Video Object Co-Segmentation and Video Vectorization

by

Chuan Wang

Department of Computer Science

The University of Hong Kong

Supervised by

Prof. Wenping Wang

Dr. Kenneth K.Y. Wong

A thesis submitted for
the Degree of Doctor of Philosophy
at The University of Hong Kong

July 6, 2015

Declaration of Authorship

I declare that this thesis represents my own work, except where due acknowledgement is made, and that it has not been previously included in a thesis, dissertation or report submitted to this University or to any other institution for a degree, diploma or other qualications.

Signed:

Chuan Wang
July 6, 2015

Abstract of thesis entitled

"Video Object Co-Segmentation and Video Vectorization"

Submitted by

Chuan Wang

for the degree of Doctor of Philosophy
at The University of Hong Kong
in July 6, 2015

Video, as a rapidly growing multimedia data, has been increasingly affecting our life in expression communication and interactions with the outer world. Compared to traditional image or text, it has better ability to convey information due to its time-sequential nature which, however, makes it a challenging task to extract information from it or process it. To keep up with its rapid extension, the research community has endeavored to develop accompanying tools that assist users analyzing and processing videos. This thesis demonstrates two systems for video analysis and process, *video object co-segmentation* and *video vectorization*.

In the system of *video object co-segmentation*, we address an issue of co-segmenting the common foreground object from a group of video sequences based on that multiple videos may share a common foreground object, such as a family member in home videos or a leading role in various clips of a movie or TV series. We propose a novel co-segmentation algorithm for video by taking full advantage of its appearance and motion features. Our algorithm is well-designed so that it can be differentiated from the algorithms for other forms of data, e.g. image or geometric shapes. We compare our method with the existing related works and our approach outperforms state-of-the-art methods.

In the system of *video vectorization*, we study a vectorization method for videos. Vector-based graphical contents are being increasingly used in smartphones and

computers, and becoming the main form of media on the Internet, demonstrated by the popularity of vectorized image editing tools such as Adobe Illustrator or CorelDraw. We realize that it would not work if simply applying existing image vectorization techniques to individual frames, because of the lack of consideration of temporal coherence between video frames that would cause unacceptable flickering. Consequently, we propose a method that treats the video as a spatial-temporal volume and uses 3D tetrahedral meshes for the vector-based representation. We present novel techniques for simplification and subdivision of a tetrahedral mesh to achieve high simplification ratio while preserving features and ensuring color fidelity. The proposed mesh simplification algorithm can be further applied to fast mesh generation for large-scale volumetric data with multiple labeled regions such as medical data. We also demonstrate the superiority of our approach by comparison with related works.

[**376 Words**]

Acknowledgements

It is my pleasure to acknowledge the people who made this thesis possible.

First I would like to thank my PhD supervisor Prof. Wenping Wang for his great guidance and continuous support on my research. His enthusiasm for research and encouragement on every progress of mine inspire me to work hard to achieve better results. This thesis would not be possible without his sound advice, encouragement and guidance during my PhD study.

Most of my work was collaborated with Dr. Yanwen Guo from Nanjing University. I wish to thank him for the excellent guidance. I would also like to thank Prof. Wei Chen and Prof. Nenghai Yu for their patient help and discussion during my visit to State Key Lab of CAD & CG in Zhejiang University and my stay in Microsoft Key Lab of Multimedia Computing and Communications in University of Science and Technology of China (USTC), when I was an undergraduate. My thanks also go to Prof. Falin Liu and Dr. Rong Zhang from USTC for their referrals during my PhD application.

I am very thankful to all my friends in Hong Kong. Without them, I could not enjoy the joyful life and learn so much during these years. My thanks go to "HKU CG Group" including Bin Chan, Yi-King Choi, Pengbo Bo, Lin Lu, Feng Sun, Yufei Li, Ruotian Ling, Yuanfeng Zhou, Li Cao, Zhan Yuan, Hao Pan, Wenni Zheng, Xinghua Zhu, Yanshu Zhu, Zhengzheng Kuang, Shuiqing He, Yongtao Hu, Wenchao Hu, Ruobing Wu, Guangmei Jing, Rui Wang, Yujing Sun, Weikai Chen, Xiaolong Zhang, Yating Yue, Lingjie Liu and Changjian Li, "HW335A fellas" including Li Ning, Jing Li, Hao Wang, Xiaoyang Li, Weida Zhang, Sirui Li, Xiangzhong Xiang, Yupeng Li, Guanbin Li, Xiaoguang Han, Wei Zhang, Yatong An, as well as my roommates Guodong Han and Li Zhang.

Finally I would like to express my sincere and deep appreciation to all my families. My thanks go to my paternal grandparents for their early education to me; go to my maternal grandmother who lives in my mind forever and my maternal grandfather for their everlasting love and company. My thanks also go to my wife Yaman Zhao for her long-time waiting for over seven years. Lastly, my special thanks go to my parents for their unconditional and never-ending love, support as well as encouragement during all my life. Without them, I would not have come so far in my education.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

ADM	**A**lternating **D**irection **M**ethod
ALM	**A**ugmented **L**agrange **M**ultiplier
AMF	**A**ppearance-**M**otion-**F**used
HD	**H**igh-**d**efinition
LRR	**L**ow **R**ank **R**epresentation
MRF	**M**arkov **R**andom **F**ield
QEM	**Q**uadric **E**rror **M**etric
QPBO	**Q**uadratic **P**seudo-**B**oolean **O**ptimization
SIFT	**S**cale-**i**nvariant **F**eature **T**ransform
TSP	**T**emporal **S**uper**p**ixel
TV	**T**ele**v**ision

Dedicated to my parents.

Chapter 1

Introduction

1.1 Motivation

We are entering an era of video. Pervasive mobile devices make video acquisition effortless with little cost. With social networks, people create online video albums for personal travels, parties or advertising. According to the statistics data, 100 hours of video are uploaded to YouTube, one of the largest online video website every minute. As a new visual media, video plays an increasingly important role in our communication, interaction and expression with the outer world. To keep up with its rapid expansion, the research community has endeavored to develop accompanying tools that assist user analyzing and processing videos. We follow this line of research and demonstrate two systems.

1.1.1 Video Object Co-Segmentation

Multiple videos may share a common foreground object, for instance a family member in home videos, or a leading role in various clips of a movie or TV series. In this thesis, we present a novel method for co-segmenting the common foreground object from a group of video sequences (Figure 1.1). The issue was seldom touched on in the literature.

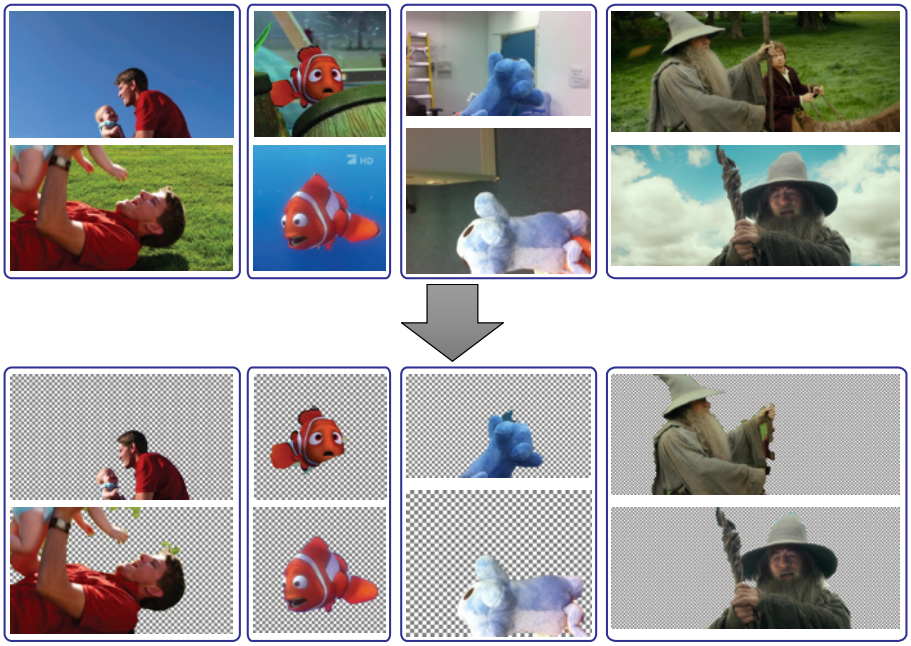


FIGURE 1.1: Common foreground object co-segmentation for a group of related videos without user interaction. Rows 1, 2 show four groups of videos, each group containing two related videos. Rows 3, 4 show the common foreground objects extracted by our method.

Starting from over-segmentation of each video into Temporal Superpixels (TSPs), we first propose a new subspace clustering algorithm which segments the videos into consistent spatiotemporal regions with multiple classes, such that the common foreground has consistent labels across different videos. The subspace clustering algorithm exploits the fact that across different videos the common foreground shares similar appearance features, while motions can be used to better differentiate regions within each video, making accurate extraction of object boundaries easier. We further formulate video object co-segmentation as a Markov Random Field (MRF) model which imposes the constraint of foreground model automatically computed or specified with little user effort. The Quadratic Pseudo-Boolean Optimization (QPBO) is used to generate the results. Experiments show that this video co-segmentation framework can achieve good quality foreground extraction results without user interaction for those videos with unrelated background, and with only moderate user interaction for those videos with

similar background. Comparisons with previous work also show the superiority of our approach.

1.1.2 Video Vectorization

Vector-based graphical contents are being increasingly used in smartphones and computers and are becoming the main form of media on the Internet. This trend is supported by many vector-based content generation tools, such as Adobe Illustrator and CorelDraw. Meanwhile, the Internet applications such as Adobe Flash have gained immense popularity in graphic design and animation. Besides compactness and editability, the scalability of a vector-based video allows it to be displayed on different devices with a wide range of resolutions, from cellular phones to tablet PC to HD TV. Therefore, converting legacy raster videos to vectorized videos is becoming an imperative issue. Driven by these demands, there has recently been a resurgence of interest in the research of image vectorization techniques [25, 29, 36, 46, 58] which aim to convert a raster image into a vector graphics that is compact, scalable, editable and easy-to-manipulate. But to the best of our knowledge, there is little work in the literature on video vectorization.

To this end, we present a video vectorization method that generates a video in vector representation from an input video in raster representation. A vector-based video representation offers the benefits of vector graphics, such as compactness and scalability. The vector video we generate is represented by a simplified tetrahedral control mesh over the spatial-temporal video volume, with color attributes defined at the mesh vertices. We present novel techniques for simplification and subdivision of a tetrahedral mesh to achieve high simplification ratio while preserving features and ensuring color fidelity. From an input raster video, our method is capable of generating a compact video in vector representation that allows a faithful reconstruction with low reconstruction errors. Figure 1.2 illustrates the vectorized results for one frame in a video. Our proposed mesh simplification algorithm can also be applied to fast mesh generation for large-scale volumetric data with multiple labeled regions but without color attributes such as medical data.

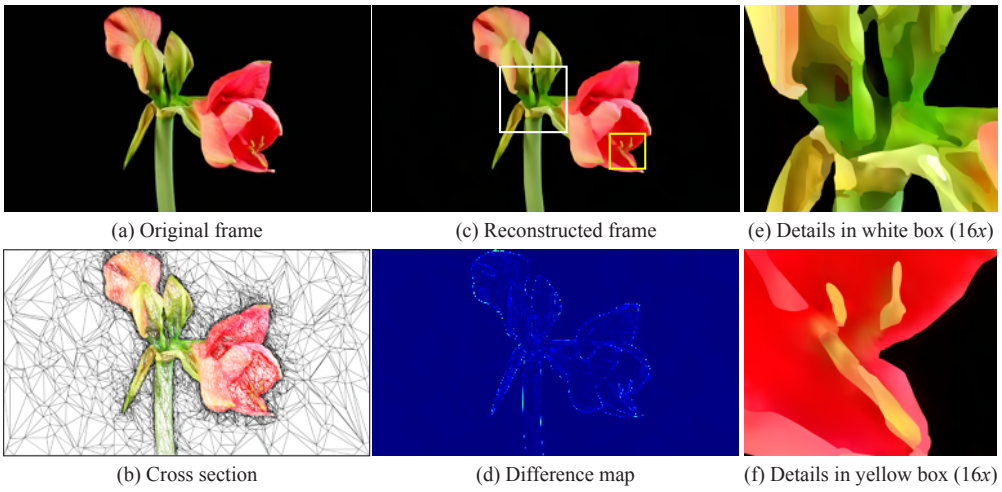


FIGURE 1.2: Vectorization of the video BLOOMING I, at frame 26. (a) The original frame. (b) The cross section of the simplified control meshes at the same position of frame 26. (c) The reconstructed frame from the vectorized video representation. (d) Difference map between the reconstructed frame and original frame. Figures (e) and (f) are the zoom-in views of the box regions in (c), magnified at 16× resolutions.

1.2 Outline

This thesis addresses two aforementioned problems of generic video analyzing and processing and it is organized as follows:

Chapter 2 proposes a novel method for co-segmenting common foreground objects in multiple videos, using subspace clustering and QPBO in an MRF framework. Our method can extract the common foreground with no user interactions for a group of videos with unrelated background, and with moderate user interactions for those with similar background. Our approach outperforms previous work in both clustering and foreground cutout results. (The presented material has been published in [53].)

Chapter 3 proposes an approach for generating vector-based representation given a raster video. Although image vectorization techniques have been developed in the past few years, it is still challenging for vectorizing a video because simply applying existing image vectorization techniques to individual frames would lead

to unacceptable flickering due to the lack of consideration of temporal coherence between video frames. In this chapter, we treat the input video as a spatial-temporal volume and use 3D sparse tetrahedral meshes to serve as the vectorized representation. The experiments show that our approach can produce vectorized video with smooth time coherence and low reconstruction errors. We also show experimental results of the aforementioned application of fast mesh generation in this chapter.

Chapter 4 summarizes and concludes the thesis.

Chapter 2

Video Object Co-Segmentation via Subspace Clustering and Quadratic Pseudo-Boolean Optimization

2.1 Introduction

Upon its release in 2012, the famous *Titanic 3D*, as its 2D version released in 1997, achieved critical and commercial success. Rolling Stone film critic Peter Travers rated the reissue 3.5 stars out of 4, and said *"The 3D intensifies Titanic."* and *"Caught up like never before in an intimate epic that earns its place in the movie time capsule"*. Behind this success the huge efforts are the cooperative endeavors of a technical team consisting of hundreds of artists and computer engineers. It was reported that the conversion from 2D to 3D took about 60 weeks and $18 million, most of which were spent on segmenting the video frames and extracting prominent foreground and background objects for assigning them

depth. Undoubtedly, segmentation of video data into semantic parts is a fundamental research topic for its wide applications, such as visual tracking, video retrieval, compression, and human-computer interaction.

Most existing efforts [3][17][27][38][37][60] concentrate on a single video as input. Segmentation of a single video is a challenging problem, and usually needs user assistance for supplying the sampling of foreground and background or correcting segmentation errors. Moreover, within an individual video accidental similarities in appearance or motion among foreground and background might be so deceptive that lead to ambiguous and even incorrect results easily. For the production of *Titanic 3D*, different clips in the movie generally share common foreground objects with similar appearance. Such a scenario holds for many other video data, for instance a family member in home videos, a hero/heroine in different shots of a TV series, and a famous player in sports videos. This induces us to explore the possibility of segmenting simultaneously the common foreground object from a group of videos. Exploiting the common or similar appearance across different videos may facilitate segmentation, since a group of related videos may provide more information which can be applied to better inference of the foreground. We refer to this problem as video co-segmentation, and co-segmentation of the common foreground objects from multiple related videos is the goal of this paper.

Video co-segmentation, as an emerging research problem, is receiving increasing attention recently. To the best of our knowledge, there are only a few methods [42][9][10] specifically designed to address this problem till now. These methods, however, either phrase it as a multi-class labeling problem [10] thus are not competent for the task of the common object co-segmentation we concern, or make strong assumptions about object motions which significantly limit the applicability. For instance, they usually overuse motions by assuming that several videos contain the common object with similar motions in addition to similar appearance [42], or these videos can be roughly grouped into foreground and background regions according to motion similarity within each video [9]. They generally ignore the fact that in multiple videos the common object rarely implies

consistent motions as the appearance. Therefore, video object co-segmentation is still a problem that needs to be intensively explored.

In this paper, we propose a novel framework for segmenting the common foreground objects in a group of videos consistently. This is accomplished by subspace clustering on Temporal Superpixels (TSPs) of the input videos and a subsequent Quadratic Pseudo-Boolean Optimization (QPBO) procedure using a Markov Random Field (MRF) model defined on videos. Our framework is applicable to the generic scenario of video object co-segmentation since we do not make further assumptions on the appearance and motions of video foreground and background as previous methods have done.

The common foreground object of different videos should have similar statistics of appearance features. To model foreground appearance, we develop an appearance-motion-fused subspace clustering algorithm to yield initial multi-label clustering results. Multiple appearance and motion features are fed into the subspace clustering algorithm. Although foreground objects across different videos may have quite different motion characteristics, motions can be used to better distinguish different parts within each video, making accurate foreground extraction within each video easier, especially around those ambiguous and low contrast object boundaries. We then for each video build a bag-of-words like descriptor. This implicitly links the common foreground objects with the same semantics across videos. We further formulate video object co-segmentation as an MRF model which imposes the constraint of appearance model of the foreground and is optimized by QPBO.

Contributions. To summarize, our contributions are:

- A novel framework that consistently segments the common foreground objects in a group of videos, and is applicable to generic videos.
- An appearance-motion-fused (amf) co-segmentation algorithm that leverages the appearance and motion features, based on subspace clustering.

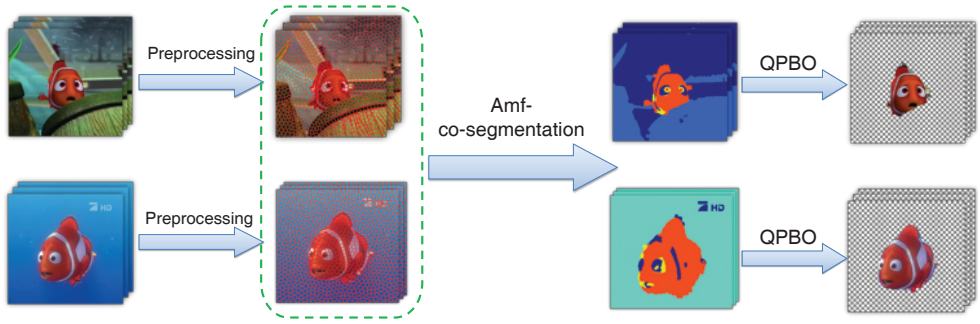


FIGURE 2.1: Workflow of our object co-segmentation framework. Columns from left to right: a video set as input, over-segmentation by TSP, clustering results by our amf-co-segmentation algorithm and object cutout results.

The remainder of the paper is organized as follows. The related work is introduced in Section 2.2. Section 2.3 gives a high-level overview of our framework. The key components of our framework, preprocessing, subspace clustering, and object co-segmentation by QPBO are presented in Sections 2.4, 2.5 and 2.6 respectively. We evaluate our method in Section 2.7 and conclude the paper finally.

2.2 Related Work

Our work is inspired by previous work on video segmentation and co-segmentation, image co-segmentation, as well as 3D shape co-segmentation.

2.2.1 Video segmentation and co-segmentation

Extraction of video object has received considerable attention over the past decade [50][11][2][28]. User interactions are more or less required to specify the sampling of foreground and background, and to remove errors caused by inseparable statistics of the foreground and background and temporal discontinuities. Early video cutout methods are generally based on global classifiers [28][54]. Recent efforts seek to extract accurate object boundaries by using local, directional, or combined classifiers [3][62], followed by foreground matting used to remove remaining errors.

Video co-segmentation has received increasing attention recently. Multi-class video co-segmentation is enabled in [10] by a generative multi-video model. This method realizes multi-class clustering where the number of classes is unknown beforehand, but cannot provide sufficiently accurate segmentation for a common foreground object of different videos. The problem of video object co-segmentation is addressed in [42][9]. Co-segmentation is posed as an optimization problem under a probabilistic framework in [42]. However, its applicability is limited by the dependency on objectness and saliency based initial estimation of foreground as well as the requirement that the foreground object undergoes similar motions across different videos. We compare our video co-segmentation approach with [10][42] through experimenting with a variety of video examples in the experiments. In [9], the intra-video motion cues and the inter-video appearance model together are taken into account for segmenting the common object in a pair of video sequences. The method may fail for the videos with similar foreground and background motions since it relies on motion-based video grouping for identifying candidate object within each video first. It can be seen that video object co-segmentation is still an emerging research problem to be intensively investigated.

2.2.2 Image and 3D shape co-segmentation

The problem of image co-segmentation was first studied by Rother et al. [41] and has gained considerable attention in the last few years [4][23][34][12][35][13]. The basic goal is to segment a common salient foreground object from two or more images. Consistency between the extracted object regions is ensured by imposing a global constraint which penalizes variations between the objects' respective histograms or appearance models. However, direct generalization of image co-segmentation methods to video co-segmentation is infeasible since it remains challenging to build a comprehensive foreground model for videos. Furthermore, the scale of video co-segmentation problem itself makes direct extension of image co-segmentation unpractical.

More recently, the concept of co-segmentation has been generalized to the 3D shape segmentation problem [24][16][44][21]. In [21], Hu et al. consider co-segmentation as a clustering problem, which is well solved by subspace clustering fed with multiple geometric features. This inspires us to explore the possibility of realizing video co-segmentation by subspace clustering.

2.3 Overview

Our video object co-segmentation framework takes a group of videos as input and produces their common foreground as output. It runs in the following steps (See Figure 2.1).

Preprocessing. To improve the efficiency of subsequent steps, we over-segment each video into Temporal Superpixels (TSPs). We also compute statistical appearance and motion features on each TSP as its descriptors as will be described in Section 2.4.

Amf-co-segmentation via subspace clustering. With the feature descriptors, our amf-co-segmentation algorithm integrates the appearance and motion information, and uses subspace clustering to cluster all TSPs by solving an objective function defined over a unified affinity matrix. This yields multi-label clustering results which in essence are similar to the output of multi-class video co-segmentation problem defined and pursued by [10]. However, we further rely on the output of this step to build a bag-of-words like histogram descriptor for each video, by which the common foreground is linked across different videos implicitly. Our amf-co-segmentation algorithm will be introduced in Section 2.5.

Object co-segmentation via QPBO. We formulate video object co-segmentation as an MRF model which imposes constraint of the common foreground model. Upon the MRF model, we build an undirected graph whose nodes are all TSPs

and edges are constructed by considering spatio-temporal consistency in both appearance and motion. We extract the common foreground by Quadratic Pseudo-Boolean Optimization in this MRF framework, and the details are described in Section 2.6.

2.4 Preprocessing

In the preprocessing stage, we over-segment each video into Temporal Superpixels (TSPs) [7] over which multiple statistical appearance and motion features are computed. This process also allows the subsequence steps to operate efficiently.

2.4.1 Over-segmentation with TSP

Even short video sequences contain a large number of pixels. The scale of this problem makes it computationally infeasible to process the data at pixel level. Normally superpixel or supervoxel, as an important preprocessing step, is applied to videos by various algorithms. However, as indicated in [7], off-the-shelf superpixel algorithms running independently on each frame will produce superpixels that are unrelated across time, and supervoxel is not specifically designed for video data.

In this paper, we use the generative probabilistic model [7] to over-segment each video into Temporal Superpixels (TSPs). Each TSP is a set of local video pixels in space and tracks the same part of an object across time. Note that, to accommodate large motions, SIFT flow [31] instead of optical flow originally applied by [7] is used to relate segments between two successive frames. This benefits us since SIFT flow establishes more robust feature correspondences than optical flow. Thus more consistent TSPs across frames and more accurate motion trajectories of them can be obtained.

2.4.2 Feature description

We extract for each TSP the appearance features and its motion trajectory.

Appearance features. We compute the raw features including 17-D texture by Winn filter bank [57] as well as HSV color at pixel level. Gaussian mixture model and discretization of feature space are then applied to the texture and color features, separately. We thus have two kinds of feature distributions, each of which describes the TSP with a histogram. An appearance feature vector is formed on each TSP by concatenating the two histograms. It is observed that TSPs from the common object of interest have similar feature distributions. As a result, these feature vectors are likely to be in common subspaces generated by standard basis corresponding to the nonzero bins. Figure 2.2 (b) shows an example.

Motion trajectory. Multiple spatio-temporal regions corresponding to different motions in a video can be separated by subspace clustering [52] because the 2D trajectories associated with a single rigid motion live in a 3D subspace under the affine camera model. In this paper, given a video sequence with F frames, the 2D trajectories of all TSPs are represented as a $2F \times N$ matrix

$$T = \begin{bmatrix} c_{11} & \cdots & c_{1N} \\ \vdots & \vdots & \vdots \\ c_{F1} & \cdots & c_{FN} \end{bmatrix} \tag{2.1}$$

where i-th column is the trajectory of its corresponding TSP, and c_{fi} is the center $[c_{fi_x}, c_{fi_y}]^{\mathsf{T}}$ of the intersection of TSP and frame-f. Noted that TSPs produced by [7] cannot guarantee each one passing through all frames due to newly appeared or vanished objects, or large motions not always tracked. For the remaining incomplete trajectories, we simply fill the empty entries by copying their nearest non-empty entry in the corresponding column. This kind of extrapolation is simple but works well in most cases proven by our experiments.

2.5 Appearance-motion-fused Co-segmentation via Subspace Clustering

In spite of possible variance due to changes in illumination, view angles, and non-rigid object motions, the foreground objects in different videos should have similar statistics of appearance features, which is a vital cue to relate them with each other. Besides, object motions within each video can facilitate differentiating foreground from background even though they may be inconsistent across different videos. To fully utilize the two kinds of features, we have developed an appearance-motion-fused (amf) co-segmentation algorithm whose core is subspace clustering. The TSPs of all videos are grouped into clusters with which a bag-of-words like histogram descriptor for the common video object can be obtained.

We first briefly introduce the background of subspace clustering. Then our appearance-motion-fused co-segmentation algorithm which is applicable to a group of videos is described in detail.

2.5.1 General subspace clustering

The problem of subspace clustering aims at clustering data vectors into multiple subspaces and finding a low-dimensional subspace fitting each cluster of vectors [51].

Notation. Given a feature data matrix $X = [x_1, x_2, ..., x_N]$ each column of which is a feature sample $x_i \in \mathbb{R}^D$ drawn from a union of P subspaces $\{\mathcal{S}_i\}_{p=1}^{P}$ of unknown dimensions, subspace clustering aims to segment the data vectors into their respective subspaces. Figure 2.2 (a) illustrates subspace clustering in $\mathbb{R}^3$.

Low-Rank Representation (LRR) solution. Since the data vectors of X are drawn from a union of P subspaces, each of them can be represented by the linear combination of X itself as the basis such that $X = XZ$ with $Z =$

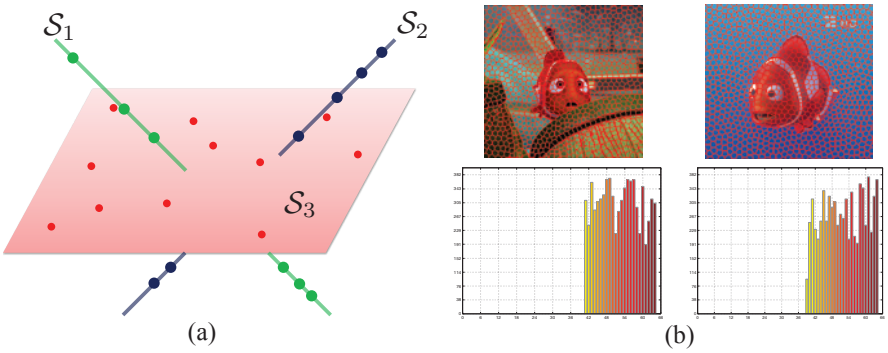


FIGURE 2.2: Subspace clustering. (a) An illustration of subspace clustering in $\mathbb{R}^3$. (b) The appearance feature, i.e. HSV histograms of two TSPs belonging to the orange clownfish. Due to the similar distributions, these two feature vectors are in a common subspace spanned by standard basis corresponding to the nonzero bins.

$[z_1, z_2, ..., z_N]$. The LRR algorithm [32] is based on the observation that Z should be low-rank because each data vector can always be represented by a sparse linear combination of the ones belonging to the same linear subspace. Consequently, the low-rank representation can be obtained by solving the following problem

$$\min_{Z,E} \|Z\|_* + \lambda \|E\|_{2,1}, \qquad \text{s.t.} \quad X = XZ + E \tag{2.2}$$

where $\|\cdot\|_*$ represents the nuclear norm (sum of the singular values) and $\|\cdot\|_{2,1}$ denotes the $\ell_{2,1}$ -norm [32] for characterizing noise E. $\lambda > 0$ is a parameter balancing the influences of the two parts.

According to [32], an affinity matrix that encodes the pairwise affinities among data vectors naturally derives from the solution Z^* of problem (2.2). Specifically, the affinity $(\mathbf{S})_{ij}$ of two data vectors x_i and x_j can be calculated by

$$(\mathbf{S})_{ij} = |(Z^*)_{ij}| + |(Z^*)_{ji}| \tag{2.3}$$

where $(\cdot)_{ij}$ denotes the (i, j)-th entry of the matrix. With such an affinity matrix, spectral clustering algorithm NCut [43] is applied to get the final clustering result.

2.5.2 Amf-co-segmentation to a video group

LRR algorithm as mentioned above, is originally designed to handle a single type of feature. In our formulation of the video co-segmentation problem, since appearance and motion features need to be taken into account simultaneously but treated unequally, it cannot be directly used here. For ease of exposition, we first formulate the video co-segmentation problem as follows.

Given L videos, each of which has been over-segmented into n_l TSPs, $l = 1, 2, ..., L$. Thus there are $N = \sum_{l=1}^{L} n_l$ TSPs in total. For each TSP, we compute K features including appearance and motion as mentioned in Section 2.4.2, so as to get K feature matrices $\{X_k\}(k = 1, 2, ..., K)$, where X_k is $D_k \times N$ and D_k is the dimension of k-th feature. Recall that we compute the 17-D texture, HSV color histogram and motion trajectories as the features of each TSP, then K is set to 3 in all our experiments. To distinguish X_k and Z_k belonging to appearance and motion features, we tag them with $\mathcal{A}$ or $\mathcal{M}$ so that X_k or Z_k belonging to appearance/motion feature can be written as X_k or $Z_k \in \mathcal{A}$ / $\mathcal{M}$. The goal of our amf-co-segmentation is to separate all TSPs into P clusters with all X_k as input, such that the common foreground has consistent labels of clusters across different videos.

To fully utilize appearance and motion features jointly within each video but treat them differently across videos, our amf-co-segmentation algorithm is proposed to consider multiple features with a penalty term imposed on them by solving the following optimization problem:

$$\min_{\substack{Z_1,...,Z_K \\ E_1,...,E_K}} \sum_{k=1}^{K} (\|Z_k\|_* + \lambda \|E_k\|_{2,1}) + \alpha \, \mathcal{P}_{\text{amf}}(Z_1, ..., Z_K) \tag{2.4}$$
$$\text{s.t.} \quad X_k = X_k Z_k + E_k, \; k = 1, \ldots, K$$

where $\alpha > 0$ is a parameter set to 1×10^{-5} in our experiments and $\mathcal{P}_{\text{amf}}$ is the consistent penalty term. The part of summation in Problem (2.4) is to apply LRR to each feature. If no other constraint is introduced, these features will actually work independently, causing each pair of data vectors to have various

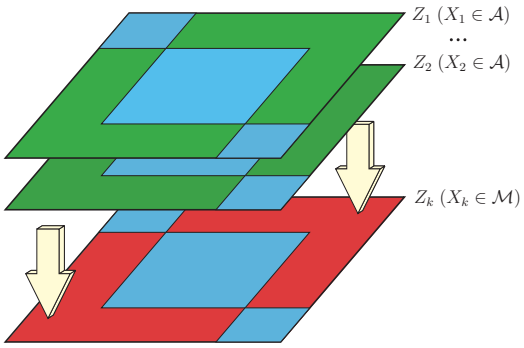


FIGURE 2.3: Construction of the matrix $\mathbf{Z}$. Blue represents the diagonal blocks corresponding to intra-video affinities $Z_k^{(l,l)}$. Green and red denote inter-video affinities $Z_k^{(l,m)}(l \neq m)$ corresponding to appearance and motion features. When constructing $\mathbf{Z}$, the inter-video affinities of matrices Z_k corresponding to appearance features replace those belonging to Z_k of motion features, which means that motion affinities take effect as appearance ones within each video only.

affinities corresponding to K features. The introduced $\mathcal{P}_{\text{amf}}$ aims to infer a unified affinity matrix by seeking the sparsity-consistent low-rank affinities Z_k over appearance and motion feature matrices jointly but freezing the effect of motion feature within each video simultaneously.

The structure of X_k and Z_k. Since every feature matrix X_k is constructed with concatenated features $X_k^{(l)}$ belonging to the l-th video, it can be represented as a block matrix as follows

$$X_k = [X_k^{(1)}, X_k^{(2)}, ..., X_k^{(L)}] \tag{2.5}$$

Then $X_k Z_k$ can be written as

$$X_k Z_k = \left[X_k^{(1)}, \cdots, X_k^{(L)}\right] \begin{bmatrix} Z_k^{(1,1)} & \cdots & Z_k^{(1,L)} \\ \vdots & \ddots & \vdots \\ Z_k^{(L,1)} & \cdots & Z_k^{(L,L)} \end{bmatrix} \tag{2.6}$$

where Z_k is also represented as a block matrix. The diagonal blocks in Z_k, i.e. $Z_k^{(l,l)}(l = 1, 2, ..., L)$ encode the affinities of data vectors within l-th video; and the

non-diagonal ones, i.e. $Z_k^{(l,m)}(l \neq m)$ encode the affinities of data vectors across the l-th and m-th videos. As aforementioned, across multiple videos appearance features should be similar but there is no guarantee on motion, therefore for $Z_k \in \mathcal{A}$, its diagonal and non-diagonal blocks should have equal status, which is however not true for $Z_k \in \mathcal{M}$. Specifically, in this case, since the motion similarity across videos is not reliable, all the affinity information in blocks $Z_k^{(l,m)}(l \neq m)$ cannot be treated equally as $Z_k^{(l,l)}$. To reserve the valid information only, we use a mask matrix which is of the same size as Z_k

$$(M)_{ij} = \begin{cases} 1 & \text{if } (Z_k)_{ij} \text{ locates at diagonal block,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

Let $\circ$ be the element-wise product, then $M \circ Z_k$ sets all non-diagonal blocks in Z_k to zero, meaning that motion affinities across different videos are filtered out.

The penalty $\mathcal{P}_{\text{amf}}$. Within each video, the affinity of a pair of data vectors should be consistent under all feature descriptors including motion. Across different videos, the affinity should be consistent only on appearance features. We therefore define the penalty term $\mathcal{P}_{\text{amf}}$ as follows,

$$\mathcal{P}_{\text{amf}} = \|\mathbf{Z}\|_{2,1} \tag{2.8}$$

where

$$\mathbf{Z} = \begin{bmatrix} (\mathcal{T}Z_1)_{11} & (\mathcal{T}Z_1)_{12} & \cdots & (\mathcal{T}Z_1)_{NN} \\ (\mathcal{T}Z_2)_{11} & (\mathcal{T}Z_2)_{12} & \cdots & (\mathcal{T}Z_2)_{NN} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathcal{T}Z_K)_{11} & (\mathcal{T}Z_K)_{12} & \cdots & (\mathcal{T}Z_K)_{NN} \end{bmatrix} \tag{2.9}$$

is a $K \times N^2$ matrix formed by concatenating $\mathcal{T}Z_k$. $\mathcal{T}Z_k$ is defined as

$$\mathcal{T}Z_k = \begin{cases} Z_k & \text{if } Z_k \in \mathcal{A} \\ M \circ Z_k + \frac{(\mathbf{J}-M)\circ\sum_{X_l \in \mathcal{A}} Z_l}{|\mathcal{A}|} & \text{otherwise} \end{cases} \tag{2.10}$$

where $\mathbf{J}$ is an all-one matrix and $|\cdot|$ is the cardinality of a set.

$\mathcal{T}$ operates on Z_k in the following manner. For the affinities $Z_k \in \mathcal{A}$, the output is actually Z_k itself, meaning that all the entries in appearance affinity matrix are equally valid. However for $Z_k \in \mathcal{M}$, since its inter-video affinities are not reliable, we first use the mask defined by Equation (2.7) to filter out the intra-video motion affinities, setting the inter-video affinities to zero. Then we further use the average inter-video appearance affinities to fill up the yielded zero entries. The filling step after filtering is necessary because the yielded zero affinity in $Z_k \in \mathcal{M}$ will strongly keep other $Z_k \in \mathcal{A}$ from taking effect across videos, due to its strong potential tend informing that the corresponding data vectors are not similar. With the operation $\mathcal{T}$, the intra-video parts of $Z_k \in \mathcal{M}$ remain unchanged while the inter-video parts (non-diagonal blocks) resemble their counterparts in $Z_k \in \mathcal{A}$.

The $\ell_{2,1}$ norm on $\mathbf{Z}$ is defined by

$$\|\mathbf{Z}\|_{2,1} = \sum_{j=1}^{N^2} \|\mathbf{Z}(*, j)\|_2 \tag{2.11}$$

where $\mathbf{Z}(*, j)$ is the j-th column of $\mathbf{Z}$ and $\|\cdot\|_2$ is the ℓ_2 norm. $\ell_{2,1}$ norm can induce column sparsity of $\mathbf{Z}$, meaning each pair of data vectors should have consistent affinities in terms of appearance and motion features. As a result, the sparsity-consistency of $\mathcal{T}Z_k(k = 1, 2, ..., K)$ is guaranteed. At the same time, the inter-video motion affinities are shielded, only leaving the intra-video ones assist to distinguish foreground from background.

Figure 2.3 illustrates the construction of $\mathbf{Z}$.

Optimization. Problem (2.4) is convex and can be solved with the augmented Lagrange multiplier (ALM) method [30]. First, it is converted into the following

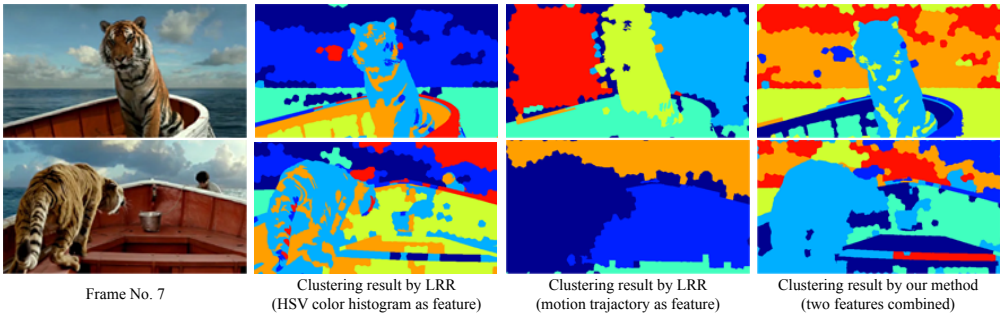


FIGURE 2.4: An example of the amf-co-segmentation results on a pair of video shots from the film *Life of Pi*. Columns from left to right: two frames from each video, clustering results by LRR with appearance feature only, results by LRR with motion feature only, and the results by our amf-co-segmentation algorithm. In each column, the same color represents the same class. Obviously, our amf-co-segmentation generates more consistent clustering results on the common foreground tiger.

equivalent problem

$$\min_{\substack{\{J_k\},\{S_k\}\\ \{Z_k\},\{E_k\}}} \quad \sum_{k=1}^{K}(\|J_k\|_* + \lambda\|E_k\|_{2,1}) + \alpha\|\mathbf{Z}\|_{2,1} \tag{2.12}$$

$$\text{s.t.} \quad X_k = X_k S_k + E_k, \quad Z_k = J_k, \quad Z_k = S_k, \quad k = 1, 2, ..., K \tag{2.13}$$

Then Problem (2.12) can be solved by the so-called alternating direction method (ADM) [30], listed in Algorithm 1. Note that the sub-problems of the algorithm are convex with closed-form solutions. Step 1 is solved via the singular value thresholding operator [6], while steps 3 and 4 are solved via Lemma 4.1 of [32].

Let $(Z_1^*, Z_2^*, \ldots, Z_K^*)$ represent the optimal solution to the objective function (2.4), the unified affinity matrix $\mathbf{S}$ is constructed with

$$(\mathbf{S})_{ij} = \frac{1}{2}\left(\sqrt{\sum_{k=1}^{K}(\mathcal{T}Z_k^*)_{ij}^2} + \sqrt{\sum_{k=1}^{K}(\mathcal{T}Z_k^*)_{ji}^2}\right) \tag{2.14}$$

NCut is applied to this affinity matrix to produce the co-segmentation result that groups all TSPs into clusters corresponding to the subspaces.

Figure 2.4 shows an simple example by our algorithm. Note that in column 2, multiple classes are produced on the tiger, when only HSV color histogram is used as feature. In column 3, when only motion trajectory is taken as the feature, it distinguishes different semantic regions in each video, but lacks consistent labels across the input videos. The tiger is labeled as yellow in the top row but as deep blue in the bottom. This shows that the common foreground do not necessarily presents consistent motions across different videos. In column 4, our amf-co-segmentation yields more consistent clustering results across the two videos. The tiger is labeled as blue in both videos. We further compare the final cutout results by motion-excluded and motion-included features in the experiment of unsupervised object co-segmentation with more examples, please refer to Section 2.7.2.1 for more details. Note that as mentioned in Section 2.4.2, an issue for applying motion trajectory as feature is, in practical experiments, the trajectories detected may be incomplete. For these cases, we just extrapolate the empty entries of the trajectory using the closest entries. Our experiments show that this method works well for most cases, yet we have to reduce this kind of incompletion. If large parts of trajectories are lost such as occlusion occurs, the clustering result will be much downgraded so that our method fails.

2.6 Object Co-segmentation

The amf-co-segmentation stage actually yields a bag-of-words like histogram description for each video. We further formulate video object co-segmentation as a binary labeling problem that aims to extract the common foreground simultaneously. It is achieved by defining a Markov Random Field (MRF) model on TSPs, while imposing the constraint of foreground model. Object co-segmentation is achieved by Quadratic Pseudo-Boolean Optimization (QPBO) under this MRF framework.

Algorithm 1 Solving Problem (2.12) by ADM

Require: Feature Matrices $\{X_k\}(k = 1, 2, ..., K)$, parameters λ and α

while not converged **do**

1. Fix the others and update $\{J_k\}$ $(k = 1, 2, ..., K)$ by

$$J_k = \arg\min_{J_k} \|J_k\|_* / \mu + \|J_k - (Z_k + W_k/\mu)\|_F^2$$

2. Fix the others and update $\{S_k\}$ $(k = 1, 2, ..., K)$ by

$$S_k = (\mathbf{I} + X_k^T X_k)^{-1}(X_k^T(X_k - E_k) + Z_k + (X_k^T Y_k + V_k - W_k)/\mu$$

3. Fix the others and update $\mathbf{Z}$ by

$$\mathbf{Z} = \arg\min_{\mathbf{Z}} \frac{\alpha}{\mu} \|\mathbf{Z}\|_{2,1} + \|\mathbf{Z} - \mathbf{Q}\|_F^2$$

where $\mathbf{Q}$ is a $K \times N^2$ matrix formed as follows:

$$\mathbf{Q} = \begin{bmatrix} (\mathcal{T}Q_1)_{11} & (\mathcal{T}Q_1)_{12} & \cdots & (\mathcal{T}Q_1)_{NN} \\ (\mathcal{T}Q_2)_{11} & (\mathcal{T}Q_2)_{12} & \cdots & (\mathcal{T}Q_2)_{NN} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathcal{T}Q_K)_{11} & (\mathcal{T}Q_K)_{12} & \cdots & (\mathcal{T}Q_K)_{NN} \end{bmatrix}$$

where $Q_k = (J_k + S_k - (W_k + V_k)/\mu)/2,\ k = 1, 2, ..., K$ and $\mathcal{T}$ is the same defined as Equation (2.10).

4. Fix the others and update $\{E_k\}$ $(k = 1, 2, ..., K)$ by

$$E_k = \arg\min_{E_k} \frac{\lambda}{\mu} \|E_k\|_{2,1} + \left\| E_k - (X_k - X_k S_k + \frac{Y_k}{\mu}) \right\|_F^2$$

5. Update the multipliers

$$\begin{aligned} Y_k &\leftarrow Y_k + \mu(X_k - X_k S_k - E_k) \\ W_k &\leftarrow W_k + \mu(Z_k - J_k) \\ V_k &\leftarrow V_k + \mu(Z_k - S_k) \end{aligned}$$

6. $\mu \leftarrow \min(1.1\mu, 10^{10})$.

7. Check the convergence conditions:

$$(X_k - X_k S_k - E_k) \to 0,\ (Z_k - J_k) \to 0,\ (Z_k - S_k) \to 0,\ k = 1, 2, ...K$$

end while

Ensure: Z

2.6.1 Estimation of video-level foreground histogram

All TSPs of the video group have been clustered into P classes corresponding to the P subspaces after the amf-co-segmentation stage. This results in a bag-of-words [45] like histogram description for each video. Let us define a histogram matrix H^l of size $P \times n_l$ for video-l, $l = 1, 2, ..., L$ such that

$$(H^l)_{bj} = \begin{cases} 1 & \text{if the } j\text{-th TSP is in the } b\text{-th class} \\ 0 & \text{otherwise.} \end{cases} \tag{2.15}$$

Its histogram description h^l is thus $h^l = H^l\mathbf{1}$ where $\mathbf{1}$ is a $n_l \times 1$ column vector of all ones. The foreground histogram h^l_f is $H^l y^l$ where y^l is a binary $n_l \times 1$ column vector for $(y^l)_i = 1$ if the i-th TSP is foreground and $(y^l)_i = 0$ otherwise.

The common foreground of all videos is supposed to be nearly identical by eliminating the scale difference. That is to say, if we stack all the latent foreground histograms into a matrix $\mathbf{H}_f = [h^1_f, h^2_f, ..., h^L_f]$, it should be closely rank-one. We further apply Rank One Decomposition to $\mathbf{H} = [h^1, h^2, ..., h^L]$ to get the approximation of $\mathbf{H}_f$ noted as $\hat{\mathbf{H}} = [\hat{h}^1, \hat{h}^2, ..., \hat{h}^L]$, where each column is an estimation of the foreground histogram of video-l.

2.6.2 QPBO in the MRF framework

For each video sequence l, an undirect graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is built whose nodes $\mathcal{V}$ are TSPs and edges $\mathcal{E}$ connect relevant TSPs. There are two cases that a pair of TSPs are linked by an edge. The first is that they fall into the same class by our amf-co-segmentation algorithm. The second is that they are spatio-temporally adjacent to each other.

The binary labelling problem is to assign a unique label $y_i \in \{0, 1\}, i = 1, 2, ..., n_l$ for each node such that the energy $E^l(y)$ can be minimized:

$$\min_{y^l} E^l(y^l) = \underbrace{\sum_{i \in \mathcal{V}} E^l_{\mathrm{d}}(y^l_i) + \sum_{(i,j) \in \mathcal{E}} E^l_{\mathrm{s}}(y^l_i, y^l_j)}_{\text{MRF Gibbs Energy}} + \gamma \left\| H^l y^l - \hat{h}^l \right\|_2^2 \tag{2.16}$$

where the first two terms on the left are data and smooth terms respectively, derived from the traditional MRF Gibbs energy. $\gamma > 0$ is a parameter penalizing the variation between the latent foreground histogram $H^l y^l$ and the given estimation $\hat{h}^l$. Obviously, the rightmost term aims to reinforce the similarity between the foreground histogram $H^l y^l$ induced by the optimal y^l and $\hat{h}^l$. With this term we show that with moderate or even no user intervention, the common foreground can be extracted simultaneously from a group of videos.

Equation (2.16) is called the Quadratic Pseudo-Boolean function, which is submodular and can be solved with roof duality by the QPBO algorithm. We refer readers to [40] for a detailed description of QPBO. Note that roof duality may produce unlabeled TSPs. In implementation as we found unlabeled TSPs usually belong to background, we just labelled them as background in all our experiments. This method is simple but works well in practice.

In our implementation, the smooth term is set as follows

$$E^l_{\mathrm{s}}(y^l_i, y^l_j) = \beta_{ij} |y^l_i - y^l_j| \tag{2.17}$$

where β_{ij} is set to the average color similarity of TSP i and TSP j. The data term is set to be a constant if no user interaction is involved. While for the sake of interactivity of our program, it will be set by the following manner if the user specifies strokes indicating the sampling of foreground and background on frames

$$E^l_{\mathrm{d}}(y^l_i) = \begin{cases} p(x^l_i|\mathcal{F}) / (p(x^l_i|\mathcal{F}) + p(x^l_i|\mathcal{B})) & \text{if } y^l_i = 0 \\ 1 - E^l_{\mathrm{d}}(0) & \text{otherwise.} \end{cases} \tag{2.18}$$

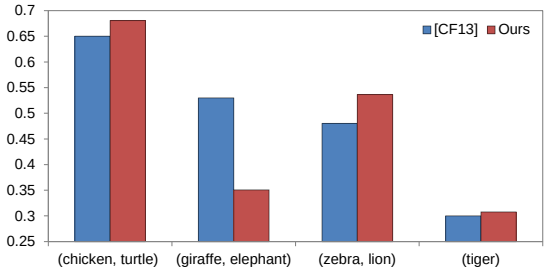


FIGURE 2.5: Comparison of co-segmentation accuracies between our method and [10] on MOViCS dataset.

	Preprocessing		Clustering	
	[10]#	Ours*	[10]	Ours
(chicken, turtle)	1h 40m	1h 10m	1h 12m	1h 04m 13s
(giraffe, elephant)	1h 33m	54m	1h 22m	13m 15s
(zebra, lion)	3h 19m	2h 14m	3h 13m	1h 23m 20s
(tiger)	1h 11m	48m	59m	30m 18s

TABLE 2.1: Runtime comparison between [10] and our method. #: including optical flow + superpixel + features. *: including sift flow + TSP + features. The data of [10] is got from its supplementary material.

where $\mathcal{F}$ and $\mathcal{B}$ represent foreground and background models learned from user-specified samples and $p(x_i^l|\cdot)$ denotes the likelihood of TSP i under the corresponding model, based on its color feature x_i^l.

2.7 Experiments

The video demos for this section can be found at the link `https://goo.gl/3S3w6K`

2.7.1 Evaluation on the clustering results

Sincewe formulate video co-segmentation as a clustering problem in Section 2.5, even though it is not the final goal of our purposed algorithm, we still evaluate the effectiveness of our approach by comparing our results against those by [10] at the very beginning.

In [10], a multi-class video co-segmentation dataset MOViCS which contains 11 videos belonging to 4 groups is released and an evaluation method that measures the average accuracy of the best matching clusters to each classes in ground truth is proposed. Specifically, the metric is defined as

$$\text{Score} = \frac{1}{C}\sum_{j}\max_{i}\frac{S_i \cap G_j}{S_i \cup G_j} \tag{2.19}$$

where S_i is a set of segments belonging to Class i, G_j is the set of segments of Class j in ground truth, and C is the number of classes in ground truth.

In our experiments, we use the same criterion to compare the clustering results. Besides, we compare the runtime under the same computer configuration Intel Core 2 Duo E8500 @ 3.16GHz with 8GB RAM as used in [10]. Figure 2.5 illustrates the performance scores and Table 2.1 shows the runtime. From Figure 2.5, we can see for 3 out of 4 video groups our method produces better, or at least comparable clustering results except the 2nd video group, which will be further discussed as a failure case in the later subsection. However, as our method treats the TSPs instead of superpixels in all frames as the basic units so that data size is much reduced, it runs much faster than [10], as shown in Table 2.1.

2.7.2 Unsupervised object co-segmentation

Our framework can produce object co-segmentation results for videos with unrelated backgrounds, in a fully unsupervised manner. Figures 2.6 and 2.7 show eight groups of results on a wide range of videos by our system, without any user guidance. Each of the video groups *BlackCar*, *FuzzyToy*, *Hobbits*, *Baby* and *Nemo* consists of two source videos, while each of the rest ones comprises three videos. The resolutions of input videos range from 344×327 to 1280×536 and numbers of frames range from 30 to 127. Most foreground objects have distinct motions from the background, nevertheless, foreground motions across different videos in the same group are not necessarily similar. Rows 7 and 8 in Figure 2.6 show such an example where the father and his baby present quite different

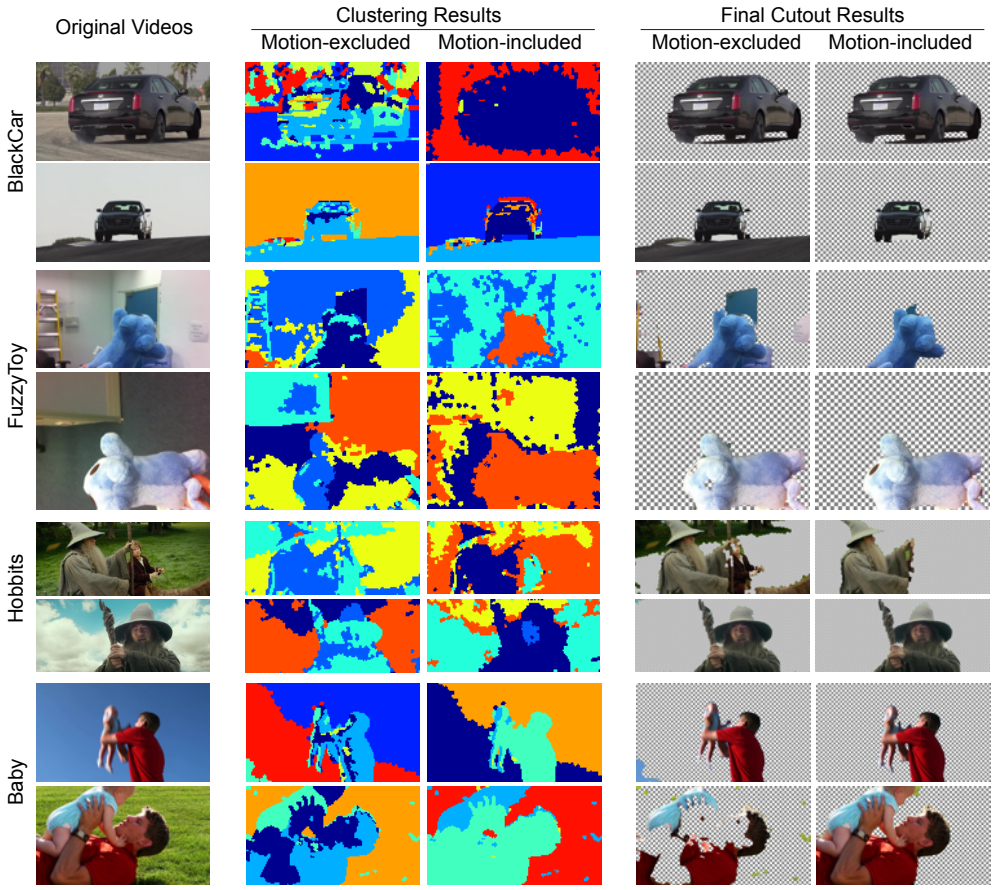


FIGURE 2.6: Unsupervised object co-segmentation results by motion-excluded vs. motion-included features on four video groups. From top to bottom: video groups *BlackCar*, *FuzzyToy*, *Hobbits* and *Baby*. In each group, from left to right: original videos, clustering as well as final cutout results with motion-excluded and motion-included features separately.

motions in the two source videos. All the results are shown in our accompanying video.

2.7.2.1 Motion-excluded vs. Motion-included

As a complement to appearance, motion information is fed into the subspace clustering algorithm for better differentiating foreground and background within each video. To validate this, Figure 2.6 shows four groups of results in each

	Motion-excluded		Motion-included	
	Precision	Recall	Precision	Recall
BlackCar	0.8423	0.8957	0.9634	0.9703
FuzzyToy	0.7989	0.8943	0.9821	0.9857
Hobbits	0.7960	0.8480	0.9729	0.9729
Baby	0.8916	0.8137	0.9748	0.9904

TABLE 2.2: Precision and recall of cutout results by motion-excluded and motion-included features.

group of which we compare the performance of subspace clustering and the final cutout results with and without motion feature used. Note that in the video group *BlackCar*, the black running car in the 2nd video has similar colors and textures to the ground due to the impact of backlight shot, causing it difficult to differentiate them with appearance features only. Accordingly, we can see from the 2nd column showing the clustering results, the ground and the car are clustered together. Furthermore, they are extracted simultaneously in the final cutout results as shown in the 4th column. By comparison, they are grouped into different clusters by our motion-included subspace clustering, and as a result, the car is successfully segmented from the video. Such a case also holds for the rest three video groups. Table 2.2 shows that incorporating motion into our framework improves both precision and recall on the four video groups over the motion-excluded implementation. The performance of both clustering and cutout is improved by ten to twenty percents, varying according to different video groups.

2.7.2.2 The method in [42] vs. ours

We further compare our algorithm with the video object co-segmentation method purposed in [42]. Unlike [42] which only accepts the videos depicting an object with similar motions in addition to similar appearance, and heavily relies on an initial estimation of the foreground and background labelling based on objectness and saliency detection, our method is more suitable for generic video data which

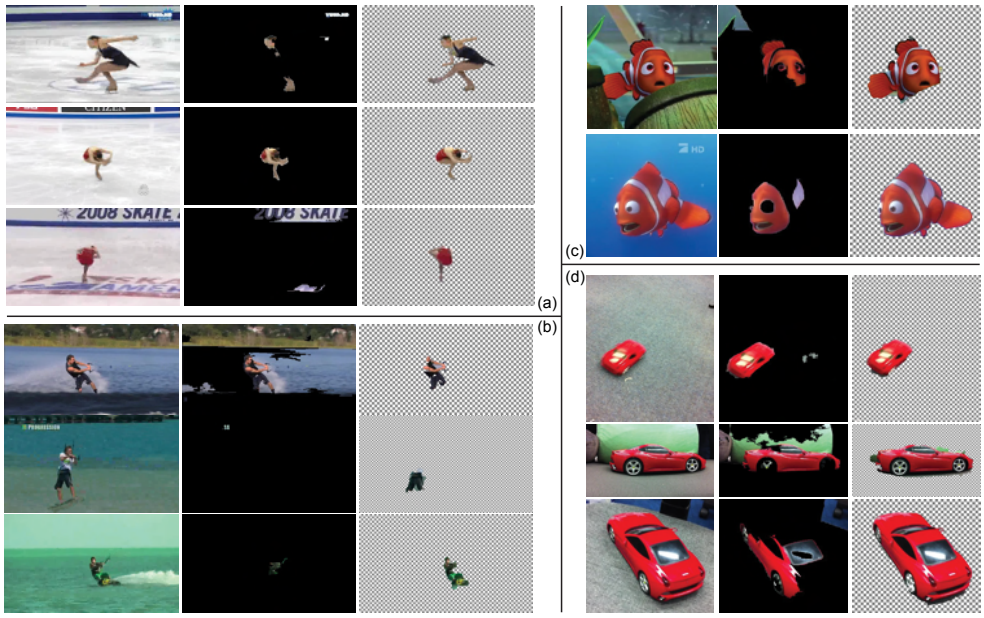


FIGURE 2.7: Unsupervised object co-segmentation results on four groups of videos. (a)∼(d): video groups *IceSkater*, *KiteSurfer*, *Nemo* and *Toycar*. In each group, each row corresponds to a video and columns from left to right show raw frames, results by [42] (black means background) and by our method. See the accompanying video.

	[42]		Ours	
	Precision	Recall	Precision	Recall
Nemo	0.2073	0.7113	0.9571	0.9808
IceSkater	0.3907	0.6308	0.9206	0.8784
KiteSurfer	0.0696	0.5822	0.7919	0.5364
Toycar	0.4785	0.5593	0.9111	0.9635

TABLE 2.3: Precision and recall of unsupervised object co-segmentation results by [42] and our method.

only requires similar visual appearance of the common foreground in the video group.

Figure 2.7 shows another four groups of videos, and each group is co-segmented by [42] and our method. Experimental results show that our method can produce cutout with much higher accuracy than [42]. Note that the videos *IceSkater* and

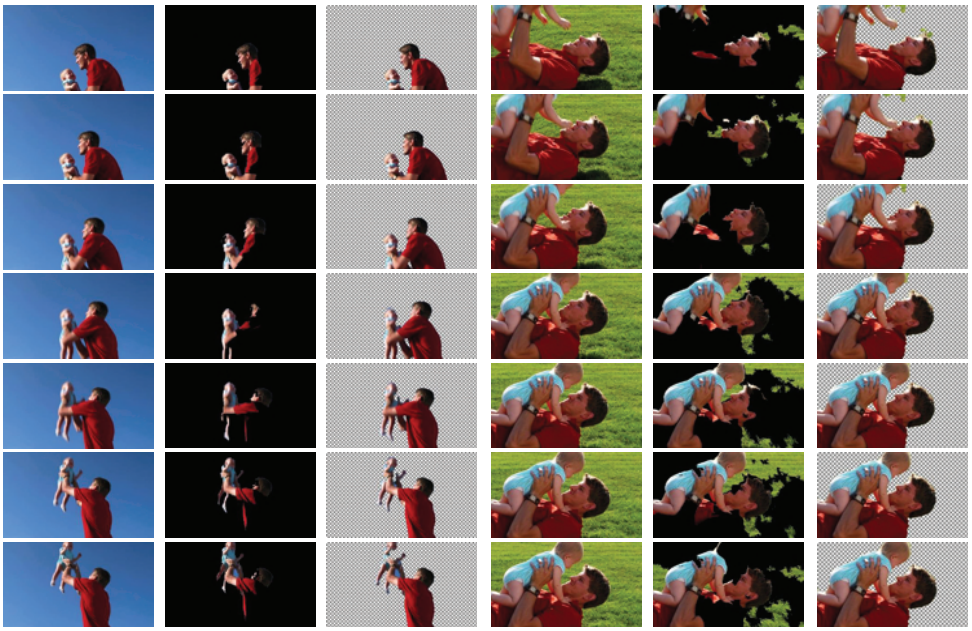


FIGURE 2.8: Cutout results of more frames in video group *Baby*. Columns 1, 4 are the original video frames, columns 2, 5 are the results by [42] and columns 3, 6 are the results by our method.

KiteSurfer are from [42] and both video groups present little appearance similarities on foreground, which are not fully suitable for our formulation of video co-segmentation problem. In [42], an initial common foreground was detected by saliency and objectness and was used to guide the following co-segmentation task. So here for fair comparison, we also provide an initial foreground estimation for our implementation using the same manner to avoid the ambiguity of foreground and background. Yet due to heterogeneous foreground involved, the third term in Equation (2.16) is violated to some extent. Even so, the experimental results clearly show that the algorithm of [42] presents large segmentation errors on most video examples. Table 2.3 compares the precision and recall on the video examples of [42] and our method, based on the ground truth we manually obtained using Adobe After Effects [1]. We also conduct the comparison using the video group *Baby*, and illustrate more frames in Figure 2.8. Please see our accompanying video for the live demo. A high resolution version can be browsed or downloaded from the link YouTube and GoogleDrive.

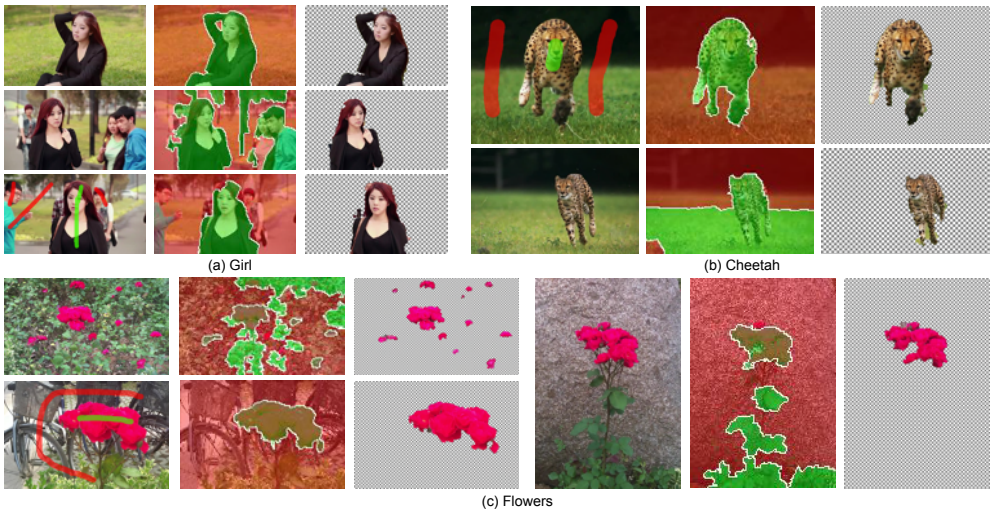


FIGURE 2.9: Object co-segmentation with moderate user guidance. (a)∼(c): video groups *Girl*, *Cheetah* and *Flowers*. In each group, from left to right: original video frame, results by image co-segmentation method [23] and by ours. For each video group, we just draw strokes (green: foreground, red: background) on no more than 3 frames of one video only. See also the accompanying video.

2.7.3 Object co-segmentation with moderate user guidance

Our framework can also produce object co-segmentation results for those videos with similar background, under which circumstance user guidance has to be involved to assist the segmentation. However, our system does not require the user to draw strokes on every video sequence in a group. For each video group, very limited number of strokes on a few frames in one input video are enough for our system to generate satisfactory results. This avoids much repetitive work, and significantly reduces users efforts.

Figure 2.9 shows some of the co-segmentation results with moderate user guidance. In the video group *Flowers*, the upper-left video contains scattered foreground objects with different sizes, some of which are small and hard to draw strokes on. Traditional video cutout tools like Rotobrush of Adobe After Effects often require users to specify strokes on all the foreground objects, and this work needs to be done on all videos separately. While with our video object co-segmentation method, users just need to draw strokes on one frame in

	Precision	Recall
Girl	0.9861	0.7926
Cheetah	0.9716	0.6358
Flowers	0.9537	0.9596

TABLE 2.4: Precision and recall of object co-segmentation results by our method, with moderate user guidance.

any video source, and then the rest videos will be co-segmented, guided by the prior knowledge on foreground thus obtained. For the group *Girl*, although background clutter and color ambiguity are apparent in the source videos, we only draw strokes on 3 frames of one video source even though the background motion is more complex. The cutout results are comparable to ground truth(see Table 2.4). In Figure 2.10, we also show cutout results of more frames of one of the examples, the video group *Girl*. Please see our accompanying video for the live demo. We also compare our video cutout results with the ones generated by applying one of the state-of-the-art image co-segmentation approaches [23] to all the video frames in each of the video groups. For fair comparison, the publicly available code[1] is slightly modified for accepting input foreground/background labels. We feed the implementation of this approach with ground-truth of the frames we draw strokes on as the guidance for segmentation. Furthermore, due to the high computational complexity of this approach, each video frame is down-sampled to a resolution so that width and height do not exceed 160 pixels. As shown in Figure 2.9, the foreground and background produced by the image co-segmentation approach are shown in green and red separately. Obviously, our video co-segmentation framework performs consistently better on the three video groups even though we give aforementioned advantages to image co-segmentation approach. Besides the potential lack of robustness, the reason might also be that the image co-segmentation working in this manner ignores the intrinsic nature of spatio-temporal consistency of the video, and pays little attention to the consistency of cutout results of neighboring video frames. In comparison, working on TSPs which is locally consistent over frames, our approach successfully generates

[1] `http://ai.stanford.edu/ ajoulin/index.php?page=coseg`

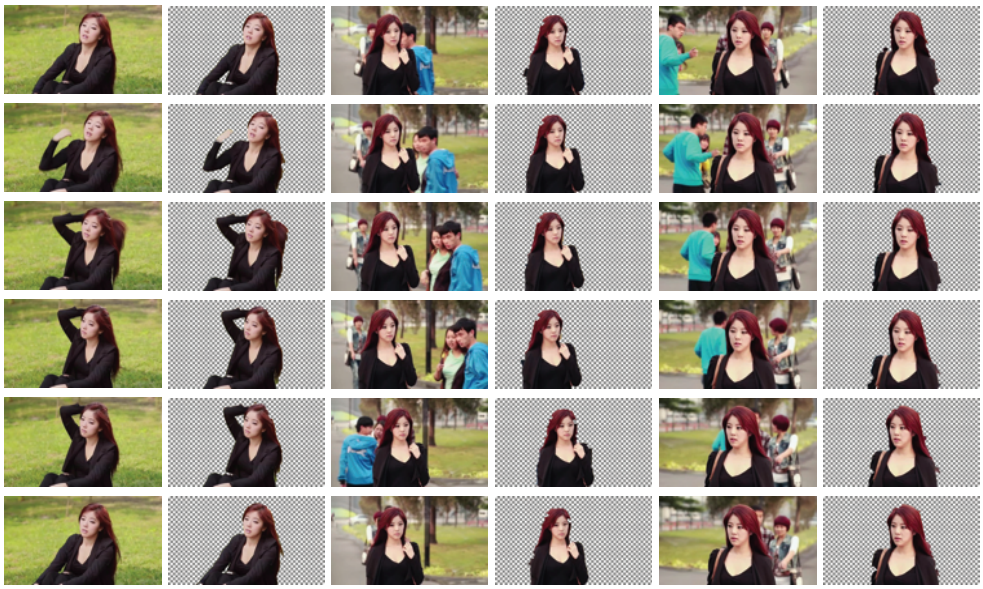


FIGURE 2.10: Cutout results of more frames in video group *Girl*. Columns 1, 3, 5 are original video frames, and Columns 2, 4, 6 are results by our method.

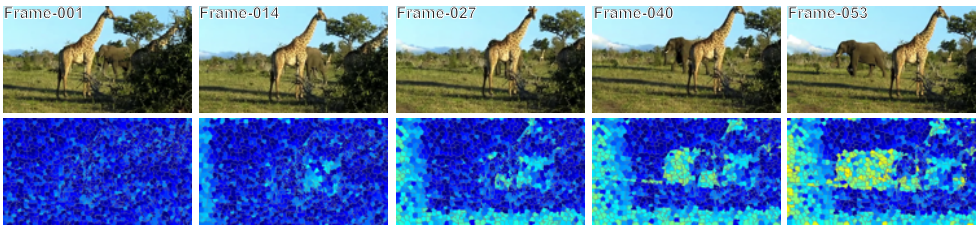


FIGURE 2.11: Label changes of the TSPs on the elephant over time, in the second video of the group. The second row visualizes the TSPs grouped into different clusters. When the elephant is occluded gradually by the giraffe, the TSPs on it vanish. New TSPs are assigned to it when it re-appears in the scene. This causes failure in obtaining the complete and consistent trajectories for the TSPs on the elephant, and leads to the poor performance of clustering.

high-quality results by leveraging the motion coherence within each video and foreground consistency across different videos.

2.7.4 Limitations

Our method takes TSP as the basic unit during the procedures of clustering and the successive object cutout. This highly reduces the amount of data to be

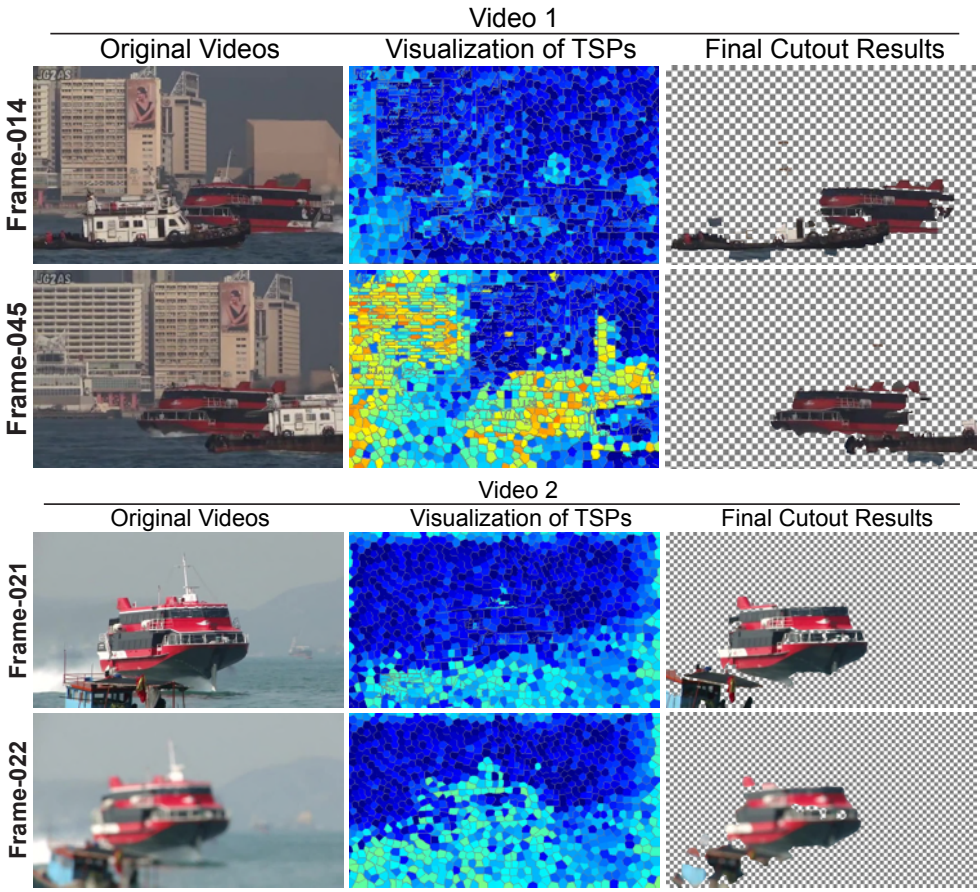


FIGURE 2.12: Object co-segmentation results on video group *Jetfoil* by our method. In Video 1, the ship occludes the jetfoil when it passes by; In Video 2, defocus blur occurs in Frame 22; Each situation causes the changing of TSPs on the jetfoil, as visualized in columns 2 and 4 respectively. Besides, color similarity commonly exists in parts of the jetfoil and the background. All of the factors result in bad cutout results as shown in columns 3 and 6.

processed so that the efficiency is improved. Motion information, characterized by the trajectories of TSPs, is incorporated into the subspace clustering for better differentiating foreground from background. However, if severe occlusion happens, it is often challenging to get a complete trajectory which is consistent over all video frames. This case probably occurs if the video clips are relatively long. This will degrade the performance of clustering. Figure 2.11 visualizes the changing of all TSPs over time for the second video in the video group *giraffe, elephant*. Besides the fact that the elephant, giraffe and meadow have similar

color distribution, TSPs on the elephant have inconsistent labels before and after the occlusion by the giraffe. Moreover, the giraffe is always static throughout the sequence, resulting in motion unable to distinguish it from the background. These factors lead to the poor performance of clustering.

We further apply our approach to a more challenging video group *Jetfoil* (Figure 2.12). In this video group, besides complicated background such as the ships and buildings which contain similar black color as the jetfoil, severe occlusion (the ship occludes the jetfoil in Video 1) and defocus blur (Frame 22 in Video 2) exist, so that the TSPs are more likely to be inconsistent. Under these conditions, the degraded motion trajectory loses its power to distinguish various object motions, causing some of the background and foreground regions are grouped and extracted together. Although in our current system the quality of results may be improved by adding more user inputs, more advanced motion segmentation techniques such as exploring the application of low-rank representation to the contaminated motion trajectories can be potentially coupled with the amf-co-segmentation to achieve more robust results in the future.

2.8 Conclusions and Future Work

We have presented a novel framework for video object co-segmentation. Our experiments show that with moderate or even no user guidance, common foreground can be segmented simultaneously from a group of videos. Our approach can make full use of the appearance and motion information embodied in the videos for co-segmentation. This is realized by a new appearance-motion-fused video co-segmentation algorithm via subspace clustering, which yields consistent labeling of the common foreground across different videos. Furthermore, we define video object co-segmentation as a binary labeling problem that can be solved by QPBO in an MRF framework. The framework imposes the constraint of the foreground model automatically computed or specified with little user effort.

Our framework realizes video object co-segmentation in a manner of global optimization on the videos, but it lacks a fine-tuning mechanism for locally refining

the results. Since the common foreground of different videos would complement each other, in the future we plan to take a co-refinement step that corrects the errors and refines the result in a video by exploiting the good result in another one. For example, the user first drags a rectangle around the area of segmentation errors in a video frame. The rectangle is then propagated to the successive frames, and the result is refined within these frames by local optimization, while imposing the constraint of good results of other videos in the same group.

Another solution is to adopt the local classifiers as [3][62] to handle the problem of inseparable statistics. Exploring the possibility of combining our framework with the local classifier-based video segmentation techniques is another interesting work. In addition, we plan to solve the matte of the foreground and remove remaining errors around object boundaries through an additive step of coherent matting which has been proven effective by previous video cutout systems [3][54][62].

Chapter 3

Video Vectorization via Tetrahedral Remeshing

3.1 Introduction

Vector-based graphical contents are being increasingly used in smartphones and computers and are becoming the main form of media on the Internet. This trend is supported by many vector-based content generation tools, such as Adobe Illustrator and CorelDraw. Meanwhile, the Internet applications such as Adobe Flash have gained immense popularity in graphic design and animation. Besides compactness and editability, the scalability of a vector-based video allows it to be displayed on different devices with a wide range of resolutions, from cellular phones to tablet PC to HD TV. Therefore, converting legacy raster videos to vectorized videos is becoming an imperative issue. Driven by these demands, there has recently been a resurgence of interest in the research of image vectorization techniques [25, 29, 36, 46, 58] which aim to convert a raster image into a vector graphics that is compact, scalable, editable and easy-to-manipulate. But to the best of our knowledge, there is little work in the literature on video vectorization.

Generating a vectorized video from a raster video is a challenging task. Simply applying existing image vectorization techniques to individual frames would not

work because the lack of consideration of temporal coherence between video frames would lead to unacceptable flickering. It is thus natural to treat the input video as a spatial-temporal volume and consider what is the most suitable geometric representation for the purpose of video vectorization. Inspired by existing image vectorization techniques which are mostly based on a 2D mesh representation, as the gradient quadrilateral mesh [25, 46] and triangular patches with curved or straight boundaries [29, 58], we propose to use 3D tetrahedral meshes for video vectorization. Note that, however, this extension from 2D meshes to 3D meshes for video vectorization is far from straightforward and a number of technical challenges need to be addressed to develop an effective video vectorization method.

3.1.1 Our approach

We present an effective method for generating a resolution-independent vector-based video from an input raster video. Our vector representation is based on a sparse tetrahedral mesh with colors defined at its vertices. The core of our method consists of new techniques for simplification and subdivision of tetrahedral meshes defined over the spatial-temporal volume of the input video.

A basic requirement for vectorized video representation is feature preservation. To meet the requirement, we first over-segment the input video into a set of *temporal super-pixels* (TSPs) whose boundaries conform with important visual feature and region boundaries, thereby preserving the features. An initial regular tetrahedral mesh is built over the piecewise defined, possibly discontinuous color field defined on the TSPs. Then we simplify the initial mesh to obtain a feature-preserving sparse tetrahedral mesh while preserving features, which constitutes the vector representation of the video. Our mesh simplification algorithm uses iterative edge contractions based on an extended quadric error metrics (QEM) as approximation error. Finally, for video displaying we propose a tetrahedral mesh subdivision method for reconstructing details from the sparse control mesh to produce a raster video. Figure 3.14 shows an example of a vectorized video

by our method. We validate the effectiveness of our approach with extensive experiments.

In summary, we present an effective video vectorization algorithm based on simplification and subdivision of tetrahedral meshes. This is the first vectorization method for generic videos. Our method achieves very high simplification ratio (typically 1%) and well preserves visual features and color fidelity.

3.2 Related Work

3.2.1 Image vectorization

There has been much work on vectorization of non-photographic images, such as fonts, cartoon [64] and line drawings [19, 22]. Recently, vectorization of photographic images has aroused a lot of interest [25, 26, 29, 36, 36, 39, 46, 58]. Some methods produce explicitly a parametric surface mesh associated with color as the vector-based image representation. The geometric primitives include the gradient mesh consisting of topologically planar Ferguson patches with meshlines [25, 46], triangular patches with curved boundaries [58], and the traditional triangular meshes with straight edges [29]. Other methods adopt a mesh-free representation. For instance, *diffusion curves* [36] creates curves with color and blur attributes, and models the color variation as a diffusion from these curves by solving a Poisson equation. [59] automatically generates sparse diffusion curve vectorizations of raster images by fitting curves in the Laplacian domain.

We note that it is infeasible to simply extend the above image vectorization techniques to each video frame independently because spatial coherence of the video cannot be guaranteed in this way. In contrast, our method aims to produce a *three-dimensional*, vector-based video representation by working directly on the spatial-temporal video volume.

3.2.2 Cartoon vectorization

Some methods work on cartoon vectorization [48, 49]. Zhang et al. [61] proposed a layered representation of cartoon animations in which the pre-segmented regions and decorative lines are vectorized separately. Our approach differs from this method in several aspects. First, our approach is suitable for generic video vectorization and can be applied to cartoon animation as well, while this method is only applicable to vectorization of cartoon animations by taking advantage of the particular nature of cartoons, such as easy image decomposition into background and foreground, and distinctive decorative lines, which are not possessed by generic videos. Therefore, this method is not general enough to be applicable to full-color videos. Second, our method yields a unified vector-based video representation, while the output of this method is the vectorization of segmented regions, decorative lines, together with their motions across frames. Third, our method works automatically, while user-assistance is required in [61] to complete the static background by this method.

3.2.3 Vectorization for other applications

It is noted that vector representation has also been used to model solid textures [56] and volumetric objects [55], and to achieve resolution independent texture mapping [47].

3.3 Overview

Our video vectorization framework takes a raster video as input and produces a tetrahedral mesh as the final vector representation. Specifically, the pipeline consists of the following steps: for an input raster video, we first segment the video into multiple temporal superpixels. Then we create an initial dense tetrahedral mesh by regular mesh generation and cleaving. Further, simplification and color optimization are performed to generate the final simplified tetrahedral control mesh, that is our proposed vector-based representation of the video. Finally,

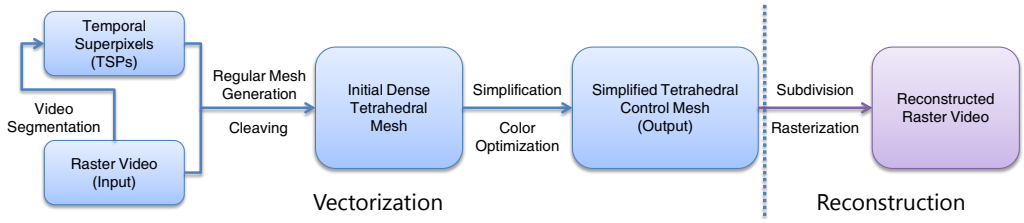


FIGURE 3.1: Pipeline of our video vectorization (left of dashed line) and reconstruction (right of dashed line) framework.

subdivision and rasterization are used to reconstruct raster video for display. (See Figure 3.1).

3.3.1 Feature detection via video segmentation

To extract the video features in 3D video volume, we first perform video segmentation to partition a video clip into *temporal superpixels* (TSPs). This step can be achieved using the methods in [17]. All the TSP are assumed to have smooth internal color changes. Also, each pixel is tagged with the ID of the TSP containing it.

3.3.2 Regular tetrahedral mesh generation and cleaving

To generate an initial tetrahedral mesh, we perform the following three steps. First, we view the video as a 3D volume and create a regular 3D lattice with all the vertices located at video pixels. Then for each neighboring $2 \times 2 \times 2$ vertices forming a cell, we split it into six lattice tetrahedra with equal volume as Figure 3.2(g) shows. Further, we apply a multi-material tetrahedral meshing algorithm Cleaver [5] to the lattice tetrahedral mesh. Here each TSP is of the same material. The Cleaver algorithm splits each tetrahedra whose four vertices have different TSP IDs into several smaller tetrahedra. Specifically, there are five unique interface patterns that Cleaver handles, corresponding to different patterns of TSP IDs tagged at the vertices (Figure 3.2 (a)$\sim$(e)). For each pattern, Cleaver remeshes the tetrahedron into smaller tetrahedra considering

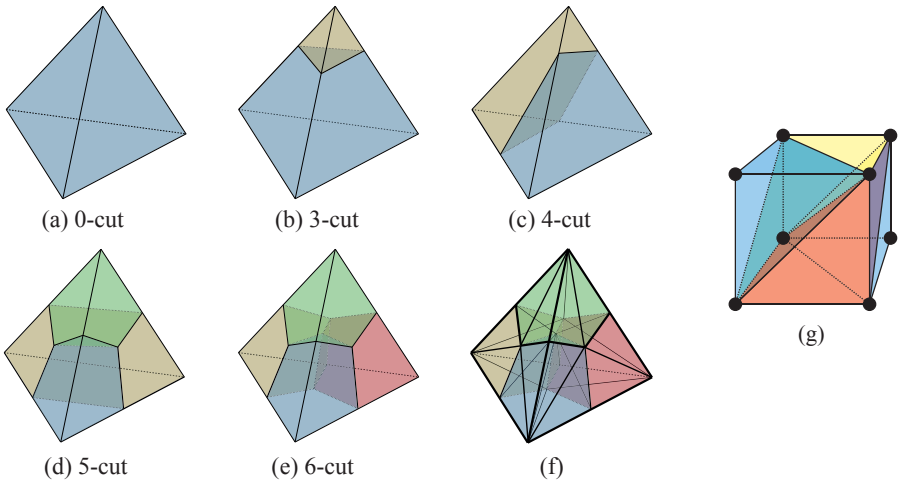


FIGURE 3.2: (a)∼(e) The five patterns of TSP IDs tagged at a tetrahedron's vertices. (a) Single TSP ID. (b) and (c) 2 TSP IDs. (d) 3 TSP IDs. (e) 4 TSP IDs. (f) The remeshing result of a 6-cut case. Figures courtesy of Bronson et al. [5]. (g) The initial meshing result for a cube of neighboring 8 vertices, split into 6 tetrahedra.

topology consistency of adjacent tetrahedra (see Figure 3.2). After being processed by Cleaver, the tetrahedral mesh will be treated as an initial dense mesh for subsequent simplification. This initial mesh has the following properties:

- The tetrahedral mesh is cut into several parts. Each part corresponds to a TSP of the original video. We call these parts "*sub-tet-mesh*" in the rest of this paper.
- Each tetrahedron has a unique TSP ID, meaning that it belongs to only one TSP.
- Each original vertex is located at the internal region of a tetrahedron, meaning that it has a unique TSP ID, and we call it an "*internal vertex*".
- The newly-added vertices on the cutpoints are at sub-pixel locations so they are shared by tetrahedra with different TSP IDs. Such a vertex is called "K-class *boundary vertex*", where K is the number of the TSPs sharing the vertex.

3.3.3 Tetrahedral mesh simplification and color optimization

With colors defined at mesh vertices, the initial tetrahedral mesh is a dense geometric representation of the input raster video. We shall simplify this mesh to obtain a compact representation while maintaining an accurate approximation of the color field. Our edge contraction operation is guided by an error metric similar to that used in the QEM mesh framework in [14]. However, unlike QEM, we introduce an additional term to encourage mesh uniformity. After mesh simplification, we perform boundary surface fitting to improve the boundary accuracy of the sub-tet-mesh. In addition, the colors on vertices are optimized so that the reconstructed video produces the minimal reconstruction error.

3.3.4 Reconstruction

Given the tetrahedral control mesh associated with color information, we reconstruct the video in two steps: mesh subdivision and rasterization. For mesh subdivision, the tetrahedral control mesh is subdivided by a method that extends the subdivision scheme [20, 33] from surface to the 3D volume. This subdivision scheme ensures topology consistency of the boundary surfaces of sub-tet-meshes, while preserving color discontinuities across different sub-tet-meshes. To rasterize the subdivided mesh with some specified resolutions, we first find the tetrahedron that a pixel belongs to and compute its color by linearly interpolating the colors at the vertices of the tetrahedron.

3.4 Simplification of Tetrahedral Mesh

We first segment the input video into N_T TSPs so that the initial tetrahedral mesh consisting of N_T sub-tet-meshes. Each vertex v_i of the video mesh is associated with 6 values, its three coordinates $\mathbf{x} = (x, y, t)$ and the three channels $\mathbf{c} = (r, g, b)$ of its color. we simplify this initial dense mesh by extended version of the quadric error metrics (QEM) [14, 15] by viewing a video as a 3D manifold in 6D space, since each vertex v is a 6D vector $[x, y, t, r, g, b]^\top = [\mathbf{x}, \mathbf{c}]^\top$.

3.4.1 QEM-based Tetrahedral Mesh Simplification

3.4.1.1 QEM

In [15], edge contraction, also called edge collapse, is proposed as the atomic mesh decimation operation. With a contraction $v_j \to v_i$, v_j and all tetrahedra incident to e_{ij} are removed, and v_i will move to a new position v_i^* minimizing a quadratic error function. The quadric error of a vertex is the sum of squared distances from the vertex to the 3D *flats* (i.e affine subspaces) spanned by the tetrahedra that are associated with the vertex. Specifically, suppose that a vertex v is associated with P tetrahedra. Then its quadric error is

$$\Delta(v^*) = \sum_{s \in S(v)} D_s^2(v^*) = \sum_{s \in S(v)} \left(\sum_j n_j^{s\top} (v^* - v) \right)^2 \tag{3.1}$$

where $S(v)$ is the set of subspaces associated with v, each of which is spanned by one of the P tetrahedra. $D_s(v)$ is the distance from v to subspace s, and the n_j^s are the unit norms of the subspace s. Using homogenous coordinate $\tilde{v}^* = (v^*, 1)$, Equation 3.1 can be written as a quadratic form $\Delta(\tilde{v}^*) = \tilde{v}^* \mathbf{Q} \tilde{v}^*$, where the quadric error is encoded in a 7×7 matrix $\mathbf{Q}$.

The optimal contraction position v^* for an edge e_{ij} connecting v_i and v_j is found by minimizing the following objective function, that is

$$\min_v \quad v^\top (\mathbf{Q}_i + \mathbf{Q}_j) v \tag{3.2}$$

Let $\tilde{v}^* = [x^*, y^*, t^*, r^*, g^*, b^*, 1]^\top$ denote the solution to the above minimization problem. Then the cost of edge contraction of e_{ij} is

$$\text{Cost}(e_{ij}) = \tilde{v}^{*\top} (\mathbf{Q}_i + \mathbf{Q}_j) \tilde{v}^* \tag{3.3}$$

After contraction, the accumulated quadratic error $\mathbf{Q} = \mathbf{Q}_i + \mathbf{Q}_j$ is inherited by the merged vertex v_i.

3.4.1.2 Extended QEM

To produce high quality approximations, Garland and Zhou [15] use a greedy strategy and maintain a minimal heap keyed on the error cost of edges. In each iteration, the algorithm selects an edge with the minimal cost from the heap and performs edge contraction if this contraction will not cause topology error or tetrahedron flipping, otherwise the algorithm simply ignores it and tries for another one. In practice, when applying the conventional QEM to a region with constant or linear color change, all the QEM error terms become zero so they provide no guidance on how to select edges to contract. As a result, such regions will be over-simplified, causing the entire topology of tetrahedral mesh to fall in a locked state so that edges cannot be contracted any more without flipping tetrahedra. To address this problem with QEM, we introduce a new quadric error terms in our extended QEM framework. Specially, the quadric error defined for a vertex not only includes the original quadric error defined in Eq. 3.1, but also embodies the squared distance from it to the average position $\bar{\mathbf{x}}$ of the vertices it inherits from iterative edge collapse. That is, if a vertex v moves to v^*, the quadric error is defined as

$$\Delta'(v^*) = (1-\eta)\Delta([\mathbf{x}^*, \mathbf{c}^*]^\top) + \eta \left\|\mathbf{x}^* - \bar{\mathbf{x}}\right\|_2^2 \tag{3.4}$$

The introduction of the second term is based on the intuition that when contracting an edge located at a region where color field is near constant or linear, we expect the simplified mesh to be as uniform as possible. The parameter η serves to balance the two terms. In our experiments η is set to 0.1. Clearly, Equation 3.4 can be written in a quadratic form. We initialize the error matrix for each vertex before edge contraction as follows,

$$\mathbf{Q}' = \mathbf{Q} + \eta \begin{pmatrix} \mathbf{I}_{3\times3} & \mathbf{O}_{3\times3} & -\mathbf{x} \\ \mathbf{O}_{3\times3} & \mathbf{O}_{3\times3} & \mathbf{O}_{3\times1} \\ -\mathbf{x}^\top & \mathbf{O}_{1\times3} & \mathbf{x}^\top\mathbf{x} \end{pmatrix} \tag{3.5}$$

To measure the degree of simplification, we define *simplification ratio* as the value of the number of vertices in the simplified control mesh N_c over that in the initial

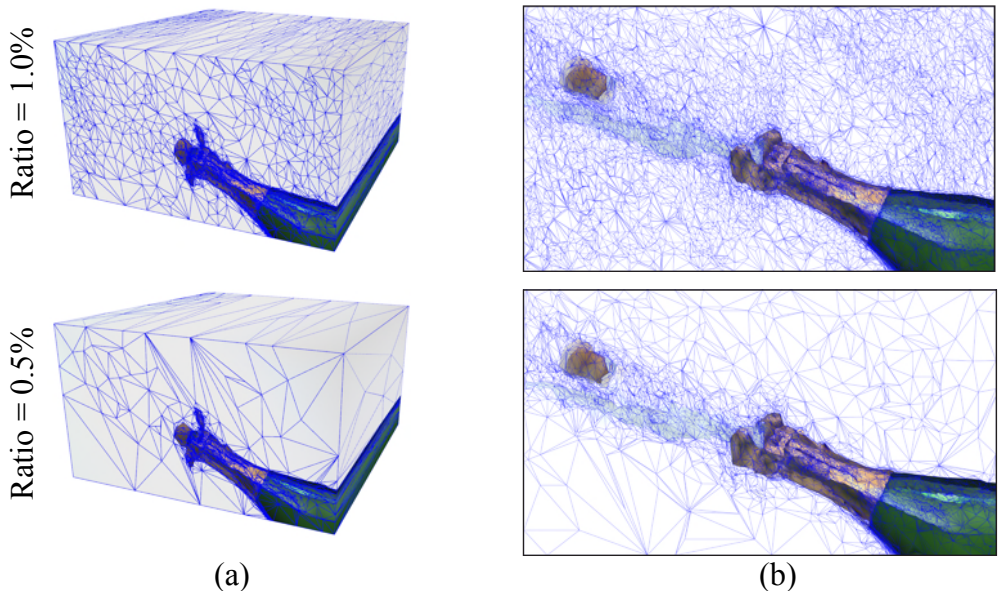


FIGURE 3.3: The simplified results for video CHAMPAGNE, at simplification ratios 1.0% (Row 1) and 0.5% (Row 2). (a) The simplified control meshes for the video. (b) The cross sections at frame 22.

dense mesh N_v. Our approach can simplify a tetrahedral mesh to up to 1% of its original size (see Table 3.2). Figure 3.3 shows two simplified results for the video CHAMPAGNE, at simplification ratios 1.0% and 0.5% respectively. For the constant colored background regions, our method generates large and uniform tetrahedra and for the feature edges, more tiny tetrahedra exist to characterize the details. With simplification going on, large tetrahedra in the background are processed before those near the features being generated so that the details are faithfully preserved after simplification.

3.4.1.3 Preserving boundaries

There are two kinds of boundaries to be preserved during simplification. One is "*internal boundaries*", i.e. the boundaries shared by multiple sub-tet-meshes, and the other is "*external boundaries*", corresponding to 6 outer faces of the video cuboid in 3D space. The vertices on the *external boundaries* are further categorized into "*face vertex*", "*edge vertex*" and "*corner vertex*". During edge

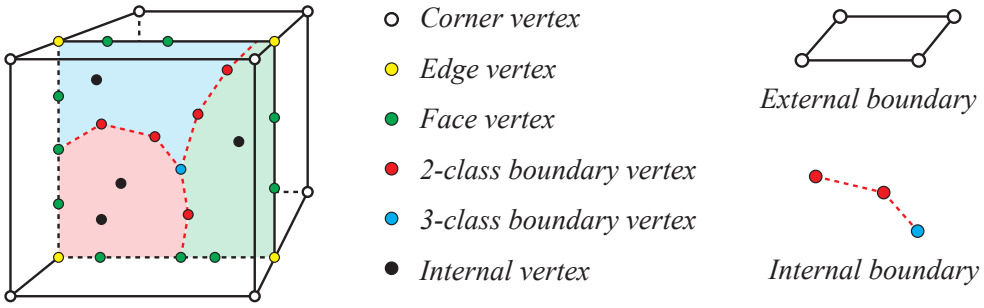


FIGURE 3.4: Vertex categories, marked with separated colors. The tetrahedral mesh for a video is a cuboid, its 6 faces are "external boundaries". The red dashed line segments represent "internal boundaries" shared by multiple sub-tet-meshes (marked as 3 colored regions in the figure).

contraction, if two vertices on the edge belong to different categories, care must be taken to where to select the optimal contraction position. We assign a priority to each vertex according to its category to reflect the vertex's feature importance, and the optimal position will be determined by the priorities of the two vertices. Specifically, vertex on the *external boundaries* is prior to *non-external*, while in *non-external boundary*, *internal boundary* is prior to *non-internal boundary*. The contraction position tends to favor those vertices with higher priority. We illustrate the vertex categories in Fig. 3.4 and please refer to Table 3.1 for the details of contraction position involving boundary vertices.

3.4.1.4 Application to Fast Mesh Generation for Volumetric Data with Multiple Labeled Regions

If color attributes are constant or not defined on the video pixels, the raster input video is degenerated into volumetric data with multiple labeled regions only. This kind of data abound in medical imaging, e.g. CT, MRI or arbitary segmented 3D data. Our extended QEM-based simplification algorithm can also applied to fast mesh generation for such kind of data. It corresponds to the case that $\eta = 1.0$ in Equation 3.4, where the goal for the generated mesh is uniformity only. A tetrahedral mesh with uniform distribution of vertices is desirable for

v_i		v_j		Contract Position
EB	face/edge/corner	Non-EB	IB(K-class, $K > 1$) / Non-IB	v_i
EB	corner	EB	corner	-
	corner		edges/faces containing the corner	v_i
	corner		edges/faces not containing the corner	-
	edge		same edge	$1/2(v_i + v_j)$
			different edges	-
	edge		faces containing the edge	v_i
			faces not containing the edge	-
	face		same face	$1/2(v_i + v_j)$
			different faces	-
Non-EB	IB (K_i-class, $K_i > 1$)	Non-EB	IB (K_j-class, $K_j > 1$)	Further determined by K_i and K_j
	IB		Non-IB	v_i
	Non-IB		Non-IB	QEM-based optimal position

TABLE 3.1: Target position of contraction determined by the boundary properties of the two vertices. External Boundary, Internal Boundary are abbreviated as EB and IB separately. "-" means not to contract the edge.

subsequent processing tasks, such as finite element simulation that require good shape quality of the tetrahedral elements.

Figure 3.5 shows the simplification results of connectome data with four hundred materials with no other attributes defined on it. The four hundred materials are acquired by pre-segmentation, and the simpilification ratios are set progressively lower in four experiments, from 5.0% to 0.5%. With the simplification ratio becomes gradually lower, the uniformity of the tetrahedra as well as the boundaries between difference materials are faithfully preserved (see Figure 3.5(b)∼(e)). The similar experimental results are produced in walnut data as Figure 3.6 shows, where the data contains four materials and ratio is set to 1.0%.

Simplification with varying densities. Our extended QEM-based simplification algorithm can be further applied to generating adaptive mesh, i.e. meshes with a varying density of the mesh vertices, which are demanded in many application where a tradeoff is needed between the complexity of the mesh and the preservation of the geometric details, especially near domain boundary. Now we discuss how to extend our method for simplifying a mesh where the final

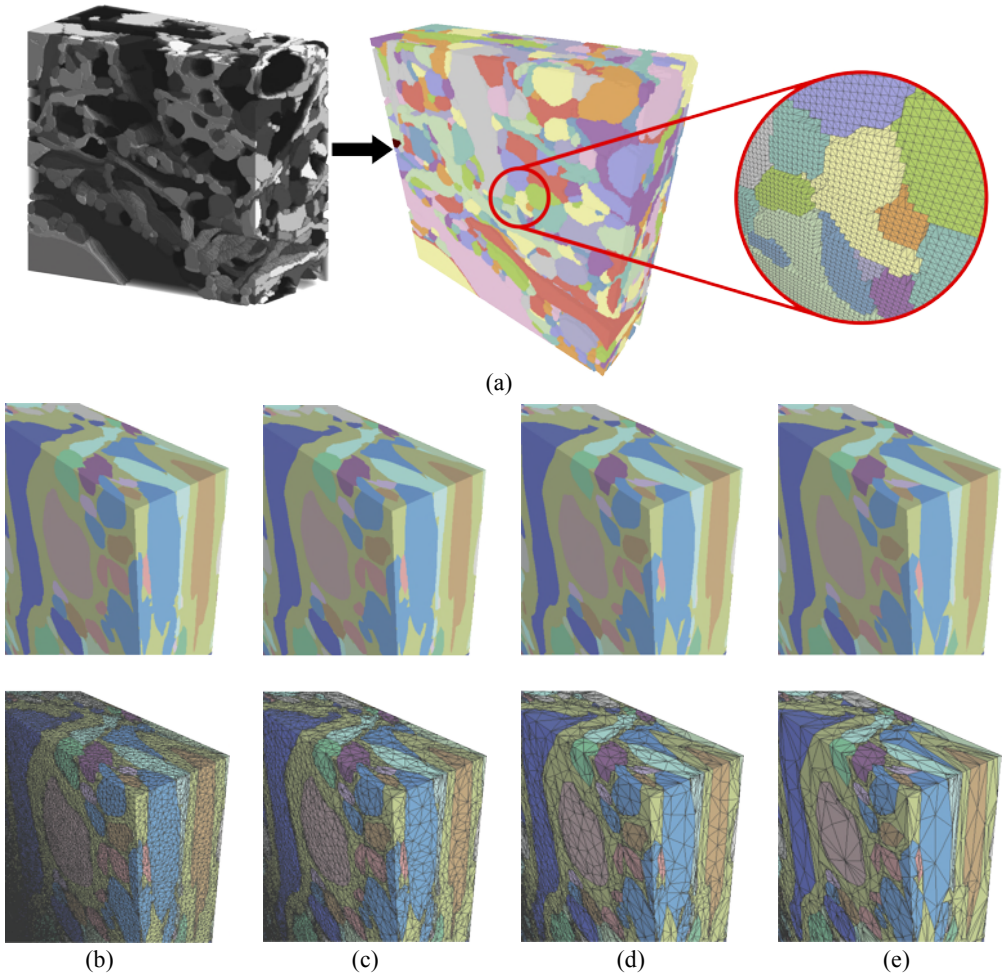


FIGURE 3.5: Simplification results of connectome data with 400 materials. (a) The initial mesh with 629,145,600 tetrahedra. (b) 5.0% (31,457,280 tetrahedra). (c) 2.0% (12,582,912 tetrahedra). (d) 1.0% (6,291,456 tetrahedra). (e) 0.5% (3,145,728 tetrahedrons).

simplified mesh has its mesh vertex density conforming with some given density function. This density function can be derived from the properties of the original data or a user-specified density function. Suppose that a density function $w(x, y, z) > 0$ is defined on the domain of the mesh to be simplified. Then we only need to modify the initial value of the error matrix $\mathbf{Q}$ of each vertex v by setting it to be $w \cdot \mathbf{Q}$, where w is the value of the density function at the position of the vertex v. This is the only modification needed and the other parts of the

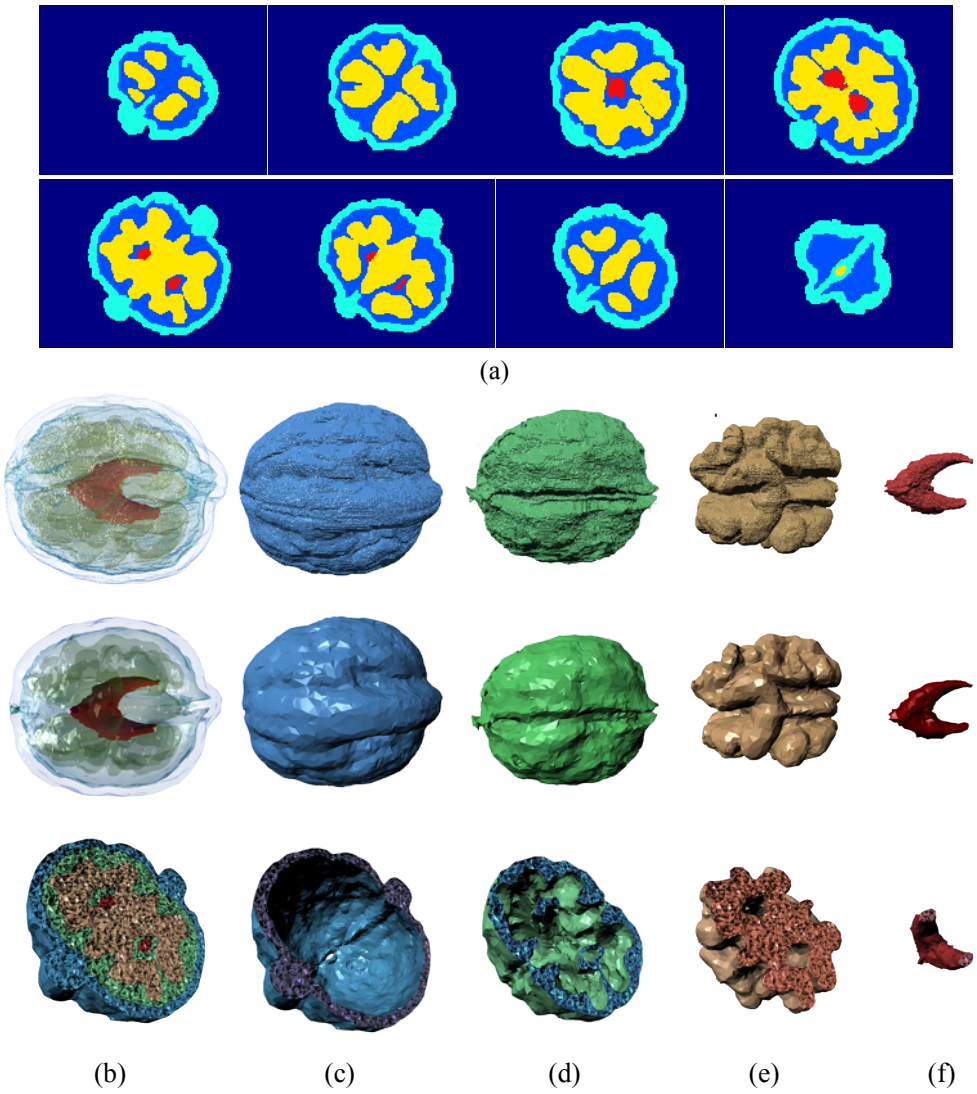


FIGURE 3.6: Simplification results of the walnut data set. (a) The visualization of the volumetric data (8 of 350 slices, from left to right, top to bottom), with each color representing each regions separatedly. In (b)∼(f), Row 1 is the initial mesh with 134,290,259 tetrahedrons. Row 2 is the simplification results with ratio 1.0%. Row 3 is the cut-off view of the simplification results. (b) The input mesh. (c)-(f) All the different regions materials from the outer layer to the inner layer.

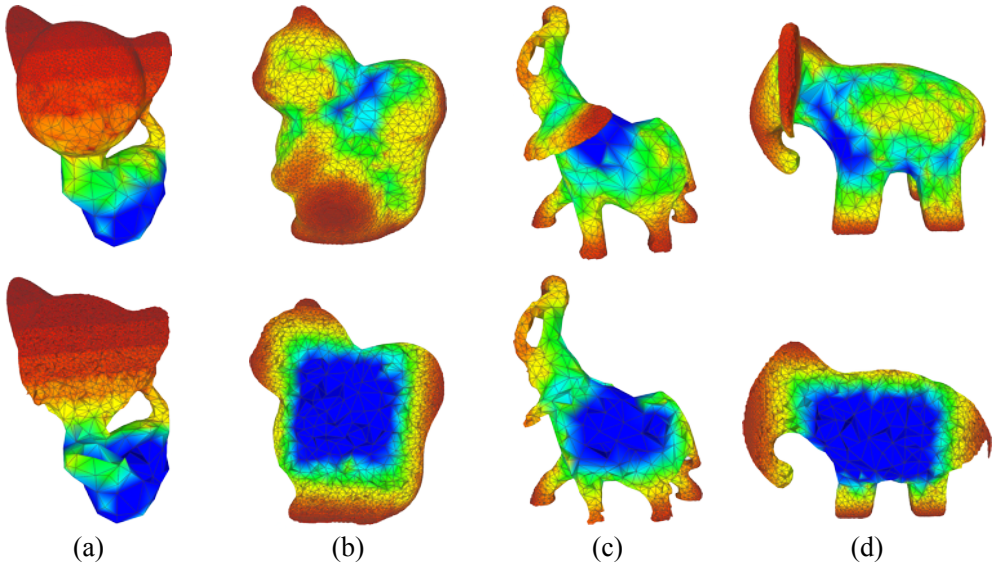


FIGURE 3.7: Simplification with varying densities on volume data of several scanned models. Red and blue mean high and low density. The second row is the cut view of the first row. (a) Density becomes larger from bottom to top. (b)(c)(d) Density becomes larger from inside to outside.

method remain the same.

Figure 3.7 shows some examples of simplification of tetrahedral mesh with varying densities. The density function of Figure 3.7(a) is a linear function with the largest weights on the upside, the rest are linear functions with the largest weights on the outer boundaries. Furthermore, the density function can also be applied to the mesh with multiple materials. As Figure 3.8 shows, a molar volume data consisting of two materials is applied with densities changing from low to high, from its bottom to the top. The interface of the two materials is a smooth surface. After simplification, it is well preserved with the variation of the density.

Large-scale data simplification. It is a challenging problem to process large-scale data altogether because the data set is often too large to be read into memory at once. We address this issue by developing a simple block-by-block approach for tet-mesh simplification. Specially, we first cut the input data

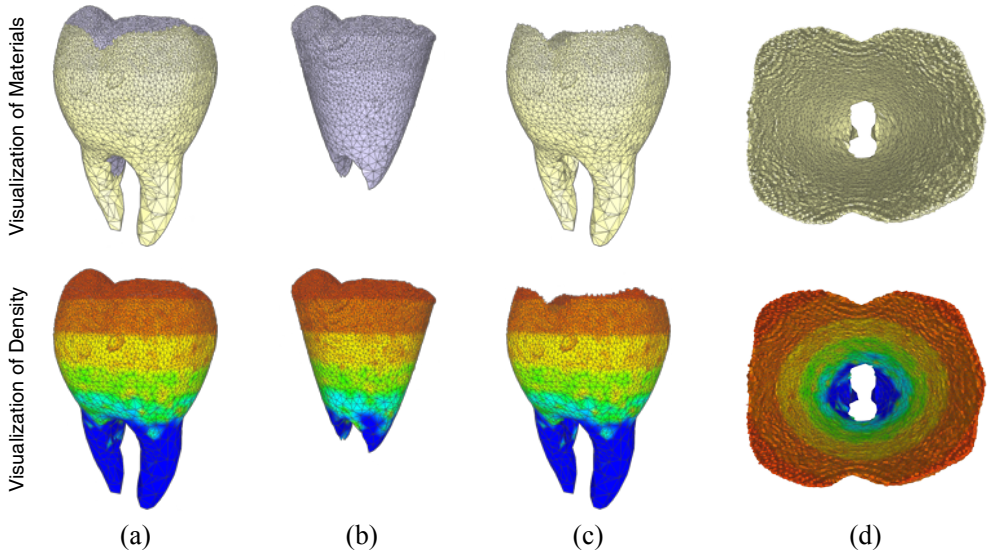


FIGURE 3.8: Density variation on mesh with multiple materials for molar scanning data. Red and blue means high and low densities. The first row is the visualization of different materials. The second row is the visualization of density. (a) The simplification result. (b)(c) The two materials of the data. (d) Top view of (c), showing the interface between the two materials.

cube into several small cubic blocks and simplify each block separately. For the ease of merging, the planar boundary surfaces of the small blocks are not simplified. After simplifying all the blocks, they are merged into a block of the original dimension and this merge large block is further simplified to generate the final simplified mesh. Figure 3.9 shows the pipeline, using a simple illustrative example.

3.4.2 Boundary Surface Fitting

A sparse control mesh after simplification produced by the previous simplification step might have its internal boundaries deviate significantly from their original positions in the input video. To improve the accuracy of the internal boundaries, we formulate and solved a surface fitting problem by minimizing the following objective function optimize the geometric positions of the vertices on the control

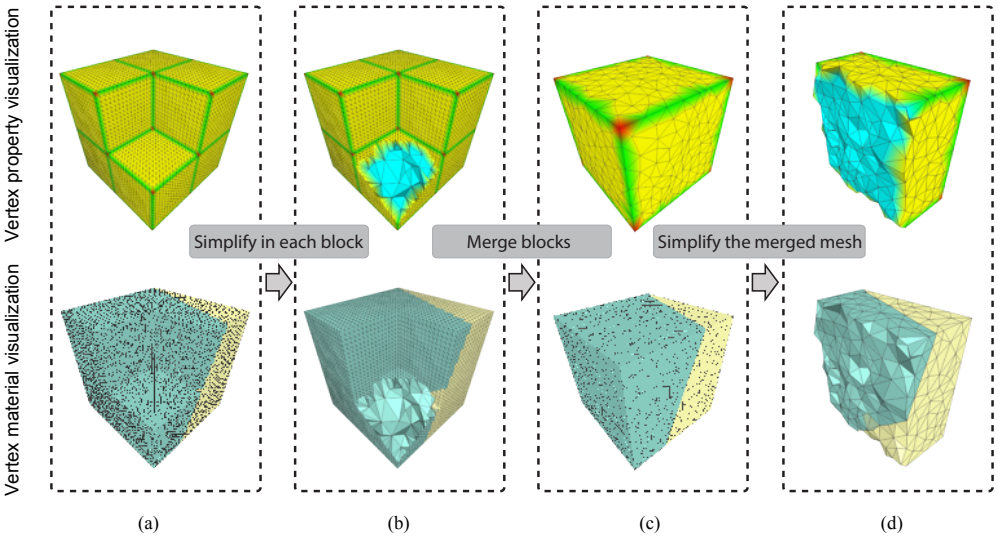


FIGURE 3.9: Block-by-block tet-mesh simplification method. The initial mesh is a $30 \times 30 \times 30$ tetrahedral mesh with 2 materials. (a) The original data is split into eight blocks of size $15 \times 15 \times 15$, with each one simplified to near 30%. In the four figures on the left, the front block is hidden so that some of the boundaries between adjacent blocks are visible, which are fixed during simplification. (b) The interior of one block. (c) The final simplified result from the merged blocks, with simplification ratio being 3.1%. (d) The interior view of the final result, boundary of each block has been simplified. Row 1: Different properties of each vertex in different colors: red $\sim$ corner, yellow $\sim$ face, green $\sim$ edge, cyan $\sim$ internal vertices. Row 2: Different materials in different colors.

mesh as follows,

$$\min_{\mathbf{v}_c} \quad E(\mathbf{v}_c) = E_F(\mathbf{v}_c) + \lambda E_L(\mathbf{v}_c) \tag{3.6}$$

where $\mathbf{v}_c$ are the vertices of the control mesh, λ a parameter balancing the energy term on fitting error E_F and mesh deformation error term E_L.

The fitting error E_F is the sum of squared distances from vertices on the internal boundaries in the densely subdivided mesh and their target positions on the original surfaces,

$$E_F(\mathbf{v}_c) = \sum_{k=1}^{N_s} \left\| v_k^t - v_k \right\|_2^2 = \sum_{k=1}^{N_s} \left\| v_k^t - \alpha_k \mathbf{v}_c \right\|_2^2 \tag{3.7}$$

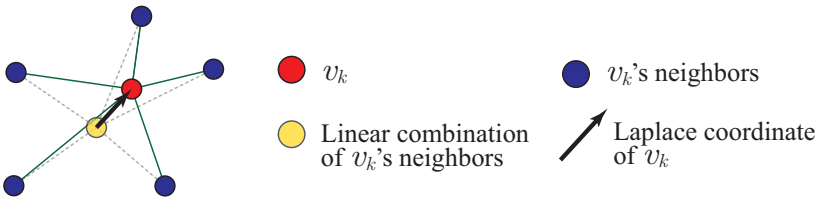


FIGURE 3.10: Laplacian coordinate of a vertex.

where v_k and v_k^t are respectively the position in the subdivided mesh and the target position of a boundary vertex. N_s is the number of such vertices. Every v_k can be substituted as a linear combination α_k of vertices on the control mesh as defined by the subdivision masks. The target position v_k^t is its projection onto the original boundary surface. Minimizing E_F only may cause flipped tetrahedra because the internal vertices remain unchanged relative to those boundary vertices. To solve this issue, we generalize the volumetric graph Laplacian [63] to the tetrahedral mesh and control the mesh deformation by constraining the changes of Laplacian, i.e. the mesh deformation error E_L. The Laplacian coordinate of a vertex encodes the difference between the vertex and a linear combination of its neighboring vertex (Figure 3.10). E_L is therefore expressed as

$$E_L(\mathbf{v}_c) = \sum_{k=1}^{N_c} \left\| \delta_k - \tilde{\delta}_k \right\| = \sum_{k=1}^{N_c} \| l_k \mathbf{v}_c - l_k \tilde{\mathbf{v}}_c \|_2^2 \tag{3.8}$$

where $\tilde{\delta}_k$ and δ_k are the Laplacian coordinates of v_k before and after optimization separately. N_c denotes vertex number of the control mesh. $\tilde{\mathbf{v}}_c$ represents positions of the vertices before optimization. l_k is a $1 \times N_c$ vector encoding the coefficients to compute δ_k. We set δ_k as

$$\delta_k = v_k - \frac{1}{|\mathcal{N}_k|} \sum_{v_j \in \mathcal{N}_k} v_j \tag{3.9}$$

where $\mathcal{N}_k$ is the set of neighbors of v_k and $|\mathcal{N}_k|$ is its cardinality.

The solution to Equation 3.6 is

$$\mathbf{v}_c = (\mathbf{A}^\top \mathbf{A} + \lambda \mathbf{L}^\top \mathbf{L})^{-1} (\mathbf{A}^\top \mathbf{v}^t + \lambda \mathbf{L}^\top \mathbf{L} \tilde{\mathbf{v}}_c) \tag{3.10}$$

where $\mathbf{A}$ is a $N_s \times N_c$ matrix with each row being α_k. $\mathbf{L}$ is a $N_c \times N_c$ matrix with each row being l_k. $\mathbf{v}^t$ is a $N_s \times 3$ matrix composed of all v_k^t with $k = 1, 2, ..., N_c$. λ is set to 1.0 in implementation. The energy function Equation 3.6 can be optimized by iterations to get an optimal and stable solution. We use 5 iterations in our implementation.

3.4.3 Color Optimization

The above steps simplify and optimize only the positions of all mesh vertices. After that, we globally optimize the colors of these vertices. We optimize an energy term E_C formulated similar to E_F in 3.7, but we use $\mathbf{v}_c$ to encode colors instead of geometric positions, while N_s becomes the total number of vertices in the subdivided mesh, and v_k^t represents the color sampled at the 3D position of the nearest vertex to v_k within the sub-tet-mesh it belongs to. We use the Conjugate Gradient (CG) algorithm to solve a sparse linear system for minimizing this energy.

3.5 Subdivision of Tetrahedral Mesh

We shall introduce in this section a tetrahedral mesh subdivision technique for reconstructing a raster video from a vectorized video represented as a tetrahedral control mesh. The simplified control mesh occupies the whole 3D video volume and consists of N_T sub-tet-meshes yielded by the Cleaver algorithm where N_T also denotes the number of TSPs. The 3 color channels of RGB are viewed as 3 scalar fields separately defined on the 3D domain. Since the simplified control mesh is sparse, direct rasterization would lead to undesirable visual artifacts. Besides, color discontinuities across sub-tet-meshes should be preserved as they essentially indicate visual features. We use mesh subdivision to address this issue.

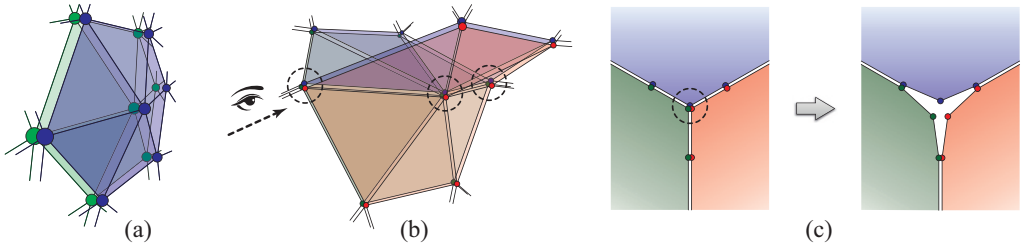


FIGURE 3.11: Subdivision at the boundary surfaces of sub-tet-meshes. Each color represents a sub-tet-mesh and points close to each other denote duplicates of a boundary vertex in its incident sub-tet-meshes. (a) A 2-class boundary vertex (the middle one), the adjacent relations of the two sub-tet-meshes (the navy and green) are the same. (b) 3 sub-tet-meshes meet and result in 3-class boundary vertices (marked by dashed circles). (c) The interfaces seen from the view angle as in (b). For a 3-class vertex (the middle one), its 3 duplicates (navy, green, red) will update the positions with the affine combination of the neighbors in their own sub-tet-meshes, yielding an inconsistent position and a "gap" among the sub-tet-meshes (right).

3.5.1 Subdivision Method

We extend the method in [33] as follows. Let $M = M^0$ represent the simplified control mesh. Our subdivision procedure $M^s \to M^{s+1}$ works by first computing vertex positions of M^{s+1} as affine combinations of their adjacent vertices in M^s, according to a set of subdivision *masks* defined on vertices and edges so that each vertex in M^{s+1} is either updated by a vertex point or derived from an edge from M^s. Then we replace each tetrahedron by 8 tetrahedra. We use the masks defined in [8] in implementation (see Figures 3.12(a) and 3.12(c))

$$\text{Vertex point:} \quad \hat{v}_i = \frac{18}{32} v_i + \frac{14}{32k} \sum_{v_j \in \mathcal{N}_i} v_j,$$

$$\text{Edge point:} \quad \hat{v}_{ij} = \frac{10}{32}(v_i + v_j) + \frac{12}{32k} \sum_{v_l \in \mathcal{N}_{ij}} v_l$$

where the $\hat{v}_i$ and $\hat{v}_{ij}$ are vertex and edge point respectively, $\mathcal{N}_i$ and $\mathcal{N}_{ij}$ the set of neighboring vertices of vertex v_i and the set of edge e_{ij}, respectively, and k is the number of elements in the corresponding set. The above scheme smoothes both the mesh vertices as well as their associated colors. However, the boundary vertices tend to be pulled inward by the internal ones if we apply

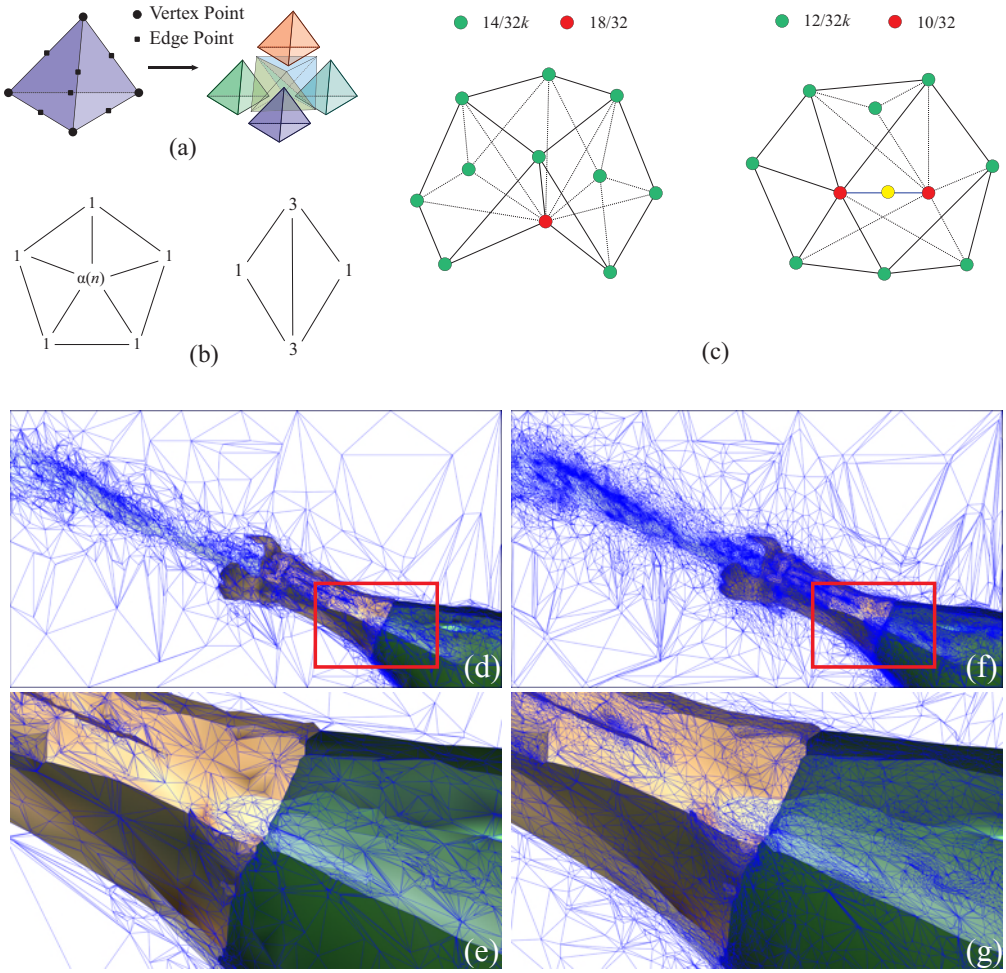


FIGURE 3.12: Illustration of tetrahedral mesh subdivision. (a) A tetrahedron is replaced by 8 tetrahedra. (b) Subdivision masks defined on boundary vertices with left for vertex points and right for edge points. (c) Subdivision masks defined on internal vertices with left for vertex points and right for edge points. (d) and (f) are the cross section of the control mesh and its subdivided mesh at frame 28 in video CHAMPAGNE, respectively. (e) and (g) are their corresponding zoomed-in views.

the masks in [8] to both kinds of vertices indiscriminately. This will cause each sub-tet-mesh to shrink and finally lead to "gaps" between neighboring sub-tet-meshes. To overcome this problem, we apply Loop's subdivision masks [33] to the boundary vertices instead so that positions of the boundary vertices are updated independent of the internal ones during subdivision.

3.5.2 Discontinuity Preservation at Boundaries

With the above subdivision method, at the boundaries of sub-tet-meshes the color will be non-smooth, though still continuous, since the boundary vertices are often shared by multiple sub-tet-meshes. To model *color discontinuities* across sub-tet-meshes, we further explicitly split the tetrahedral mesh at boundary surfaces of all the sub-tet-meshes. Specifically, we create K_i duplicates (including the original one) for each boundary vertex v_i^B that is shared by K_i sub-tet-meshes, and assign duplicate to each of its incident sub-tet-meshes. To make all the duplicates of v_i^B always meet at a common position, we impose the constraint for K-class boundary vertices ($K > 2$) that their positions should keep unchanged during subdivision. On the contrary, for a K-class boundary vertex with $K \leq 2$, the positions can be updated independently, since the adjacent relations are the same at each side of the sub-tet-meshes (Figure 3.11(a)).

Figures 3.12(d)∼3.12(g) shows the cross sections of a control mesh and its subdivision mesh at frame 28 of video CHAMPAGNE. By subdividing the control mesh, the boundaries and color within all the regions become smooth while the color discontinuity at the boundaries is still faithfully preserved.

3.6 Experimental Results

We have implemented our video vectorization algorithm and tested it on a variety of raster videos. Our results can be found in Figures 3.14, 3.13, 3.15, 3.17 and 3.18, as well as the accompanying video linked at `https://goo.gl/7AGrFe`.

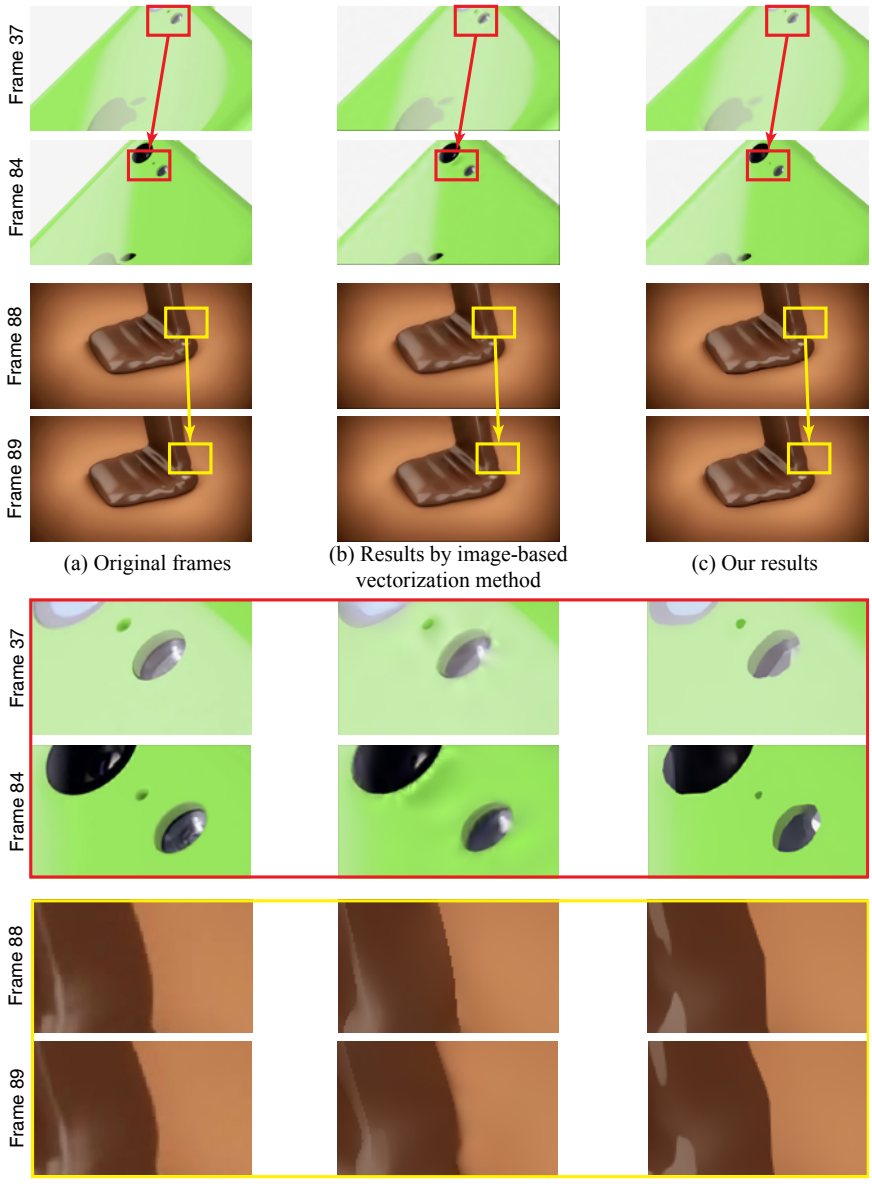


FIGURE 3.13: Reconstructed frames in video IPHONE and CHOCOLATE. (a) The original frames. (b) Image-based vectorization results by [29]. (c) Our results. (d) is zoom-in views of the regions circled by the same color boxes in (a), (b) and (c). Results show that our method faithfully reconstructs the details and more importantly ensures the temporal coherence. Taking the IPHONE video as an example, from Frame 37 to 84, the dark spot near the flash is lost by the image vectorization method although the same parameter is set for both frames. But it remains in our results (red boxes). Likewise, our method also produces consistent vectorized details for the main edges in the video CHOCOLATE, while blurry edge is produced in Frame 89 by Liao et al.'s method (yellow boxes).

We first choose two synthesized videos to test our algorithm. Figure 3.13 shows two video clips from TV ads, which contain sharp visual features, considerable object motions, as well as details and glossy changes. For example in the video iPhone, the dark spot near the flash is so tiny making it easily lost when the video is represented as a sparse mesh. While in the video Chocolate, frequent glossy changes make it difficult to generate a compact, yet still accurate vectorized video. However, in our reconstructed video these features are faithfully preserved and frame transition is smooth. In terms of the visual quality of individual frame, our method also produces reasonably good results, which can be compared with those generated by image vectorization method [29].

In Figures 1.2, 3.14 and 3.15, we show the results for two videos recording the process of flower blooming. Unlike the two synthesized videos above, both videos contain many irregular motions of the foreground objects as well as occlusion and visibility change. The two factors indeed make it more challenging for us to simplify the tetrahedral mesh and to keep the temporal coherence. Although the simplification ratio is set to 1.0%, our approach still produces faithfully reconstructed videos without apparent visual artifacts. Since it is resolution-independent, a vectorized video can be magnified at arbitrary resolution. To illustrate the sparsity of the control mesh, we visualize the cross sections of the mesh at the corresponding frame positions. Large tetrahedra are generated by our approach to represent those flat background, while more detailed tetrahedra are produced near the region containing substantial color variation which are used to characterize local details, e.g. the flower cores, the boundaries of petals and stems, etc. Some zoom-in views of the cross sections of the subdivided meshes are also illustrated for better visualization.

3.6.1 Comparison

3.6.1.1 Temporal Coherence

To show the importance of frame coherence, we first compare our method with a state-of-the-art image vectorization method [29], which is applied to each video

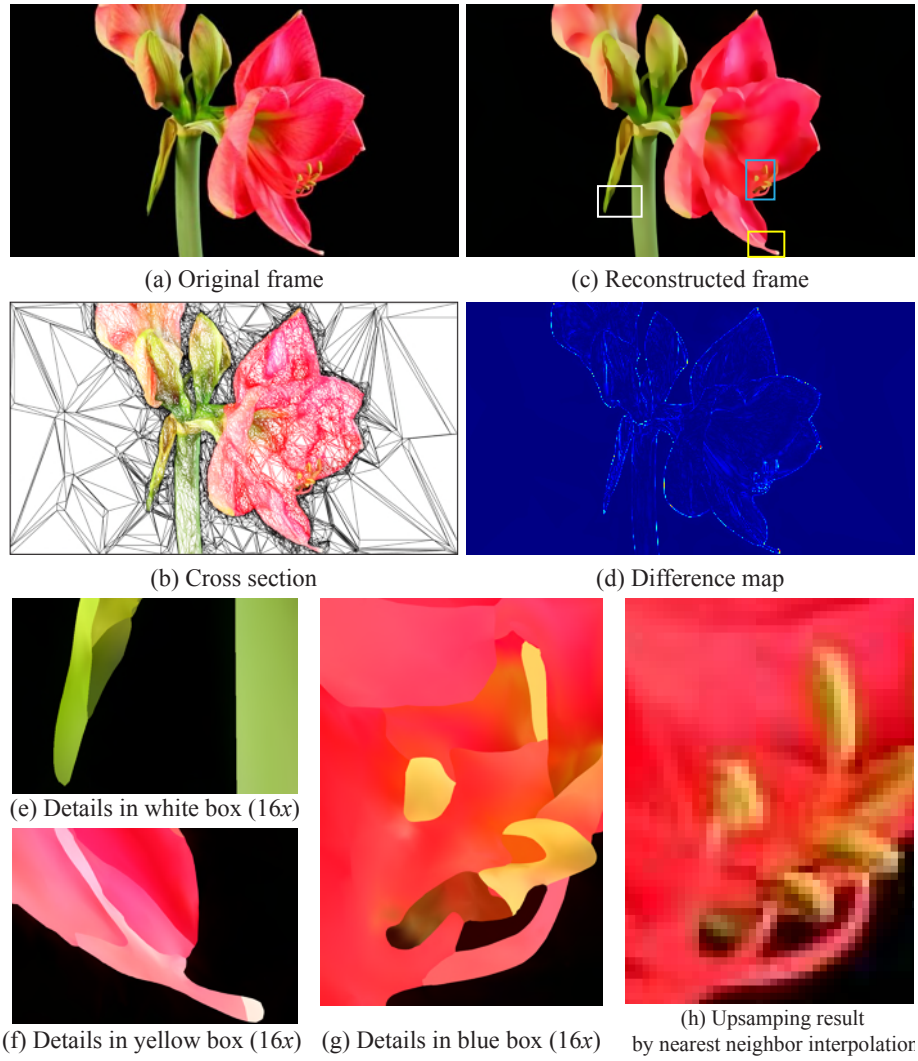


FIGURE 3.14: Vectorization of the video BLOOMING I, at frame 67. (a) The original frame. (b) The cross section of the simplified control meshes at the same position of frame 67. (c) The reconstructed frame from the vectorized video representation. (d) Difference map between the reconstructed frame and original frame. Figures (e), (f), and (g) are the zoomed-in views of the box regions in (c), magnified at 16× resolutions. (h) The upsampling result of the region in the original frame, corresponding to the blue box in (c), by nearest neighbor interpolation (16×). Please refer to the accompanying video for this and the other results in this paper.

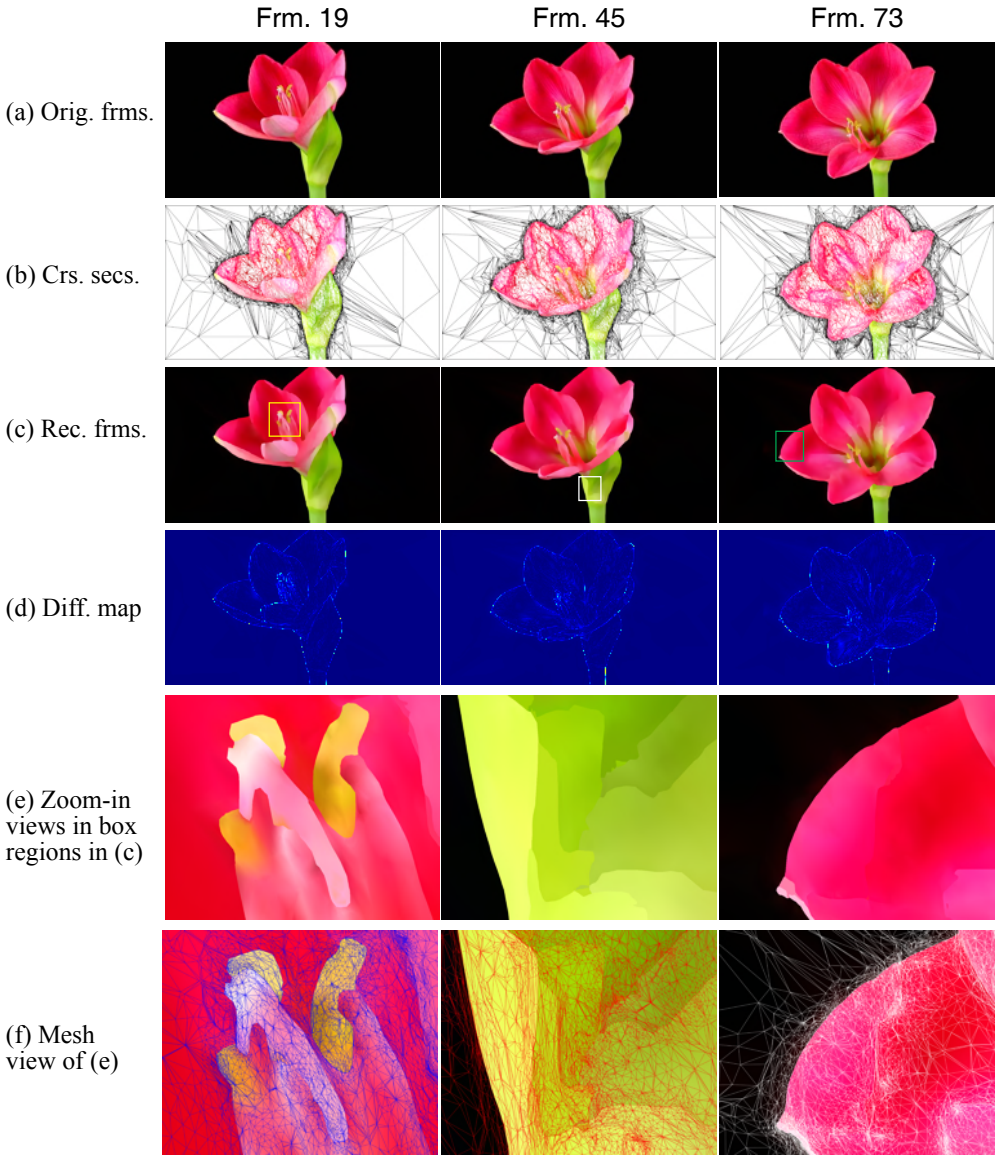


FIGURE 3.15: Results of video BLOOMING II at frames 19, 45 and 73 (Rows 1∼3). The original video is of size $640 \times 360 \times 95$. (a) Original frames. (b) The cross sections of the control tetrahedral meshes at corresponding frames. (c) Our reconstructed frames (d) The difference maps. (e) and (f) are the magnified details (16×) of the box regions shown in (c) and their corresponding cross sections of the subdivided meshes.

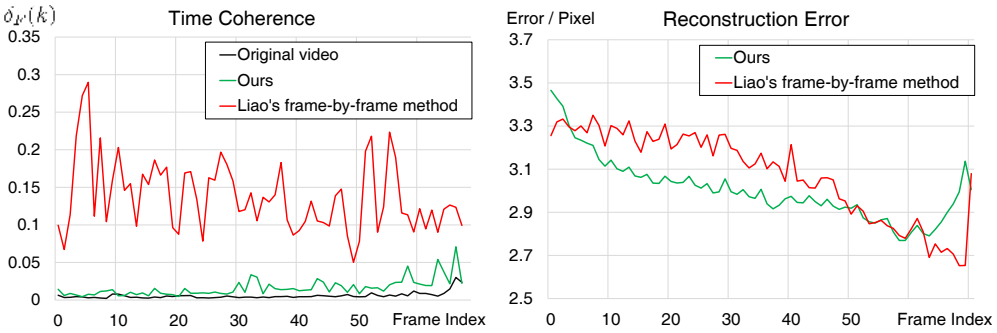
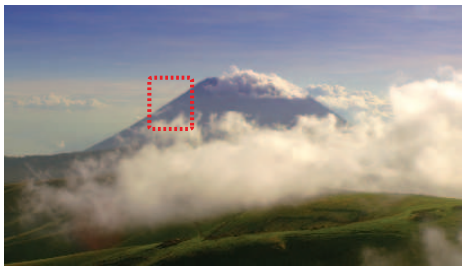


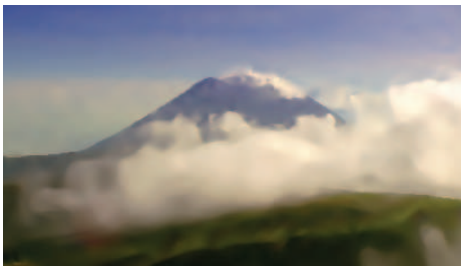
FIGURE 3.16: $\delta_F(k)$ and reconstruction error, for video MOUNTAIN.

frame individually, frame by frame. Such an image vectorization method is expected to produce higher quality result, but when applying it to each of the video frames, we have found that the reconstructed video is unstable and annoying visual artifacts like flicker present. This is because applying the image vectorization method to each video frame directly cannot preserve the spatio-temporal coherence of the video, even after a carefully parameter is tuned. For example, for the video IPHONE in Fig. 3.13, frame 37 in Liao et al.'s results is rather good as the edges around the camera is sharp and smooth; while in frame 84, some edges and a small black spot are lost. Such visual features, however, are well preserved in our results.

In order to weaken the influence of complex motion to the results, in Fig. 3.16 we show the results for a video sequence with smooth motion, MOUNTAIN. In the video the cloud moves slowly, while the mountain is almost static in the distance. Even with such relatively simple settings, the frame-by-frame method [29] produces unstable results with flickering, as the accompanying video shows. In contrast, our method yields very stable results and sharp features are well preserved, with a 2.0% simplification ratio. To quantitatively characterize the temporal coherence, we compute the change of optical flow throughout the entire video sequence. Specifically, we first accumulate the summed square distance $\delta_F(k)$ between the optical flow maps of every two successive frames of the video, f_k and f_{k+1}. The sum further regularized by the total number of video pixels is taken as the measure of temporal smoothness for the entire video frames, noted

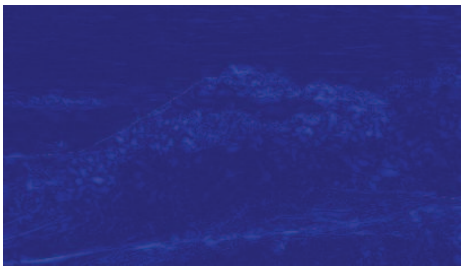


(a) Original frame

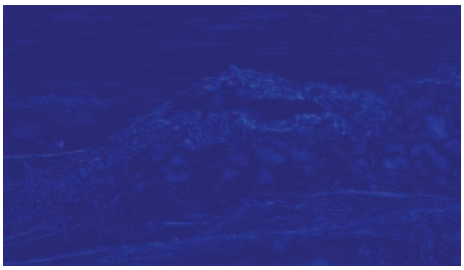


(b) Liao et al.'s result

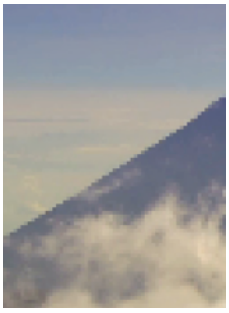
(c) Our result



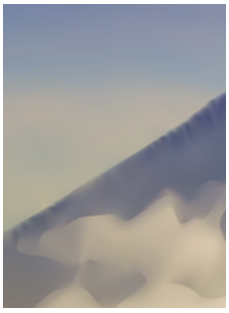
(d) Difference map of (b)



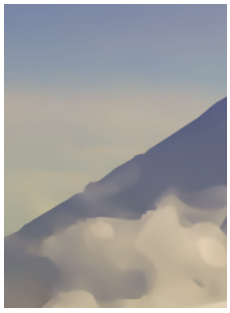
(e) Difference map of (c)



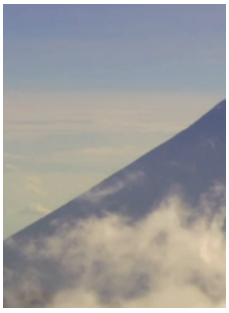
(f) Nearest neighbor



(g) Liao et al's result ($8x$)



(h) Our result ($8x$)



(i) Original frame in HD version

FIGURE 3.17: Results of video MOUNTAIN. (a) Original frame. (b)(c) Reconstructed frames by Liao et al's and our methods. (d)(e) Difference maps of (b)(c) respectively. (f)(g) and (h) are zoom-in views of the red box regions in (a), by nearest neighbor interpolation, Liao et al.'s and our methods respectively, rastered at $8\times$ resolution. (i) is the red box region in the HD-version original frame. Compared with (i), our result (h) preserves the sharp feature at the mountain ridge.

as Δ_F. Intuitively, severe flickering will lead to a high Δ_F since optical flow will change frequently. To visualize the results, Fig. 3.16 plots $\delta_F(k)$ for every two successive optical flows on the results of MOUNTAIN by our and Liao et al.'s method, comparing with the original video. We can see our result is as stable as the input video.

3.6.1.2 Reconstruction Error

We further evaluate the reconstruction quality in terms of per-pixel reconstruction error, which is also used by previous image vectorization methods.

The results show that our method achieves lower reconstruction error compared to [29] with the same simplification ratio. In Figure 3.16 right, we plot the reconstruction error for each frame of the video MOUNTAIN. The simplification ratio in this example is set to 2.0% for both methods. The curve illustrates that for most frames our method yields lower reconstruction error than Liao et al.'s approach, which also means more accurate reconstruction. Figure 3.17 also reveals some details in the results of both methods. Although the highly textured regions like the lawn become blurry that are beyond the capability of vector representation, our method still preserves those sharp features. For example, our rasterized video faithfully reconstructs the mountain ridge. The overall reconstruction errors are 3.00/pixel and 3.07/pixel for our and Liao et al.'s results respectively.

Figure 3.18 shows results for three additional real videos and one animation, DOG. The reconstruction error of our results is typically around 2.0/pixel, and simplification ratios are set no larger than 2.0% except for the animation containing large motions in the foreground (the dog) and textured background, which is 5.0% instead. All our results look smooth without flicking artifacts. We list the statistics data in Table 3.2 for detailed references.

For the amount of storage required, we have applied the streaming compression algorithm specifically designed for tetrahedral volume meshes [18] to our generated tetrahedral mesh. We achieve comparable results to those produced by

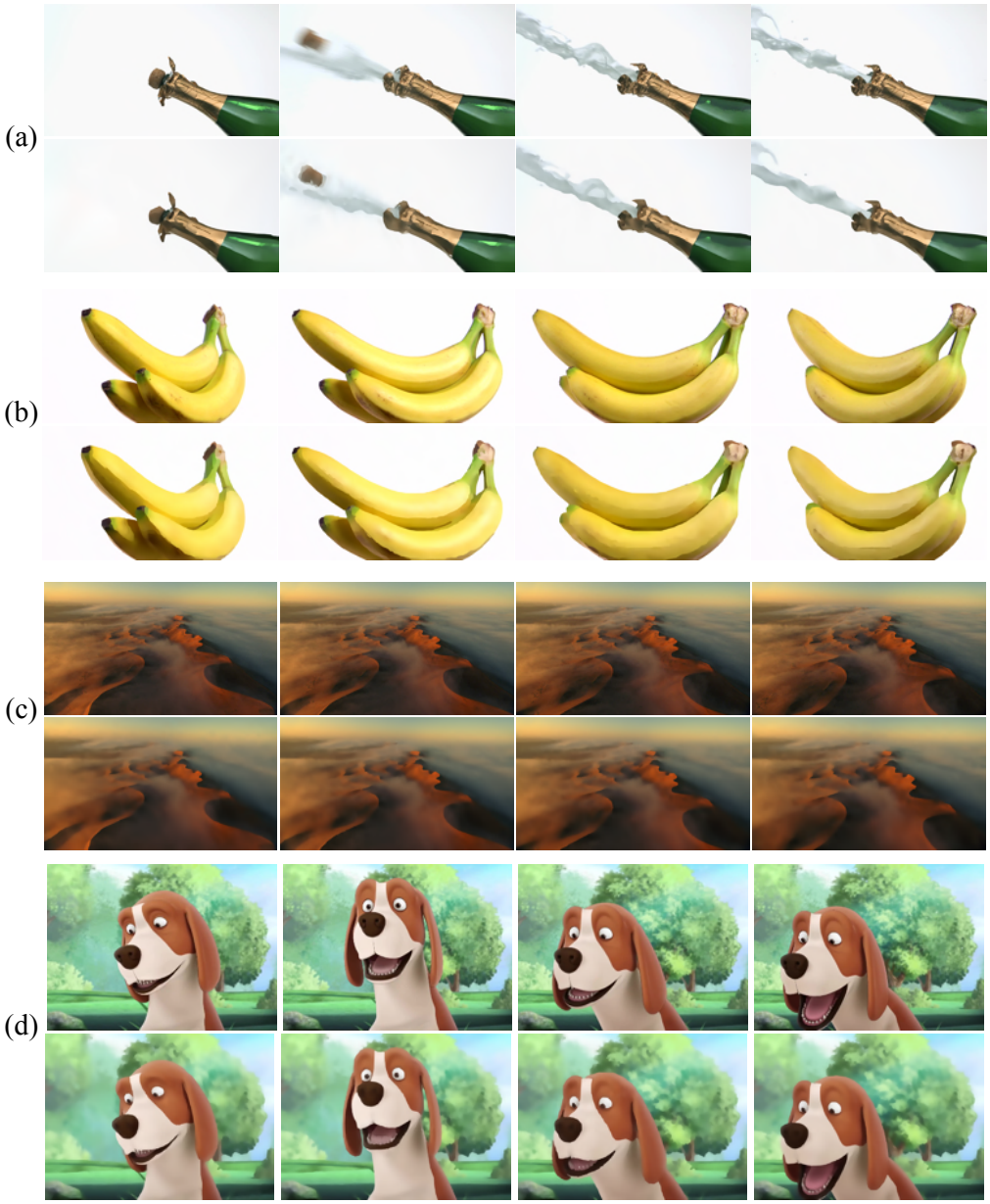


FIGURE 3.18: Reconstructed frames from our vector video representation for four videos. (a)∼(d) CHAMPAGNE, BANANA, DESERT and DOG. The odd rows show the original frames and even rows show the corresponding frames from our reconstructed videos. See our accompanying video for more details.

	Resolution	Ours			Liao et al.'s		
		S.R.	R.E.	Storage	S.R.	R.E.	Storage
BLOOMING I	640×360×70	1.0%	2.163	2.51MB	1.0%	2.936	2.36MB
BLOOMING II	640×360×95	1.1%	2.862	3.73MB	1.5%	3.042	3.25MB
DOG	373×270×57	5.0%	4.293	3.70MB	5.1%	5.014	4.12MB
IPHONE	480×270×100	0.5%	1.325	813KB	1.1%	1.379	876KB
CHAMPAGNE	480×270×100	0.5%	2.914	933KB	1.0%	3.899	1.02MB
CHOCOLATE	448×252×100	1.0%	1.523	1.48MB	1.0%	1.571	1.62MB
MOUNTAIN	406×231×70	2.0%	3.005	1.88MB	2.0%	3.079	1.92MB
DESERT	508×288×80	1.0%	2.637	1.62MB	1.1%	3.047	1.87MB
BANANA	450×256×160	2.0%	2.866	5.20MB	2.0%	2.928	5.04MB

TABLE 3.2: Simplification ratio (S.R.) and per pixel reconstruction error (R.E.) of testing videos, by our and Liao et al.'s methods. Note that for the amount of storage of Liao et al.'s method, it is evaluated under the same simplification ratios specified to our method.

applying zip compression to the triangular meshes of the video frames generated by Liao et al's method, shown in Table 3.2. For time complexity and memory cost, we run our method on an Intel Xeon E5-4650 2.7GHz machine with 100G RAM. Our C++ implementation takes 45 minutes on average to process a 70 frames of 640×360 video, and the memory cost is around 70GB.

3.6.2 Limitations

We have shown that our method can deal with a wide variety of videos. One limitation for vectorizing a video is that it may not be suitable to represent video objects with fine and rapidly changing details, as image vectorization methods are vulnerable to highly detailed image regions such as textures. For such cases, the reconstruction error will be relatively large. For example for a video of a running rhino shown in Figure 3.19, the segmented TSPs contain sharp color change because of the rapid motion. This leads to color mix-up across sharp features.

FIGURE 3.19: Failure case caused by rapid motion in the video RHINO. Frame No. 14 of the original frame (left) and our constructed result (right). In this example, the control mesh is simplified to a ratio of 5.0%. From AFRICA (2013) (©BBC).

3.7 Conclusions

We have presented a vector-based video representation and its associated video vectorization and reconstruction methods. The core of our method is the simplification and subdivision of tetrahedral meshes defined over the spatial-temporal video volume. Our vector-based video representation offers the benefits usually found in vector graphics, such as compactness and scalability, as indicated by our experiments and comparisons. Since editability is another advantage of vector graphics compared with raster graphics, as future work we plan to explore the applications such as shape editing and color editing that will benefit from the editability of our video vector representation.

Chapter 4

Conclusion and Future Research

4.1 Principal Contributions

In this thesis, we focus on two issues on video analyzing and processing, one is to extract the common foreground object from a group of related videos, and the other is to convert a raster video into a vector-based representation. The contributions of the present works are as follows:

For video object co-segmentation:

- We present a novel framework that consistently segments the common foreground objects in a group of videos, and is applicable to generic videos. For videos with different background, our method can work without user interaction, and for videos with similar background, our method can also work with moderate user guidance.
- We develop an appearance-motion-fused (amf) co-segmentation algorithm that leverages the appearance and motion features, based on subspace clustering, which also enables our algorithm well differentiated from image co-segmentation ones.

For video vectorization:

- We present an effective video vectorization algorithm based on simplification and subdivision of tetrahedral meshes, which is the first vectorization method for generic videos.
- We extend QEM-based simplification algorithm and apply it to generating the sparse tetrahedral mesh, which is the vector-based representation for video. Our extended QEM-based simplification algorithm can achieve very high simplification ratio and well preserve visual features and color fidelity. Such an extended simplification algorithm can also handle volumetric data with no color attributes defined and generates sparse mesh with uniform vertex distribution.
- The experimental results show that our method can generate vectorized videos with smooth time coherence, and its reconstruction error is comparable with image-based method on individual frame. The main visual features are faithfully preserved.

4.2 Future Research

There are many interesting issues worth further studying. Some of these issues have been mentioned in preceding chapters.

- In video object co-segmentation, our framework realizes the problem in a manner of global optimization on the videos, but it lacks a fine-tuning mechanism for locally refining the results. Since the common foreground of different videos would complement each other, in the future we plan to take a co-refinement step that corrects the errors and refines the result in a video by exploiting the good result in another one. For example, the user first drags a rectangle around the area of segmentation errors in a video frame. The rectangle is then propagated to the successive frames, and the result is refined within these frames by local optimization, while imposing the constraint of good results of other videos in the same group.

- In video vectorization, our proposed framework is also a global processing solution. This solution may cost large computer resources and cannot well handle videos with fast motions. In the future research, we may consider to use a propagation scheme, that is to vectorize some key-frames first and then propagate the generated meshes to neighboring frames. Although many video applications are developed with such kind of propagation idea, how to propagate a triangular mesh to successive frames and how to keep the time coherence of the meshes is still a challenging task and seldom touched on in the literature. In our future work, we may explore the solution following this way.

Bibliography

[1] Adobe after effects.

[2] Aseem Agarwala, Aaron Hertzmann, David H Salesin, and Steven M Seitz. Keyframe-based tracking for rotoscoping and animation. In *ACM Trans. Graphics*, volume 23, pages 584–591. ACM, 2004.

[3] Xue Bai, Jue Wang, David Simons, and Guillermo Sapiro. Video snapcut: robust video object cutout using localized classifiers. In *ACM Transactions on Graphics (TOG)*, volume 28, page 70. ACM, 2009.

[4] Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo, and Tsuhan Chen. icoseg: Interactive co-segmentation with intelligent scribble guidance. In *IEEE CVPR*, pages 3169–3176. IEEE, 2010.

[5] Jonathan R Bronson, Joshua A Levine, and Ross T Whitaker. Lattice cleaving: Conforming tetrahedral meshes of multimaterial domains with bounded quality. In *Proceedings of the 21st International Meshing Roundtable*, pages 191–209. Springer, 2013.

[6] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[7] Jason Chang, Donglai Wei, and John W. Fisher III. A video representation using temporal superpixels. In *IEEE CVPR*, June 2013.

[8] Yu-Sung Chang, Kevin T McDonnell, and Hong Qin. A new solid subdivision scheme based on box splines. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 226–233. ACM, 2002.

[9] Ding-Jie Chen, Hwann-Tzong Chen, and Long-Wen Chang. Video object cosegmentation. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 805–808. ACM, 2012.

[10] Wei-Chen Chiu and Mario Fritz. Multi-class video co-segmentation with a generative multi-video model. In *IEEE CVPR*, Portland, OR, USA, 2013. IEEE, IEEE.

[11] Yung-Yu Chuang, Aseem Agarwala, Brian Curless, David H Salesin, and Richard Szeliski. Video matting of complex scenes. In *ACM Trans. Graphics*, volume 21, pages 243–248. ACM, 2002.

[12] Maxwell D. Collins, Jia Xu, Leo Grady, and Vikas Singh. Random walks based multi-image segmentation: Quasiconvexity results and gpu-based solutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, Rhode Island, June 2012.

[13] Yanwei Fu and Yanwen Guo. Content-sensitive collection snapping. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.

[14] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.

[15] Michael Garland and Yuan Zhou. Quadric-based simplification in any dimension. *ACM Transactions on Graphics (TOG)*, 24(2):209–239, 2005.

[16] Aleksey Golovinskiy and Thomas Funkhouser. Consistent segmentation of 3d models. *Computers and Graphics*, 33(3):262–269, 2009.

[17] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient hierarchical graph-based video segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2141–2148. IEEE, 2010.

[18] Stefan Gumhold, Stefan Guthe, and Wolfgang Straßer. Tetrahedral mesh compression with the cut-border machine. In *Proceedings of the conference on Visualization'99: celebrating ten years*, pages 51–58. IEEE Computer Society Press, 1999.

[19] Xavier Hilaire and Karl Tombre. Robust and accurate vectorization of line drawings. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(6):890–904, 2006.

[20] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 295–302. ACM, 1994.

[21] Ruizhen Hu, Lubin Fan, and Ligang Liu. Co-segmentation of 3d shapes via subspace clustering. In *Computer Graphics Forum*, volume 31, pages 1703–1713. Wiley Online Library, 2012.

[22] Rik DT Janssen and Albert M Vossepoel. Adaptive vectorization of line drawing images. *Computer vision and image understanding*, 65(1):38–56, 1997.

[23] Armand Joulin, Francis Bach, and Jean Ponce. Multi-class cosegmentation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 542–549. IEEE, 2012.

[24] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3d mesh segmentation and labeling. *ACM Trans. Graphics*, 29(4):102, 2010.

[25] Yu-Kun Lai, Shi-Min Hu, and Ralph R Martin. Automatic and topology-preserving gradient mesh generation for image vectorization. In *ACM Transactions on Graphics (TOG)*, volume 28, page 85. ACM, 2009.

[26] Gregory Lecot and Bruno Lévy. Ardeco: Automatic region detection and conversion. In *Proceedings of the 17th Eurographics conference on Rendering Techniques*, pages 349–360. Eurographics Association, 2006.

[27] Yong Jae Lee, Jaechul Kim, and Kristen Grauman. Key-segments for video object segmentation. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 1995–2002. IEEE, 2011.

[28] Yin Li, Jian Sun, and Heung-Yeung Shum. Video object cut and paste. *ACM Trans. Graphics*, 24(3):595–600, 2005.

[29] Zicheng Liao, Hugues Hoppe, David Forsyth, and Yizhou Yu. A subdivision-based representation for vector image editing. *Visualization and Computer Graphics, IEEE Transactions on*, 18(11):1858–1867, 2012.

[30] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

[31] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE Trans. PAMI*, 33(5):978–994, 2011.

[32] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. PAMI*, 35:171–184, 2013.

[33] Charles Loop. Smooth subdivision surfaces based on triangles. 1987.

[34] Lopamudra Mukherjee, Vikas Singh, and Jiming Peng. Scale invariant cosegmentation for image groups. In *IEEE CVPR*, pages 1881–1888. IEEE, 2011.

[35] Lopamudra Mukherjee, Vikas Singh, Jia Xu, and Maxwell D Collins. Analyzing the subspace structure of related images: concurrent segmentation of image sets. In *Computer Vision–ECCV 2012*, pages 128–142. Springer, 2012.

[36] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. Diffusion curves: A vector representation

for smooth-shaded images. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008)*, volume 27, 2008.

[37] Ochs Peter and Brox Thomas. Higher order motion models and spectral clustering. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 614–621. IEEE, 2012.

[38] P.Ochs and T.Brox. Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[39] Brian Price and William Barrett. Object-based vectorization for interactive image editing. *The Visual Computer*, 22(9-11):661–670, 2006.

[40] Carsten Rother, Vladimir Kolmogorov, Victor Lempitsky, and Martin Szummer. Optimizing binary mrfs via extended roof duality. In *IEEE CVPR*, pages 1–8. IEEE, 2007.

[41] Carsten Rother, Tom Minka, Andrew Blake, and Vladimir Kolmogorov. Cosegmentation of image pairs by histogram matching-incorporating a global constraint into mrfs. In *IEEE CVPR*, volume 1, pages 993–1000. IEEE, 2006.

[42] Jose C Rubio, Joan Serrat, and Antonio López. Video co-segmentation. In *ACCV*, 2012.

[43] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. PAMI*, 22(8):888–905, 2000.

[44] Oana Sidi, Oliver van Kaick, Yanir Kleiman, Hao Zhang, and Daniel Cohen-Or. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. In *ACM Trans. Graphics*, volume 30, page 126. ACM, 2011.

[45] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE, 2003.

[46] Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum. Image vectorization using optimized gradient meshes. In *ACM Transactions on Graphics (TOG)*, volume 26, page 11. ACM, 2007.

[47] Xin Sun, Guofu Xie, Yue Dong, Stephen Lin, Weiwei Xu, Wencheng Wang, Xin Tong, and Baining Guo. Diffusion curve textures for resolution independent texture mapping. *ACM Trans. Graph.*, 31(4):74, 2012.

[48] Daniel Sỳkora, Jan Buriánek, and Jiří Žára. Sketching cartoons by example. In *Proceedings of Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 27–34, 2005.

[49] Daniel Sỳkora, Jan Buriánek, and Jiří Žára. Video codec for classical cartoon animations with hardware accelerated playback. In *Advances in Visual Computing*, pages 43–50. Springer, 2005.

[50] Ruo-Feng Tong, Yun Zhang, and Meng Ding. Video brush: A novel interface for efficient video cutout. In *Computer Graphics Forum*, volume 30, pages 2049–2057. Wiley Online Library, 2011.

[51] René Vidal. A tutorial on subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2010.

[52] René Vidal and Yi Ma. A unified algebraic approach to 2-d and 3-d motion segmentation. In *Proc. ECCV*, pages 1–15. Springer, 2004.

[53] Chuan Wang, Yanwen Guo, Jie Zhu, Linbo Wang, and Wenping Wang. Video object co-segmentation via subspace clustering and quadratic pseudo-boolean optimization in an mrf framework. *IEEE Transactions on Multimedia*, pages –, 2014.

[54] Jue Wang, Pravin Bhat, R Alex Colburn, Maneesh Agrawala, and Michael F Cohen. Interactive video cutout. In *ACM Trans. Graphics*, volume 24, pages 585–594. ACM, 2005.

[55] Lvdi Wang, Yizhou Yu, Kun Zhou, and Baining Guo. Multiscale vector volumes. *ACM Transactions on Graphics (TOG)*, 30(6):167, 2011.

[56] Lvdi Wang, Kun Zhou, Yizhou Yu, and Baining Guo. Vector solid textures. *ACM Transactions on Graphics (TOG)*, 29(4):86, 2010.

[57] John Winn, Antonio Criminisi, and Thomas Minka. Object categorization by learned universal visual dictionary. In *IEEE ICCV*, volume 2, pages 1800–1807. IEEE, 2005.

[58] Tian Xia, Binbin Liao, and Yizhou Yu. Patch-based image vectorization with automatic curvilinear feature alignment. In *ACM Transactions on Graphics (TOG)*, volume 28, page 115. ACM, 2009.

[59] Guofu Xie, Xin Sun, Xin Tong, and Derek Nowrouzezahrai. Hierarchical diffusion curves for accurate automatic image vectorization. *ACM Trans. Graph.*, 33(6):230:1–230:11, 2014.

[60] Chenliang Xu, Caiming Xiong, and Jason J Corso. Streaming hierarchical video segmentation. In *Proc. ECCV*, pages 626–639. Springer, 2012.

[61] Song-Hai Zhang, Tao Chen, Yi-Fei Zhang, Shi-Min Hu, and Ralph R Martin. Vectorizing cartoon animations. *Visualization and Computer Graphics, IEEE Transactions on*, 15(4):618–629, 2009.

[62] Fan Zhong, Xueying Qin, Qunsheng Peng, and Xiangxu Meng. Discontinuity-aware video object cutout. *ACM Trans. Graphics (SIGGRAPH Asia)*, 31(6):175:1–175:10, 2012.

[63] Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Large mesh deformation using the volumetric graph laplacian. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 496–503. ACM, 2005.

[64] Ju Jia Zou and Hong Yan. Cartoon image vectorization based on shape subdivision. In *Computer Graphics International 2001. Proceedings*, pages 225–231. IEEE, 2001.