localization. It is worth noting that all these mentioned methods are proposed and evaluated on the images datasets. When they are transferred to video datasets, the accuracy drops significantly [58]. The superficial reason is that the state-of-the-art detection tools can not detect the objects continuously on the video datasets [58], while the actual reason is that the image for research have better quality than the frames from videos on average. To overcome the limitations, connecting the visual objects by leveraging the spatial-temporal continuity is necessary. Therefore, some researchers design the track-by-detection tracker to connect the visual objects discovered by object detection [9]. The process firstly uses detector to get the objects' positions then tracks them in the following frames.

But the tracking-by-detection trackers have the same problem as the general trackers. They cannot stop tracking automatically when the objects already disappear. It means they will connect the visual objects wrongly for several frames [9]. Such behavior is fine in the tracking benchmark [114] because the objects in the corresponding test videos seldom disappear for a long time. However, in the real world videos, the duration of object disappearance is uncertain. Simply performing such tracker will generate a lot of false object occurrences into the frame-level filtering process.

2.5.2 Video-level Filtering

Video-level filtering using non-textual content vector has been also studied for many years in multimedia and computer vision area. It has a lot of applications, such as action recognition [107], video classification [52], event detection [116] and so on. The research hotspots are content vector generation [82, 50], filtering model [29, 11, 52] and multiple content fusion [4, 117, 123]. The content vector generation has been discussed in Section 2.3. In this section, we mainly review the recent works on filtering model and multiple content fusion.

To filter the videos with content vector, the filtering models need to accept content vectors as input and assign scores to the videos for ranking purpose. The widely used filtering models are support vector machine (svm) [29, 11], ridge regression [123] and neural networks [52]. Among them, svm and ridge regression are linear model while neural networks are non-linear model. At beginning of the video-level filtering, only linear models and BOW vectors are available for large scale video data [108]. To make the linear model aware the non-linear pattern in the content vectors, the chi-square distances between content vectors are precomputed and stored in a kernel matrix [108]. After that, svm or ridge regression is applied on the kernel matrix to train the filtering model [108, 4]. The difference between the linear models is that, svm [29] tries to separate the exemplar videos and the background videos, while ridge regression tries to assign high scores to the exemplar videos. The state-of-the-art video-level filtering methods changed after FV and VLAD were proposed. FV and VLAD vectors are of high dimension so they can approximate the complex non-linear relations in a linear space. Under this direction, linear svm [29] has been widely used. Recent tendency tries to use convolutional neural networks (CNN) to filtering the videos [52, 116, 97]. Compared to the traditional methods, the functions of feature extraction and encoding are gradually replaced by CNN [56]. CNN has multiple convolutional layers to perform non-linear learning but its last layer is linear for filtering purpose.

Multiple content fusion is another hotspot. As we introduce in Section 2.3, there are multiple content vectors extracted for representing the videos [4, 109, 116]. However, the filtering models we introduced are naturally oriented to single content type. Accordingly, many studies on multiple content fusion has been conducted in recent years [17, 76, 99]. Based on existing works, there are two branches of fusion strategies, one is early fusion which applies fusion on the inputs of the filtering models, and the other one is late fusion which applies fusion on the outputs of the filtering models. The simple form of early fusion is summing the kernel matrices up. It is applied in the system proposed in [60]. Another way to perform early fusion is to learn a shared space of multiple content vectors [76, 99]. The simple form is concatenating different content vectors into big vectors, then feeding the big vectors into the existing filtering models. More complicated models are proposed in [76] and [99]. In [76], various neural networks structures were proposed to perform early fusion. The most effective design is to establish a neural network for each content type and add a canonical component analysis (CCA) layer on the top of all the neural networks to perform early fusion. In [99], the shared space was learned by restricted Boltzmann machine (RBM). Compared to the supervised model in [76], RBM is an unsupervised model and uses contractive divergence (CD) [43] to train the model rather than back propagation.

The simple form of late fusion is average fusion. It collects the scores derived from different features and averages them to obtain the final score. It is fast and it does not modify filtering model as the early fusion. A enhanced version of average fusion is proposed in [59]. It treats the scores from early fusion as one score source during the calculation. Learning-to-rank [10] is the most popular late fusion method nowadays. It treats the scores derived from different content vectors as the new features. Then, it applies svm or regression to weight the scores from different content vectors. It has been widely used in document retrieval. In [17], a feature correlation tree is constructed to model the relations between different content vectors. Then, conditional random field method was applied on the tree to perform the late fusion.

Recent studies [17, 4, 60] show that both fusion strategies push the video-level filtering accuracy in to a high level. Sometimes, the high accuracy means the highly ranked ones could be used as exemplar videos. This encourages the studies on how to leverage the highly ranked videos to improve the accuracy further, which are called pseudo relevance feedback (PRF) [60, 123].

2.5.3 User-level Filtering

The user-level filtering, namely personalized recommendation, exploits rating matrix and video content vectors to perform filtering. It aims to generate personalized top- k videos for each user. The rating matrix is derived from user implicit feedbacks [47, 87], while the video content vectors are same as those used in video-level filtering methods. The rating matrix records the user behaviors on the videos such as likes and clicks [47] and transforms these behaviors into binary values. According to whether the videos are included by the rating matrix, the user-level filtering can be further divided for two scenarios: in-matrix and out-of-matrix. In the in-matrix scenario, the filtering methods generate the top- k videos which have not been rated by the target user but have been rated by other users [106]. Based on the co-rating behaviors from similar users, state-of-the-art methods [47, 106, 78, 110] use collaborative filtering (CF) to generate the personalized recommendation. In out-of-matrix scenario, the filtering methods generate top- k new videos that have not been rated by any user [106]. In this scenario, CF-based methods are ineffective, whereas content-based methods perform well.

Weighted matrix factorization (WMF) [54] and Bayesian personalized ranking (BPR) [87] represent the state-of-the-art user-level filtering methods in in-matrix scenario. Both of them are matrix factorization models and are derived from collaborative filtering (CF). They learn a latent vector to predict each user's rating on each item, for each user and item in turn, and then select the top ranked items with the highest predicted ratings. The major difference between them is the optimization objective. The WMF model [54] learns the latent factors by minimizing the rating prediction loss on the training data, while the BPR model [87] learns the latent factors by preserving the personalized rankings. Recently, both WMF and BPR were extended to incorporate content features, so they can learn a latent vector to represent both in-matrix and out-of-matrix items, and hence be applied to both in-matrix and out-of-matrix scenarios. The representative WMF-based models include collaborative topic regression (CTR) [106], deep content-based music recommendation model (DPM) [78] and collaborative deep learning (CDL) [110]. CTR and CDL only integrate the textual features of items, while DPM only considers non-textual features. The representative BPR-based models are visual Bayesian personalized ranking (VBPR) [42] and collaborative knowledge base embedding (CKE) [124]. VBPR is designed to incorporate visual features, and CKE fuses both structural and non-structural features from the knowledge base.

2.6 Summary

In this chapter we reviewed some related research areas to our thesis, including visual object detection and tracking, video content vectors as well as video filtering methods from different levels. We notice the limitations of the existing methods from different levels even though the non-textual contents are in use, and we try to overcome the limitations in our study. In the next three chapters, we will describe how we improve the multi-level video filtering using the non-textual contents.

Chapter 3

Frame-level Filtering using Detected Visual Objects

3.1 Introduction

There exist numerous visual objects in the videos. They not only have various appearances but also associate with accurate time information within the videos. These properties make the frame-level filtering with visual objects return more precise results [53, 24], which is different from the low-level filtering [44, 77] and the semantic filtering [83, 120].

To manage the visual objects for frame-level filtering, existing works usually maintain two tables simultaneously as Figure 3.1: one is the unique object table which stores the globally identified objects appear in the videos, the other one is the occurrence table which stores the occurrences of the globally identified objects in the videos [57, 22, 63, 49]. Ideally, all the occurrences should be connected to the identified objects. With these connections, the frame-level filtering is easy to implement: given a set of visual objects, the filtering process firstly performs lookup in the unique object table; if all the given visual objects are hit, the filtering process will fetch the frames according to the connections and merge the adjacent frames into video clips.

In previous works [57, 22, 63, 49], the construction of these two tables is a top-down process: first, experts are hired to generate the unique object table; second, more labors are hired to identify the unique visual objects and connect their corresponding occurrences in the videos; third, the connections are gathered to generate the occurrence table. After the top-down process, the globally identified objects are stored by the unique object table, while their occurrences as well as the connections are

stored by the occurrence table.

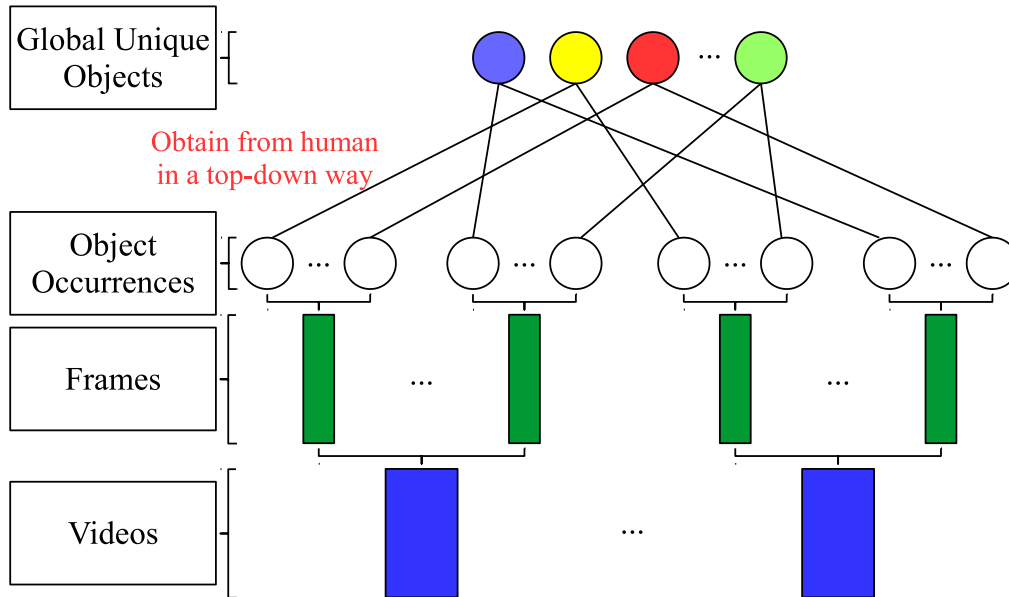


FIGURE 3.1: The data structure for supporting frame-level filtering.

There is one major limitation that all the existing works [57, 22, 63, 49] have. That is, the construction process of these two tables heavily depends on human labors [23, 53]. It means that, every time new visual objects or new videos appended into the filtering system, enormous human annotations on the videos should be conducted to update these tables. Obviously, it is inefficient for obtaining these two tables on the real world large-scale dynamic video datasets. There requires a more efficient way to help the frame-level filtering get rid of intensively labeling.

Recent years' progress on object detector has shown that machine is very possible to locate visual objects as precise as human in the near future [9, 48, 86]. Accordingly, one promising way to improve the efficiency is using the detector instead of human labors to annotate visual objects as much as possible. However, it is worth noting that precise detector is not enough to relieve labors from annotations. This is because the detector has two drawbacks compared to human: (1) detector is unable to identify the visual objects in the videos [9], which is inappropriate for generating the unique object table; (2) it also fails to connect the object occurrences in the continuous frames [9, 58], which decreases the number of true occurrences in the occurrence table.

According to existing works, there exist three assistant methods to equip object detection with identifying or connecting. They are recognizing-based, matching-based and tracking-based methods. According to Table 3.1, their respective functionalities are explained as below:

- **Recognizing-based** method leverages the recognized information to identify and connect the visual objects. For example, if two car plates are detected and recognized to have the same plate number, they will be identified as the same. The limitation of the recognizing-based method is that the recognized information is not always reliable and available.
- **Matching-based** method leverages low-level visual features to identify and connect the visual objects. Compared to recognizing-based method, it is more versatile and it performs well when the visual objects are clear in the videos. The limitation of matching-based method is that it performs bad when the visual objects are blurred, transformed or occluded.
- **Tracking-based** method leverages the spatial-temporal continuity to connect visual objects. The tracking-based method accepts a visual object and its location as input. It connects the occurrences of the same object within the videos. When the visual objects are blurred, transformed or occluded, it performs better than matching-based method. However, the tracking-based method has one non-negligible drawback. That is, the tracking-based method cannot stop connecting when the target visual object disappears.

TABLE 3.1: Differences between the assistant methods where \checkmark denotes yes and \times denotes no

Method	Connect	Identify	Precision	Recall
human	\checkmark	\checkmark	very high	very high
recognizing	\checkmark	\checkmark	high	very low
matching	\checkmark	\checkmark	high	low
tracking	\checkmark	\times	very low	high
our method	\checkmark	\checkmark	high	high

In addition to the functionalities, each method's connecting accuracy is different. The human annotation obviously achieves very high accuracy in terms of precision and recall. Recognizing-based and matching-based methods have high precision but their recalls are quite low. On the contrary, the tracking-based method achieves high recall but its precision is very low. Since the drawbacks of each assistant method is too significant, there calls for a method which can achieve both high precision and recall.

Based on the above analysis, we propose a hybrid method to generate the unique object table and the occurrence table in our work. The proposed method blends the matching-based and tracking-based methods to identify and connect visual objects based on object detection. The process has three steps. The first step is local merge. It begins with applying the detector on the frames to get the visual objects from each video. Then, the detected visual objects are identified locally by the matching-based method. The second step is propagation. It connects the visual objects within the videos by tracking-based method. It is worth noting that not all the tracked occurrences are reserved in our method. Only those pass the check by the matching-based method will be left. After that, the connections between the unique visual objects and their occurrences are stored locally. The third step is global merge. It identifies and connects the visual objects globally from different videos. Our experiments show that the proposed hybrid method is more accurate than the three assistant methods in both identifying and connecting visual objects. In addition to that, the experiments show that the proposed method cost less time than object detection but generates more connected object occurrences.

In summary, our contributions in this work are as follows:

1. We propose to leverage automatically detected objects to support frame-level filtering. The new proposal makes the original top-down process changed into bottom-up for generating the unique object table and the occurrence table;
2. We find object detection alone is not enough to generate the two table. Accordingly, we study three candidate assistant methods, namely, recognizing-based, matching-based and tracking-based methods, for object detection to identify and connect visual objects. We show that all of them are not suitable for table generation by mechanism analysis and experiments;
3. Based on above analysis, we propose a hybrid method to generate the tables. It blends the matching-based and tracking-based methods, and has three steps, namely local merge, propagation and global merge. The evaluation in terms of accuracy and efficiency shows that our proposed method is more suitable to assist object detection than the three assistant methods for supporting frame-level filtering.

3.2 Problem Statement

In this section, we present some preliminary concepts and give an overview of the proposed method. Table 3.2 summarizes the major notations used in the rest of the section.

TABLE 3.2: Summary of notations

Notation	Definition	Notation	Definition
A	a set of unidentified visual objects	a	an unidentified visual object
O	a set of identified visual objects	o	an identified visual object
V	a set of videos	v	a video

Given a set of visual objects, the frame-level filtering should return the video clips which contain all the given visual objects. Accordingly, the research problem is defined as follow:

$$V_{out} = filter(O_{in}, V_{all}) \quad (3.1)$$

where O_{in} is the set of input visual objects, V_{all} is the set of videos to be filtered and V_{out} is the set of videos containing all the objects in O_{in} . It is worth noting that a video in V_{out} might be a part of a video in V_{all} .

3.2.1 Preliminaries

The real-world motion is continuous. It is simulated by the frame switching in the videos. Therefore, the motion in the video is discrete. The switching speed is defined by frames per second (fps). For example, if fps is 24, it means one second in the video has 24 frames on average. According to this, if the beginning frame of the video is assigned the frame number 1, the n^{th} frame in the sequence will be assigned the frame number n . These frame numbers are used for calculating video duration in our method. For example, if fps is r and frame numbers are from m to n ($m < n$), the duration of video clip is calculated as $\frac{n-m}{r}$ seconds.

The frame-level filtering aims at selecting the relevant video frames given some constraints and concatenating the remaining frames to form the video clips as return [24]. In our work, the frame-level filtering is designed to use the detected visual objects as constraints. To simplify the irrelevant details, we have following definitions for our further discussions:

Definition 3.1. *Visual object* is a real-world 3D object's projection on the frame.

In our work, we assume the shape of visual object is rectangle, which has been applied by many previous works [31, 86]. It means that the visual object can be located by the minimum bounding rectangle (MBR) such as Fig. 3.2. The appearance of the visual object then can be represented by the image cropping from the frame.

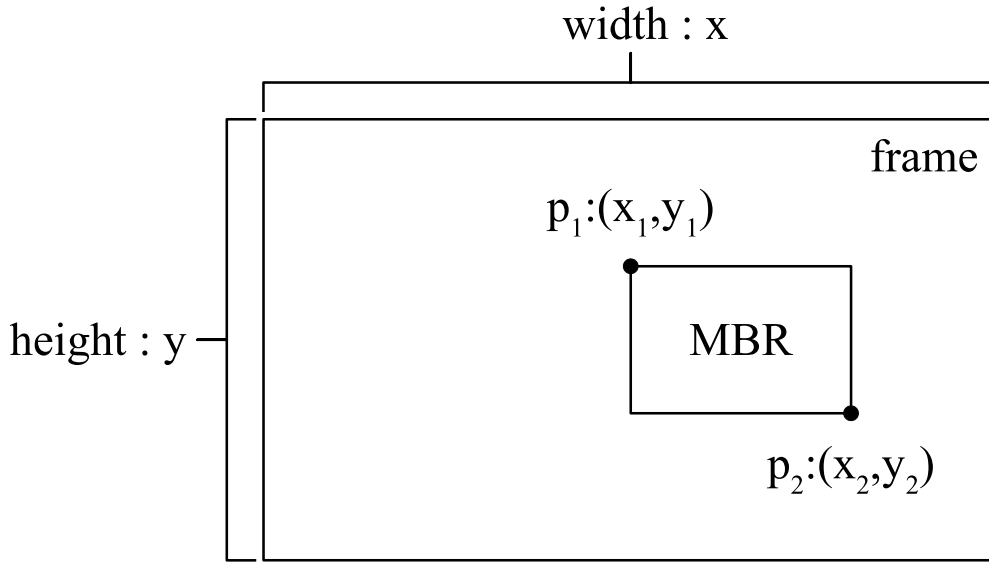


FIGURE 3.2: Visual object is located by a minimum bounding rectangle on a video frame

Definition 3.2. *Visual object detector* is a function which locates the visual objects on the frames using MBRs.

Definition 3.3. *Visual object tracker* is a function which connects the visual objects in the videos using MBR sequences.

The differences between detector and tracker are two folds: first, the input of a detector is frame only, while the input of a tracker in addition needs the MBR of the visual object that needs to be tracked; second, the visual objects discovered by a detector have no identifications, while the visual objects discovered by a tracker have identifications within the videos.

Definition 3.4. *Unidentified object* is a visual object having no identification within or across videos.

The visual objects discovered by detector have none identifications. Because they have no connections to other visual objects within or across the videos. Every unidentified object is associated with an image, a MBR and the frame number and the video id.

Definition 3.5. *Locally identified object* is a visual object having identification within videos.

The visual objects discovered by tracker have local identifications. Because they are connected by the tracker within the videos. Same to unidentified object, every locally identified object is at least associated with an image, a MBR, the frame number and the video id.

Definition 3.6. *Globally Identified object* is a visual object having identification within and across videos.

The identified objects have connections to more than one unidentified objects not only within the videos but also across the videos. This makes them have global identifications. Every identified object is associated with an image and a global unique id.

Definition 3.7. *Occurrence table* is a set of unidentified visual objects with their connections to identified visual objects.

The tuple in occurrence table representing an unidentified visual object is denoted as $(aid, oid, img, c, s, mbr, v, t)$. In each tuple, aid is the primary key of the occurrence table, oid is the foreigner key to the corresponding identified object, img is the appearance image of the unidentified object, c is the recognized content, s is the confidence of localization, mbr is the location, v is the video id and t is the frame number.

Definition 3.8. *Unique object table* is a set of identified visual objects.

The tuple in unique object table representing an identified visual object is denoted as (uid, img, c, s) where uid is the primary key of the unique object table and img is the appearance image of the identified object. In our propose method, each video will have one local unique object table and one local occurrence table temporally. After all the processes are done, there will be only one global unique object table and one occurrence table for supporting frame-level filtering.

3.2.2 Method Overview

Our method aims at generating the unique object table and the occurrence table based on object detection. It consists of four steps and its data stream is described in Fig. 3.3:

- 0) **Detection** detects the unidentified objects from the frames of the videos. These detected visual objects are stored in-memory temporally;

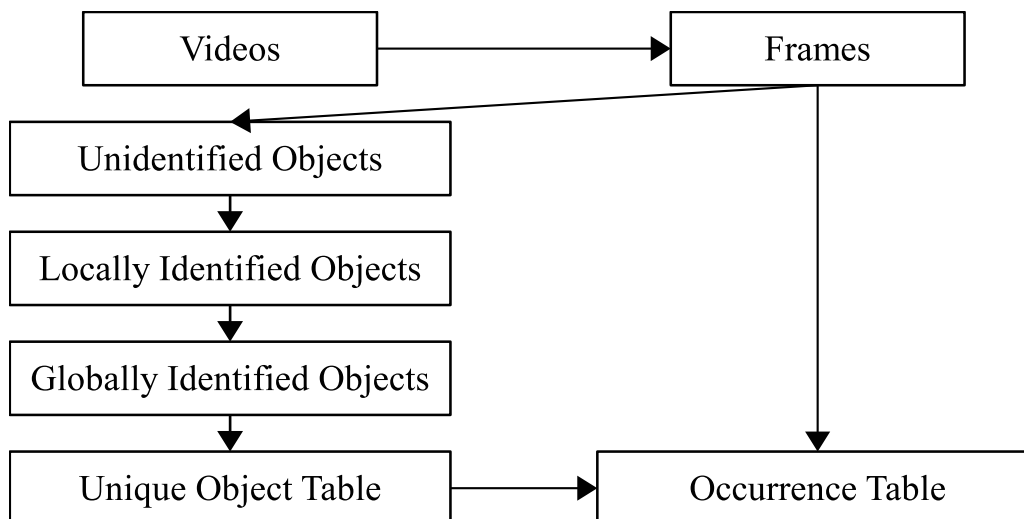


FIGURE 3.3: The pipeline of the proposed method.

- 1) **Local Merge** identifies and connects the visual objects within the videos. It generates the local identified object set and the local occurrence set for each video based on the detected visual objects;
- 2) **Propagation** enriches the local occurrence set by propagating the locally identified objects on the video frames;
- 3) **Global Merge** identifies and connects the visual objects globally. It merges the locally identified objects from local merge to obtain the globally identified objects, and generates the unique object table. In addition, global merge also updates the connections to generate the global occurrence table.

After above steps, the globally identified visual objects and their occurrences on the frames are stored by the global unique object table and global occurrence table accordingly. The final data structure for supporting the existing frame-level filtering methods [24, 22] is summarized as Fig. 3.4.

3.3 Related Work

The frame-level filtering using non-textual contents has been studied for many years. The research mainly focuses on designing filtering queries to make a better use of the video data. Until now, three enhanced query types have been proposed, namely, low-level query, spatial-temporal query and

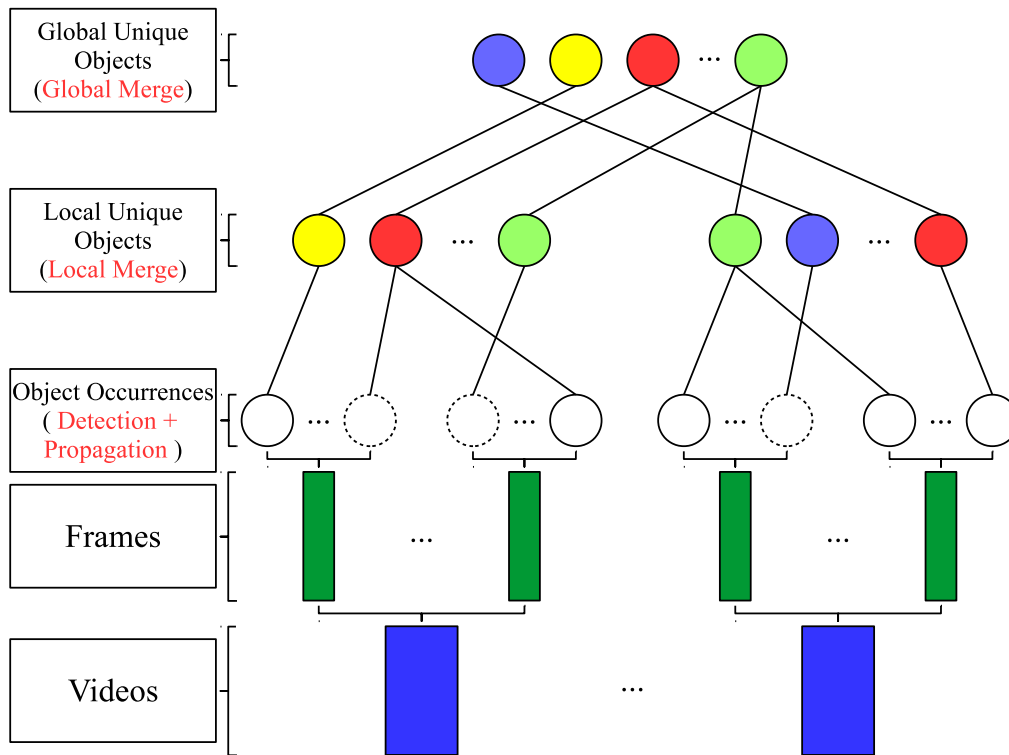


FIGURE 3.4: The revised data structure for supporting frame-level filtering.

semantic query [24, 95]. The low-level query is equivalent to content-based video retrieval [83]. Given some exemplar videos, the query leverages the low-level features to search the similar videos [46]. In recent years, the low-level query has been improved a lot by more advanced features [56] [109] [35]. However, since the low-level query only involves similarity calculation and ranking, it seldom builds index structure to guide the query. In addition to that, the granularity of low-level query is high. For instance, if a user wants to query some visual objects in the video database, the low-level query cannot provide accurate location and time information. As an improvement, the spatial-temporal query which exploits the visual objects is proposed to achieve lower granularity [53]. On one hand, it allows more flexibility in query conditions: any combination of object directional relations, topological relations, appearance and trajectory [24]. On the other hand, the spatial-temporal query accepts the visual objects as the input and returns arbitrary segments of videos [24]. The low-level query and spatial-temporal query are powerful, but they require specialist skills to work properly. The semantic query is proposed to achieve the easy-to-use purpose. It is usually decomposed into some sub-queries which in fact can be regarded a combination of the low-level query and the spatial-temporal query [1][83]. With these query types, many systems have been proposed in the past, such

as OVID [77], QBIC [33], AVIS [1], CVQL [57] and BiVideo [103]. We notice that some of them also design index structure for the enhanced query type. However, the visual objects and links for constructing index are obtained manually, which limits the scalability of these systems.

Developing automatic algorithms instead of human to locate objects visually has achieved a significant improvement by Deep Convolution Neural Network (DCNN) [100]. However, these algorithms are oriented to images. When they are applied to videos, the accuracy drops significantly [58]. One possible reason is that the images for training have better quality than the frames from videos on average. To improve this, some researchers propose the track-by-detection algorithms to increase the accuracy of object localization [9]. Such series of algorithms first try to locate the objects on some frames then track them in the following frames. But the tracking-by-detection algorithm keeps on tracking even when the objects already disappear [9]. Such behavior is fine in the tracking benchmark [114] because the visual objects in the test videos seldom disappear for a long time. However, in the real world videos, simply performing such tracker will bring a lot of false visual objects into the video database.

3.4 Proposed Method

The proposed method is divided into three steps, namely, local merge, propagation and global merge. After detection provides the initial visual objects, local merge identifies the visual objects within the videos. Then, the enriches the corresponding occurrence set. When all the locally identified objects and occurrence sets are gathered, the global merge identifies and connects the visual objects from different videos. Accordingly, the data structure for supporting frame-level filtering is obtained after global merge.

3.4.1 Detection

The initial step is detection. It is achieved by the pretrained visual object detector. To perform the detection, the frames in the videos are fetched sequentially. Then, the detectors operate on the frames and return the MBRs and the detection confidences. According to our definitions, these returns are used to represent the unidentified objects. Some visual objects might have the recognized contents such as car plate. These contents, if exist, will be collected as well.

3.4.2 Local Merge

The first step of the hybrid method is local merge. It aims at identifying and connecting the visual objects within videos. It can be performed in parallel. As we discussed, the visual objects from the detectors have no connections to the visual objects on the other frames or in the other videos, which means they are unidentified. Considering the real-world objects usually last for more than one frames in the videos, there exist redundancy in the unidentified objects. This decreases the efficiency when the video filtering need to check whether the given objects existing in the database. To improve that, we need to identify and connect these detected visual objects.

The key of identification is to decide whether two visual objects are same by comparing their appearances. To this end, we introduce the image matching function in local merge.

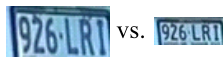





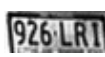









factor	example	solution	result
size	 vs. 	resize	 eq. 
illumination	 vs. 	change to grey image+ histogram equalization	 eq. 
occlusion	 vs. 	use the last appearance	 eq. 
transformation	 vs. 		 eq. 

FIGURE 3.5: Four factors prevent image matching correctly in videos.

However, directly using the image matching function does not return the correct indications in many situations. In Figure 3.5, we list 4 common factors that will make the image matching function return the false indications. The first factor is size. It makes the image matching function wrong when the input images are too small. This is because the small images do not have enough visual features to accomplish the matching [69]. A solution to that is to resize the small images to the larger ones. In Figure 3.5, the matching function returns the correct result when resized images are used. The second factor is illumination. Too strong or weak light will change the color of object appearances. To overcome this, we convert the images into gray. It is not enough because the contrast ratio needs to adjust as well. Therefore, we perform the adjustment by histogram equalization. After that, the image matching function can well adapt the illumination. The occlusion and transformation are caused by the object or camera motion. These factors cannot be well handled by the image matching function alone. Since the frames are read in sequence, a better way is to maintain the last appearance for

each unique objects during merge and use it to perform image matching. This method is similar to visual tracking task which leverages the spatial-temporal continuity of appearance in a shot [121]. An example about how to leverage spatial-temporal continuity to overcome occlusion is described in Figure 3.6. The adjacent images are easy to match because the change is small. After a gradual match progress, we get the match result between the target image and the large-occlusion image.

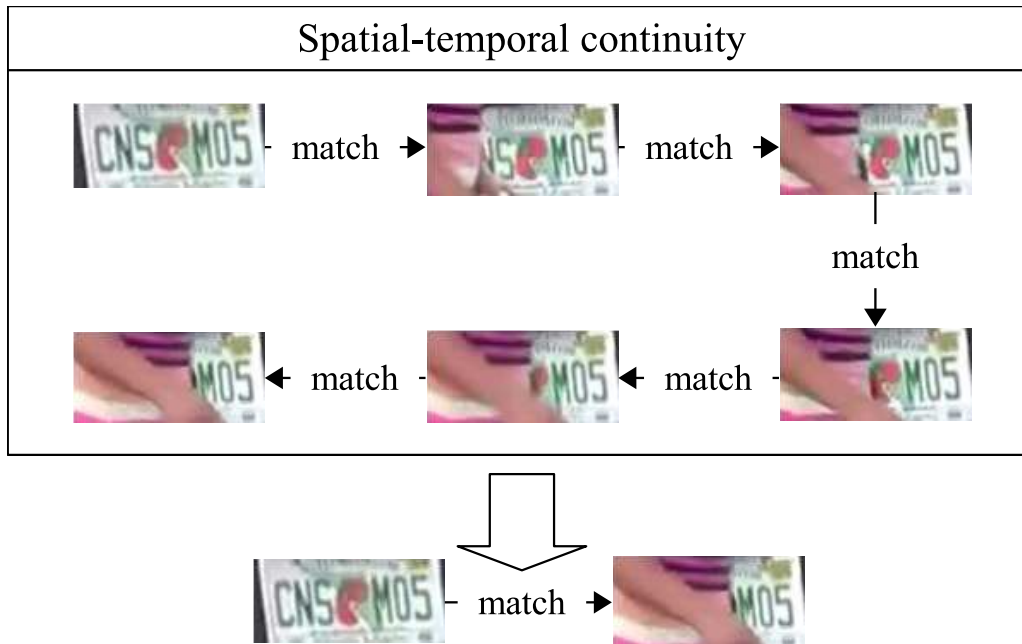


FIGURE 3.6: Use continuity to improve matching results.

The local merge is described in Algorithm 1. For each frame, the local merge uses the object detector to obtain unidentified objects at first (line 3 to line 4). These appearances are resized, changed to gray, and histogram equalized to overcome the size and illumination issues. After that, the local merge updates O_k and adds correspondences between unidentified object and locally identified objects relying on the indications from the mapping function f_{map} (line 5 to line 19). If a locally identified object is mapped to an unidentified object, its last appearances will be updated correspondingly. Otherwise, a new locally identified object will be inserted.

Each locally identified object in O_k has two appearances: img_{last} stores the last appearance, while img_{best} stores the best appearance. We keep two appearances for each locally identified object in local merge for two reasons. First, as we mentioned, keeping img_{last} makes the image match overcome the motion issues. Second, img_{last} may drift seriously as the transformation continues. In this situation, only keeping img_{last} is incorrect when the next appearances are not transformed. We need to keep the non-transformed appearance as well. After the local merge, the img_{last} of each

Algorithm 1: Local Merge

Input: video v_k , object detector f_{od} , matching function f_{match} , mapping function f_{map}

Output: local object occurrence set A_k , locally identified object set O_k

```

# Initialize the empty object sets;
1: initialize  $O_k \leftarrow \{\}$ ,  $A_k \leftarrow \{\}$ 
# Fetch the frames sequentially;
2: for all frame  $fr_k^t \in v_k$  do
    # Get the MBRs and the corresponding confident scores;
3:  $MBRs \leftarrow f_{od}(fr_k^t)$ 
4:  $A \leftarrow$  crop unidentified objects from  $fr_k^t$  by  $MBRs$ 
    # Traverse the unidentified objects in  $fr_k^t$ ;
5: for  $i \in \{1, 2, \dots, |A|\}$  do
6:    $flag, o \leftarrow f_{map}(A[i].img, O_k)$ 
7:   if  $flag$  is true then
    # Update the last appearance with current one;
8:      $o.img_{last} \leftarrow A[i].img$ 
9:      $A_k \leftarrow A_k \cup \{(A[i].aid, o.oid, A[i].img, A[i].MBR, A[i].c, A[i].s, v_k, t)\}$ 
10:    if  $A[i].s > o.s$  then
    # If current appearance is better, update the best appearance and score;
11:       $o.img_{best} \leftarrow A[i].img$ 
12:       $o.s \leftarrow A[i].s$ 
13:    end if
14:  else
    # If not mapped, add a new unique object;
15:     $o'.img_{last} \leftarrow A[i].img$ 
16:     $o'.img_{best} \leftarrow A[i].img$ 
17:     $o'.c \leftarrow A[i].c$ 
18:     $o'.s \leftarrow A[i].s$ 
19:     $O_k \leftarrow O_k \cup \{(o'.oid, o'.img_{last}, o'.img_{best}, o'.s, o'.c)\}$ 
20:  end if
21: end for
22: end for
23: return  $A_k, O_k$ 

```

Algorithm 2: Matching-Based Map**Input:** object appearance img , locally identified object set O_k , matching function f_{match} **Output:** $flag, o$

```

1:  $flag \leftarrow false$ 
2:  $n \leftarrow null$ 
3: for  $o \in O_k$  do
4:    $lflag \leftarrow f_{match}(o.img_{last}, img)$ 
5:    $bflag \leftarrow f_{match}(o.img_{best}, img)$ 
6:   if  $lflag$  is true or  $bflag$  is true then
7:      $flag \leftarrow true$ 
8:     break
9:   end if
10: end for
11: return  $flag, o$ 

```

locally identified object will be dismissed due to the possible arbitrary drift. img_{best} is assigned by the appearance from the mapped unidentified object whose confident score is the highest. It will be kept and used as the input for the global merge.

The initial mapping method is written in Algorithm 2 where it uses both best appearance and last appearances of each object in O_k to match the target object appearance. Once a locally identified object is matched, the mapping is successful and the matched object will be returned. However, the match-based map cannot guarantee to generate the correct result. This is because the image matching function does not perform exactly matching. It may cause more than one objects to be matched to the target object appearance in Algorithm 2, which means that if a locally identified object is not same as the target it also has the chance to be returned. With or without the break statement in line 9 only influences whether first or last matched unique object will be returned.

To address the drawback of the match-based map method, we propose our similarity-based mapping method in Algorithm 3. We make this change based on the assumption that the correct match exists in the most similar object. The similarity-based mapping function first finds the most similar locally identified object. Then, it compares the target appearance with the most similar locally identified object instead of all the objects in O_k . This ensures only one object is mapped to the target object

Algorithm 3: Similarity-Based Map**Input:** object appearance img , unique object set O_k , matching function f_{match} **Output:** $flag, o$

```

1:  $flag \leftarrow false$ 
2:  $o \leftarrow null$ 
3: find the most similar  $o'$  in  $O_k$  given  $img$ 
4:  $lflag \leftarrow f_{match}(o'.img_{last}, img)$ 
5:  $bflag \leftarrow f_{match}(o'.img_{best}, img)$ 
6: if  $lflag$  is true or  $bflag$  is true then
7:    $flag \leftarrow true$ 
8:    $o \leftarrow o'$ 
9: end if
10: return  $flag, o$ 

```

appearance at most. If the match is successful, the most similar object will be hid for the other comparisons on the same frame and returned. Otherwise, the match is unsuccessful and non locally identified object will be returned. The similarity-based mapping works better than the matching-based one. It is worth noting that the similarity calculation is important for the improved map method. We will discuss the details of similarity calculation in our experiments.

3.4.3 Propagation

The local object occurrence set misses some occurrences due to the limitation of detection. To recover some missing occurrences, we propose the propagation step. The propagation aims to enrich the occurrences caused by the locally identified objects in the videos. It can be performed in parallel. The process logic is described in Algorithm 4. It leverages the visual object tracker and our proposed similarity-based mapping method to achieve the recovering.

In propagation, the videos are read again to fetch the frames sequentially. The first step is filtering off the visual objects generated by detector (line 4 to line 12) to reduce unnecessary propagations: if some of the locally identified objects have been already detected by the object detectors on current frame, the propagation will not propagate them but use their occurrences to update the last appearances. After this step, the propagation uses the visual object tracker to recover the filtered locally

Algorithm 4: Propagation

Input: video v_k , locally identified object set O_k , local occurrence set A_k ,
 matching function f_{match} , object tracker f_{track} , mapping function f_{map}

Output: local occurrence set A_k

```

1: for all frame  $F_k^t \in v_k$  do
2:    $A \leftarrow$  get object occurrences on  $F_k^t$  from  $A_k$ ,  $O \leftarrow \{\}$ 
3:   for  $a \in A$  do
4:      $flag, o \leftarrow f_{map}(a.img, O_k, f_{match})$ 
5:     if  $flag$  is true then
6:        $o.img_{last} \leftarrow a.img$ 
7:     else
8:        $O \leftarrow O \cup \{o\}$ 
9:     end if
10:  end for
11:  for all  $o \in O$  do
12:     $MBR \leftarrow f_{track}(o.img_{last}, F_k^t)$ 
13:     $a \leftarrow$  get object occurrence from  $F_k^t$  by  $MBR$ 
14:     $lflag, o' \leftarrow f_{map}(a.img, O_k, f_{match})$ 
15:    if  $lflag$  is true and  $o$  is  $o'$  then
16:       $A_k \leftarrow A_k \cup \{(a.aid, o.oid, a.img, a.MBR, o.c, o.s, v_k, t)\}$ 
17:       $o.img_{last} \leftarrow a.img$ 
18:    else
19:       $MBR \leftarrow f_{track}(o.img_{best}, F_k^t)$ 
20:       $a \leftarrow$  get object occurrence from  $F_k^t$  by  $MBR$ 
21:       $bflag, o' \leftarrow f_{map}(a.img, O_k, f_{match})$ 
22:      if  $bflag$  is true and  $o$  is  $o'$  then
23:         $A_k \leftarrow A_k \cup \{(a.aid, o.oid, a.img, a.MBR, o.c, o.s, v_k, t)\}$ 
24:         $o.img_{last} \leftarrow a.img$ 
25:      end if
26:    end if
27:  end for
28: end for
29: return  $A_k$ 

```

identified objects' occurrences (line 13 to line 30). Different from the object detector, the object tracker always returns a MBR from the frame based on the appearance of the target object. So it obviously has much higher recall rate than the object detector. However, generating a MBR does not ensure that the locally identified object exists in the frame. It means that only relying on the tracker will add many false positive occurrences. Therefore, after the tracking, the propagation leverages the similarity-based mapping to get a mapped locally identified object for the returned MBR (line 15 and line 22). If the mapped locally identified object's appearance equals to the input to tracker at start, the propagation is successful and the occurrence is recorded. Same as local merge, the propagation also maintains the best and last appearances for each locally identified object during the process. It makes propagation very robust to the occlusion and transformation. It is worth noting that the last appearances are not set before the propagation performs successfully once. The default last appearance is initialized by the best appearance. When a propagation operation is successful, the local occurrence table is update accordingly.

The Algorithm 4 contains the major idea of propagation. However, applying the tracker on the whole frame is usually unnecessary. This is because the object motion often occurs in a small region between two adjacent frames, which reveals the essence of spatial-temporal continuity. The propagation therefore can be accelerated by taking this advantage. In experiments, we will introduce how we leverage the spatial-temporal continuity to improve propagation.

3.4.4 Global Merge

The global merge aims to identify and connect the visual objects from different videos globally. It is responsible for generating the unique object table and the occurrence table. The global merge is described in Algorithm 5 where only the best object appearances are in use to perform mapping. Different from local merge, the parallel version of global merge needs to add read and write locks on the globally identified object set, because their copies need to be consistent. The global merge generates the object occurrence table. This requires to update the connections in the local object occurrence sets, when the corresponding locally identified objects are merged. After global merge, the generated data structure can be instantly used for frame-level filtering.

Algorithm 5: Global Merge**Input:** locally identified object sets O_1, \dots, O_k ,locally object occurrence sets A_1, \dots, A_k ,matching function f_{match} , mapping function f_{map} **Output:** globally identified object set O , object occurrence set A

```

1: initialize  $O \leftarrow \{\}$ ,  $A \leftarrow \{\}$ 
2: for all  $i \in \{1, \dots, k\}$  do
3:   for all  $o' \in O_i$  do
4:      $flag, o \leftarrow f_{map}(o'.img_{best}, O, f_{match})$ 
5:     if  $flag$  is false then
6:        $O \leftarrow O \cup \{(o'.oid, o'.img_{best}, o'.c, o'.s)\}$ 
7:     else
8:       update  $a \in A_k$  where  $a.oid$  eq.  $o'.oid$  set  $a.oid \leftarrow o.oid$ 
9:     end if
10:  end for
11:   $A \leftarrow A \cup A_k$ 
12: end for
13: return  $O, A$ 

```

3.4.5 Complexity Analysis

In this part, we analyze the complexity of merge and propagation. We use $T(f)$ to represent the time complexity of a function f . For local merge, we assume there are x unidentified objects, y locally identified objects and z frames for a video after process. According to Algorithm 1, the local merge needs to perform z detections and $2(x-1)$ matches constantly. The time complexity only varies with the frequency of calling similarity function. We therefore get the time complexities of the best case and the worst case. In the best case, the first $x-y+1$ unidentified objects are the same and the last $y-1$ unidentified objects are different to each other. In this situation, the local merge needs to call the similarity function $\underbrace{0 + \dots + 0}_{x-y \text{ times}} + 2 + 3 + \dots + y - 1 = \frac{y(y-1)}{2} - 1$ times. Therefore, the time complexity of the best case is :

$$2(x-1)T(f_{match}) + \left(\frac{y(y-1)}{2} - 1\right)T(f_{sim}) + zT(f_{od})$$

In the worst case, the first y unidentified objects are different to each other. The local merge needs to call the similarity function $\frac{y(y-1)}{2} - 1 + \underbrace{y + \dots + y}_{x-y \text{ times}} = \frac{y(2x-y-1)}{2} - 1$ times. Therefore, the time complexity of the worst case is :

$$2(x-1)T(f_{match}) + \left(\frac{y(2x-y-1)}{2} - 1\right)T(f_{sim}) + zT(f_{od})$$

The time complexity of global merge is same to that of local merge. Here, we omit the deduction of its time complexity.

For propagation, its time complexity is correlated to the unidentified objects on each frame. We use x_i to denote the amount of the detected unidentified objects on the i^{th} frame. According to Algorithm 4, the propagation firstly tries to filter x_i detected unidentified objects by mapping them to the locally identified objects. The propagation therefore performs $x_i y$ similarity calculations and x_i matches. After filtering, the propagation performs various amounts of tracking and matching based on cases: in the best case where all the recovered object occurrences are matched by last appearances, the propagation perform tracking $y - x_i$ times and matching $y - x_i$ times; in the worst case where all the recovered object occurrences are matched by best appearances, it performs tracking $2(y - x_i)$ times and matching $2(y - x_i)$ times. Notice that $x = \sum_{i=1}^z x_i$. The overall time complexity of propagation is :

$$\begin{aligned} \text{best} &: \sum_{i=1}^z yT' + (y - x_i)T(f_{rd}) \\ &= yzT' + (yz - x)T(f_{rd}) \\ \text{worst} &: \sum_{i=1}^z (2y - x_i)T' + 2(y - x_i)T(f_{rd}) \\ &= (2yz - x)T' + 2(yz - x)T(f_{rd}) \end{aligned}$$

where $T' = T(f_{match}) + yT(f_{sim})$.

In our study, the I/O time cost is not a big issue because videos are usually efficiently encoded. This makes the time cost on disk access is small, which is negligible compared to perform detecting, matching and tracking on the frames.

3.4.6 Running Example

In this section, we give an illustrative example to display how the frame-level filtering performs using the visual objects. In our case, the visual object is the image of the car plate. The process is depicted

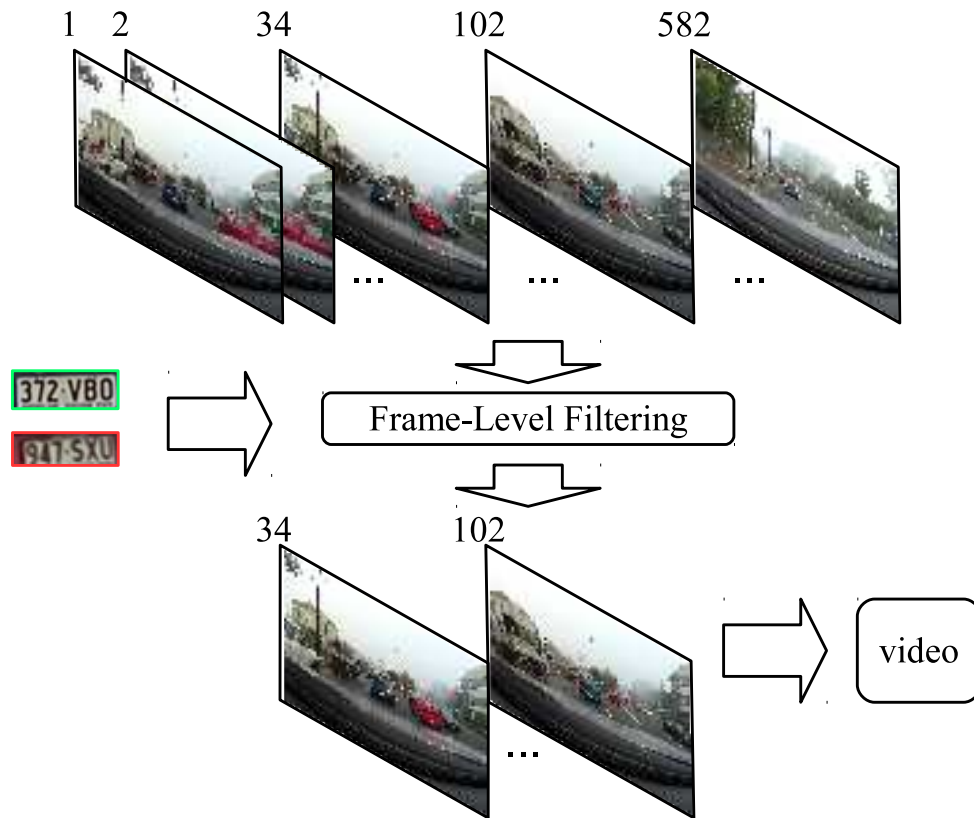


FIGURE 3.7: A running example to display how the frame-level filtering is performed

in Figure 3.7 where the video frames are annotated with their numbers. The process has three stages. In the first stage, two visual objects are presented to start the filtering. Then, in the second stage, the frame-level filtering is performed to filter the frames. In Figure 3.7, not all the frames contain the exemplar plates. Accordingly, after filtering, only the frames which record the co-occurrences of these two visual objects are left. In Figure 3.7, we can see the frames whose numbers are from 34 to 102 are selected. These frames form the final video clip in the third stage.

3.5 Experiment

3.5.1 Settings

Detection Tools

The accuracy of the ready-made detectors is a disturbance term for us to examine the proposed methods. More specifically, the false positive results will add unnecessary unique objects during the merge and cause a lot meaningless propagations, which will influence the evaluation. In order to make the

experiment get rid of it to some extent, we require the detectors generating the false positive results as few as possible. In addition to that, we need the detector to recognize the visual objects by texts to examine the recognizing-based method. For these reasons, we choose the plate detector in the following experiments. It is one of the most mature detection applications in the real world. The detector is downloaded from the OpenALPR repository ¹. Here, we use the release version 2.1. It is a stable version and it supports the plate format in Australia. The width of the minimum detected plate is set to 45 pixels to ensure most of the plates can be detected.

Data

The videos used in our experiments are recorded by the **SONY X1000V 4K Action Cam with Wi-Fi & GPS**². The collection consists of 393 videos which record the daily drives on the road. Due to the change of the speed and surrounding environment, all the influential factors discussed in Figure 3.5 are involved in these videos. The durations of the videos are around 10 seconds and fps is either 30 or 60. And they are all recorded in FULL HD where the frame width is 1920 pixels and the frame height is 1080 pixels. We use the FULL HD videos because the plate detector cannot perform well on the lower resolution videos in our scenario. After the videos are collected, we notice that the total size is 22GB while the individual sizes range from 54 MB to 62 MB. This means the video sizes do not vary with the fps significantly. On the other hand, the plate images used in the image match function must be resized. Based on our observation, we choose to resize the images whose widths are below 80 pixels. The resize is performed by a bicubic interpolation over 4x4 pixel neighborhood. After resizing, their widths will be adjusted to 80 pixels and heights will be adjusted as well according to the original aspect ratio.

We manually label 100 car plates appear in the videos and their 56160 occurrences. They are used as ground truth in our evaluation.

Environment

We run the experiments on a server. Its main specifications are in Table 3.3. The OpenALPR is much faster in the GPU mode than in the CPU mode. Therefore, we installed a Tesla K40 on the server which was donated by the NVIDIA Corporation. It is worth noting that Tesla K40 only has

¹<https://github.com/openalpr/openalpr>

²<http://www.sony.com/electronics/actioncam/fdr-x1000v-body-kit>

TABLE 3.3: Server Configuration

OS	Debian 3.2.57-3
CPU	2×Intel Xeon E5-2690 v2
#CORES	20
#THREADS	40
MEMORY	256GB
GPU	1×NVIDIA Tesla K40
DRIVER VERSION	346.46
CUDA VERSION	7.0
OPENCV VERSION	2.4.11
PYTHON VERSION	2.7.7

15 multiprocessors. This means that the detections can only use 15 threads in parallel. In order to ensure the parallel does not interfere the evaluation of the time efficiency, we also use 15 threads in the parallel of local merge and propagation. The whole project is implemented in Python. And we package the OpenALPR into a shared library called in Python instead of calling it from command line. This reduces a lot of the time cost on function call, which makes the time measurement on the detection more accurate.

Functions

In this part, we discuss the low-level functions serve merge and propagation in details. They are the image matching function f_{match} , the similarity function f_{sim} and the tracker f_{track} .

The image matching function f_{match} which is described by Algorithm 6 accepts two images as input then returns an indicator judging whether these images are same. Based on [73] and [72], the threshold is set to 10 in our implementation. f_{match} has two important components. The first one is the visual feature. Different visual features might cause different matching results. Accordingly, we make f_{match} work with three kinds of visual features as comparison in the evaluation:

1. **SIFT** [104] is used to capture the texture information in images. It is invariant to scale and slight transformation;
2. **ORB** [91] is used to detect and describe the corners in the images. ORB can be regarded as

Algorithm 6: Image Match Function f_{match} **Input:** Images img_1 and img_2 , threshold θ **Output:** IsMatched

```

1: feats1 ←  $f_{feat}(img_1)$ 
2: feats2 ←  $f_{feat}(img_2)$ 
3: count ←  $f_{pc}(feats_1, feats_2)$ 
4: if count >  $\theta$  then
5:   IsMatched ← True
6: else
7:   IsMatched ← False
8: end if
9: return IsMatched

```

an extension of FAST [89]. ORB has all the properties that SIFT has and it is also invariant to rotation. However, since ORB uses binary descriptor, it loses accuracy on object matching [72];

3. **BRISK** [64] is also used to capture the corners in the images. In spite of BRISK’s high efficiency, it also loses accuracy on object matching [72].

In addition to visual feature, the other crucial component is the counting function f_{pc} . Its return value directly influences the matching result. Recent counting functions [73, 72] often exploit fast library of approximate nearest neighbors (FLANN) to count the pairs. Accordingly, f_{pc} operates as Algorithm 7 in our implementation. According to [73], ρ is set to 0.7 in the evaluation. In Algorithm 7, the search of the nearest neighbors is based on different distance metrics for different feature type. For SIFT, as suggest in [104], we use Euclidean distances. For ORB and BRISK, we use normalized Hamming distances referring to [91, 64]. FLANN is efficient but cannot ensure the counting result is symmetrical (pair count from $feats_1$ to $feats_2$ is not equal to that from $feats_2$ to $feats_1$). So Algorithm 7 performs FLANN twice then returns the maximum counting result. Our merge method is implemented based SIFT because its higher accuracy of image matching [72].

The feature pair counting function f_{pc} is also served as the similarity function f_{sim} in our implementation. Accordingly, the similarity-based map in Algorithm 3 (line 3) regards the candidate which has the largest counting value as the most similar one. This trick makes the image matching in

Algorithm 7: Feature Pair Counting f_{pc}

Input: feats₁ and feats₂, ratio ρ **Output:** count

```

1: count1 ← 0
2: for feat in feats1 do
3:   find 2 nearest neighbors  $n_1$  and  $n_2$  for feat from feats2
4:   if distance(feat,  $n_1$ ) >  $\rho \cdot$  distance(feat,  $n_2$ ) then
5:     count1 = count1 + 1
6:   end if
7: end for
8: count2 ← 0
9: for feat in feats2 do
10:  find 2 nearest neighbors  $n_1$  and  $n_2$  for feat from feats1
11:  if distance(feat,  $n_1$ ) >  $\rho \cdot$  distance(feat,  $n_2$ ) then
12:    count2 = count2 + 1
13:  end if
14: end for
15: count ← max(count1, count2)
16: return count

```

similarity-based map simplify because the counting value can be directly re-used to indicate whether the image pair is same.

The tracker f_{track} is the most important component for object connecting. The related research has achieved significant progress in recent years [114]. In our connecting evaluation, we investigate three state-of-the-art trackers which are highly cited and have source code released. Their details are as follow:

1. **STRUCK** [38] leverages multiple feature kernels to learn a robust appearance model for tracking. It requires the MBR on the initial frame to track the visual object;
2. **TLD** [51] is a tracking-by-detection tracker. Compared to the original method [9], TLD develops positive and negative learner to generate exemplars for improving the accuracy of tracker. Unlike STRUCK, TLD only requires the object detector to start tracking. Accordingly, we

adapt the car plate detector into TLD;

3. **SO-DLT** [111] is an enhanced version of deep learning tracker (DLT) [112]. SO-DLT leverages stacked de-noising auto encoder (SDAE) to learn a robust appearance model. Same to STRUCK, SO-DLT requires the MBR on the initial frame to track the visual object.

Our propagation method is implemented based on STRUCK due to its usability.

In previous section, we mentioned a way to accelerate the tracker when the last appearance is used. It aims at reducing the search space during tracking. To do so, we leverage the spatial-temporal continuity to restrict the search space. The new search space is a subspace of the whole frame. Its center is indicated by the last propagation, and its area is nine times to the last detected area. Compared to the whole frame, the new search space is reduced by a factor of 700 roughly. It avoids the exhausting search in the whole frame so that the efficiency is improved.

3.5.2 Evaluation Plan

The whole evaluation is divided into four parts, namely, accuracy, efficiency, impact of factor and analysis. in accuracy evaluation, we study the accuracy of object identifying and connecting. Specifically, for object identifying, we will count the amount of unique objects each methods generated and their overlaps with the ground truth. The overlap rate is calculated by Equation 3.2.

$$Overlap\ Rate = \frac{|O_{gt} \cap O_{merge}|}{|O_{gt} \cup O_{merge}|} \quad (3.2)$$

The method which has closer amount and higher overlap rate to the ground truth is more accurate. For object connecting, we use precision and recall to evaluate the accuracy. The calculations of precision and recall are listed in Equation 3.3:

$$\begin{aligned} Precision &= \frac{\text{The amount of correct fixed occurrences}}{\text{The amount of total fixed occurrences}} \\ Recall &= \frac{\text{The amount of correct fixed occurrences}}{\text{The amount of ground truth occurrences}} \end{aligned} \quad (3.3)$$

Besides that, we also use F1 measure to reflect the overall performance. The calculation of F1 score is expressed in Equation 3.4.

$$F1\ score = \frac{Precision \times Recall}{Precision + Recall} \quad (3.4)$$

. In the perspective of precision, recall and F1 measure, the method which obtains higher values is more accurate.

In efficiency evaluation, we study time and space cost respectively. As the I/O cost is negligible, we only monitor the in-memory cost in the evaluation. For time cost, we record the execution time of each step and display their average time costs over all the videos. For memory cost, we use object wrapper to make the data structure persistent and record the storage cost. The baseline in the efficiency experiment is the costs from detection.

In impact of factor evaluation, we study how the motion, transformation and illumination influence the accuracy of object identifying and connecting. We apply the variance control strategy: each time, we remove one factor and re-run the whole process; after that, we count and record the corresponding result.

In analysis part, we study how the hybrid method benefits the frame-level filtering. We collect and compare the numbers of identified objects and occurrences. The comparison shows that the proposed method supports the frame-level filtering more sufficient.

3.5.3 Accuracy

The first experiment evaluates the accuracy of object identifying. We use object detector to generate the local occurrence sets, then apply local merge and global merge to obtain the globally identified objects. For comparison, we have evaluated following methods against the proposed method:

1. **Recognizing based merge** leverages the recognized texts from the car plate to identify the visual objects and merge them to generate the globally identified objects;
2. **Matching based merge** leverages the visual features from the appearances to identify visual objects and perform merge.

In Table 3.4, the number of globally identified objects generated by different methods are listed. Since tracking-based method can only generate locally identified objects, Table 3.4 does not have its number of globally identified visual objects. Compared to recognizing-based and matching-based methods, the proposed method with similarity-based map function generates the closest amount to the ground truth. In addition to that, the corresponding overlap rate is the highest. These results show that putting all the changes in Algorithm 1 leads to the most accurate identification result. The proposed method with matching-based map achieving the second accuracy indicates that the local merge is very sensitive to the mapping function in use.

TABLE 3.4: The accuracy of object identifying

Method		#Visual Objects	Overlap Rate
Ground Truth		100	100%
Recognizing: Text		123	77%
Matching	SIFT [104]	112	86%
	ORB [91]	117	80%
	BRISK [64]	121	78%
Our Method	matching-based map (Algorithm 2)	109	89%
	similarity-based map (Algorithm 3)	102	97%

TABLE 3.5: The accuracy of object connecting

Method		Precision	Recall	F1 score
Recognizing: Text		0.9315	0.2803	0.2157
Matching: SIFT [104]		0.9372	0.4046	0.2826
Tracking	STRUK [38]	0.3218	0.9654	0.2414
	TLD [51]	0.3353	0.9814	0.2499
	SO-DLT [111]	0.3418	0.9984	0.2546
Our Method: SIFT [104]+STRUK [38]		0.9225	0.7490	0.4134

The second experiment evaluates the accuracy of object connecting. The inputs are the locally identified object sets and the local occurrence sets. For comparison, we evaluate following methods in our experiment:

1. **Recognizing based propagation** leverages the recognized texts from the car plate to connect the visual objects;
2. **Matching based propagation** leverages the visual features from the appearances to connect visual objects. Since SIFT is the most accurate in our first experiment, we only use SIFT to perform matching based propagation;

3. **Tracking based propagation** leverages the object trackers to connect the visual objects.

Table 3.5 shows the accuracy of different methods on object connecting. The results show that the recognizing-based and the matching-based methods achieve much higher precisions than the tracking-based method, while they have much lower recalls than the tracking-based method. Compared to all of the existing assistant methods, our hybrid method which achieves high precision and recall at the same time. In the perspective of F1 measure, our method has the highest overall accuracy.

3.5.4 Efficiency

Another aspect we study is the efficiency of local merge, global merge and propagation. In Table 3.5.4, we list average time cost of executing each stage per video and the overall storage cost of the outputs from each stage. The time cost comparison shows that our proposed method which consists of local merge, global merge and propagation is faster than detection. It means that the time of generating unique object table and fixing object occurrences will be finished within acceptable time. We also notice that the size of occurrence table increases significantly after propagation. Because the proposed hybrid method has high accuracy, we think these enriched object occurrences are what the detection fails to discover.

Time Cost		Storage Cost	
Process	Time	Output	Volume
detection	437.98s	initial occurrences	972.8 MB
local merge	1.76s	locally identified objects	52.4 MB
global merge	34.21s	globally identified objects	22.4 MB
propagation	304.64s	fixed occurrences	2.3 GB

3.5.5 Impact of Factor

The factor experiment examines the impact of four factors (resizing, illumination, last appearance and best appearance) in accuracy. We want to know which factor has the largest impact on the object identifying and connecting. Therefore, we set the version which considers all the factors as the baseline, and remove one factor each time to see the influence respectively. The influence is measured by the

same metrics in the accuracy experiment. For object identifying, the amount of globally identified is used. We want to see how much the identification capacity is damaged by the factor removing. Accordingly, if the amount of globally identified objects increases too much, we will regard the removal factor as the important one to object identifying. For object connecting, the number of fixed occurrences is used for evaluation. As the identification capacity is damaged, less accurate connections will be established. Accordingly, if the amount of propagations decreases sharply after removing a factor, we will regard it as the important one to object connecting. The corresponding results of each factor are recorded in Table 3.6. Three conclusions are made from Table 3.6: (1) resizing is the most important factor for object identifying and connecting. The accuracy is damaged seriously without resizing. (2) The last appearance is very important for object connecting. The amount of occurrences is decreased sharply without last appearance. (3) The best appearance has nothing to do with object connecting. The amount of the occurrences keeps unchanged after removing best appearance.

TABLE 3.6: Impact of factor

factor	#identified objects		#fixed occurrence
	locally	globally	
ground truth	221	100	23792
without resize	2523	2242	7937
without illumination	252	132	20482
without last appearance	251	121	9257
without best appearance	232	127	23792

3.5.6 Analysis

In this analysis, we want to show how the hybrid method supports the frame-level filtering. Figure 3.8 summarizes two major benefits. The first benefit is the compressed search space. If the unique object table does not exist, the frame-level filtering needs to scan the occurrence table in a brute-force manner when given a set visual objects. Fig 3.8(a) shows the huge difference between scanning on the whole occurrence table and the unique object table. It indicates that the filtering process needs to pay much more time to scan the visual objects without the unique object table. The second benefit is the enriched object occurrences. Frame-level filtering with visual objects should provide

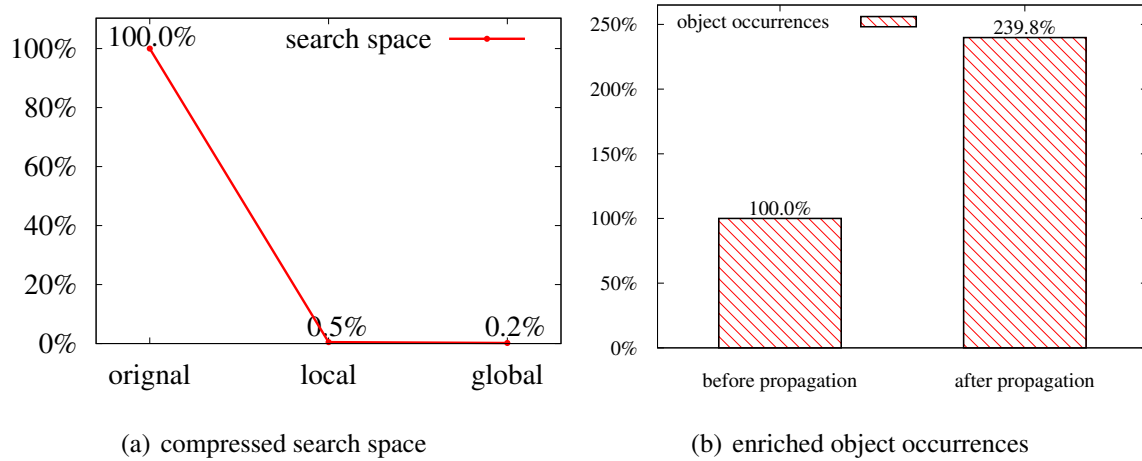


FIGURE 3.8: The benefits from unique object table and occurrence table.

precise temporal information. If most of the object occurrences are missing, the temporal information won't be precise any more. Figure 3.9(a) compares and shows how much occurrences are fixed by the proposed method. Obviously, the hybrid method significantly increases the amount of object occurrences compared to using detection only. It indicates frame-level filtering gets more precise temporal information.

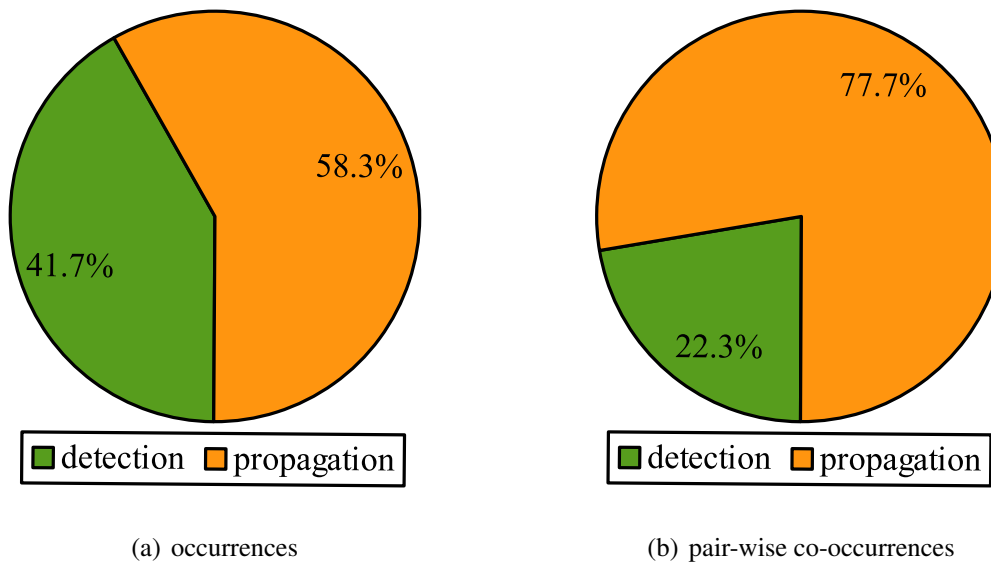


FIGURE 3.9: Share comparison

Above analysis shows that the amount of object occurrence is increased after propagation. The result is promising but we come up with another question. That is, whether the increase in occurrences would cause the increase in co-occurrences. The co-occurrences between objects are often leveraged in more complex frame-level filtering when a set of objects are given. If the amount of co-occurrences

does not increase significantly, the fixed occurrences cannot support the complex frame-level filtering well. To answer this question, we count the number of pair-wise co-occurrences between the objects which appear on the same video frames and display the ratio comparisons in Figure 3.9: the shares of detection and propagation on occurrences are revealed in Figure 3.9(a); the shares of detection and propagation on co-occurrences are revealed in Figure 3.9(b). The share results show that the amount of co-occurrences is increased significantly as well. It indicates complex frame-level filtering could be served well by the proposed hybrid method.

3.6 Summary

In this chapter, we study how to support frame-level filtering by detected visual objects. The key tasks are how to generate the unique object table and occurrence table accurately and efficiently. Based on our literature review, the visual objects used in previous methods are manually labeled. The process is top-down which heavily relies on human labors. It makes previous methods inapplicable on the dynamic or large-scale datasets. To improve these problems, we propose to use detected visual objects instead, whereas object detection fails to support object identifying and connecting as human. There are several assistant methods but all of them have many drawbacks. Accordingly, we propose a hybrid method which consists of local merge, propagation and global merge to better serve the frame-level filtering. The experiments show that the unique object table and occurrence table generated from the proposed method is better than those generated from the existing methods. Our further analysis shows that the unique object table and occurrence table generated by the proposed method supports a more accurate and efficient frame-level filtering.

Chapter 4

Video-level Filtering Using Small Non-textual Content Set

4.1 Introduction

Unlike frame-level filtering which decomposes the video into frames, video-level filtering treats the video as a whole during the process. The widely used application is keyword-based video filtering where each video in the database is associated with some texts collecting from web users or video producers. When a set of keywords are given, the videos whose texts are irrelevant to the keywords are filtered off. Keyword based filtering is inapplicable when the texts are sparse or meaningless. Accordingly, non-textual content-based filtering is introduced and has been further widely applied in video classification [52], event detection [59, 4] and so on.

The non-textual content-based filtering starts with the user specific videos which are regarded as positive exemplars in the learning process. Then, the system extract non-textual contents and perform vectorization to obtain the vectors. The classification model is trained after the vector generation using some background videos' content vectors as negatives. After that, the classification model is used to predict scores for all the videos, and the top-k videos of the highest scores are selected as the result. Usually, different content types cause different predication scores so as to the dissimilar rankings. Therefore, the fusion process is applied when the filtering process uses one more content types. Recent systems [60, 4, 123] exploit the content types as many as possible. In other word, all of them try to exploit rich content set to perform video-level filtering.

In some specific areas such as surveillance, the rich content set is not applicable for video-level

TABLE 4.1: Differences between normal and surveillance videos

	Normal	Surveillance
untrimmed	×	✓
muted	×	✓
scene independent	×	✓
noise from crowd	little	much

filtering. In Table 4.1, we list the major differences between normal and surveillance videos, which make video-level filtering cannot exploit rich content set [59, 4, 52]:

- **Surveillance videos are untrimmed:** In previous works [59, 4, 52], the input videos are trimmed. It means that the non-textual contents from the videos can be totally used as positive or negative. Differently, the surveillance videos are untrimmed. It means that a video may contain positive and negative contents at the same time, which damages the discriminative ability of the content vectors;
- **Surveillance videos are muted:** Audio is a important content source for video-level filtering [123]. However, surveillance videos are usually muted that disable the audio contents. This makes many audio content vectors cannot be used for video-level filtering;
- **Surveillance videos are scene independent:** The scene contents are often exploited for video classification [94, 116]. They are useful because many events correlate to the scene such as playing football and playground, swimming and swimming pool. However, in surveillance videos, the scene is always same under the same camera. Therefore, the scene content vectors are useless for video-level surveillance filtering;
- **Surveillance videos have noise from crowd:** Surveillance videos record the daily activities under the certain cameras. If the filtering process try to remain some video clips correlate to some specific individual activities, it is inevitably interfered by the noise from the crowd.

One of the video-level filtering applications is surveillance event detection (SED). It aims at alarming the predefined events in the surveillance videos when they occur. However, the differences in

Table 4.1 make the filtering process difficult on surveillance videos, because many content types exploited by previous works are inapplicable. This makes the motion contents become the only choice for video-level surveillance filtering. In state-of-the-art system proposed in [60], two motion contents are used. They are spatial temporal interest points (STIP) [62] and motion SIFT (MoSIFT) [12]. These motion contents leverage sparse sampling method to extract interest points from the frames, and calculate the optical flow between the temporally adjacent points to describe motions. They are ineffective when the motions are complex in the SED videos. To improve this problem, we introduced the new content set which consists of dense trajectory (DT) [107] and improved dense trajectory (IDT) [109] to improve the accuracy of SED. Our internal experiments show that the new content set significantly improves the accuracy and the conclusion helps us win the competition of TRECVID SED in 2015.

In summary, we have following contributions in this work:

- Through analyzing the characters of surveillance videos and uncovering the mechanism of recent motion features, we push the accuracy of video-level surveillance video filtering to a new level by exploiting new content set which consists of improved dense trajectory (IDT) and dense trajectory (DT). The new content set beats all the previous content sets on recent five-year TRECVID SED competition.
- We conduct extensive experiments and show how different settings influence the accuracy of video-level surveillance video filtering, which provides performance benchmark to the future followers.

4.2 Problem Statement

4.2.1 Preliminaries

The basic elements of video-level surveillance filtering system are videos and annotations. The videos are untrimmed so they cannot be used as negative or positive instantly. Additionally, they are usually captured from several cameras so the scenes are not helpful for the filtering. The annotations are classified by the events. They contain the event intervals in the videos. According to the annotations, the negative and positive could be parsed. In real-world surveillance, the amount of the annotations

is usually small. To perform video-level surveillance filtering, the users need to provide the annotations of the events on the training videos. The system then learns the prediction models to filter the irrelevant parts on the test videos and returns the high confident parts.

4.2.2 System Overview

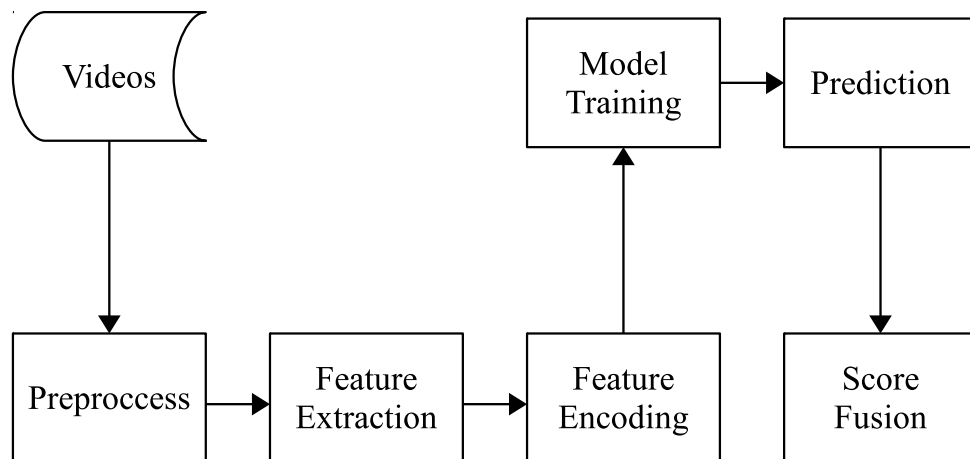


FIGURE 4.1: The pipeline of proposed video-level filtering system

The proposed system consists of five components : preprocess, feature extraction, feature encoding, model training and score fusion. The data stream flows as Figure 4.1. In this part, we will briefly introduce their functionalities.

- 1) **Preprocess** slides the training videos to generate the negative and positive instances according to the annotations, as well as the test videos for prediction. The preprocess only performs once. All the videos are delivered for feature extraction;
- 2) **Feature Extraction** extracts raw features of different content types from the video clips and normalizes them for feature encoding. The number of extraction process is equal to the number of content types used in the system;
- 3) **Feature Encoding** learns the codebooks and uses the codebooks to encode the raw features into content vectors. The number of encoding process is equal to the number of extraction process times the number of encoding settings;

- 3) **Model Training** trains the detection models based on the content vectors of positives and negatives. The scores are required to transform into $[0, 1]$ to represent the possibilities.

- 3) **Score Fusion** fuses multiple ranking scores of the video clips from different models into unified scores. The unified scores are used to filtering the irrelevant video clips finally.

4.3 Related Work

The key of video-level filtering is calculating the ranking scores given the exemplar videos. In early works, the scores are calculated by texts, click rate or demographic data [68], which require human efforts. In recent years, the research focuses on how to leverage the videos themselves to calculate the scores. As a result, many non-textual video contents have been introduced [69, 12, 109] to provide more accurate scores without human efforts. In addition to that, recent studies [4, 123] show that fusing multiple scores wisely could provide more accurate filtering results. Therefore, various learning based fusion methods such as double fusion [59], feature weighting [117] have been proposed recently.

However, due to the limitations of the surveillance videos, the existing methods for general video filtering tasks are usually poor in SED. In order to improve that, the research on SED has drawn considerable attention in the past. One early system uses MoSIFT features and BOW encoding method is proposed in [13]. Since there is only one content type in use, there is no fusion process in this system. The drawback of the system is the large amount of false alarms. To reduce them, the researchers design a cascade method to reject the positives inferred by the predictions. In [14], the system is enhanced by introducing STIP features and FV encoding method. Compared to [13], the new system gets more useful information from the encoding and the fusion. The accuracy has been improved accordingly. The system in [123] introduces IDT features for SED. However, the IDT brings numerous false positives into the results. Even though fusion could reduce the amount, but the final results are inferior to those in [14].

4.4 Proposed System

4.4.1 Preprocessing

The video preprocessing prepares the input for the feature extraction. As we mentioned, the original inputs are surveillance videos and annotations. To make the following processes efficiently and sufficiently, the preprocessing is made up of following steps in the proposed system:

- **Resizing.** The input videos are recorded in full HD. The resolution is too high for fast feature extraction. In order to accelerate the extraction, the original videos are resized to resolution 320×240 in the proposed system. In our experiments, we will show the resized videos are more suitable for SED than the original ones;
- **Sliding.** The input videos are untrimmed, which cannot provide the positive and negative exemplar videos instantly. The original videos can be trimmed by the annotations. However, due to the small amount of the annotations, the trimmed positive and negative exemplar videos are usually not enough to train robust detectors. In the proposed system, the videos are not trimmed by the annotations directly. Instead, they are slided uniformly at first. We adopt the window size of 60 frames and the step size of 30 frames to slide the videos. Considering the features in use are DT and IDT which need continuous 15 frames to track the feature points on current frames, we also append extra 15 frames at the end of each slided videos. This results in the maximum length of each video clip is 75 frames in the end;
- **Pruning.** The sliding only tries to slide the videos into short pieces. However, when the slided video clips are too small, the feature extraction fail to extract features anymore. To avoid the failures, we prune the videos whose length is below 15 length;
- **Labeling.** Since the video clips are slided uniformly, they do not have labels as those slided according to the annotations. To assign the labels, the video clips which have more than half length overlapped with the annotations are regarded as the positives. Otherwise, they are regarded as the negatives. This process enriches the amount of non-duplicate positives and stabilizes the accuracy of the proposed system on the test data.

The preprocessing applies all the above steps on the training videos, while applies the first three steps on the test videos. In addition to that, it can fast generate many short video clips in parallel.

4.4.2 Feature Extraction

TABLE 4.2: Differences between IDT, STIP and MoSIFT

	IDT	MoSIFT & STIP
sampling strategy	dense	sparse
motion removal	✓	×
description method	HOG, HOF and MBH	HOF, HOG and SIFT

The feature extraction extracts raw features from the video clips. In [14], STIP [61] and MoSIFT [12] are extracted from the surveillance videos for filtering. Recent findings in [109] and [80] show that improved dense trajectory is better than STIP and MoSIFT for many tasks. But the experiments in [123] show that IDT performs worse than STIP and MoSIFT for video-level surveillance filtering. In addition to that, IDT’s poor performance is not improved even though it is fused with STIP and MoSIFT. To figure out what make IDT worse, we check the source codes of IDT, STIP and MoSIFT. We list their major differences in Table 4.2. The first major difference is that IDT applies the dense sampling strategy, while STIP and MoSIFT apply the sparse strategy. This makes the keypoints used in STIP and MoSIFT are a subset of those used in IDT. Therefore, the fusion may not work. The second major difference is that IDT applies motion removal. This is achieved by estimating dominant motion between two adjacent frames through homography by RANSAC [32, 30], and removing the dominant motion from the motion descriptors [109]. In SED, there is no camera motion in the videos. When there are a crowd of people appear in the camera, the dominant motion is caused by the crowd. Extracting IDT in this situation can remove the interference motion contents from the irrelevant crowd.

However, the surveillance videos also have the situation where few people are moving around. In such situation, removing the dominant motion is not wise because it may contain the useful motion contents. According to this, in our proposed system, we introduce dense trajectory (DT) [107] as the complementary content to IDT instead of STIP and MoSIFT. DT is an early version of IDT which does not remove the camera motions. We think the small content set which made up of DT and IDT is more powerful than STIP and MoSIFT because the situations of both few and many persons are taken into consideration simultaneous.

After the small content set is fixed by using DT and IDT, the feature extraction in the proposed system has two steps:

- **Extraction.** The feature extraction needs to process thousands of videos clips for different content types. Even though the extractions can be parallel, the I/O issues will slow down the overall speed. To accelerate the extraction, we limit the amount of concurrency to the number of the available cores. In addition, we do not perform persistence for all the raw features. Only the raw features used for codebook learning are preserved;
- **Normalization.** The raw features can cause more accurate detection if correct normalization is applied. In the proposed system, we follow [5] to normalize the raw features by SSR. The computational cost can be neglected compared to feature extraction and encoding.

As comparison, we will also extract MoSIFT and CNN features according to [14] and [116] respectively. We will use them to show whether many features can not be used for the SED.

4.4.3 Feature Encoding

The feature encoding encodes the raw features into content vectors. In SED, this step is necessary because it makes the content vectors more discriminative. The state-of-the-art encoding method is the spatial-temporal fisher vector (SFV) [55]. Compared to the standard fisher vector (FV), this method also encodes the spatial-temporal information of the feature points into the content vectors.

The proposed system applies four steps to generate the content vectors:

- **Dimension reduction on raw features.** This steps learns a projection matrix on the sampled features by Principle Components Analysis (PCA). Then, use the projection matrix to map the raw features into a low-dimensional space. This step is necessary because following steps assume there are no co-variances between different feature dimensions.
- **Codebook learning on sampled features.** This steps learns the codebook for encoding. In fisher vector, the codebook is formed by Gaussian Mixture Model (GMM), which leverage multiple Gaussian components to reconstruct the feature space. In our system, the learning is initialized by k-means clustering and optimized by EM algorithm.

- **Fisher encoding.** This step performs the encoding actually and generates the content vectors. It leverages the means and variances from the Gaussian components from the codebook to calculate the derivatives. Since these derivatives describe different gradient information, they are concatenated to form a high-dimensional vector. Given K components and one D dimensional feature point, the fisher encoding results in a $2KD$ dimensional vector. The high dimensional vectors from multiple feature points are dimension-wise averaged. Therefore, each video clip generates one content vector in the end.
- **Normalization.** This step further enhances the discrimination of content vectors. After encoding, some dimensions will dominate the similarity computation between two vectors, which harms the discrimination. In order to improve that, the content vectors are firstly applied signed square root (SSR) to weaken the dominant dimensions. In addition to that, during the model training, it is recommended to make the instances be the most similar to themselves. To achieve that, the length of each content vector is normalized to 1 by applying power normalization.

In previous systems [14, 123], the projection matrix in dimension reduction is learned by standard PCA. Recent studies in [80] show that whitened PCA could improve the detection accuracy further. Therefore, in the proposed system, we also applied whitened PCA to perform the dimension reduction. This results in another group of content vectors in the proposed system. The effects of whitened PCA will be examined in the experiments.

4.4.4 Model Training

The model training creates detectors for each event. Because there exist multiple cameras, the detectors are further trained for different cameras. The training has three steps:

- **Detector Training.** This step uses classification model to train the detectors. In previous works [60, 123], the widely used model is kernel svm [11]. It selects the support vectors during the training and decides whether the new instances are positives according to the distances between them. As the vector dimension increases by fisher encoding, the number of support vectors increases as well. This makes keeping the support vectors no longer a good choice. For instance, in our system, each content vector occupies 0.5MB space no matter on disk or in memory. Each detector usually keeps roughly 5000 support vectors. Adding all the support vectors results in 2.5GB cost on disk or in memory. Considering the detectors are trained for

different events and cameras, the overall space cost is amazingly huge. To reduce the space cost, in our proposed system, we apply linear svm [29] to train the detector. Instead of keeping support vectors, linear svm transforms these vectors into weights for prediction. After that, each detector only occupies 0.5MB on disk or in memory.

- **Score Learning.** The public available source code of linear svm [29] is older than that of kernel svm [11]. This makes the detectors trained by linear svm only generate distances rather than probabilities. The distances usually are not in same scale. If the fusion is directly applied on the distances, less accurate results could be generated. To improve that, there requires a method to map the distances into probability scores. To do so, we refer to [85] to learn the probabilities from the scores. Our python implementation is open available ¹. We verify this code by reproducing the action recognition experiment in [109].
- **Parameter Selection.** The regularization parameter of svm need to tune during the training. Since the search space is infinite, we empirically apply enumeration method to select the regularization parameter. The search space of the parameter is defined as [0.01, 0.1, 1, 10, 100], and selected by two-fold cross validation. It means we divide the whole training data into two disjoin parts. When one fold is used for training, the other fold is used for evaluation. The best parameter is selected by the best average performance on the tow folds.
- **Positive Selection.** The initial outputs of the detectors are probabilities, which indicate the relevances between the video clips and events. They cannot indicate whether and when the events happened. To parse these final outputs from the initial results, the first step is to learn the threshold for binarizing the probabilities, which is achieved by cross validation. After this step, the system gets the positive indicators on the time intervals predefined by the sliding process. The second step is to compress the positives. The positive indicators may appear on several continuous intervals, which potentially increases the amount of false positives. In order to amend this problem, we introduce non-maximum suppression (NMS) [75] to prune the positives. After that, the system returns the remaining positives as output.

¹<https://github.com/domainxz/pytools.git>

4.4.5 Score Fusion

Given T content types, C cameras and E events, the model training generates $T \times C \times E$ binary detectors in total and results in the ranking lists of same amount. For system outputs, there only require $C \times E$ ranking lists. The system outputs can be generated by selecting one content type. But it is not recommended due to the low accuracy [14, 123]. The more suitable process is score fusion which transforms the ranking lists derived from different content types into one. In our proposed system, we select the direct but efficient way to perform score fusion, which averages the probabilities and thresholds from different selected content types to form the final results. After that, the key issue is which content types should be fused together. After apply the detectors on the cross-validation data, we get four group of detection scores at hand. They are predicted by the fisher vectors in terms of dense trajectory with normal PCA (dt_{fv}), dense trajectory with whiten PCA (dt_{wfv}), improved dense trajectory with normal PCA (idt_{fv}) and improved dense trajectory with whiten PCA (idt_{wfv}). We enumerate all the possible small sets to select the best one, and find the combination of dt_{wfv} and idt_{wfv} is the best. The details are displayed in the experiments.

4.5 Experiments

4.5.1 Dataset

The data for video-level surveillance filtering is from NIST SED competition [79], which consist of raw videos and annotations. The raw videos are from London airport, which are recorded by 5 cameras. They are divided into two parts. The first part is training videos which contain 10-day surveillance. The second part is test videos which are recorded after the training videos. The total duration of the test videos are 10 hours. The events required to detect are provided by annotations. There are seven events in total, namely, CellToEar, Embrace, ObjectPut, PeopleMeet, PeopleSplitUp, PersonRun and Pointing. The annotations contain the durations of different events on the training videos. In Table 4.3, we list the statistics of the seven events, which include the amounts and the duration medians. PersonRuns, PeopleMeet, PeopleSplitUp and Embrace have more than 2-second duration on average. We call them as long-duration events. The rest events are called as short-duration events.

TABLE 4.3: The statistics of events on training videos

Event	Amount	Duration Median
PersonRuns	632	2.68s
CellToEar	488	0.80s
ObjectPut	2457	1.00s
PeopleMeet	1102	3.44s
PeopleSplitUp	1469	6.36s
Embrace	878	2.88s
Pointing	2588	1.28s

4.5.2 Evaluation Plan

The filtering accuracy is measured by counting the system errors. The metric in used is detection cost rate (DCR). It is made up of two sub-metrics after the predictions from the system are aligned to the ground truth. The first sub-metric is positive miss (P_{miss}). If one ground truth event has not hit any positive predictions, a P_{miss} will be reported. The second sub-metric is false alarm. If the positive predictions have 50% less overlap with the corresponding ground truth, a false alarm (FA) will be reported. After counting all the P_{miss} and FA of one specific event, the DCR is calculated by Equation 4.1.

$$DCR = \frac{P_{miss}}{N_P} + 0.005 \cdot FA \quad (4.1)$$

Since DCR measures the system' errors, a smaller DCR value means higher accuracy. If the system does not perform detection, the DCR will be 1. If the system does not have errors, the DCR will be 0,. Currently, none of the existing systems can perform error-free detections.

The value of DCR is decided by the threshold. Therefore, there are two values for references in the experiments. The first value is $aDCR$ which is the actual detection error of the proposed system. The other value is $mDCR$ which is the ideal detection error of the proposed system. $mDCR$ is usually smaller than $aDCR$ because it searches the best threshold according to the ground truth, which is impossible for the proposed system leveraging.

4.5.3 Experimental Results

Small Content Set Comparison

The first experiment tries to select the best small content set. In order to compare with previous works, the number of content types is set to 2. The best 4 combinations are recorded in Table 4.4. We can clearly see the small set which consist of dt_{wfv} and idt_{wfv} is the most accurate.

TABLE 4.4: Evaluations for fusion strategy

Event	$dt_{fv} + idt_{fv}$	$dt_{fv} + idt_{wfv}$	$idt_{fv} + idt_{wfv}$	$dt_{wfv} + idt_{wfv}$
CellToEar	1.0058	1.0013	1.0036	1.0040
Embrace	1.0068	0.9253	0.9197	0.9105
ObjectPut	1.0042	1.0023	1.0026	1.0020
PeopleMeet	0.9520	0.9238	0.9369	0.9297
PeopleSplitUp	0.9613	0.8931	0.9036	0.8861
PersonRuns	0.6440	0.6478	0.6549	0.6299
Pointing	1.0140	0.9920	0.9891	0.9858

Resized Videos vs. Original Videos

The first experiment display the impact of small feature set. In our second experiment, we want to examine how the resized videos influences the accuracy of surveillance video filtering. Table 4.5 shows the aDCR from our cross validation. The results clearly indicate that original videos could improve the filtering accuracy of Embrace and PersonRuns events.

Based on the aDCR from individual content types, we further fuse them according to the first experiment to see whether the accuracy could be remained after fusion. The aDCR values of the fusion on validation and the feedback from NIST are displayed in 4.6. The results in Table 4.6 show the poor performance of detection on original videos for formal submission, while the performance of detection on resized videos is quite consistent.

To figure out why the performance divergence of detection on original videos is so high, we look inside the aDCR and extract the positive miss (P_{miss}) and false alarm (FA). The sub-metrics show that detection on original videos increase the amount of FA significantly. This drawback ruins the

TABLE 4.5: Filtering accuracy with resized and original videos

event	resized		original	
	dtwfv	idtwfv	dtwfv	idtwfv
CellToEar	1.0009	1.0018	1.0026	1.0022
Embrace	1.0050	1.0131	0.9197	0.9774
ObjectPut	1.0057	1.0059	1.0154	1.0218
PeopleMeet	0.9589	0.9516	0.9634	0.9710
PeopleSplitUp	0.9610	0.9595	0.9798	0.9752
PersonRuns	0.6634	0.6458	0.6193	0.6119
Pointing	0.9890	0.9956	0.9887	0.9908

TABLE 4.6: Fusion under resized videos and original videos

event	validation		submission	
	resized	original	resized	original
CellToEar	1.0013	1.0008	1.0046	1.0140
Embrace	0.9251	0.8409	0.8680	0.8646
ObjectPut	1.0034	1.0133	1.0160	1.0044
PeopleMeet	0.9172	0.9200	0.8939	0.9269
PeopleSplitUp	0.8821	0.8712	0.8934	0.8909
PersonRuns	0.6426	0.5325	0.5768	1.0303
Pointing	0.9869	0.9826	1.0140	1.0057

overall performance of the detection. We think it is because the motion contents in original videos are richer than those in resized videos. This brings much more noise than useful signals. To this end, the performance of detection on original system is inferior to that on resized videos.

TABLE 4.7: Ground True (GT), Positive Miss (P_{miss}) and False Alarm (FP) comparison

event	Resized			Original		
	GT	P_{miss}	FA	GT	P_{miss}	FA
CellToEar	54	54	8	77	77	28
Embrace	138	76	552	173	131	215
ObjectPut	289	282	70	348	345	26
PeopleMeet	256	156	495	323	231	424
PeopleSplitUp	152	95	467	176	135	248
PersonRuns	50	22	238	63	36	1237
Pointing	794	759	101	929	899	76

Comparison with Other Systems

We use the conclusions from above experiments to generate submission for the TRECVID SED 2015 competition. The results are listed in Table 4.8. Our overall retrospective results (no human assistance) outperform the other retrospective systems. Besides that, the proposed system can also beat the results from the systems with human assistance.

TABLE 4.8: Competition results in TRECVID SED 2015

Event	Our retro results		Best retro results		Best inter results	
	aDCR	mDCR	aDCR	mDCR	aDCR	mDCR
CellToEar	1.0046	1.0006	1.3071	1.0006	2.1010	1.0006
Embrace	0.8680	0.8453	0.7909	0.7909	0.8540	0.8540
ObjectPut	1.0160	0.9884	1.0120	0.9965	0.9930	0.9867
PeopleMeet	0.8939	0.8848	1.0426	0.9981	0.9978	0.9919
PeopleSplitUp	0.8934	0.8785	0.9387	0.9253	0.9164	0.9164
PersonRuns	0.5768	0.5466	0.9700	0.9545	0.9411	0.9411
Pointing	1.0140	0.9940	1.0040	0.9989	0.9939	0.9939

In addition to that, we achieve the state-of-the-art aDCR in PersonRuns event. We compare our

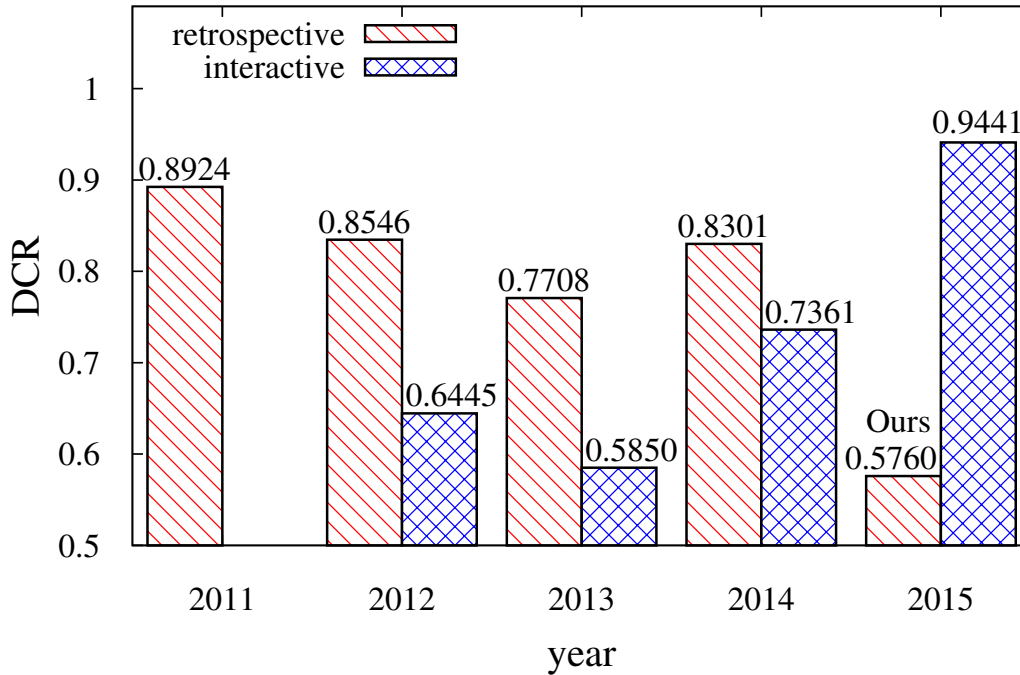


FIGURE 4.2: best aDCR for PersonRuns in recent five years' retrospective and interactive systems.

result to recent five years' best results in Fig. 4.2. We find this score achieves the new record in recent years' SED competitions, even though the test data become harder since 2014.

4.6 Summary

In this chapter, we study how to perform video-level filtering in a surveillance environment. Due the differences between normal and surveillance videos, the filtering methods cannot exploit rich content set in this situation. It only allows to use a small motion content set to filter the videos. The existing content set is made up of MoSIFT and STIP which are sparse sampling motion features. In our work, we try to introduce the dense sampling motion features as replacement. The challenge of this study is that improve dense trajectory (IDT) alone performs poor in the surveillance filtering. We need to find a good complementary feature to work with IDT. This cannot be achieved by MoSIFT or STIP in previous works. Therefore, we introduce dense trajectory (DT) in our study by analyzing what make IDT imperfect on surveillance videos. Our study further uncovers the importance of resizing, encoding and fusion, which helps us win the TRECVID SED 2015 retrospective competition.

Chapter 5

User-level Filtering Using Rich Content Set

5.1 Introduction

Watching online videos has become one of the indispensable entertainment activities in daily life. Many famous websites, such as YouTube, Netflix and Hulu, host a tremendous number of videos to meet such demand. However, these massive video repositories place an enormous burden on users when trying to find videos of interest [2, 88]. To improve this situation, most video websites have adopted user-level video filtering service, namely recommender systems, as an effective way to help users explore the world of videos [20, 36]. Existing user-level filtering methods can be categorized into three classes [8, 2]: content-based methods, collaborative filtering (CF)-based methods, and hybrid methods. Content-based methods make use of user profiles and item descriptions, e.g., item contents, for filtering videos. CF-based methods use the historical user activity or feedback, such as user ratings, but not user or item content information. Hybrid methods [3, 45] seek the best of both worlds by combining both content and CF-based methods.

With the rapid expansion of video websites and platforms, and a dramatic increase in the amount of available videos, existing video recommender systems are confronted with two critical problems: data sparsity and cold start. The number of videos a user can watch is limited, and most videos receive a small number of ratings. The user-video interaction/rating matrix is thus extremely sparse, which significantly limits the performance of CF-based methods. Moreover, thousands of new videos are uploaded to video websites very day. Collaborative filtering and matrix factorization methods, which use only user-video matrix information without any content information, are not effective for recommending new videos. These new videos are called cold-start videos or out-of-matrix videos. To

tackle these problems, hybrid recommendation methods, which combine collaborative filtering and auxiliary information such as item content, usually achieves more accurate filtering results and have gained increasing popularity in recent years.

Most existing hybrid recommender systems [110, 106, 71, 45] integrate textual content to improve recommendations. However, the scarcity of textual content, especially for user-generated videos, makes these hybrid recommendation methods ineffective. For example, plenty of videos on Youtube only have titles. A few recent works [118, 78, 42] have tried to exploit non-textual content features (i.e., multimedia features) for video, music and product recommendation, but have only focused on in-matrix recommendation scenarios. In these cases, the user-item interaction matrix information actually dominates the model learning process, and the effect of non-textual content is not significant. As such, whether non-textual features can really benefit out-of-matrix recommendations, is still unexplored, and is an important issue for video recommendation given the fast pace of today's video generation.

Given traditional video features such as normalized color histogram and aural tempos, have proven to be unhelpful for improving the video recommendation [118], we first introduce several new non-textual content features to represent videos. Intuitively, users might be interested in a video for many reasons. We thus propose to use MFCC [4], SIFT [12, 104], improved dense trajectory (IDT) [109] and convolutional neural network (CNN) [56] to extract and quantize the audio, scene and action information contained in the videos. Encoding these non-textual content features with the state-of-the-art methods [82, 55, 50] will enable generation of more effective and expressive content features [4, 109, 116].

Using both the widely used textual content features and these new non-textual content features, we first reproduced and tested the state-of-the-art hybrid recommendation methods [106, 78, 110, 42] in both in-matrix and out-of-matrix scenarios. The results showed that none of these methods achieved high recommendation accuracy in both scenarios. In particular, we observed that weighted matrix factorization (WMF)-based methods achieved better performance in the in-matrix scenario, while Bayesian personalized ranking (BPR)-based methods generated more accurate recommendations in the out-of-matrix scenario. To improve that, we propose a collaborative embedding regression method (CER) based on WMF in this work. Unlike existing WMF-based methods [106, 78, 110] which apply non-linear learning on the content features, CER applies linear learning instead, considering that (1) the non-textual content features are encoded to work with the linear learning models [82, 55, 50];

and (2) the content features are usually of high dimensionality, and linear learning is more efficient than the non-linear learning. The experimental results show that, for any individual content feature (either non-textual or textual), CER performs slightly better than other WMF-based methods in the in-matrix scenario, and significantly outperforms both WMF and BPR-based methods in the out-of-matrix scenario. Moreover, CER's model training is more efficient and more scalable to large datasets than the other methods'.

In addition, observing that different content features have significantly diverse performance in out-of-matrix recommendation, we have also studied how to use multiple content features of videos to further improve top- k recommendations in the out-of-matrix scenario. In recent years, designing fusion strategies of multiple features has become a major research trend and different techniques have been proposed. There are two widely accepted yet independent strategies to fuse multiple features [17]: early fusion and late fusion. Most works on early fusion try to map multiple feature spaces to a unified one. For example, in [98, 124, 113], multiple original features are mapped to a latent space with lower dimensionality based on neural networks. Although some interactions among features can be captured by such a framework, a number of problems exist. First, a unified feature space is often built according to global statistical information using deep learning models, which incurs extremely high computational costs for large-scale video databases, each with tens of thousands dimensions. Second, the textual, audio, visual and action information contained in videos are widely diverse and heterogeneous. It is almost infeasible to construct a shared latent space for recommendation without losing some important and meaningful feature information.

The other line of research focuses on the late fusion of multiple features. This fusion strategy uses separate result lists derived from different features, and carries out fusion using the candidate results [101, 84]. Learning-to-rank techniques (e.g., ranking SVM) represent the state-of-the-art of late fusion mechanisms [68, 4, 116]; however, as supervised learning techniques, learning-to-rank models can only be trained based on user-video interaction matrix in our problem. The feature weights learned in in-matrix setting are not suitable for out-of-matrix setting, as these two settings have disparate characteristics and intrinsically different. Also, training learning-to-rank models is time-consuming. Instead, we propose a novel unsupervised late fusion method to compute the feature weights that does not depend on user-video interaction information.

To summarize, the contributions of this work include:

- To the best of our knowledge, this is the first effort to leverage MFCC, SIFT, IDT as well as

CNN features for video recommendations and to study their effect in improving out-of-matrix recommendations.

- We propose a novel hybrid video recommender model, CER, to effectively combine collaborative filtering with both textual and non-textual content features in a unified way. We also study how to fuse multiple types of content features to further improve out-of-matrix recommendation and propose a novel fusion method.
- We conduct extensive experiments to evaluate both the proposed CER and the unsupervised late fusion method. The results reveal that our approaches significantly outperform competitor methods.

5.2 Problem Statement

5.2.1 Preliminaries

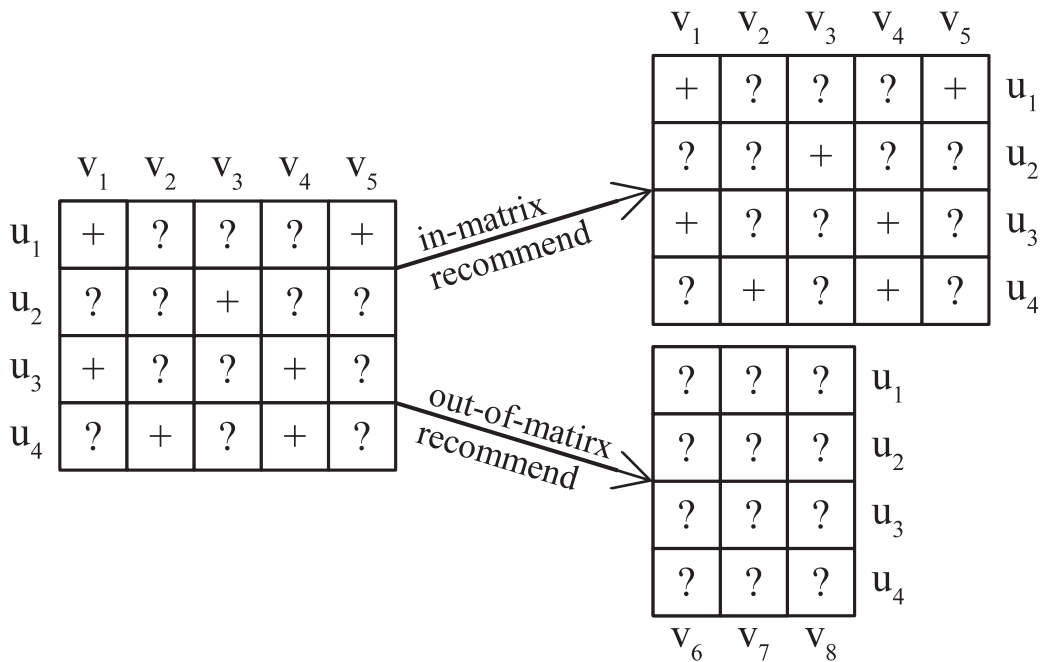


FIGURE 5.1: Implicit rating matrix for in-matrix and out-of-matrix recommendation.

The basic elements of a video recommender system are users and videos. Assume there are m users and n videos in total. As shown in Figure 5.1, we use $r_{ij} \in \{?, +\}$ to denote the i^{th} user's implicit rating/feedback on the j^{th} video: $r_{ij} = +$ means the i^{th} user likes the j^{th} video; $r_{ij} = ?$

means the i^{th} user dislikes the j^{th} video or is not aware of the j^{th} video. As a convention[87], we map $\{?, +\}$ to $\{0, 1\}$.

Given a target user, the video recommender system aims to find the top- k videos that the user is potentially interested in. The video recommendation can be further divided into two settings: in-matrix and out-of-matrix recommendations. In the in-matrix setting, the recommender system recommends the top- k videos which have not been rated by the target user but have been rated by other users[106]. Based on the co-rating behaviors of similar users, state-of-the-art methods[47, 106, 78, 110] use collaborative filtering (CF) to generate recommendations. In out-of-matrix setting, the recommender system suggests top- k new videos that have not been rated by any user[106] (i.e., cold-start recommendation). In this setting, CF-based methods become ineffective, whereas content-based methods perform well.

5.2.2 System Overview

The proposed systems aims to perform user-level video filtering with rich content vectors. It consists of several steps:

- 1) **Content Vector Generation** extracts raw features from the videos and use feature encoding methods to transform the raw features into vectors. The content types in use are MFCC, OSIFT, MoSIFT, IDT, CNN. The encoding methods in use are fisher vector (FV) and VLAD.
- 2) **Model Training** learns the score predictors based on the rating matrix and the content vectors. It tries to obtain the latent factors from both user collaboration and video contents, and uses the latent factors for both in-matrix and out-matrix score predictions.
- 3) **Multiple Content Fusion** fuses the scores of a video from different contents into the uniformed one. It has two strategies, namely, early fusion and late fusion. The method in the proposed system is late fusion because it has superior performance to early fusion.

5.3 Related Work

For top- k recommendation, weighted matrix factorization (WMF) [54] and Bayesian personalized ranking (BPR)[87] represent the state-of-the-art performance in in-matrix setting. Both of them are

matrix factorization models and are derived from collaborative filtering (CF). They learn a latent vector to predict each user's rating on each item, for each user and item in turn, and then select the top ranked items with the highest predicted ratings. The major difference between them is the optimization objective. The WMF model[54] learns the latent factors by minimizing the rating prediction loss on the training data, while the BPR model[87] learns the latent factors by preserving the pair-wise personalized rankings.

Recently, both WMF and BPR were extended to incorporate content features, so they can learn a latent vector to represent both in-matrix and out-of-matrix items, and hence be applied to both in-matrix and out-of-matrix recommendation scenarios. The representative WMF-based models include collaborative topic regression (CTR)[106], deep content-based music recommendation model (DPM)[78] and collaborative deep learning (CDL)[110]. CTR and CDL only integrate the textual features of items, while DPM only considers non-textual features. The models for learning the content latent vectors are latent Dirichlet allocation (LDA), stack de-noising auto-encoder (SDAE) and multiple layer perception (MLP) respectively. The representative BPR-based models are visual Bayesian personalized ranking (VBPR)[42], collaborative knowledge base embedding (CKE)[124] and Visual-CLiMF[90]. VBPR and Visual-CLiMF are designed to incorporate with single feature, while CKE works with both structural and non-structural features from the knowledge base by adding them up. Visual-CLiMF enhances VBPR by optimizing the approximate reciprocal rank instead of pair-wise rank. All of these BPR variants use linear embedding to learn the latent content vectors.

5.4 Proposed System

5.4.1 Content Vector Generation

This section describes how the content features, including both textual and non-textual, are extracted for video recommendation. They are used for content-based inference in Figure 5.2.

The proposed system not only generate the textual content vectors but also the non-textual contents. Two kinds of textual content vectors are generated. They are word vector and meta vector. In addition to that, six kinds of non-textual content vectors are generated in the proposed system. The raw features in use are MFCC, OSIFT, MoSIFT, IDT and CNN. Unlike MFCC, MoSIFT and IDT which take the whole audio or video file as input, OSIFT and CNN are applied to the frames sampled

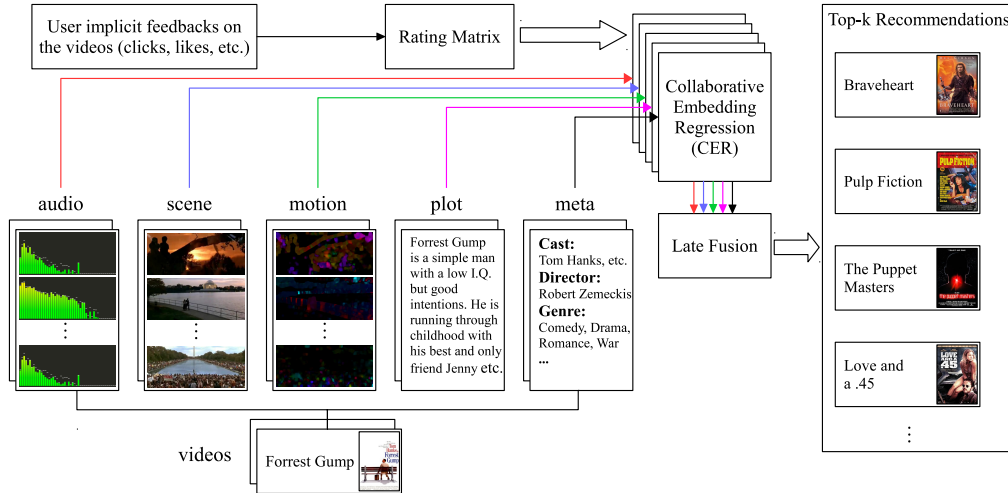


FIGURE 5.2: The flowchart of exploiting rich contents to recommend videos.

from the video. Following[4, 116], we fetch 5 frames per second from the video. Besides, there is usually a normalization process on the raw features. We apply SSR (signed squared root) to normalize all the raw features[5, 4].

Feature	Encoder	Dimension
MFCC	FV	10240
OSIFT		98304
MoSIFT		68608
IDT		128304
CNN		131072
	VLAD	65536

TABLE 5.1: The dimensions of the encoded non-textual content vectors.

We obtain a group of vectors for each non-textual content feature. These feature vectors need to be transformed into one feature vector to be incorporated into collaborative filtering[106, 78, 110, 42]. An intuitive transformation method is to simply average the feature vectors by dimension. But this is not a good choice due to its limited representative capacity[80]. Recent studies show that it is better to transform the feature vectors using an encoding process[82, 4, 80, 116]. As a result, we apply two state-of-the-art encoding methods, Fisher vector (FV)[82] and VLAD[50], to transform a group of feature vectors to one vector. The encoding methods and the resulting dimensions for each non-textual feature in Table 5.1. We notice that the dimensions of all the encoded feature vectors are very

high. This high dimensionality makes it infeasible to integrate with collaborative filtering (i.e., latent factor models). Thus, we apply PCA to reduce the dimension of each feature to 4000, following[81].

5.4.2 Video Recommendation

In this section, we first reproduce existing recommender models and analyze their performance in both in-matrix and out-of-matrix settings. Based on the results, we study the possible reasons why these state-of-the-art recommender models cannot deliver effective video recommendations with non-textual features. Inspired by the results, we propose an improved recommender model, CER, followed by a novel late fusion strategy to fuse the recommendation lists from different content features to further improve recommendation accuracy.

Recent Methods on Various Features

Methods	Contents
WMF, BPR	N/A
CDL, VBPR, CTR	WORD, META
CDL, VBPR, DPM	MFCC
CDL, VBPR	CNNFV

TABLE 5.2: The state-of-the-art recommender models and the corresponding content features in use.

Given the set of extracted content features associated with videos, an interesting question naturally arises: how do state-of-the-art recommender models perform with these features in top- k recommendations. To answer this question, we reproduce the WMF and BPR-based recommender models using the Movielens 10M dataset[39]. We adopt the optimal parameter settings proposed in[47, 87, 106, 78, 110, 42]. Additionally, we extend CDL[110] and VBPR[42] to work with vectors from MFCC and CNNFV. The recommender models and the corresponding content features are listed in Table 5.2.

The recommender models listed in Table 5.2 were tested in both in-matrix and out-of-matrix settings with their optimal parameters. More details about the dataset splits and evaluation metrics are discussed in the experiments section. The results are presented in Figure 5.3 where the subscripts of

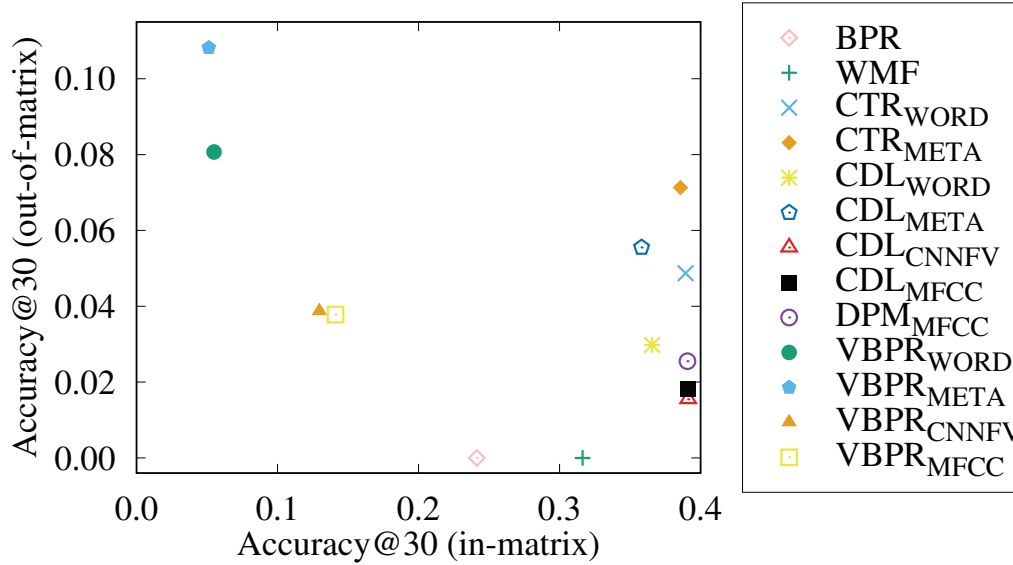


FIGURE 5.3: Performance of the state-of-the-art methods in both in-matrix and out-of-matrix settings. To clearly display the methods which only support in-matrix recommendation, we shift the origin of the vertical axis to a higher position.

the models denote the content features in use. To clearly show the differences between these methods, we use the evaluation metric Accuracy@30. Figure 5.3 provides the following observations:

1. **WMF-based recommender models yielded more accurate recommendation than BPR-based models in the in-matrix test.** In Figure 5.3, all the WMF-based models (i.e. WMF, CTR, DPM and CDL) are located to the right of the BPR-based models (i.e. BPR and VBPR). Additionally, the performance of the WMF-based models (e.g., CDL) in the in-matrix test do not vary significantly with respect to the different types of content features, while the introduction of content features improves the basic WMF. All these facts indicate that the WMF-based models in the in-matrix scenario are mainly dominated by their collaborative filtering component WMF, and they are insensitive to feature types.
2. **VBPR achieved the best performance in the out-of-matrix test.** In Figure 5.3, given a particular content feature, the position of VBPR is always higher than that of all the other methods. This shows that VBPR is the most effective method in the out-of-matrix scenario, and the content-based components in the existing WMF-based models are not suitable for out-of-matrix recommendations.

In summary, our reproduction experiment shows that none of the existing recommender models achieve high recommendation accuracy in both in-matrix and out-of-matrix scenarios. To address this

problem, we propose a new WMF-based recommender model CER in this work.

Collaborative Embedding Regression

All recent WMF-based models[106, 78, 110] follow a similar rating generation process. The major difference among them is the way they generate content latent vectors. CTR[106] incorporates textual features with WMF and generates the content latent vectors using latent Dirichlet allocation (LDA). Since the optimization of LDA is based on word count only, CTR naturally fails to support non-textual features that are real values. Compared to CTR, DPM[78] and CDL[110] can generate various content latent vectors from both textual and non-textual features. They achieve this by respectively applying multiple layer perception (MLP) and stacked de-noising auto-encoder (SDAE) as generation functions. However, the results in Figure 5.3 show that neither of them perform well in out-of-matrix setting, especially those with non-textual features. This is because the non-textual features are encoded for linear learning[82, 55, 50]. Thus, MLP and SDAE that perform non-linear learning degrade the performance of the encoded non-textual features. On the other hand, the excellent performance of VBPR actually benefits from its adoption of the linear embedding method[42]. Based on above analysis, we propose a novel recommender model, collaborative embedding regression (CER), to work with both textual and non-textual features. Let d denote the dimension of the content feature and k denote the dimension of the latent vector. The whole generation process of CER with an individual content feature is described below.

1. For each user i , draw a user latent vector $w_i \in \mathcal{R}^{k \times 1}$:

$$w_i \sim \mathcal{N}(0, \lambda_u^{-1} I). \quad (5.1)$$

2. Generate an embedding matrix $E \sim \mathcal{N}(0, \lambda_e^{-1} I)$.

3. For each video j :

- (a) Generate a content latent vector $h'_j \in \mathcal{R}^{k \times 1}$:

$$h'_j = E^T f_j. \quad (5.2)$$

- (b) Draw a latent video offset vector $\epsilon_j \sim \mathcal{N}(0, \lambda_v^{-1} I)$, and then set the video latent vector as:

$$h_j = h'_j + \epsilon_j. \quad (5.3)$$

4. For each user-video pair (i, j) , draw the rating:

$$r_{ij} \sim \mathcal{N}(w_i^T h_j, c_{ij}^{-1}). \quad (5.4)$$

where I is an identity matrix, $f_j \in \mathcal{R}^{d \times 1}$ is a feature vector, $E \in \mathcal{R}^{d \times k}$ is an embedding matrix, and c_{ij} is the confidence parameter for the user-item pair (i, j) . Following[106, 110], the value of c_{ij} is defined below:

$$c_{ij} = \begin{cases} 1, & \text{if } r_{ij} = 1 \\ 0.01, & \text{if } r_{ij} = 0 \end{cases} \quad (5.5)$$

Note that, in step 3(a), we use linear embedding instead of non-linear learning adopted by CTR, DPM and CDL. This is more suitable for learning the content latent vectors from the non-textual features[82, 4, 109]. In step 3(b), h'_j serves as the bridge between the implicit feedback preference and the video content features.

Learning the parameters. To predict the rating, the latent vectors and the embedding matrix need to be learned. As computing the full posterior of the parameters is intractable and maximizing the posterior probability of W , H and E is equivalent to maximizing the log-likelihood, we follow[106] to minimize the negative log-likelihood as follows:

$$\begin{aligned} & \sum_{i=1}^m \sum_{j=1}^n \frac{c_{ij}}{2} (w_i^T h_j - r_{ij})^2 + \frac{\lambda_u}{2} \sum_{i=1}^m w_i^T w_i + \\ & \frac{\lambda_v}{2} \sum_{j=1}^n (h_j - E^T f_j)^T (h_j - E^T f_j) + \frac{\lambda_e}{2} \|E\|_F^2, \end{aligned} \quad (5.6)$$

where λ_u , λ_v and λ_e are the hyper parameters and $\|\cdot\|_F$ denotes the Frobenius norm. When these hyper parameters are fixed, the optimal latent vectors w_i and h_j as well as the embedding matrix E are obtained by performing the alternating least squares (ALS), following[106, 110]. Specifically, in each iteration, given the current estimation of E , the derivatives with respect to w_i and h_j are computed and set to zero. We then derive the following updating formulas for w_i and h_j :

$$\begin{aligned} w_i & \leftarrow (HC_i H^T + \lambda_u I_k)^{-1} HC_i R_i \\ h_j & \leftarrow (WC_j W^T + \lambda_v I_k)^{-1} (WC_j R_j + \lambda_v E^T f_j) \end{aligned} \quad (5.7)$$

where $W = (w_i)_{i=1}^m \in \mathcal{R}^{k \times m}$ is the matrix formed by user latent vectors, $H = (h_j)_{j=1}^n \in \mathcal{R}^{k \times n}$ is the matrix formed by video latent vectors, and $F = (f_j)_{j=1}^n \in \mathcal{R}^{d \times n}$ is the content matrix. For user i , $C_i \in \mathcal{R}^{n \times n}$ is a diagonal matrix with c_{ij} , $j = 1 \cdots, n$ as the diagonal elements, $R_i \in \{0, 1\}^{n \times 1}$ is a vector with r_{ij} , $j = 1 \cdots, n$ as its elements. For video j , C_j and R_j are similarly defined.

Then, we fix the current estimation of H , and the derivatives with respect to E are computed and set to zero. We derive the following updating formula for E :

$$E \leftarrow (\lambda_v F F^T + \lambda_e I_d)^{-1} (\lambda_v F H^T). \quad (5.8)$$

Similar to CTR and CDL, CER supports both in-matrix and out-of-matrix rating prediction. For in-matrix predictions, given a user-video pair (i, j) , the rating \hat{r}_{ij} is estimated as $w_i^T (E^T f_j + \epsilon_j)$. For out-of-matrix prediction, the rating \hat{r}_{ij} is predicted as $w_i^T E^T f_j$ since no offset is observed. In summary, the rating predictor is defined as:

$$\hat{r}_{ij} = \begin{cases} w_i^T h_j, & \text{in-matrix setting} \\ w_i^T E^T f_j, & \text{out-of-matrix setting} \end{cases} \quad (5.9)$$

Multiple Feature Fusion

The CER model presented in the previous subsection is designed to work with a single type of feature, just like most of the recent hybrid recommender models[106, 78, 110, 42]. In this subsection, we will study how to leverage rich and diverse content features to further improve the video recommendation. Specially, we present three feature fusion methods to facilitate CER to work with multiple types of features.

The first method concatenates all the feature vectors associated with a video into one big vector and then feeds the big vectors into CER. Assuming there are L features in total, the concatenation is performed as follow:

$$f_j \leftarrow [f_j^1, f_j^2, \dots, f_j^L]. \quad (5.10)$$

This fusion method is expected to learn the shared latent factors among the concatenated features. It does not introduce any modification on the objective function of CER, but it will significantly increase the training time of the CER because the time complexity of CER's optimization is proportional to the dimension of the feature vector f_j .

The second method adds all the content latent vectors h_j^l together, as done in CKE[124]. The content latent vectors in the generation process of CER are redefined as:

$$h_j' = \sum_{l=1}^L h_j^l = \sum_{l=1}^L E^l f_j^l. \quad (5.11)$$

Compared to the first method, the second method compresses the dimension so that the training is faster, but it needs to modify the objective function of the CER by adding the regularization terms of

all the embedding matrices, and the updating formulas of the model parameters also need to change accordingly.

The above two methods are early fusion methods. They try to map multiple feature spaces to a unified one. However, the textual, audio, visual and action information contained in videos are widely diverse and heterogeneous. It is almost infeasible to construct a shared latent space for recommendation without losing some important and meaningful feature information. Besides, early fusion methods require re-training models when the features in use are changed (e.g., adding new features). From these two perspectives, the early fusion tends to be inferior to the late fusion which directly works on the results obtained from each type of feature.

As shown in Figure 5.3, the performance of WMF-based models in the out-of-matrix scenario varies greatly with respect to the different types of content features. In a recent video retrieval system[4], such divergence is leveraged by fusing multiple ranking lists to obtain a more relevant ranking list. Inspired by[4], we consider the late fusion has the potential to improve the video recommender system as well. We thus propose the third method to fuse the top- k recommendations generated from multiple content features.

Inspired by the success of learning-to-rank techniques[4, 116], in the third method, we first compute a weight for each feature and then apply the weighted sum strategy to implement the late fusion. The fused estimation rating is computed as follows:

$$\bar{r}_{ij} = \sum_{l=1}^L \pi_l \hat{r}_{ij}^l, \quad (5.12)$$

where L is the number of content features; π_l is the weight of the l^{th} content feature; \hat{r}_{ij}^l is the predicted rating based on the l^{th} content feature. The challenge of the above fusion mechanism is how to compute the weights.

A naive solution is to treat each content feature equally, namely average fusion. Recall that Figure 5.3 shows a large performance divergence between different content features. The average fusion method neglects this divergence, which would lead to inferior performance. Another solution is to learn the weights using a learning-to-rank method [10]. However, learning-to-rank models are supervised learning and can only be on a user-video interaction matrix in our problem. Thus, the feature weights are learned in the in-matrix setting, which are not suitable for the out-of-matrix setting, as these two settings are intrinsically different. Moreover, training learning-to-rank models is time-consuming. Accordingly, we propose an efficient unsupervised method to decide the weights. We

first rank all content features based on their performances in the in-matrix or out-of-matrix settings on the validation dataset, then the weight of l^{th} content feature is computed as $\pi_l = p(1-p)^{l-1}$ where $p \in [0.5, 1)$ is a hyper parameter. Note that, for any rank position t (i.e., $\forall t > 0$), the inequality $\sum_{l=t+1}^L \pi_l \leq \pi_t$ holds in our method. This strategy ensures that the l^{th} content feature has higher weight than the total weight of the remaining less powerful content features. In other words, the proposed method allows the more effective features have much more impact in the final rating which is consistent with the observation in Figure 5.3.

To clearly illustrate the calculation of the weights, we present an example in Table 5.3 where four features are given and ranked. Note that, as WMF-based models (including our CER) with different content features achieve almost the same recommendation results in the in-matrix setting, as shown in the experiment section, we only apply our proposed fusion method to the out-of-matrix recommendation.

Feature	META	CNNFV	IDT	MFCC
l	1	2	3	4
π_l	0.5	0.25	0.125	0.0625

TABLE 5.3: An example of the weights generated in the late fusion method when p is set to 0.5.

5.5 Experiments

In this section, we first describe the setup of experiments and then demonstrate the experimental results.

5.5.1 Dataset Description

We used the MovieLens 10M [39] as the base dataset for our empirical studies. The MovieLens dataset does not itself contain videos or links for downloading. So we attempted to collect the videos from YouTube by ourselves. However, as most full-length videos are not available to download for free due to copyright restrictions, we downloaded the trailers according to the movie titles with the dataset. After a manual check to ensure the trailers matched the original full-length videos, a small fraction of the movies still not have trailers sourced from YouTube and we used other available clips

instead. By these means, we collected 10380 videos of the 10682 movies in the Movielens 10M dataset. The ratings associated with the missing 302 videos were removed, which slightly decreased the number of ratings from 10,000,054 to 9,988,676. The collected videos are resized to accelerate the content feature extraction: their widths were reduced to 240 pixels and their heights were adjusted proportionally.

The Movielens dataset also provides the movie IDs that correspond to IMDB¹. Based on these IDs, we crawled the movie plots, actors, directors, companies, languages and genres. Each movie’s title and plot were concatenated into a document. The top 20000 words were selected as the vocabulary according to global TF-IDF values, following[106, 110]. Then, a word vector for each movie was generated by word frequency. The other textual information including actors, directors, languages, companies, genres and other meta items formed another meta vector. To make the textual features of the videos have the same dimensions, the top 20000 meta items are selected as the codebook of meta vectors.

Similar to[106, 110], to be consistent with an implicit feedback setting, we transformed the ratings in the dataset into $\{0, 1\}$. Specifically, we mapped rating 5 to 1 and all the other ratings to 0. As a result, 1,543,593 positive ratings were generated, which only used 0.2% of all elements in the rating matrix. To make our experiment repeatable, both our collected dataset and the code is publicly available².

5.5.2 Experimental Settings

Comparison Methods

We compared our proposed CER model with the following six state-of-the-art recommender models.

Weighted Matrix Factorization (WMF)[54] only works in in-matrix setting, and achieves its best performance with $\lambda_u = 0.01, \lambda_v = 0.01$.

Collaborative Topic Regression (CTR)[106] learns the content latent vectors from word vectors using LDA. We trained CTR with both word and meta vectors. CTR achieves its best performance with $\lambda_u = 0.1, \lambda_v = 10$.

DeepMusic (DPM)[78] uses MLP to learn content latent vectors from MFCC. We extended DPM[78] to work with all the content features introduced in this section. DPM achieves its best

¹<http://www.imdb.com/>

²<https://github.com/domainxz/top-k-rec>

performance with $\lambda_u = 0.1$ and $\lambda_v = 10$.

Collaborative Deep Learning (CDL)[110] learns content latent vectors using SDAE from word vectors. Replacing the binary visible layer with Gaussian visible layer, SDAE can accept non-textual content vectors as input. We therefore extend CDL to work with both textual and non-textual features. CDL achieves its best performance with $\lambda_u = 0.1$, $\lambda_v = 10$ and $\lambda_n = 1000$.

Bayesian Personalized Ranking (BPR)[87] can be only applied to in-matrix recommendation setting, and its best performance is obtained with $\lambda_u = 0.0025$, $\lambda_i = 0.0025$, $\lambda_j = 0.00025$ and $\lambda_b = 0.0$.

Visual Bayesian Personalized Ranking (VBPR)[42] is an extension of BPR to combine visual contents with the CF. VBPR can work with all content features. Its optimal parameter settings are $\lambda_u = 0.0025$, $\lambda_p = 0.0025$, $\lambda_i = 0.0025$, $\lambda_j = 0.00025$, $\lambda_b = 0.0$ and $\lambda_e = 0.0$.

Note that CTR, DPM, CDL and VBPR can work in both in-matrix and out-of-matrix settings. The dimension of the latent vectors in all the methods is set to 50 for fair comparison. Our proposed CER achieves its best performance with $\lambda_u = 0.1$, $\lambda_v = 10$ and $\lambda_e = 1000$.

We also compare our proposed late fusion method with three state-of-the-art late fusion methods and two early fusion methods as follow:

Average fusion (AF) averages the predicted ratings from different content features.

Ranking SVM (SF) is a classic learning-to-rank model based on SVM[68].

Ranking BPR (BF) computes the feature weights in a learning-to-rank way by BPR[87].

EFC is the first early fusion method presented in Section 5.4.2 that concatenates all the feature vectors.

EFS is the second early fusion method presented in Section 5.4.2 that sums up all the content latent vectors to get a unified content latent vector.

Our proposed fusion method is denoted as **PF**. The ranking of the content features is obtained on the validation dataset. Given the ranking list, we find PF achieves its best performance with $p = 0.5$.

Data Split

Following the previous works [106, 110], we applied 5-fold cross validation to test the recommendation accuracy of each method in both in-matrix and out-of-matrix settings. Specifically, we divided the dataset into the training set, in-matrix test set and out-of-matrix test set with a split of 60%, 20%, 20% of the total positive ratings, respectively. To achieve this, all videos were first split into five folds

randomly and uniformly. Then, the corresponding ratings are also split into five folds. When one fold of videos was used to simulate new videos, its corresponding rating fold was chosen as the out-of-matrix test set, and the rest of the four rating folds were mixed together and re-split into four folds uniformly and randomly. Three of the re-split rating folds were used as the training set and the rest of the rating fold was used as the in-matrix test set. Note that we randomly chose 5% of the ratings from each test set as validation data to tune the model hyper-parameters.

In previous works [106], the in-matrix and out-of-matrix tests were conducted separately. The training data would change when the test scenario switches, which actually makes the two recommendation scenarios incomparable. Our proposed split protocol improves this situation so we can exactly compare the performance of each recommendation method in both in-matrix and out-of-matrix settings.

Evaluation Protocol

We adopt the evaluation methodology and measurement $\text{Accuracy}@k$ in [16, 122] to evaluate the top- k video recommendation accuracy. According to our data and the split protocol described in Section 5.2.2, each user will have roughly 8000 unrated videos in the in-matrix test and 2000 unrated videos in the out-of-matrix test. We computed the ratings based on the latent vectors or the content vectors, then generated a ranking list of the unrated videos for each user according to the predicted ratings. The top- k videos from the ranking list were returned as the personalized recommendation. For each user-video pair (i, j) in the test set D_{test} , if video j is in user i 's recommendation, we have a hit (i.e., the ground truth video is recommended to the user), otherwise we have a miss.

All the methods were evaluated by $\text{Accuracy}@k$ where a higher value means better performance. Its calculation proceeds as follows. We define $\text{Hit}@k$ for a single test case as either the value 1, if the ground truth video is in a user's top- k video recommendation, or the value 0 if otherwise, if otherwise. The overall $\text{Accuracy}@k$ is defined by averaging all the test cases:

$$\text{Accuracy}@k = \frac{\#\text{Hit}@k}{|D_{test}|} \quad (5.13)$$

where $\#\text{Hit}@k$ denotes the total number of hits in the test set, and $|D_{test}|$ is the number of all test cases. The experimental results were validated by means of a standard 5-fold cross validation. In previous works [106] [110], the value of k was selected from $\{50, 100, 150, 200, 250, 300\}$. However, such values of k were too large for a user to receive at once in a real world recommender system[36].

Therefore, k was selected from $\{5, 10, 15, 20, 25, 30\}$ in this section.

5.5.3 Experimental Results and Analysis

In this subsection, we evaluate the performance of our proposed CER in both in-matrix and out-of-matrix settings. We also study whether our proposed feature fusion method can improve the out-of-matrix recommendation. Recommendation efficiency is also studied.

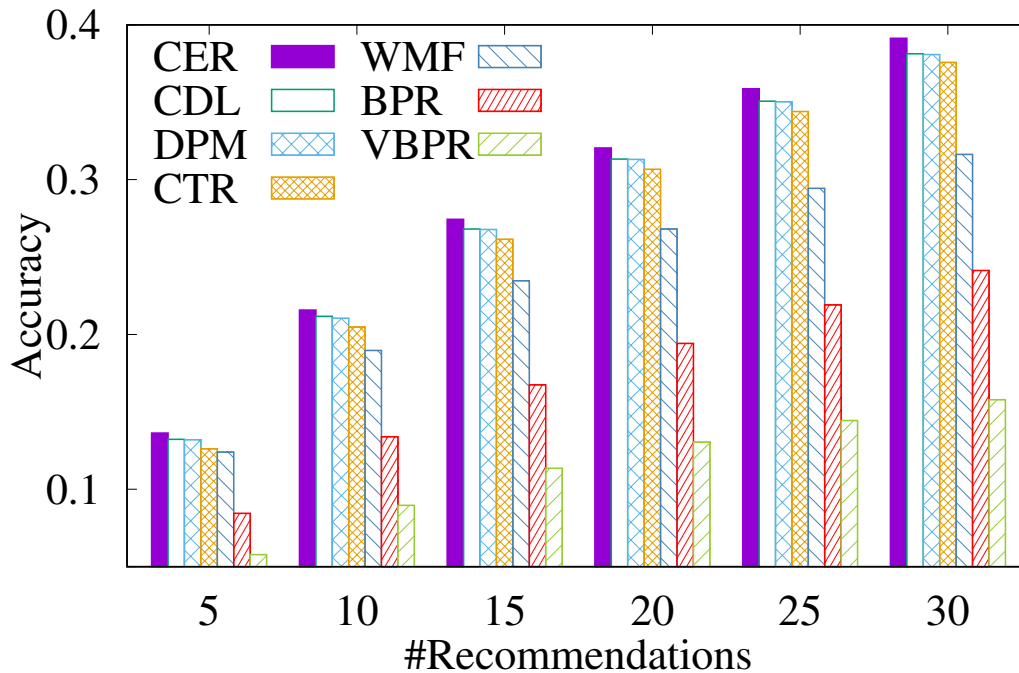


FIGURE 5.4: Accuracy@k of different methods under in-matrix setting

In-matrix Recommendation Effectiveness

In this experiment, we study recommendation effectiveness in the in-matrix setting and present the experimental results in Figure 5.4. For each recommender model, we notice that the performance difference incurred by using different content features can be ignorable. Therefore, we only present the one with the highest accuracy. Overall, our proposed CER achieved the highest recommendation accuracy, although its superiority is not visually obvious in Figure 5.4. Another observation is that the performance gap between the BPR-based models and WMF-based models are significant. This indicates the WMF-based models are more effective for top- k recommendation in in-matrix setting.

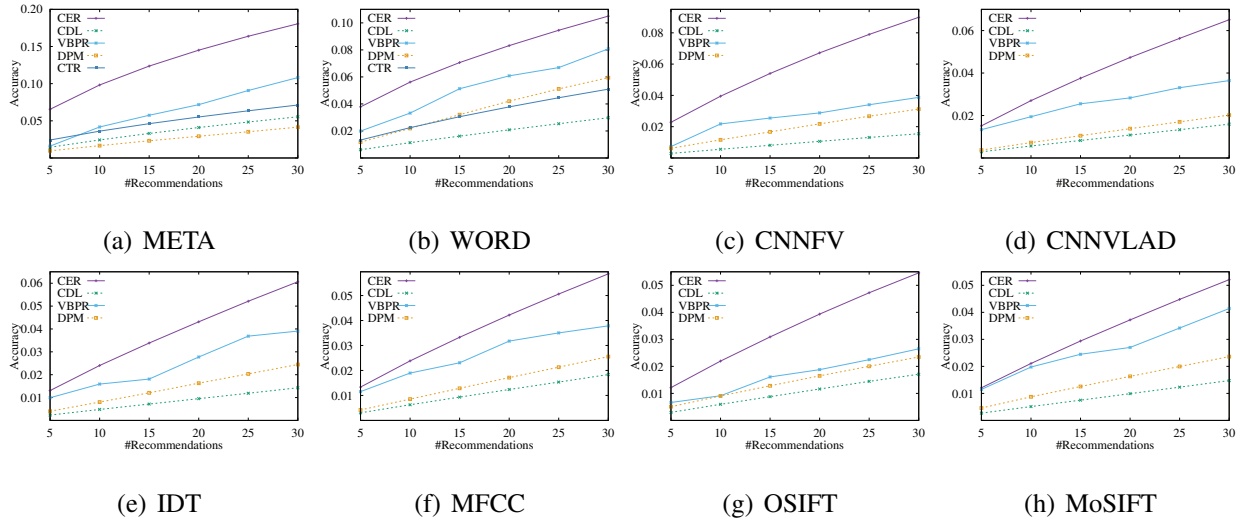


FIGURE 5.5: Accuracy@k of different methods and features in out-of-matrix setting.

Additionally, the differences between pure WMF and its variants (CTR, DPM, CDL, CER) are non-negligible. This indicates content information is beneficial for the in-matrix recommendation.

Out-of-matrix Recommendation Effectiveness

In this experiment, we study the performance of all recommendation methods in out-of-matrix setting. Since out-of-matrix recommendation accuracy is heavily dependent on the types of content features, we show the performance of all recommendation methods with different types of content features in Figure 5.5. The sub-figures are sorted in descending order according to the performance of our CER. From the results, we observe that our CER model significantly outperformed the other WMF-based models consistently with each feature. Moreover, our CER also achieves higher recommendation accuracy than VBPR which is the most effective baseline method in the out-of-matrix setting. This indicates that linear embedding is more suitable for generating latent content vectors in the video recommendation. However, the results in these figures also indicate that textual features (i.e. Figure 5.5(a) & 5.5(b)) are still the most powerful for out-of-matrix recommendation, while our introduced non-textual feature CNNFV (i.e. Figure 5.5(c)) achieves comparable performance. This finding suggests that, for user generated/uploaded videos without sufficient textual contents, the video recommender system is still able to produce accurate recommendations if the effective non-textual features are exploited and leveraged. Another observation is that recommendation accuracy in out-of-matrix setting is not as high as in in-matrix setting. This is because out-of-matrix recommendation is more challenging than in-matrix recommendation[106, 110, 42].

fusion on all non-textual features						
Method	k=5	k=10	k=15	k=20	k=25	k=30
AF	0.021243	0.037600	0.051699	0.064683	0.076871	0.088040
BF	0.021241	0.037602	0.051707	0.064690	0.076864	0.088044
SF	0.021498	0.037748	0.051894	0.064918	0.077049	0.088096
EFC	0.021830	0.038222	0.051606	0.065720	0.077664	0.089460
EFS	0.021439	0.037826	0.051994	0.064998	0.077119	0.088350
PF (p=0.5)	0.023090	0.040390	0.055746	0.069401	0.081788	0.093239
CNNFV	0.022809	0.039488	0.054017	0.067184	0.078983	0.089915
fusion on all features						
Method	k=5	k=10	k=15	k=20	k=25	k=30
AF	0.063132	0.093990	0.118933	0.140554	0.159595	0.176996
BF	0.061949	0.092322	0.116883	0.138128	0.156966	0.174059
SF	0.067023	0.100991	0.127185	0.149112	0.168508	0.186059
EFC	0.041733	0.063908	0.081737	0.097422	0.111366	0.124557
EFS	0.068546	0.101244	0.127280	0.149542	0.169221	0.187168
PF (p=0.5)	0.070157	0.104109	0.130914	0.153513	0.173169	0.190906
META	0.065530	0.098272	0.123640	0.145093	0.163806	0.180630

TABLE 5.4: Fusion results on different feature combinations

Test of Multiple Feature Fusion

In this experiment, we study whether fusing multiple types of features can further improve the out-of-matrix recommendations.

We report the recommendation accuracy of each fusion method with different feature combinations in Table 5.4. Since our CER achieved the best performance on all types of features, all the fusion methods were performed based on our CER. To clearly illustrate the improvement, we also present the highest recommendation accuracy achieved by our CER on a single feature in the last row.

As shown in Table 5.4, AF and BF fail to improve the recommendation accuracy with the combination of either the non-textual features or all the features. SF improves out-of-matrix accuracy with all the features, but it does not improve the accuracy with non-textual features. The only method that

improves recommendation accuracy with both feature combinations is our proposed PF. The failure of AF is due to the huge performance gap among different content features. In Figure 5.5, the highest out-of-matrix accuracy of CER is achieved with META vectors, while the lowest accuracy of CER is achieved with MoSIFT vectors. The highest accuracy is three times of the lowest accuracy. In this situation, averaging the ratings weakens the predictability of the most powerful feature. Both BF and SF are learning-to-rank methods and they learn the feature weights in a supervised way. In other words, the weights can only be learned based on user-video interaction matrix (i.e., in the in-matrix setting). The feature weights learned in the in-matrix setting, however, are not applicable to the out-of-matrix setting, as the importance of the same feature is different in these two different settings. In contrast, our proposed PF computes the weights in an unsupervised manner, thus the weights can still be computed even in the out-of-matrix setting.

The early fusion method EFS achieves the consistent performance in both feature combination settings. It performs better than the late fusion methods AF, BF and SF but worse than our proposed late fusion method PF. On the contrary, EFC achieves different performance in different feature combination settings. It achieves higher recommendation accuracy than AF, BF, SF and EFS when the fusion is applied on the non-textual features, but lower accuracy than all the methods when the fusion is performed on all the features. The results show that concatenating feature vectors then learning the shared latent vectors may be infeasible when the input features are heterogeneous. Summing up the latent vectors may overcome the heterogeneous problem, but it is not as good as the late fusion method which leverages the accuracy divergences.

Test in Text Sparsity Setting

In this experiment, we study whether the out-of-matrix recommendation accuracy can be improved by the non-textual features when few texts are available. Based on the fact that many user uploaded videos on Youtube having titles only, we simulate a text sparsity setting where the construction of the word vectors for the videos only uses titles. Figure 5.6 shows the performance of our CER with sparse text features and the fusion of all the non-textual features, respectively. From the figure, we observe that our CER achieves much higher out-matrix accuracy with the fusion of non-textual features than with the sparse text features. It shows that more accurate recommendation is achieved with non-textual features when there are few texts available, which indicates non-textual features are better for out-of-matrix recommendation in a text sparsity scenario.

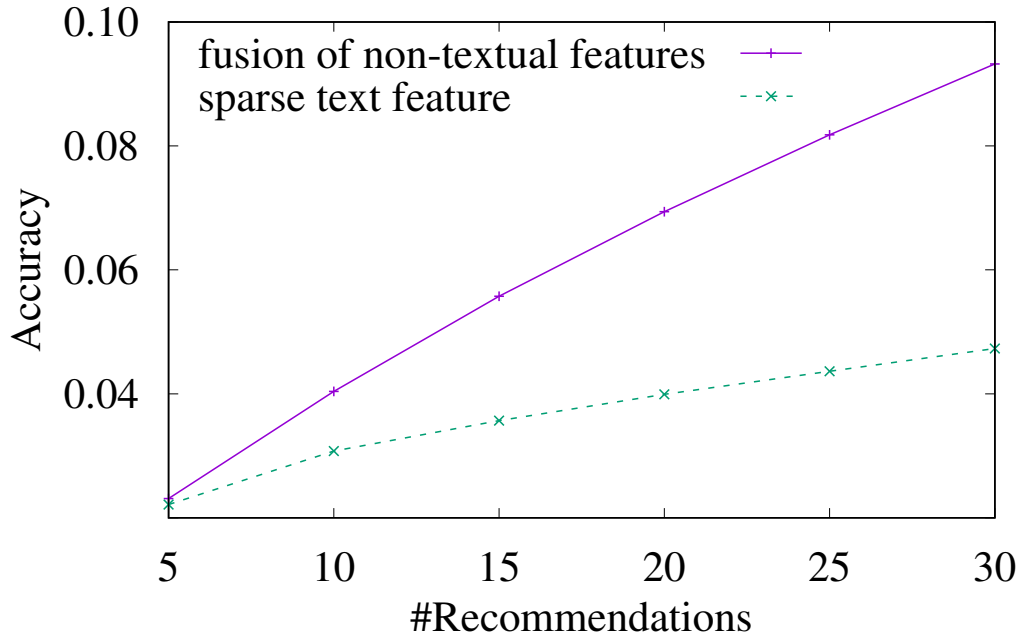


FIGURE 5.6: Out-of-matrix recommendation accuracy in the text sparsity setting.

Training Time Comparison

In this experiment, we compare the model-training efficiency of different WMF-based models. Table 5.5 reports the time cost of each iteration. Obviously, WMF costs the least time per iteration. This is because WMF is a pure CF method which does not involve the content vector generation process. CTR is the fastest of all the methods that use a content latent vector generation process. Our proposed CER is the second fastest. This is because CER needs to update embedding matrix, which requires more computation resource than the updates in LDA. However, CTR cannot support non-textual contents. Therefore, among the methods that can work with all types of features, CER is the most efficient. CDL is the slowest because it must pre-train SDAE before regression. Its time cost per iteration is slightly higher than DPM when regression begins.

Method	WMF	CTR	DPM	CDL	CER
Time cost	8.52s	11.57s	19.18s	41.77s	13.82s

TABLE 5.5: Training efficiency of WMF-based models.

5.6 Summary

In this section, we investigated how to leverage the rich textual and non-textual content information associated with videos to improve recommendation quality, especially in the out-of-matrix scenario. We first extracted and encoded multiple content features including word vectors, meta vectors, MFCC, SIFT, IDT and CNN. Then, we proposed a collaborative embedding regression model (CER) to incorporate these content features with collaborative filtering. We also studied how to fuse multiple content features to further improve video recommendation and proposed a novel late fusion strategy to fuse both non-textual and textual features. To evaluate the performance of our proposed recommender model CER and feature fusion method, extensive experiments were conducted on a large video dataset collected through multiple sources. The results show that our CER achieved the best performance in both in-matrix and out-of-matrix recommendation settings, and our proposed unsupervised feature fusion method significantly outperforms existing both early fusion and late fusion methods.

Chapter 6

Conclusion and future work

6.1 Conclusion

In this thesis, we first reviewed the relevant techniques for frame-level, video-level and user-level video filtering. Enlightened by that the non-textual contents are complementary to the texts, we study how to improve multi-level video filtering by using non-textual contents:

- In Chapter 3, we study frame-level filtering using detected visual objects. In previous works, the visual objects are obtained manually with a top-down process, which is very costly. In our study, we proposed to leverage the detected visual objects instead and designed a bottom-up method. The proposed method is superior to the existing methods in our experiments in terms of accuracy and efficiency.
- In Chapter 4, we study video-level filtering using small non-textual content set. In previous works, the small content set is made up of STIP and MoSIFT. The state-of-the-art motion feature IDT is not beneficial for the surveillance video filtering. In our study, we found that DT is complementary to IDT. Fusing them together overwhelmed the state-of-the-art content set on most events. Our findings helped us win the 2015 competition of TRECVID SED.
- In Chapter 5, we study user-level filtering using rich contents. In previous works, texts have been widely used. When the texts are scarce, the accuracy of filtering is low. In our study, we leveraged the non-textual contents to improve the text sparsity problem. Additionally, we studied how to fuse multiple contents to form more accurate filtering. The experiments show

that the proposed method is superior to the state-of-the-art methods and the fusion method is superior to widely used early and late fusion methods.

6.2 Future work

In the future, we plan to continue to explore our research work on video filtering along the following directions:

- We plan to extend the frame-level filtering method to more visual objects classes. In order to achieve that, we plan to use learning-based matching method instead of current hand-crafted method. The input features will be replaced in the future as well to adapt to more visual objects classes. In addition to that, we will gradually try to apply detection on some frames rather than all frames in the future. This is because the detection is very time-consuming in our proposed pipeline. In order to avoid missing unique visual objects, the frame sampling strategy should be carefully selected. We will explore different strategies and design a better one in the future work.
- We also plan to introduce the frame-level filtering method into video surveillance. This is because some events are pose-related which the motion contents could not correctly capture. In order to achieve that, we will apply pedestrian detection on the surveillance videos and label a small dataset of different pose-related events to train the detection model. After that, we will explore how to use the frame-level filtering method to improve the detection accuracy.
- For video-level filtering, we plan to explore another feature set in the future. In recent studies, CNN based feature extraction has attracted considerable attention even though its performance is still inferior to IDT in a pure motion-oriented dataset. However, as more powerful computing devices and algorithm appear, CNN based feature extraction has the potential to exceed IDT. We want to be the pioneer in this area. After that, we will try to design a real-time system instead of the current retrospective system, and try to port the system onto CCTV in the future.
- For user-level filtering, we plan to explore user privacy preservation in the future. Existing methods including our proposed method require to identify user in the system. It is not realistic in many cases. In the future work in user-level filtering, we will try to study how to perform it

without user identification. Related works have been done by using recurrent neural networks (RNN). We want to make it more flexible and accurate with rich contents.

References

- [1] S. Adali, K. S. Candan, S. Chen, K. Erol, and V. S. Subrahmanian. The advanced video information system: Data structures and query processing. *MMS*, 4(4):172–186, 1996.
- [2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDE*, 17(6):734–749, 2005.
- [3] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD*, pages 19–28, 2009.
- [4] R. Aly, R. Arandjelovic, K. Chatfield, M. Douze, B. Fernando, Z. Harchaoui, K. McGuiness, N. O’Connor, D. Oneata, et al. The axes submissions at trecvid 2013. In *TRECVID Workshop*, 2013.
- [5] R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, pages 2911–2918, 2012.
- [6] B. Babenko, M. Yang, and S. J. Belongie. Visual tracking with online multiple instance learning. In *CVPR*, pages 983–990, 2009.
- [7] H. Bay, T. Tuytelaars, and L. J. V. Gool. SURF: speeded up robust features. In *ECCV*, pages 404–417, 2006.
- [8] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Know.-Based Syst.*, 46:109–132, July 2013.
- [9] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. J. V. Gool. Robust tracking-by-detection using a detector confidence particle filter. In *ICCV*, pages 1515–1522, 2009.

- [10] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML*, pages 129–136, 2007.
- [11] C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM TIST*, 2(3):27, 2011.
- [12] M.-y. Chen and A. Hauptmann. Mosift: Recognizing human actions in surveillance videos. 2009.
- [13] M.-y. Chen, H. Li, and A. Hauptmann. Informedia@ trecvid 2009: Analyzing video motions. In *TRECVID Workshop*, 2009.
- [14] Q. Chen, Y. Cai, L. M. Brown, A. Datta, Q. Fan, R. S. Feris, S. Yan, A. G. Hauptmann, and S. Pankanti. Spatio-temporal fisher vector coding for surveillance event detection. In *MM*, pages 589–592, 2013.
- [15] M. Cheng, Z. Zhang, W. Lin, and P. H. S. Torr. BING: binarized normed gradients for objectness estimation at 300fps. In *CVPR*, pages 3286–3293, 2014.
- [16] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, pages 39–46, 2010.
- [17] B. Cui, A. K. Tung, C. Zhang, and Z. Zhao. Multiple feature fusion for social media applications. In *SIGMOD*, pages 435–446, 2010.
- [18] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005.
- [19] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, pages 428–441, 2006.
- [20] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. V. Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath. The youtube video recommendation system. In *RecSys*, pages 293–296, 2010.
- [21] Y. Deldjoo, M. Elahi, P. Cremonesi, F. Garzotto, P. Piazzolla, and M. Quadrana. Content-based video recommendation system based on stylistic visual features. *J. Data Semantics*, 5(2):99–113, 2016.

- [22] M. E. Dönderler, E. Saykol, U. Arslan, Ö. Ulusoy, and U. Güdükbay. Bilvideo: Design and implementation of a video database management system. *MTA*, 27(1):79–104, 2005.
- [23] M. E. Dönderler, Ö. Ulusoy, and U. Güdükbay. A rule-based video database system architecture. *Inf. Sci.*, 143(1-4):13–45, 2002.
- [24] M. E. Dönderler, Ö. Ulusoy, and U. Güdükbay. Rule-based spatiotemporal query processing for video databases. *VLDB J.*, 13(1):86–103, 2004.
- [25] X. Du, X. Li, X. Zhou, and A. Hauptmann. Ward-cmu @ trecvid 2015. In *Proceedings of TRECVID 2015*, 2015.
- [26] X. Du, Y. Yang, and X. Zhou. Ward@trecvid 2016. In *Proceedings of TRECVID 2016*, 2016.
- [27] X. Du, H. Yin, Z. Huang, Y. Yang, and X. Zhou. Using detected visual objects to index video database. In *The 27th Australasian Database Conference (ADC) 2016*, pages 333–345. Springer, 2016.
- [28] X. Du, H. Ying, L. Chen, Y. Wang, Y. Yang, and X. Zhou. Exploiting rich contents for personalized video recommendation. *arXiv preprint arXiv:1612.06935*, 2016.
- [29] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.
- [30] G. Farneback. Two-frame motion estimation based on polynomial expansion. In *SCIA*, pages 363–370, 2003.
- [31] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–1645, 2010.
- [32] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *CACM*, 24(6):381–395, 1981.
- [33] M. Flickner, H. S. Sawhney, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, 1995.

- [34] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.
- [35] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014.
- [36] C. A. Gomez-Uribe and N. Hunt. The netflix recommender system: Algorithms, business value, and innovation. *TMIS*, 6(4):13, 2015.
- [37] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *BMVC*, pages 47–56, 2006.
- [38] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *ICCV*, pages 263–270, 2011.
- [39] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *TiiS*, 5(4):19, 2016.
- [40] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.
- [41] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *PAMI*, 37(9):1904–1916, 2015.
- [42] R. He and J. McAuley. VBPR: visual bayesian personalized ranking from implicit feedback. In *AAAI*, pages 144–150, 2016.
- [43] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.
- [44] R. Hjelsvold and R. Midtstraum. Modelling and querying video data. In *VLDB*, pages 686–694, 1994.
- [45] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu. Personalized recommendation via cross-domain triadic factorization. In *WWW*, pages 595–606, 2013.
- [46] W. Hu, N. Xie, L. Li, X. Zeng, and S. J. Maybank. A survey on visual content-based video indexing and retrieval. *SMC C*, 41(6):797–819, 2011.

- [47] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.
- [48] Z. Huang, H. T. Shen, J. Shao, B. Cui, and X. Zhou. Practical online near-duplicate subsequence detection for continuous video streams. *TMM*, 12(5):386–398, 2010.
- [49] Z. Huang, H. T. Shen, J. Shao, X. Zhou, and B. Cui. Bounded coordinate system indexing for real-time video clip search. *TOIS*, 27(3), 2009.
- [50] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *PAMI*, 34(9):1704–1716, 2012.
- [51] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *PAMI*, 34(7):1409–1422, 2012.
- [52] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. Li. Large-scale video classification with convolutional neural networks. In *CVPR*, pages 1725–1732, 2014.
- [53] M. Köprülü, N. K. Cicekli, and A. Yazici. Spatio-temporal querying in video databases. *Inf. Sci.*, 160(1-4):131–152, 2004.
- [54] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.
- [55] J. Krapac, J. J. Verbeek, and F. Jurie. Modeling spatial layout with fisher vectors for image categorization. In *ICCV*, pages 1487–1494, 2011.
- [56] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS.*, pages 1106–1114, 2012.
- [57] T. C. T. Kuo and A. L. P. Chen. Content-based query processing for video databases. *TMM*, 2(1):1–13, 2000.
- [58] A. Kuznetsova, S. Ju Hwang, B. Rosenhahn, and L. Sigal. Expanding object detector’s horizon: Incremental learning framework for object detection in videos. In *CVPR*, pages 28–36, 2015.
- [59] Z. Lan, L. Bao, S. Yu, W. Liu, and A. G. Hauptmann. Double fusion for multimedia event detection. In *MMM*, pages 173–185, 2012.

- [60] Z.-Z. Lan, L. Jiang, S.-I. Yu, S. Rawat, Y. Cai, C. Gao, S. Xu, H. Shen, X. Li, Y. Wang, et al. Cmu-informedia at trecvid 2013 multimedia event detection. In *TRECVID 2013 Workshop*, page 5, 2013.
- [61] I. Laptev. On space-time interest points. *IJCV*, 64(2-3):107–123, 2005.
- [62] I. Laptev and T. Lindeberg. Space-time interest points. In *ICCV*, pages 432–439, 2003.
- [63] T. Le, M. Thonnat, A. Boucher, and F. Brémond. A query language combining object features and semantic events for surveillance video retrieval. In *MMM*, pages 307–317, 2008.
- [64] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: binary robust invariant scalable keypoints. In *ICCV*, pages 2548–2555, 2011.
- [65] F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, pages 524–531, 2005.
- [66] H. Li, Y. Li, and F. Porikli. Deeptrack: Learning discriminative feature representations online for robust visual tracking. *TIP*, 25(4):1834–1848, 2016.
- [67] L. Liu, L. Wang, and X. Liu. In defense of soft-assignment coding. In *ICCV*, pages 2486–2493, 2011.
- [68] T. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.
- [69] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.
- [70] M. Mathias, R. Benenson, M. Pedersoli, and L. V. Gool. Face detection without bells and whistles. In *ECCV*, pages 720–735, 2014.
- [71] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *RecSys*, pages 165–172, 2013.
- [72] O. Miksik and K. Mikolajczyk. Evaluation of local detectors and descriptors for fast feature matching. In *ICPR*, pages 2681–2684, 2012.

- [73] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP*, pages 331–340, 2009.
- [74] M. Müller. *Information retrieval for music and motion*. Springer, 2007.
- [75] A. Neubeck and L. J. V. Gool. Efficient non-maximum suppression. In *ICPR*, pages 850–855, 2006.
- [76] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *ICML*, pages 689–696, 2011.
- [77] E. Oomoto and K. Tanaka. OVID: design and implementation of a video-object database system. *TKDE*, 5(4):629–643, 1993.
- [78] A. V. D. Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *NIPS*, pages 2643–2651, 2013.
- [79] P. Over, G. Awad, M. Michel, J. Fiscus, W. Kraaij, A. F. Smeaton, G. Quenot, and R. Ordelman. Trecvid 2015 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proceedings of TRECVID 2015*. NIST, USA, 2015.
- [80] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *CoRR*, abs/1405.4506, 2014.
- [81] F. Perronnin and D. Larlus. Fisher vectors meet neural networks: A hybrid classification architecture. In *CVPR*, pages 3743–3752, 2015.
- [82] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, pages 143–156, 2010.
- [83] M. Petkovic and W. Jonker. *Content-Based Video Retrieval - A Database Perspective*, volume 25 of *Multimedia systems and applications*. Springer, 2003.
- [84] J. Pickens and G. Golovchinsky. Ranked feature fusion models for ad hoc retrieval. In *CIKM*, pages 893–900, 2008.

- [85] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.
- [86] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015.
- [87] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.
- [88] F. Ricci, L. Rokach, and B. Shapira, editors. *Recommender Systems Handbook*. Springer, 2015.
- [89] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *TPAMI*, 32(1):105–119, 2010.
- [90] S. Roy and S. C. Guntuku. Latent factor representations for cold-start video recommendation. In *RecSys*, pages 99–106, 2016.
- [91] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski. ORB: an efficient alternative to SIFT or SURF. In *ICCV*, pages 2564–2571, 2011.
- [92] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, pages 1–42, April 2015.
- [93] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.
- [94] J. Sánchez, F. Perronnin, T. Mensink, and J. J. Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 105(3):222–245, 2013.
- [95] H. T. Shen, J. Shao, Z. Huang, and X. Zhou. Effective and efficient query processing for video subsequence identification. *TKDE*, 21(3):321–334, 2009.
- [96] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

- [97] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *ICML*, pages 843–852, 2015.
- [98] N. Srivastava and R. Salakhutdinov. Learning representations for multimodal data with deep belief nets. In *ICML*, 2012.
- [99] N. Srivastava and R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *NIPS*, pages 2231–2239, 2012.
- [100] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *NIPS*, pages 2553–2561, 2013.
- [101] D. R. Turnbull, L. Barrington, G. Lanckriet, and M. Yazdani. Combining audio content and social context for semantic music discovery. In *SIGIR*, pages 387–394, 2009.
- [102] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.
- [103] Ö. Ulusoy, U. Güdükbay, M. E. Dönderler, E. Saykol, and C. Alper. Bilvideo video database management system. In *VLDB*, pages 1373–1376, 2004.
- [104] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *PAMI*, 32(9):1582–1596, 2010.
- [105] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, 2001.
- [106] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *SIGKDD*, pages 448–456, 2011.
- [107] H. Wang, A. Kläser, C. Schmid, and C. Liu. Action recognition by dense trajectories. In *CVPR*, pages 3169–3176, 2011.
- [108] H. Wang, A. Kläser, C. Schmid, and C. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103(1):60–79, 2013.
- [109] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, pages 3551–3558, 2013.

- [110] H. Wang, N. Wang, and D. Yeung. Collaborative deep learning for recommender systems. In *SIGKDD*, pages 1235–1244, 2015.
- [111] N. Wang, S. Li, A. Gupta, and D. Yeung. Transferring rich feature hierarchies for robust visual tracking. *CoRR*, abs/1501.04587, 2015.
- [112] N. Wang and D. Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, pages 809–817, 2013.
- [113] W. Wang, B. C. Ooi, X. Yang, D. Zhang, and Y. Zhuang. Effective multi-modal retrieval based on stacked auto-encoders. *Proc. VLDB Endow.*, 7(8):649–660, Apr. 2014.
- [114] Y. Wu, J. Lim, and M. Yang. Object tracking benchmark. *PAMI*, 37(9):1834–1848, 2015.
- [115] Z. Wu, G. Xu, Y. Zhang, Z. Cao, G. Li, and Z. Hu. GMQL: A graphical multimedia query language. *KBS*, 26:135–143, 2012.
- [116] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative CNN video representation for event detection. In *CVPR*, pages 1798–1807, 2015.
- [117] Z. Xu, Y. Yang, I. W. Tsang, N. Sebe, and A. G. Hauptmann. Feature weighting via optimal thresholding for video analysis. In *ICCV*, pages 3440–3447, 2013.
- [118] B. Yang, T. Mei, X. Hua, L. Yang, S. Yang, and M. Li. Online video recommendation based on multimodal fusion and relevance feedback. In *CIVR*, pages 73–80, 2007.
- [119] J. Yang, Y. Jiang, A. G. Hauptmann, and C. Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the 9th ACM SIGMM International Workshop on Multimedia Information Retrieval, MIR 2007, Augsburg, Bavaria, Germany, September 24-29, 2007*, pages 197–206, 2007.
- [120] Y. Yang, Z. Huang, H. T. Shen, and X. Zhou. Mining multi-tag association for image tagging. *WWWJ*, 14(2):133–156, 2011.
- [121] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), 2006.

-
- [122] H. Yin, B. Cui, J. Li, J. Yao, and C. Chen. Challenging the long tail recommendation. *PVLDB*, 5(9):896–907, 2012.
- [123] S.-I. Yu, L. Jiang, Z. Mao, X. Chang, X. Du, C. Gan, Z. Lan, Z. Xu, X. Li, Y. Cai, et al. Informedia@ trecvid 2014 med and mer. In *NIST TRECVID Video Retrieval Evaluation Workshop*, 2014.
- [124] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma. Collaborative knowledge base embedding for recommender systems. In *KDD*, pages 353–362, 2016.
- [125] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, pages 1529–1537, 2015.