Scale-Adaptive Video Understanding

by

Chenliang Xu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2016

Doctoral Committee:

Associate Professor Jason J. Corso, Chair
Assistant Professor Jia Deng
Professor Irfan Essa, Georgia Institute of Technology
Assistant Professor Matthew K. Johnson-Roberson
Professor Benjamin Kuipers

For my family and friends

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Professor Jason J. Corso, for all the advices and support over the past five years. I could not have imagined having a better advisor for my Ph.D. study. I would also like to thank Professor Irfan Essa, Professor Benjamin Kuipers, Professor Jia Deng and Professor Matthew Johnson-Roberson for their roles on my thesis committee.

My sincere thanks also goes to Dr. Manmohan Chandraker, who has provided numerous guidance and mentorship during my internship and beyond.

I would like to thank my fellow lab members for their help in the past: Albert Y. C. Chen, Jeff Delmerico, Caiming Xiong, Pradipto Das, Ran Xu, Wei Chen, Gang Chen, Suren Kumar, David Johnson, Vikas Dhiman, Richard F. Doell, Shao-Hang Hsieh, Parker Koch, Ryan Szeto and Theodore Nowak.

Finally, I would like to thank my wife, Yankun, my family and my many friends for their support, encouragement, and company during these years.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Scale-Adaptive Video Understanding

by

Chenliang Xu

Chair: Jason J. Corso

The recent rise of large-scale, diverse video data has urged a new era of high-level video understanding. It is increasingly critical for intelligent systems to extract semantics from videos. In this dissertation, we explore the use of supervoxel hierarchies as a type of video representation for high-level video understanding. The supervoxel hierarchies contain rich multiscale decompositions of video content, where various structures can be found at various levels. However, no single level of scale contains all the desired structures we need. It is essential to adaptively choose the scales for subsequent video analysis. Thus, we present a set of tools to manipulate scales in supervoxel hierarchies including both scale generation and scale selection methods.

In our scale generation work, we evaluate a set of seven supervoxel methods in the context of what we consider to be a good supervoxel for video representation. We address a key limitation that has traditionally prevented supervoxel scale generation on long videos. We do so by proposing an approximation framework for streaming hierarchical scale generation that is able to generate multiscale decompositions for arbitrarily-long videos using constant memory.

Subsequently, we present two scale selection methods that are able to adaptively choose the scales according to application needs. The first method flattens the entire supervoxel hierarchy into a single segmentation that overcomes the limitation induced by trivial selection of a single scale. We show that the selection can be driven by various post hoc feature criteria. The second scale selection method combines the supervoxel hierarchy with a conditional random field for the task of labeling actors and actions in videos. We formulate the scale selection problem and the video labeling problem in a joint framework. Experiments on a novel large-scale video dataset demonstrate the effectiveness of the explicit consideration of scale selection in video understanding.

Aside from the computational methods, we present a visual psychophysical study to quantify how well the actor and action semantics in high-level video understanding are retained in supervoxel hierarchies. The ultimate findings suggest that some semantics are well-retained in the supervoxel hierarchies and can be used for further video analysis.

CHAPTER I

Introduction

1.1 Motivation

Recent advances in technology and rapid growth of consumer electronics have made video capturing ever easier and more diverse than before. In addition to traditional devices, such as surveillance cameras and camcorders, modern devices such as smartphones, action cameras (e.g. GoPro Hero), wearable devices (e.g. Google Glass), dashboard cameras and even drones (e.g. DJI Phantom) are also capturing videos. The cost of a mobile consumer video recording device is also negligible compared to what it was ten years ago, which makes video recording accessible to almost everyone on the earth. For example, the number of smartphone subscriptions alone has reached 3.2 billion worldwide in 2015 and is anticipated to reach 6.3 billion in 2021 according to the annual Mobility Report from Ericsson [123].

In addition to this ease in capturing video, the sharing of video content is becoming more straightforward and popular. For example, YouTube gets more than 400 hours of uploaded video content every minute as of July 2015[1]. The Ericsson report [123] again shows that video content currently accounts for 41% - 55% of mobile data traffic, and that it will account for over two-thirds of mobile data traffic by 2021.

[1]Industry keynote speech by Susan Wojcicki at VidCon 2015, `https://www.youtube.com/watch?v=O6JPxCBlBh8`.

This tremendous growth of video data has urged a new era of video understanding, which requires an intelligent system to understand the underlying semantics in videos that are consistent with human perceptions. Not only is it needed for off-line systems that index and retrieve videos, but also for real-time interactive systems (agents) that communicate with humans and the environment.

However, comprehensive video understanding is complicated and requires both in-video evidence and out-of-video knowledge. Figure 1.1 shows an example of human descriptions of videos from YouTube. Human descriptions not only include common focuses such as actions (e.g. *crawling*, *walking* and *flying*), actors performing actions (e.g. *baby*, *dog* and *cat*) and their spatiotemporal relationships (e.g. *along*, *next to* and *toward*), but also richer information such as location (e.g. *floor*, *beach* and *sky*), attributes (e.g. *carpeted*, *little* and *in red*), inferred relationships (e.g. *his dog and his cat*) and fine-grained categories (e.g. *seagulls*). At the same time, video content poses many challenges: the video recording is often subject to irregular motion and changes in lighting; objects and actions can appear at various spatiotemporal locations and scales; and the dynamics of actions and interactions are computationally demanding to model.

Clearly, video understanding has a huge assortment of problems that need to be addressed. This dissertation presents a small but steady step towards achieving the goal of comprehensive video understanding. We study the basics of video representation, and in particular, we focus on supervoxel hierarchies (see Sec. 1.1.1 for the definition of supervoxels). We motivate the task of scale selection in Sec. 1.1.2. We propose tools for both scale generation and scale selection. Our efforts lead to improved performance in joint modeling of actors, actions and their spatiotemporal locations in video understanding.

1. A dog and a male baby crawl across the floor.
2. A little boy is crawling on the ground along with a dog.
3. A baby and a brown dog are crawling across a carpeted floor.
4. A small child and a dog are crawling along the floor.
5. A dog and a kid crawl on a green floor.

(a)

1. A man is walking with his dog and cat on the beach.
2. A man in red is walking on a sandy beach with a cat and a white dog.
3. A man is walking his dog and his cat across the sand.
4. A man walks a dog and a cat walks alongside them.
5. A man walks a white dog next to a cat on the beach.

(b)

1. One baby stands near a fence playing with a green toy, while another baby walks toward the fence.
2. A little boy is putting a toy into his mouth while another boy is walking towards him.
3. A baby is walking with his hands up towards a gate and another baby who is teething something.
4. Two babies are standing and one is eating something green.
5. Two babies, one with a brown suit walk near a fence.

(c)

1. A flock of seagulls flying in a heavy wind on a cloudy day.
2. A group of seabirds are flying together under the blue and cloudy sky.
3. A flock of birds is flying through a cloudy sky.
4. Many birds are flying in the sky.
5. Seagulls fly on a partly cloudy day.

(d)

Figure 1.1: Example of human descriptions of YouTube videos. Each video has five one-sentence descriptions written independently by human annotators.

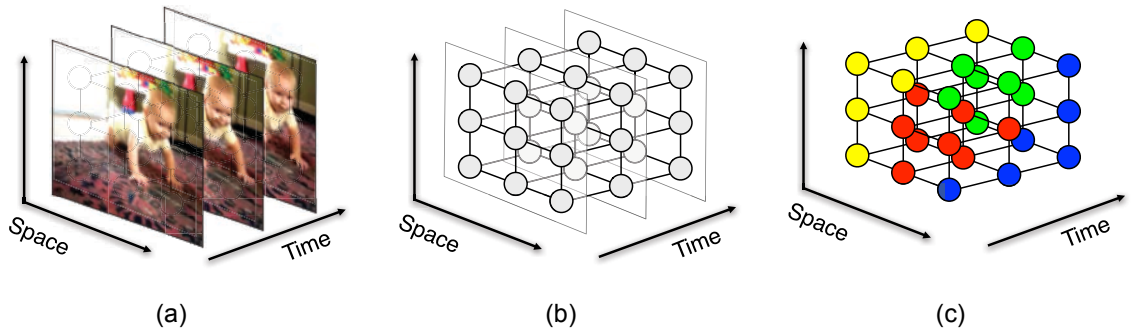


Figure 1.2: Illustration of supervoxel segmentation. (a) A video contains a sequence of frames. (b) A 3D lattice is built with nodes as voxels and edges linking voxels within a frame and across frames. (c) Voxels are being segmented into four groups, i.e. supervoxels, denoted by distinct colors.

1.1.1 Space-Time Groupings of Voxels

Images have many pixels; videos have more. It has thus become standard practice to first preprocess images and videos into more tractable sets by either extracting salient points [165] or oversegmenting the image into superpixels [158]. They are more perceptually meaningful than raw pixels, which are merely a consequence of digital sampling [158].

In this dissertation, we explore a mid-level video representation—the *supervoxel segmentation* (see Chapter II for a survey of video representations). We believe that supervoxel segmentation has great potential to benefit early video analysis. We define a *supervoxel* as the video analog to a superpixel. Concretely, given a 3D lattice Λ^3 on all voxels[2], a supervoxel v is a subset of the lattice $v \subset \Lambda^3$ such that the union of all supervoxels comprises the lattice and they are pairwise disjoint: $\bigcup_i v_i = \Lambda^3 \wedge v_i \bigcap v_j = \varnothing \; \forall i, j$ pairs. Figure. 1.2 illustrates this concept.

Different supervoxel algorithms treat space and time differently in a video and apply various grouping strategies. There are many plausible supervoxel methods and yet, the literature contains little understanding as to when and where each is most

[2]The video analog to pixels.

appropriate. Indeed, aside from our work [214, 216], we are not aware of a single comparative study on supervoxel segmentation. Thus, we explore the set of criteria that makes a good supervoxel algorithm and how well do the existing algorithms perform. Furthermore, we quantify their performance in a recognition task as a proxy for subsequent high-level use of supervoxels.

Although videos are orders of magnitude bigger than images, most methods in literature require all voxels in a video to be loaded into memory before computing supervoxels, which is clearly prohibitive for even medium length videos. For example, for a 480p video running 20 seconds at a typical frame rate of 30 frames-per-second, the resultant number of pixels is over 180 million, which requires large amount of memory to load and process the video. Furthermore, in many applications such as surveillance and interaction, it is impractical to expect the whole video to be available. To address these limitations, data stream algorithms provide a possible solution: each video frame is processed only once and does not change the segmentation of previous frames. Therefore, we investigate such possibilities in an approximation framework for streaming hierarchical video segmentation. Notice that there are a few methods [145, 65] in literature that process a video in a streaming fashion, but none of them is able to maintain a hierarchy.

1.1.2 Scale Selection in Supervoxel Hierarchies

Supervoxels are typically computed by unsupervised grouping processes. No matter if the algorithm is hierarchical or flat, it can output multiple segmentations for an input video and they are spread in a fine-to-coarse scale space, where coarser levels/scales contain larger supervoxels (w.r.t. the number of voxels in the grouping) and finer levels contain more fragmented supervoxels, and various structures can be found at various levels (see Fig. 1.3). The various scales of segmentation together define a supervoxel hierarchy. However, no single level in the hierarchy contains all the

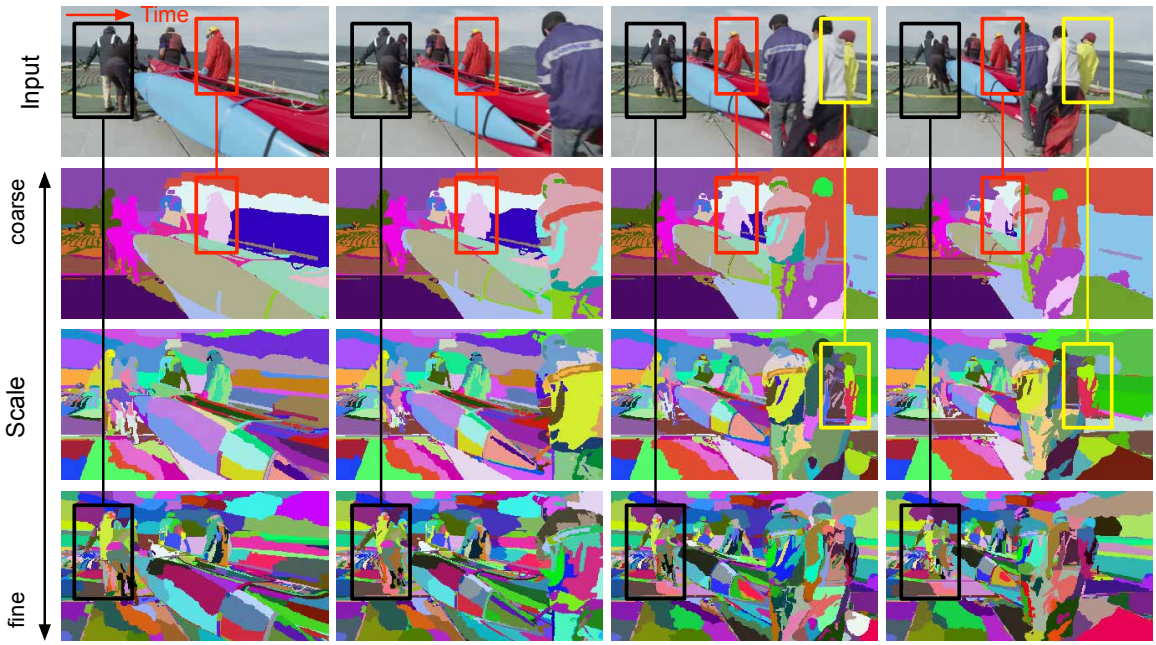


Figure 1.3: An input video and its three sampled levels from the supervoxel hierarchy. The paired boxes denote objects and their appropriate levels of scale, where going up causes undersegmentation (i.e. merge with other objects), and going down causes oversegmentation (i.e. have too many fragmented segments).

desired structures we need—it is susceptible to undersegmentation at coarser levels and oversegmentation at finer levels for different video parts. To obtain a good segmentation for a given task, we need to select and combine supervoxels from different scales.

Indeed, we are unaware of a principled approach to make use of this rich information in the hierarchies. Despite a few methods [77, 93] that do operations on hierarchies (see Chapter II for detail), most methods that use supervoxels usually pick a single level of scale [190, 120]. We believe that ignoring the intrinsic **multiscale** nature of video content is a key limitation to the use of supervoxels in video analysis.

If we look at the history of computer vision, scale is a fundamental problem and has been studied in various aspects, from features to systems. In fact, it dates

back to the 80s, where Witkin [211] discovered that a one-dimensional signal could be unambiguously segmented into a hierarchy of multiscale regions. However, it does not easily transfer to higher-dimensional data. Built upon Lindeberg's scale-space theory [110], Lowe [121] created the scale-invariant SIFT feature, where scale is handled in two ways: by sampling different image resolutions and by sampling Gaussian kernels with different variances. In a recent object detection system [46], scale is handled by sliding windows over different image resolutions.

We study the scale selection problem in supervoxel hierarchies. First, we are interested in exploring the possibilities of driving the scale selection attention by different post hoc feature criteria. The feature criteria provide a doorway to apply situation-specific guidance post hoc, and they span the spectrum of unsupervised, such as video motion, to class-specific supervised, such as human-ness and car-ness from raw detector responses. We would like to know how different selections are computed with criterion-specific foci of attention.

Second, we explore scale selection in a particular application, i.e. video labeling of actors and actions. Here, we aim to label each pixel in a video with a pair of actor and action labels. Existing methods based on conditional random fields (CRFs) for image labeling either exhaustively optimize over an entire hierarchy [92, 103], or apply Gaussian kernels for approximation [88]. It is computationally infeasible and also suboptimal to model the dynamics of actions when extending these methods to videos. Instead, we formulate the scale selection and video labeling in a joint framework, where CRFs are built dynamically for video labeling and the same constraint in our first method is used to select supervoxels. The new model has the ability to adaptively add long-ranging interactions to infer pixel labels based on supervoxel evidences from various levels in a hierarchy.

1.2 Thesis Statement

Supervoxel hierarchies provide rich multiscale decompositions of video content. No single level in the scale space contains all desired structures. We can improve performance in post hoc segmentation and video labeling tasks by adaptively choosing supervoxels from various scales in the hierarchy.

1.3 Overview

In this section, we present an overview of the dissertation by summarizing the key contributions.

1.3.1 Contribution 1: Library and Benchmark for Scale Generation

As motivated in Sec. 1.1.1, there are many plausible supervoxel scale generation algorithms and little understanding as to when and where each is most appropriate. To that end, in Chapter III, we study seven supervoxel algorithms, including both off-line and streaming methods, in the context of what we consider to be a good supervoxel: namely, spatiotemporal uniformity, object/region boundary detection, region compression and parsimony. For the evaluation we propose a comprehensive suite of seven quality metrics to measure these desirable supervoxel characteristics. In addition, we evaluate the methods in a road scene labeling task as a proxy for subsequent high-level uses of the supervoxels in video analysis. We use six existing benchmark video datasets with a variety of content-types and dense human annotations. Our findings lead us to conclusive evidence that the hierarchical graph-based (GBH), segmentation by weighted aggregation (SWA) and temporal superpixels (TSP) methods are the top-performers among the seven methods. They all perform well in terms of segmentation accuracy, but vary in regard to the other desiderata: GBH captures object boundaries best; SWA has the best potential for region compression; and TSP

achieves the best undersegmentation error.

We release our implementation of the supervoxel methods, as well as the evaluation benchmark, to the public[3]. Since our initial release in 2012, the LIBSVX benchmark has been widely used in evaluation of supervoxel methods by the computer vision community, including but not limited to [221, 220, 23, 143, 160, 196, 100, 34, 192].

1.3.2 Contribution 2: A Streaming Hierarchical Framework for Scale Generation

The use of video segmentation as an early processing step in video analysis lags behind the use of image segmentation for image analysis, despite many available video segmentation methods. As we discovered in our evaluation work, a major reason for this lag is simply that videos are orders of magnitude bigger than images; yet most methods require all voxels in the video to be loaded into memory, which is clearly prohibitive for even medium length videos.

In Chapter IV, we address this limitation by proposing an approximation framework for streaming hierarchical video segmentation motivated by data stream algorithms: each video frame is processed only once and does not change the segmentation of previous frames. We implement the graph-based hierarchical segmentation method within our streaming framework. Notice that the method we proposed is the first and remains the only one as of the time of writing this dissertation that is able to generate a hierarchy of multiscale segmentations for arbitrarily-long videos with constant memory. We perform thorough experimental analyses on the LIBSVX benchmark. Our results indicate the graph-based streaming hierarchical method outperforms other streaming video segmentation methods and performs nearly as well as the full-video hierarchical graph-based method.

[3]Link to download: `http://www.supervoxel.com`.

1.3.3 Contribution 3: A Visual Psychophysical Study of Semantic Retention

In Chapter V, we present a visual psychophysical study of semantic retention in supervoxel hierarchies. To be specific, we conduct a systematic study of how well the actor and action semantics are retained in a video supervoxel segmentation. Our study has human observers watching supervoxel segmentation videos and trying to discriminate both the actor (human or animal) and the action (one of eight everyday actions). We gather and analyze a large set of 640 human perceptions over 96 videos at 3 different supervoxel scales. Our ultimate findings suggest that some semantics are well-retained in the video supervoxel hierarchies and can be used for further video analysis.

1.3.4 Contribution 4: Scale Selection by Post Hoc Guidance

As motivated in Sec. 1.1.2, supervoxel hierarchies contain rich multiscale information and we need to adaptively choose scales for different video parts for a given task.

In Chapter VI, we present the first method of scale selection in this dissertation. It flattens the entire hierarchy into a single segmentation that overcomes the limitation induced by a trivial single-level scale selection—undersegmentation at coarser levels and oversegmentation at finer levels. Our method, called the *uniform entropy slice*, seeks a selection of supervoxels that balances the relative level of information in the selected supervoxels based on some post hoc feature criterion. For example, with a criterion of object-ness, in regions nearby objects, our method prefers finer supervoxels to capture the local details, but in regions away from any objects it prefers coarser supervoxels. We explore different types of feature criteria that span the spectrum of unsupervised to class-specific supervised, where different post hoc criteria lead to different selection attentions. We formulate the uniform entropy slice as a binary

quadratic program and implement four different feature criteria, both unsupervised and supervised, to drive the flattening. Our experiments demonstrate both strong qualitative performance and superior quantitative performance to state of the art on benchmark internet videos.

1.3.5 Contribution 5: Joint Scale Selection and Video Labeling

In Chapter VII, we present our second method of scale selection. Existing methods based on CRFs for video labeling are local and unable to capture the long-ranging interactions of video parts. We propose a new model that combines the labeling CRF with scale selection in the supervoxel hierarchy, where supervoxels at various scales provide cues for possible groupings of nodes in the CRF to encourage adaptive and long-ranging interactions. The new model defines a dynamic and continuous process of information exchange: the CRF influences what supervoxels in the hierarchy are active, and these active supervoxels, in turn, affect the connectivity in the CRF; we hence call it a *grouping process model*.

Experiments are conducted on the novel task of video labeling of actors and actions. We show that, by further incorporating the video-level evidences, the proposed method achieves a large margin of 60% relative improvement over state of the art.

1.3.6 Contribution 6: A New Problem of Actor-Action Video Understanding

Video labeling of actors and actions is a part of our actor-action video understanding work, which is described in the beginning of Chapter VII. It advances the line of research in action understanding by jointly considering various types of actors undergoing various actions. Indeed, there is no earlier work we know of on simultaneously inferring actors and actions in the video, not to mention a dataset on which to experiment. To start, we collect a dataset of 3782 videos from YouTube and label

both actors and actions at the pixel-level in each video. We formulate the general actor-action understanding problem and instantiate it at various granularities: both video-level single- and multiple-label actor-action recognition and pixel-level actor-action semantic segmentation. Our experiments demonstrate that joint inference over actors and actions outperforms independent inference over them, and concludes our argument of the value of explicit consideration of various actors in comprehensive action understanding.

1.4 Relevant Publications

This dissertation is based on the following publications:

1. Chenliang Xu and Jason J Corso. Evaluation of super-voxel methods for early video processing. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

2. Chenliang Xu, Caiming Xiong, and Jason J. Corso. Streaming hierarchical video segmentation. In *European Conference on Computer Vision*, 2012.

3. Chenliang Xu, Richard F. Doell, Stephen José Hanson, Catherine Hanson, and Jason J. Corso. Are actor and action semantics retained in video supervoxel segmentation? In *IEEE International Conference on Semantic Computing*, 2013.

4. Chenliang Xu, Richard F. Doell, Stephen José Hanson, Catherine Hanson, and Jason J. Corso. A study of actor and action semantic retention in video supervoxel segmentation. *International Journal of Semantic Computing*, 07(04):353–375, 2013.

5. Chenliang Xu, Spencer Whitt, and Jason J. Corso. Flattening supervoxel hierarchies by the uniform entropy slice. In *IEEE International Conference on Computer Vision*, 2013.

6. Chenliang Xu, Shao-Hang Hsieh, Caiming Xiong, and Jason J. Corso. Can humans fly? action understanding with multiple classes of actors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

7. Chenliang Xu and Jason J. Corso. Actor-action semantic segmentation with grouping process models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

8. Chenliang Xu and Jason J. Corso. Libsvx: A supervoxel library and benchmark for early video processing. *International Journal of Computer Vision*, 2016.

CHAPTER II

Related Work

In this chapter, we survey the work related to the complete dissertation. In future chapters, we include any work that is exclusive to the content of that chapter.

2.1 Video Representation

Comparing to images, videos contain orders of magnitude more pixels and they describe complex spatiotemporal events. Thus, it is important and non-trivial to design compact yet expressive representations for video. In this section, we discuss a few generic video representations that are related to this dissertation. They cover a wide spectrum, from key points [98] to templates [163], and from hand-crafted [203] to deeply-learned [175, 191].

Notice that these representations are often complementary rather than exclusive to each other. For example, the improved dense trajectories [204] are used in [208] to pool the deep convolutional features into more effective descriptors. Interest points and trajectories are used in our work [219, 215] as features to describe the spatiotemporal video regions of supervoxels. The use of these representations are application-oriented. Here, we try to provide a general view of these video representations without going into one specific application.

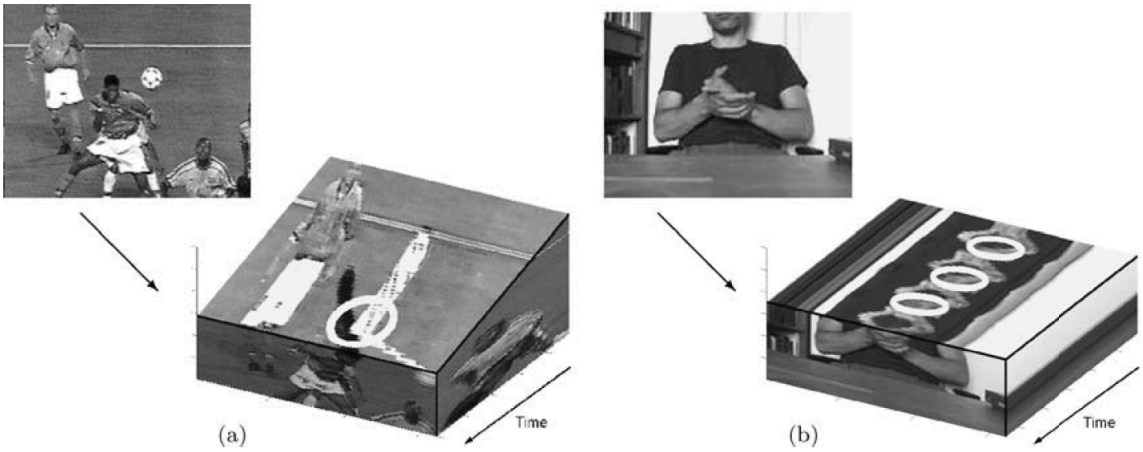


Figure 2.1: Results of detecting the strongest space-time interest points in a football sequence (a) and in a hand clapping sequence (b). The interest points correspond to space-time corners. Figure from [98].

2.1.1 Interest Points

Interest points are used to capture significant local variations. In images, they are extensively studied [49, 69, 111, 166]. They are relatively stable with respect to perspective transformation, and are used in various image analysis tasks such as stereo matching [194], image indexing [165] and object recognition [121].

Laptev [98] extends the Harris detector [69] from the 2D spatial domain to the 3D spatiotemporal domain, where interest points are required to have large variations along both spatial and temporal directions in a local 3D video volume. Such interest points are called Space-Time Interest Points (STIPs). Figure 2.1 shows this concept. Combined with histograms of oriented gradients (HOG) and histograms of optical flow (HOF) descriptors in a bag-of-feature setting, Laptev et al. [99] train SVM classifiers on STIPs to classify human actions in videos.

Other methods that extract interest points in video include but are not limited to [40, 14, 210, 213]. Furthermore, various descriptors [167, 168, 85, 224] are also explored as features to describe the spatiotemporal interest points.

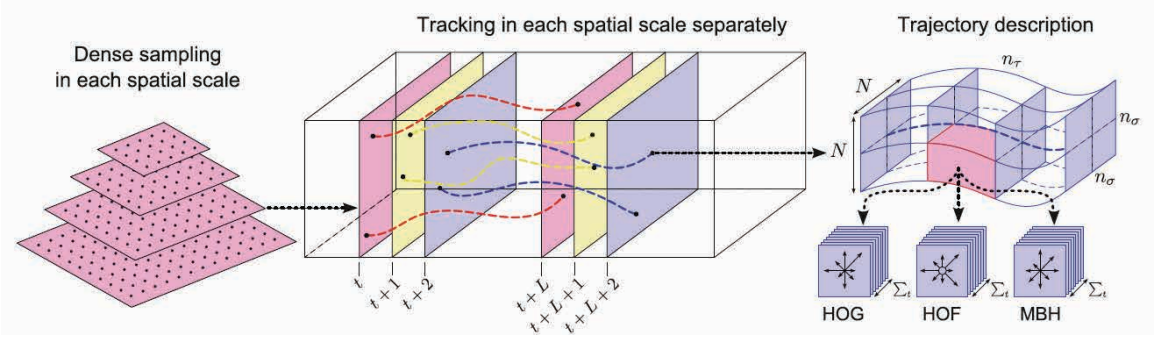


Figure 2.2: An overview of the process to extract dense trajectories. Figure from [203].

2.1.2 Trajectories

Instead of modeling a video as a 3D volume, another approach is to track interest points throughout the video. Methods applying this approach include but are not limited to [131, 129, 180, 179, 154, 203]. Among all these methods, Wang et al. [203] propose a way to extract dense trajectories, which has been shown to be the most effective in terms of modeling human actions.

We briefly describe the process of extracting dense trajectories in [203]. First, dense feature points are sampled on a grid for each spatial scale of a frame. Then these points are tracked by median filtering in a dense optical flow field at their corresponding scales to form trajectories. All the trajectories are forced to have a fixed length of 15 frames. Finally, feature descriptors such as HOG, HOF and motion boundary histograms (MBH), along the space-time location of a trajectory are extracted. Figure 2.2 depicts this process.

When put into the same framework as Laptev et al. [99], dense trajectories have shown strong performance in terms of classifying human actions and events in videos. The performance can be further improved by adding a camera motion model and human detector responses [204], or by using fisher vectors to encode bag-of-features [149].

2.1.3 Supervoxels

Segmentation is a fundamental problem in computer vision. We find the following definition of segmentation from the popular computer vision textbook [50].

> *"The core idea is collecting together pixels or pattern elements into summary representations that emphasize important, interesting, or distinctive properties."* — *Forsyth and Ponce*

Supervoxels are the oversegmentation of video voxels. Each supervoxel defines a spatiotemporal region in video that has roughly coherent color, texture and motion. There are many supervoxel methods, but they mainly fall into two categories according to the way they model space and time in video. The first set of methods [31, 65, 146, 51] model video as a 3D volume where clustering or partitioning is conducted simultaneously in space and time. The second set of methods [23, 196, 160] conduct clustering or partitioning first in space, usually on the first frame, then track or propagate the existing groupings to future frames. We survey methods from both worlds in Sec. 2.2.1.

Supervoxels are used in various video analysis tasks. Tighe and Lazebnik [190] use supervoxels for semantic road scene labeling in driving videos. Raza et al. [155] train geometric context classifiers by exploring various motion and appearance features grouped by supervoxels. Tang et al. [185] propose a weakly supervised approach to assign supervoxels with object labels in video. Moreover, supervoxels have the ability to localize objects [141] and actions [79]. They have also been used as higher-order potentials for human action segmentation [122] and video object segmentation [80].

2.1.4 Patches and Templates

Other than by points, trajectories, or supervoxels, videos can also be represented by mid-level patches and high-level templates.

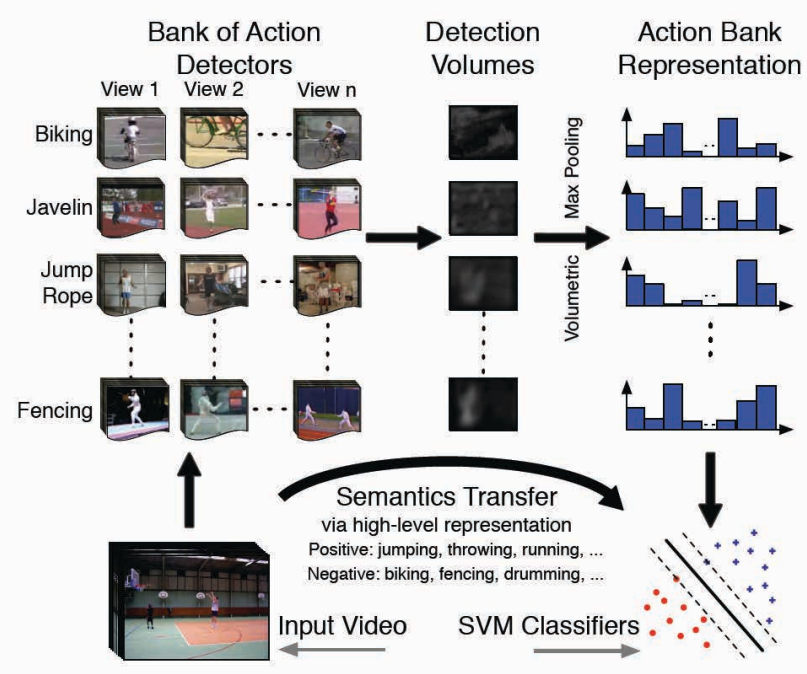


Figure 2.3: An overview of the Action Bank method. It is a high-level template-based video representation. Figure from [163].

Mid-Level Patches. Among all the methods, Zhu et al. [231] propose a mid-level action representation, called acton, that is built on top of low-level descriptors to represent a volume of interest. These actons are discovered in a weakly-supervised manner. Jain et al. [78] propose an exemplar-based clustering approach to mine discriminative and representative spatiotemporal patches. Lan et al. [97] represent videos by a hierarchy of mid-level action elements. Each action element corresponds to an action-related spatiotemporal segment discovered by a discriminative clustering algorithm. Similar methods aligned with this concept are [124, 125].

High-Level Templates. Action bank [163] defines a unique high-level template-based video representation. It contains a large set of templated-based action detectors. For a given video, correlations are computed over each detector at multiple scales and a feature vector is extracted by volumetric max-pooling. To build the bank, they manually sample action templates over a large set of action classes (50 action

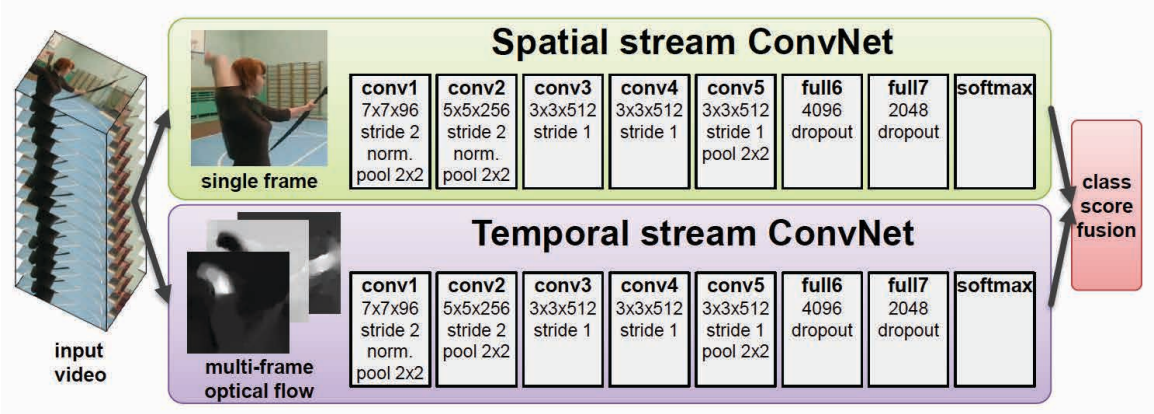


Figure 2.4: An overview of the two-stream ConvNet architecture. Figure from [175].

classes from UCF50 [156] and six action classes from KTH [167]). Within one action class, they sample three to six action templates to capture the variations (e.g. viewpoints, temporal scales). Therefore, it results in a total 205 action templates. These templates are used to construct detectors using the action spotting framework [39]. Figure 2.3 shows an overview of the process.

2.1.5 Deep Representations

Recently deep Convolutional Networks (ConvNets) have attracted much attention due to the availability of large-scale training set and the advances in GPU computation power. They achieve superior performance on image recognition tasks [89, 176, 182]. The computer vision community is also pushing to adapt ConvNets to model videos.

Simonyan and Zisserman [175] propose a two-stream ConvNet architecture, where a spatial stream ConvNet operates on individual video frames and a temporal stream ConvNet operates on multi-frame optical flow fields. Figure 2.4 depicts their architecture. Another set of works include Ji et al. [82] and Tran et al. [191], where they apply 3D convolution on a video volume. In addition, some other works use LSTM [72] to model the information in temporal domain of a video.

2.1.6 Summary

We have discussed various video representations based on interest points, trajectories, supervoxels, patches and templates, and neural networks. Among all the representations, there are, in general, two approaches to model space-time in a video. Methods such as STIPs [98], graph-based supervoxels [31, 65], and 3D ConvNets [82, 191] consider video as a 3D volume where time is modeled as a third dimension in addition to space. The other set of methods such as dense trajectories [203] and temporal superpixels [23] treat space and time differently, and often track interest points and regions over time. In our work, we study supervoxels generated by methods from both perspectives.

The various representations are complementary rather than exclusive to each other, e.g. dense trajectories can be used to pool convolutional features [208] or compute supervoxels that are consistent with long-term object motion [106]. In our work, we use supervoxels as the basic blocks for video labeling, where we also compute point and trajectory features and use them to describe supervoxels.

2.2 Video Segmentation

In this section, we discuss works in the broader domain of video segmentation, including region segmentation, object segmentation and semantic segmentation. Notice that all works we discuss here are spatiotemporal segmentation rather than temporal only segmentation [147].

2.2.1 Region Segmentation

Region segmentation aims to decompose a video into spatiotemporal regions that are consistent with human perceptions. The term *video segmentation* often refers to region segmentation. Supervoxel segmentation is a type of region segmentation with

a focus on oversegmenting a video into spatiotemporal regions, such that regions not only are more consistent with human perceptions but also have coherent color, texture and motion. Methods in video segmentation can be used to generate supervoxels in a process of supervoxelization [58, 214], and supervoxel hierarchies can also provide video segmentations [220, 65]. Thus the boundary between supervoxel segmentation and video segmentation is not salient.

We study seven segmentation methods to generate supervoxels in Chapter III including mean shift [146], graph-based [47], hierarchical graph-based [65], streaming hierarchical graph-based [221], Nyström normalized cut [171, 52, 51], segmentation by weighted aggregation [169, 170, 31] and temporal superpixels [23]. These methods broadly sample the methodology space among statistical and graph partitioning methods [5].

Notice that many other methods have been proposed in the literature for region segmentation including but not limited to: computing watersheds [201], clustering via Gaussian mixture modeling [63], tracking regions or superpixels [15, 198, 56], propagating groupings in graph [119], formulating as energy minimization [199, 84, 19], modeling by trajectories [106, 143], and spectral clustering on superpixels [57].

An early survey of spatiotemporal grouping techniques is in [130]. A recent video segmentation evaluation benchmark is proposed in [58]. We propose and establish the evaluation benchmark for supervoxel segmentation in Chapter III.

2.2.2 Object Segmentation

Video object segmentation aims to separate the foreground moving objects and the background, which is analogous to figure-ground segmentation in the image domain except that videos contain rich motion information. Figure 2.5 shows an example output. Comparing to tracking, video object segmentation gives precise object shapes. Various methods have been proposed in literature. According to the amount

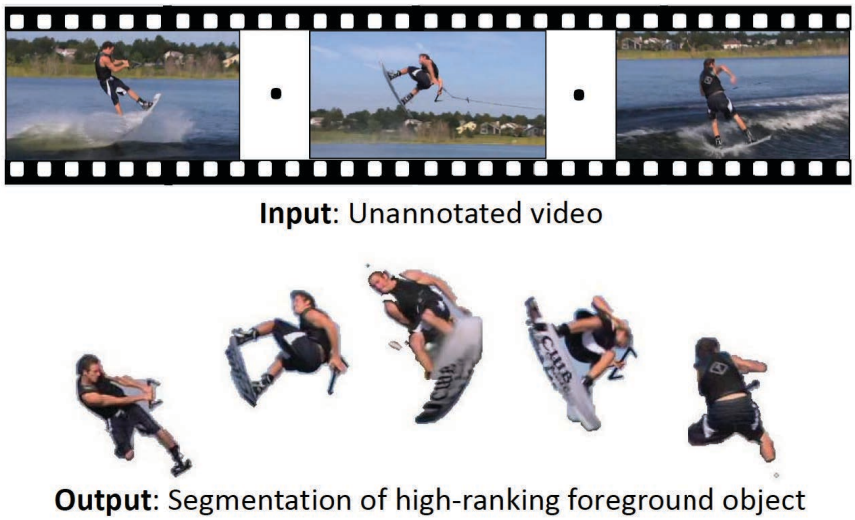


Figure 2.5: An example output of video object segmentation. Figure from [102].

of supervision they require, the methods can be organized into three categories.

Interactive or supervised. Several interactive methods, such as [7, 152, 225], require users to annotate object boundaries for a few frames. Then, they propagate the annotations to other frames in video. Another set of methods, such as [159, 28, 193, 153, 80, 197], require users to mark the object positions in the first frame, then track or propagate the annotated object throughout the whole video.

Weakly supervised. The weakly supervised methods [70, 185, 120, 230] typically require some video-level tags, then they can extract the video object segmentation along with object class information in large-scale video datasets. Methods such as [70, 185, 120] rely on a pre-processing of videos into supervoxels.

Unsupervised and fully automatic. Methods in this category generate object segmentations in an unsupervised and fully automatic fashion. They are most extensively studied in the literature due to the promising applications. Several methods [18, 140, 42, 137] explore the motion segmentation by grouping long-term motion trajectories into regions. Some other methods, such as [102, 126, 227, 144, 60] start with some object proposals, then evaluate over multiple cues from both appearance

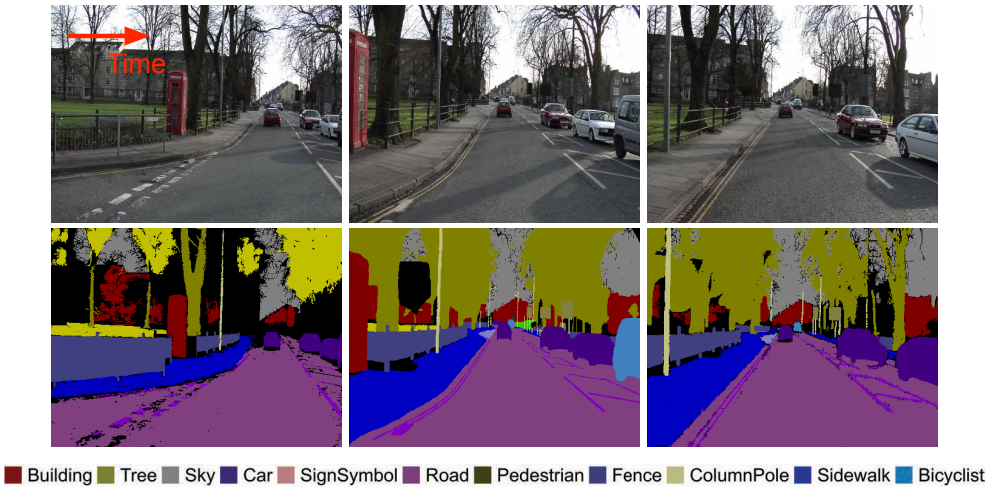


Figure 2.6: An example of the semantic video segmentation. Images are taken from the CamVid dataset [17].

and motion domains to get the final segmentation. They vary by the ways they formulate the problem, e.g. [126] model it as finding a maximum weight clique in a weighted region graph, and [227] use a layered directed acyclic graph. Another set of methods [54, 207, 228] explore this video object segmentation problem by formulating it as a co-segmentation of a group of similarly-recorded videos.

2.2.3 Semantic Segmentation

Given a video, semantic video segmentation aims to assign each pixel in video with a semantic label. For example, the set of semantic labels in a road scene setting can include both objects, e.g. *car*, *pedestrian* and *bicyclist*, and scene elements, e.g. *road*, *sidewalk*, *building* and *tree*. Figure 2.6 shows an example video of the popular CamVid driving dataset [17]. Semantic video segmentation is often referred to as video labeling as the majority methods formulate this problem as a labeling problem of conditional random field (CRF), where they differ by the way they obtain the unary and the way they construct the graph.

There are two lines of works. The first line of works is essentially image-based

and they do not model any spatiotemporal coherence. A number of methods have been proposed: [212, 94] model jointly the object reasoning and pixel labeling; [177] add features by structure from motion (SFM); [88] propose efficient inference for fully-connected CRF; [91] combine the pixel labeling with 3D reconstruction; and [45] train ConvNets to obtain initial superpixel labels.

The second line of works models the spatiotemporal coherence in the labeling field and they are considered as semantic video segmentation methods. Several methods have been proposed: [24] add dynamic temporal links to connect pixels in different frames in order to obtain consistent labeling over time; [48] add such links by SFM cues; [132] propagate labels to future frames by learning a similarity function between pixels; [190, 113, 77] build spatiotemporal CRF graphs using supervoxels; and [112] further model the object interactions on top of a supervoxel graph.

2.2.4 Summary

We have discussed various video segmentation tasks including region segmentation, object segmentation and semantic segmentation. They have different aims, e.g. object segmentation tries to extract foreground moving objects, and semantic segmentation aims to label pixels in the scene densely with a set of semantic labels. Supervoxel segmentation provides voxel groupings that can be used for both object segmentation [65, 80] and semantic segmentation [190, 155, 185].

In our work, we extend the scope of semantic segmentation to include both actors and actions, i.e. dog-running, cat-climbing and adult-eating are examples of the labels in our task. We leverage supervoxels to model various relationships among actors and actions as well as long-ranging interactions of different video parts. We do not rely on trained object trackers [30] to track objects over time, as supervoxels have already provided trajectories for video parts. Indeed, supervoxels are used in [26] to track objects, and Palou and Salembier [143] first track points into trajectories and then

group trajectories into supervoxels.

2.3 Scales in Supervoxel Hierarchy

In this section, we survey the work that uses the rich multiscale information contained in a supervoxel hierarchy. The concept of *scale* is important: objects can appear at arbitrary scales in an image, e.g. finding faces in surveillance images versus selfies; action can appear at different rates in a video, e.g. 100-meter performed by professionals versus non-professionals; and not to mention that the resolution of an image and the frame rate of a video can be easily manipulated. Indeed, the computing of scales has been heavily studied in various video representations (see Sec. 2.1). However, the use and manipulation of scales are relatively less explored, especially for supervoxel hierarchies. Most methods using supervoxels [190, 80, 113, 112] only use a single level from the hierarchy. There are only a few works in the literature that take the advantage of multiscale information in hierarchies in their modeling of image/video. They can be roughly organized into two categories according to the way they manipulate the hierarchies and we discuss them next.

The first line of works contains bottom-up methods. They either grow or prune parts in a hierarchy to propose desired regions for later processing. Uijlings et al. [195] propose image regions for object detection by examining various object-ness criteria in hierarchical clustering of image regions. Similarly, Oneata et al. [141] build a supervoxel hierarchy for spatiotemporal object detection proposals in video. Jain et al. [79] grow a hierarchy for action localization, and on the contrary, Ma et al. [125] prune a segmentation tree for the same task.

The second line of works considers the hierarchies in a graphical model energy minimization framework. The Pylon model [103] and associative hierarchical random fields [93] define a hierarchical cost function over superpixels at all levels. Jain et al. [77] propose an exact coarse-to-fine energy minimization strategy on the hierarchy

for semantic video segmentation. Lu et al. [122] consider a higher-order potential between supervoxels at different levels in the hierarchy. Ren et al. [157] combine segmentation tree path classification with a pairwise Markov random field. In addition, scales in a hierarchy can be picked during post-processing. In [45], each node is encoded by a spatial grid of feature vectors pooled in the image region and then classified to produce a histogram of categories. The final labeling is obtained by finding the optimal cover of the segmentation tree.

In our work, we propose two methods for scale selection that are different from the above two lines of research. We seek to locate a best interpretation of video content by selecting video parts from different scales in a hierarchy. We drive the scale selection by different post hoc feature criteria in the first method, and formulate a joint model of scale selection and video labeling in the second method.

CHAPTER III

Scale Generation I: Library and Benchmark

3.1 Introduction

Images have many pixels; videos have more. It has thus become standard practice to first preprocess images and videos into more tractable sets by either extraction of salient points [165] or oversegmentation into superpixels [158]. Preprocessing output—salient points or superpixels—is more perceptually meaningful than raw pixels, which are merely a consequence of digital sampling [158]. However, the same practice does not entirely exist in video analysis. Although many methods do indeed initially extract salient points or dense trajectories, e.g., [98], we are aware of few methods that rely on a supervoxel segmentation, which is the video analog to a superpixel segmentation. In fact, those papers that do preprocess video tend to rely on a per-frame superpixel segmentation, e.g., [102], or use a full-video segmentation, e.g., [65].

The basic position of this chapter is that supervoxels have great potential in advancing video analysis methods, as superpixels have for image analysis. To that end, we perform a thorough comparative evaluation of seven supervoxel methods: five off-line and two streaming methods. The off-line methods require the video to be available in advance and short enough to fit in memory. They load the whole video at once and process it afterwards. The five off-line methods we choose—segmentation by weighted

aggregation (SWA) [169, 170, 31], graph-based (GB) [47], hierarchical graph-based (GBH) [65], mean shift [146], and Nyström normalized cuts (NCut) [51, 171, 52]—broadly sample the methodology-space, and are intentionally selected to best analyze methods with differing qualities for supervoxel segmentation. For example, both SWA and NCut use the normalized cut criterion as the underlying objective function, but SWA minimizes it hierarchically whereas NCut does not. Similarly, there are two graph-based methods that optimize the same function, but one is subsequently hierarchical (GBH). We note that, of the off-line methods, only GBH had been proposed intrinsically as a supervoxel method; each other one is either sufficiently general to serve as one or has been adapted to serve as one. We also note a similar selection of segmentation methods has been used in the (2D) image boundary comparative study [5] and nonetheless our selections share a good overlap with the ones studied in the recent video segmentation benchmark [58].

In contrast, streaming methods require only constant memory (depends on the streaming window range) to execute the algorithm which makes them feasible for surveillance or to run over a long video on a less powerful machine. The two streaming methods we choose—streaming hierarchical video segmentation (streamGBH) [221] and temporal superpixels (TSP) [23] employ different strategies to treat video data. The streamGBH approximates a full video segmentation by both hierarchical and temporal Markov assumptions. Each time it segments video frames within a streaming window, and the length of the streaming window can be as short as one frame or as long as the full video, which equates it to full-video GBH segmentation. TSP represents a set of methods [23, 196, 160] that computes the superpixel segmentation on the first frame and then extends the superpixels to subsequent frames (one by one) in a video. The TSP method [23] uses a Gaussian Process for the streaming segmentation.

This chapter pits the five off-line and two streaming methods in an evaluation

on a suite of metrics designed to assess the methods on basic supervoxel desiderata (Sec. 3.2.2), such as following object boundaries and spatiotemporal coherence. The specific metrics we use are 3D undersegmentation error, 3D segmentation accuracy, boundary recall distance and label consistency. They evaluate the supervoxel segmentations against human annotations. We also use a set of human-independent metrics: explained variation, mean size variation and temporal extent of supervoxels, which directly explore the properties of each method. Finally, we compare the supervoxel methods in a particular application—supervoxel classification—that evaluates methods in a recognition task, which we consider to be a proxy to various high-level video analysis tasks in which supervoxels could be used. We use six complementary video datasets to facilitate the study: BuffaloXiph [25], SegTrack v2 [193, 107], CamVid [17], BVDS [181, 58], [115] and Middlebury Flow [8]. They span from few videos to one hundred videos, and from sparse annotations to dense frame-by-frame annotations.

A preliminary version of our work appears in [214]. Since its initial release, the LIBSVX benchmark has been widely used in supervoxel method evaluation by the community, including but not limited to [221, 220, 23, 143, 160, 196, 100, 34, 192]. In this chapter, we complement the library with the two streaming methods and a set of new benchmark metrics on new video datasets. In addition, we add a new experiment of supervoxel classification to evaluate methods in terms of a middle-level video representation towards high-level video analysis. We also note that a recent video segmentation evaluation is proposed in [58]. We distinguish our work from them by evaluating directly on supervoxel segmentation, the oversegmentation of a video, and using various datasets including densely labeled human annotations with a set of novel benchmark metrics. It is our position that evaluations of both over-segmentation and segmentation in video are necessary to establish a thorough understanding of the problem-space within the computer vision community.

Our evaluation yields conclusive evidence that GBH, SWA and TSP are the top-performers among the seven methods. They all perform well in terms of segmentation accuracy, but vary in regard to the other desiderata: GBH captures object boundaries best; SWA has the best potential for region compression; and TSP follows object parts and achieves the best undersegmentation error. Although GBH and SWA, the two offline methods, are quite distinct in formulation and may perform differently under other assumptions, we find a common feature among the two methods (and one that separates them from the others) is the manner in which coarse level features are incorporated into the hierarchical computation. TSP is the only streaming method among the three and generates supervoxels with the best spatiotemporal uniformity. Finally, the supervoxel classification experiment further supports our findings and shows a strong correlation to our benchmark evaluation.

The complete supervoxel library, benchmarking code, classification code and documentation are available for download at `http://www.supervoxels.com`. Various supervoxel results on major datasets in the community (including the existing six datasets [25, 193, 107, 17, 181, 58, 115, 8]) are also available at this location to allow for easy adoption of the supervoxel results by the community.

The rest of the chapter is organized as follows. We present a theoretical background in Sec. 3.2 and a brief description of the methods in Sec. 3.3. We introduce the datasets and processing setup in Sec. 3.4. We thoroughly discuss comparative performance in terms of benchmark in Sec. 3.5 and supervoxel classification in Sec. 3.6. Finally, we conclude this chapter in Sec. 5.5.

3.2 Background

3.2.1 Superpixels

The term *superpixel* was coined by Ren and Malik [158] in their work on learning a binary classifier that can segment natural images. The main rationale behind superpixel oversegmentation is twofold: (1) pixels are not natural elements but merely a consequence of the discrete sampling of the digital images and (2) the number of pixels is very high making optimization over sophisticated models intractable. Ren and Malik [158] use the normalized cut algorithm [171] for extracting the superpixels, with contour and texture cues incorporated. Subsequently, many superpixel methods have been proposed [105, 199, 133, 118, 226] or adopted as such [47, 201, 29] and used for a variety of applications: e.g., human pose estimation [134], semantic pixel labeling [71, 188], 3D reconstruction from a single image [73] and multiple-hypothesis video segmentation [198] to name a few. Few superpixel methods have been developed to perform well on video frames, such as [41] who base the method on minimum cost paths but do not incorporate any temporal information.

3.2.2 What makes a good supervoxel method?

First, we define a *supervoxel*—the video analog to a superpixel. Concretely, given a 3D lattice Λ^3 (the voxels in the video), a supervoxel v is a subset of the lattice $v \subset \Lambda^3$ such that the union of all supervoxels comprises the lattice and they are pairwise disjoint: $\bigcup_i v_i = \Lambda^3 \wedge v_i \bigcap v_j = \varnothing \; \forall i, j$ pairs. Obviously, various image/video features may be computed on the supervoxels, such as color histograms and textons. In this initial definition, there is no mention of certain desiderata that one may expect, such as locality, coherence, and compactness. Rather than include them in mathematical terms, we next list terms of this sort as desirable characteristics of a *good* supervoxel method.

We define a good supervoxel method based jointly on criteria for good supervoxels, which follow closely from the criteria for good segments [158], and the actual cost of generating them (videos have an order of magnitude more pixels over which to compute). Later, in our experimental evaluation, we propose a suite of benchmark metrics designed to evaluate these criteria (Section 3.5).

Spatiotemporal Uniformity. The basic property of spatiotemporal uniformity, or *conservatism* [133], encourages compact and uniformly shaped supervoxels in space-time [105]. This property embodies many of the basic Gestalt principles—proximity, continuation, closure, and symmetry—and helps simplify computation in later stages [158]. Furthermore, Veksler et al. [199] show that for the case of superpixels, compact segments perform better than those varying in size on the higher level task of salient object segmentation. For temporal uniformity (called coherence in [65]), we expect a mid-range compactness to be most appropriate for supervoxels (bigger than, say, five frames and less than the whole video).

Spatiotemporal Boundaries and Preservation. The supervoxel boundaries should align with object/region boundaries when they are present and the supervoxel boundaries should be stable when they are not present; i.e., the set of supervoxel boundaries is a superset of object/region boundaries. Similarly, every supervoxel should overlap with only one object [118]. Furthermore, the supervoxel boundaries should encourage a high-degree of *explained variation* [133] in the resulting oversegmentation. If we consider the oversegmentation by supervoxels as a compression method in which each supervoxel region is represented by the mean color, we expect the distance between the compressed and original video to have been minimized.

Computation. The computation cost of the supervoxel method should reduce the overall computation time required for the entire application in which the supervoxels are being used.

Performance. The oversegmentation into supervoxels should not reduce the

achievable performance of the application. Our evaluation will not directly evaluate this characteristic (because we study the more basic ones above).

Parsimony. The above properties should be maintained with as few supervoxels as possible [118].

3.3 Methods

We study seven supervoxel methods—mean shift [146], graph-based (GB) [47], hierarchical graph-based (GBH) [65], streaming hierarchical graph-based (streamGBH) [221], Nyström normalized cut (NCut) [171, 52, 51], segmentation by weighted aggregation (SWA) [169, 170, 31] and temporal superpixels [23]—that broadly sample the methodology-space among statistical and graph partitioning methods [5]. We have selected these seven due to their respective traits and their inter-relationships: for example, Nyström and SWA both optimize the same normalized cut criterion, and streamGBH extends GBH to handle arbitrarily long videos and still keeps the hierarchy property.

We describe the methods in some more detail below. We note that *many* other methods have been proposed in the computer vision literature for video segmentation, e.g., [201, 63, 15, 119, 198, 199, 84, 130, 19, 56], but we do not cover them in any detail in this study. We also do not cover strictly temporal segmentation, e.g. [147].

3.3.1 Mean Shift

Mean shift is a mode-seeking method, first proposed by [55]. [29] and [205] adapt the kernel to the local structure of the feature points, which is more computationally expensive but improves segmentation results. Original hierarchical mean shift in video [37, 145] improves the efficiency of (isotropic) mean-shift methods by using a streaming approach. The mean shift algorithm used in this chapter is presented by [146], who introduce Morse theory to interpret mean shift as a topological de-

composition of the feature space into density modes. A hierarchical segmentation is created by using topological persistence. Their algorithm is more efficient than previous works especially on videos and large images. We use the author-provided implementation[1] to generate a supervoxel hierarchy and then stratify the pairwise merging into a fixed-level of hierarchy.

3.3.2 Graph-Based (GB)

Felzenszwalb and Huttenlocher [47] propose a graph-based algorithm for image segmentation; it is arguably the most popular superpixel segmentation method. Their algorithm runs in time nearly linear in the number of image pixels, which makes it suitable for extension to spatiotemporal segmentation. Initially, each pixel, as a node, is placed in its own region R, connected with 8 neighbors. Edge weights measure the dissimilarity between nodes (e.g. color differences). They define the internal difference of a region, $Int(R)$, as the largest edge weight in the minimum spanning tree of R. Traversing the edges in a non-decreasing weight order, the regions R_i and R_j incident to the edge are merged if the current edge weight is less than the relaxed minimum internal difference of the two regions:

$$\min(Int(R_i) + \tau(R_i), Int(R_j) + \tau(R_j)) \ , \tag{3.1}$$

where $\tau(R) = k/|R|$ is used to trigger the algorithm and gradually makes it converge. k is a scale parameter that reflects the preferred region size. The algorithm also has an option to enforce a minimum region size by iteratively merging low-cost edges until all regions contain the minimum size of pixels. We have adapted the algorithm for video segmentation by building a 3D lattice over the spatiotemporal volume, in which voxels are nodes connected with 26 neighbors in the lattice (9 to the previous and the next frames, 8 to the current frame). One challenge in using this algorithm

[1] `http://people.csail.mit.edu/sparis/`

is the selection of an appropriate k for a given video, which the hierarchical extension (GBH, next) overcomes. We use a set of k as well as various minimum region sizes to generate the segmentation output for our experiment.

3.3.3 Hierarchical Graph-Based (GBH)

The hierarchical graph-based video segmentation algorithm is proposed by [65]. Their algorithm builds on an oversegmentation of the above spatiotemporal graph-based segmentation. It then iteratively constructs a region graph over the obtained segmentation, and forms a bottom-up hierarchical tree structure of the region (segmentation) graphs. Regions are described by local Lab histograms. At each step of the hierarchy, the edge weights are set to be the χ^2 distance between the Lab histograms of the connected two regions. They apply the same technique as above, Felzenszwalb and Huttenlocher [47], to merge regions. Each time they scale the minimum region size as well as k by a constant factor s. Their algorithm not only preserves the important region borders generated by the oversegmentation, but also allows a selection of the desired segmentation hierarchy level h, which is much better than directly manipulating k to control region size. We set a large h to output segmentations with various numbers of supervoxels.

3.3.4 Graph-Based Streaming Hierarchical (streamGBH)

Graph-based streaming hierarchical video segmentation is proposed in detail in Chapter IV and in our earlier work [221] to extend GBH [65] to handle arbitrarily long videos in a streaming fashion and still maintain the segmentation hierarchy. The algorithm approximates the full video GBH segmentations by both a hierarchical and a temporal Markov assumption, allowing a small number of frames to be loaded into a memory at any given time. Therefore the algorithm runs in a streaming fashion. In our comparison experiments, we set a fixed streaming window size (10 frames) for all

subsequences and, again, a large h as in GBH to output segmentations with various numbers of supervoxels.

3.3.5 Nyström Normalized Cut (NCut)

Nyström Normalized Cuts [171] as a graph partitioning criterion has been widely used in image segmentation. A multiple eigenvector version of normalized cuts is presented in [51]. Given a pairwise affinity matrix W, they compute the eigenvectors V and eigenvalues Γ of the system

$$(D^{-1/2}WD^{-1/2})V = V\Gamma \ , \tag{3.2}$$

where D is a diagonal matrix with entries $D_{ii} = \sum_j W_{ij}$. Each voxel is embedded in a low-dimensional Euclidean space according to the largest several eigenvectors. The k-means algorithm is then be used to do the final partitioning. To make it feasible to apply to the spatiotemporal video volume, Fowlkes et al. [52] use the Nyström approximation to solve the above eigenproblem. Their paper demonstrates segmentation on relatively low-resolution, short videos (e.g., $120 \times 120 \times 5$) and randomly samples points from the first, middle, and last frames.

However, in our experiments, NCut is not scalable as the number of supervoxels and the length of video increases. Sampling too many points makes the Nyström method require too much memory, while sampling too few gives unstable and low performance. Meanwhile, the k-means clustering algorithm is sufficient for a video segmentation with few clusters, but a more efficient clustering method is expected regarding the number of supervoxels. Therefore, we run NCut for a subset of our experiments with lower solution and we set 200 sample points. We run k-means on 20% of the total voxels and k-nearest neighbor search to assign supervoxel labels for all voxels.

3.3.6 Segmentation by Weighted Aggregation (SWA)

SWA is an alternative approach to optimizing the normalized cut criterion [169, 170, 31] that computes a hierarchy of sequentially coarser segmentations. The method uses an algebraic multigrid solver to compute the hierarchy efficiently. It recursively coarsens the initial graph by selecting a subset of nodes such that each node on the fine level is *strongly coupled* to one on the coarse level. The algorithm is nearly linear in the number of input voxels, and produces a hierarchy of segmentations, which motivates its extension to a supervoxel method. The SWA implementation is based on our earlier 3D-SWA work in the medical imaging domain [31].

3.3.7 Temporal Superpixels (TSP)

The temporal superpixels method computes the superpixel segmentation on the first frame and then extends the existing superpixels to subsequent frames in a video. Therefore, this set of methods [23, 196, 160], by their nature, are computing supervoxels in a streaming fashion, which is similar to streamGBH with a streaming window of one frame. We choose [23] as the representative method for evaluation. The algorithm first extends the SLIC [1] superpixel algorithm to form a generative model for constructing superpixels. Each pixel is modeled using five dimensional feature vector: three channel color and the 2D location in image. Superpixels are inferred by clustering with a mixture model on individual features as a Gaussian with known variance. After generating superpixels for the first frame, the algorithm applies a Gaussian Process with a bilateral kernel to model the motion between frames. We use the implementation[2] provided by the authors with the default parameters to run the algorithm in evaluation.

[2] `http://people.csail.mit.edu/jchang7/code.php`

3.4 Datasets

We make use of six video datasets for our experimental purposes, with varying characteristics. The datasets have human-annotator drawn groundtruth labels at a frame-by-frame basis (four out of six) or at densely sampled frames in the video (two out of six). The sizes of the selected datasets vary from a few videos to one hundred videos. The set of datasets we choose are BuffaloXiph [25], SegTrack v2 [107, 193], BVDS [181, 58], CamVid [17], Liu et al. [115] and Middlebury Flow [8]. The datasets are originally built solving different video challenges: BuffaloXiph is gathered for pixel label propagation in videos; SegTrack is built for object tracking; BVDS has contributed to occlusion boundary detection; CamVid is taken in driving cars for road scene understanding; and Liu et al. [115] and Middlebury Flow [8] are used for optical flow estimation. Rather than evaluating supervoxel methods on a single dataset, we conduct the evaluation on all six datasets (with only label consistency metric on Liu et al. [115] and Middlebury Flow [8]), as the datasets are complementary and we believe supervoxels have potential to be a first processing step towards various video applications and problems. We briefly describe the six datasets used in our experiments.

BuffaloXiph from [25] is a subset of the well-known `xiph.org` videos that have been supplemented with a 24-class semantic pixel labeling set (the same classes from the MSRC object segmentation dataset [174]). The eight videos in this set are densely labeled with semantic pixels that leads to a total of 638 labeled frames, with a minimum of 69 frames-per-video (fpv) and a maximum of 86 fpv. The dataset is originally used for pixel label propagation [25] and videos in the dataset are stratified according to camera motion, object motion, the presence of articulated objects, the complexity of occlusion between objects and the difficulty of label propagation. Distinct regions with the same semantic class label are not separated in this dataset.

SegTrack v2 from [107] is an updated version of the *SegTrack* dataset [193] and pro-

vides frame-by-frame pixel-level foreground objects labeling rather than the semantic class labeling as in BuffaloXiph. It contains a total of 14 video sequences with 24 objects over 947 annotated frames. The videos in the dataset are stratified according to different segmentation challenges, such as motion blur, appearance change, complex deformation, occlusion, slow motion and interacting objects.

BVDS is initially introduced in [181] for occlusion boundary detection and then used for evaluating video segmentation algorithms by [58]. It consists of 100 HD quality videos with a maximum of 121 fpv and videos in the dataset are stratified according to occlusion, object categories and sizes, and different kinds of camera motion: translational, scaling and perspective motion. Each video is labeled with multiple human annotations by a sampling rate of 20 frames. We use all 100 videos in the evaluation ignoring the training/testing split (because BVDS is used only in the unsupervised parts of our evaluation).

Furthermore, the dataset has three different groupings for videos with moving objects, non-rigid motion, and considerable camera motion. Our experimental results show that all methods preserve the same performance order over these three video groupings, except TSP has better temporal extent than GB when only using videos with considerable camera motion. We show this additional result in the supplement.

CamVid from [17] provides five long video sequences recorded at daytime and dusk from a car driving through Cambridge, England. The videos are composed by over ten minutes high quality 30Hz footage and are labeled with 11 semantic object class labels at 1Hz and in part 15Hz that leads to a total of 701 densely labeled frames. It also provides the training/test split, with two daytime and one dusk sequence for training and one daytime and one dusk sequence for testing. Therefore, this dataset in addition allows us to evaluate methods in terms of supervoxel semantic label classification. We use all videos, in total 17898 frames[3], in the evaluation in

[3]We manually exclude the corrupted frames, and organize the dataset into short clips with roughly 100 frames-per-clip. The organized short clips can be downloaded from our website.

Sec. 3.5, and follow the training/test split in Sec. 3.6.

The remaining two datasets, **Liu et al. [115]** and **Middlebury Flow** [8] are used for evaluating label consistency in Sec. 3.5.4. They are densely annotated with groundtruth flows. Liu et al. [115] contains five videos with a minimum of 14 fpv and a maximum of 76 fpv. Middlebury Flow contains eight videos, but groundtruth for only two frames (one optical flow estimate) is available. We treat it as a special case where algorithms only process two frames.

3.4.1 Processing

To adapt all seven supervoxel methods to run through all videos in the datasets within reasonable time and memory consumption, we use BuffaloXiph, SegTrack v2 and Middlebury Flow at the original resolution; Liu et al. [115] at half the original resolution; BVDS and CamVid, the two large datasets, at a quarter of the original HD resolution. One exception is NCut which runs at a fixed resolution of 240×160 on BuffaloXiph and SegTrack v2 datasets (the results are scaled up for comparison) and is not included in the experiments with BVDS and CamVid datasets due to its high computational demands. The comparison of NCut and other methods at the same downscaled resolution on BuffaloXiph and SegTrack are shown in our conference version of the chapter [214], where the relative performance is similar to here.

We compare the seven methods as fairly as possible. However, each method has its own tunable parameters; we have tuned these parameters strictly to achieve a certain desired number of supervoxels per video (or per frame, depending on the experiment); parameters are tuned per method per dataset. For hierarchical methods, such as GBH, streamGBH, SWA, a single run over a video can generate fine-to-coarse multiple levels of supervoxels. For Mean Shift, we tune the persistence threshold to get multiple stratified segmentations. For NCut, we vary the final step K-means clustering to get a set of supervoxels varying from 100 to 500 on BuffaloXiph and

SegTrack v2. We use the suggested parameters by the authors for the two other methods (Mean Shift and TSP) and we provide all parameters to reproduce our experiments.

After we have generated a range of supervoxels for each video in a dataset, we use linear interpolation to estimate each methods' metric outputs for each video densely. The performance over a dataset at a certain number of supervoxels is drawn by averaging the interpolated values from all videos at the same number of supervoxels. This strategy can better align videos in a dataset and therefore avoids outliers with too many or too few supervoxels by simply taking averaged number of supervoxels over a dataset, especially when the videos are diverse in a dataset.

3.5 Benchmark Evaluation

Rather than evaluating the supervoxel methods on a particular application, as [68] does for superpixels and image segmentation, in this section we directly consider all of the base traits described in Sec. 3.2.2 at a fundamental level. We believe these basic evaluations have a great potential to improve our understanding of when a certain supervoxel method will perform well. Nonetheless, we further evaluate the performances of the supervoxel classification on the CamVid dataset in Sec. 3.6.

We note that some quantitative superpixel evaluation metrics have been recently used in [133, 105, 199, 118, 226]. We select those most appropriate to validate our desiderata from Section 3.2.2. One way to conduct the experiments is by evaluating the frame-based measures and take the average over all the frames in the video. However, if we directly apply these methods to the supervoxel segmentation, the temporal coherence property can not be captured. Even a method without any temporal information can achieve a good performance in those 2D metrics, which have driven us to extend the above frame-based measures to the volumetric video-based measures when appropriate.

In the rest of this section, we first introduce a pair of volumetric video-based 3D metrics that score a supervoxel segmentation based on a given human annotation and they are 3D undersegmentation error (Sec. 3.5.1) and 3D segmentation accuracy (Sec. 3.5.2). We also evaluate the boundary recall distance of the supervoxel segmentation to the human drawn boundaries (Sec. 3.5.3), as well as measure the label consistency in terms of annotated groundtruth flows in a video (Sec. 3.5.4). Then we evaluate some basic properties of supervoxel segmentation that do not require human annotation, namely explained variation, mean size variation and temporal extent of supervoxels, in Sec. 3.5.5. We also report the computational cost of each supervoxel method (Sec. 3.5.6). We give visual comparison of the supervoxel segmentations against the groundtruth annotation in Fig. 3.6. Finally, we discuss our findings in Sec. 3.5.7.

3.5.1 3D Undersegmentation Error (UE3D)

Undersegmentation error in image segmentation was proposed in [105]. It measures the fraction of pixels that exceed the boundary of the groundtruth segment when overlapping the superpixels on it. We extend this concept to a spatiotemporal video volume to measure the space-time *leakage* of supervoxels when overlapping groundtruth segments. Given a video segmented into supervoxels $\mathbf{s} = \{s_1, s_2, \ldots, s_n\}$ and a set of annotated groundtruth segments $\mathbf{g} = \{g_1, g_2, \ldots, g_m\}$ in video, we define the following UE3D as the average fraction of the voxels that exceed the 3D volume of groundtruth segments:

$$\text{UE3D}(\mathbf{s}, \mathbf{g}) = \frac{1}{m} \sum_{i=1}^{m} \frac{\sum_{j=1}^{n} \text{Vol}(s_j | s_j \cap g_i \neq \emptyset) - \text{Vol}(g_i)}{\text{Vol}(g_i)} \ , \tag{3.3}$$

where $\text{Vol}(\cdot)$ denotes the amount of voxels that are contained in the 3D volume of a segment. Equation 3.3 takes the average score from all groundtruth segments $\mathbf{g}$. We

note that the score from a single groundtruth segment g_i is not bounded. The metric imposes a greater penalty when supervoxels leak on smaller groundtruth segments. For example, if a video has a very small object, it will be equally weighted with a large object (e.g. background). Missing a pixel in the small object has a greater penalty than missing a background pixel. We also note that it is possible to set different weights for groundtruth segment classes when evaluating against a dataset with pixel semantic labels (e.g. BuffaloXiph). For a dataset with multiple human annotations (e.g. BVDS), we simply take the average score, which equally weights different human perceptions.

3.5.2 3D Segmentation Accuracy (SA3D)

Segmentation accuracy measures the average fraction of groundtruth segments that is correctly covered by the supervoxels: each supervoxel belongs to only one groundtruth segment (object) as a desired property from Sec. 3.2.2. We define the volumetric SA3D as

$$\begin{aligned} &\mathrm{SA3D}(\mathbf{s}, \mathbf{g}) = \\ &\frac{1}{m} \sum_{i=1}^{m} \frac{\sum_{j=1}^{n} \mathrm{Vol}(s_j \cap g_i) \mathbb{1}[\mathrm{Vol}(s_j \cap g_i) \geq \mathrm{Vol}(s_j \cap \bar{g}_i)]}{\mathrm{Vol}(g_i)} \ , \end{aligned} \tag{3.4}$$

where $\bar{g}_i = \mathbf{g} \setminus \{g_i\}$ and the indicator function decides when there is an association of supervoxels between segment s_j and groundtruth segment g_i. Similar to UE3D, SA3D also takes the average score from all groundtruth segments $\mathbf{g}$. However, the score from a single groundtruth segment g_i is bounded in $[0, 1]$, where the extreme situations 1 and 0 are respectively define when g_i is perfectly partitioned by a set of supervoxels (e.g. Fig. 3.1(a)), and g_i is completely missed (e.g. Fig. 3.1(b)).

We note that UE3D and SA3D are complementary to evaluate an algorithm, as UE3D measures the leakage of all supervoxels touching a groundtruth segment and

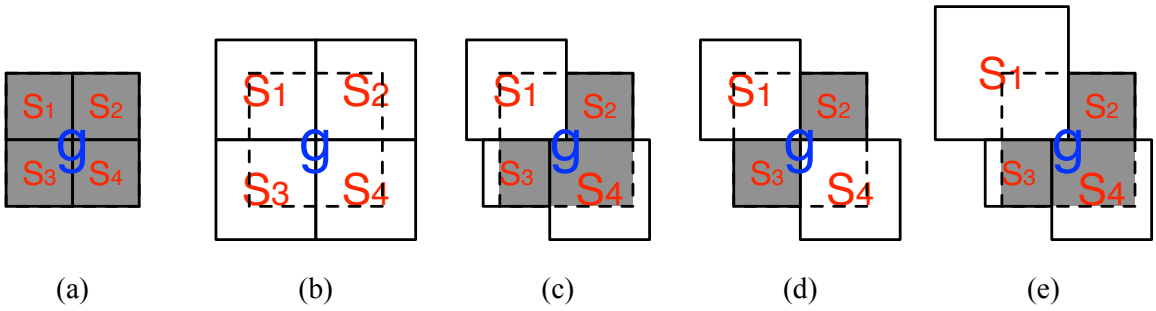


Figure 3.1: A toy example of a single groundtruth segment g with five different supervoxel segmentations. We show the example in 2D for simple illustration. We draw the groundtruth segment g as a 2x2 dashed square shape. All supervoxel segments are shown in solid square shapes and are defined in three different sizes: 1x1 (e.g. s_1 in (a)), 1.5x1.5 (e.g. s_1 in (b)), and 2x2 (e.g. s_1 in (e)). Segment s_3 in (c) and (e) is offset by 1/4. The gray areas are counted toward SA3D. The scores of UE3D, SA3D and BRD for each cases are shown in Tab. 3.1.

SA3D measures the fraction of the groundtruth segment that is correctly segmented. To further elucidate the differences between UE3D and SA3D, we show a toy example in Fig. 3.1 with scores shown in Tab. 3.1, where (c) and (d) have the same UE3D score but different SA3D scores, and (c) and (e) have the same SA3D score but different UE3D scores. (c) has the best scores for both UE3D and SA3D among all imperfect segmentation cases (b)-(e). Both the metrics are evaluated in space-time, such that they penalize supervoxels that break not only spatial boundaries but also temporal boundaries of the groundtruth segments—a good superpixel method can achieve high performance. However, it typically does so with a large number of supervoxels (the temporal extent is only one frame in this case) for the per-video basis. Therefore datasets with dense human annotations, such as BuffaloXiph and SegTrack v2, are more precise in terms of the 3D volumetric measures.

Metric	(a)	(b)	(c)	(d)	(e)
UE3D	**0.00**	1.25	**0.63**	**0.63**	1.06
SA3D	**1.00**	0.00	**0.75**	0.50	**0.75**
BRD	**0.00**	0.50	0.27	**0.25**	0.39

Table 3.1: The scores of UE3D, SA3D and BRD for the toy example in Fig. 3.1. The larger the better for SA3D, and the small the better for UE3D and BRD. The top two scores are bolded for each metric. BRD is calculated strictly for vertical boundary matching only and horizontal boundary matching only in this toy example, which is slightly different than Eq. 3.5.

3.5.3 Boundary Recall Distance (BRD)

So far we have introduced a pair of 3D metrics defined by the set of groundtruth segments. They intrinsically use the groundtruth boundaries for locating the volume of the segments. We now directly evaluate the boundary recall distance, which measures how well the groundtruth boundaries are successfully retrieved by the supervoxel boundaries. We use BRD proposed in [23] to calculate the average distance from points on groundtruth boundaries to the nearest ones on supervoxel boundaries frame-by-frame in a video. It does not require a fixed amount of dilation for boundary matching as in typical boundary recall measures to offset small localization errors. The specific metric is defined as follows:

$$\mathrm{BRD}(\mathbf{s}, \mathbf{g}) = \frac{1}{\sum_t |\mathcal{B}(\mathbf{g}^t)|} \sum_{t=1}^{T} \sum_{i \in \mathcal{B}(\mathbf{g}^t)} \min_{j \in \mathcal{B}(\mathbf{s}^t)} d(i, j) \ , \tag{3.5}$$

where $\mathcal{B}(\cdot)$ returns the 2D boundaries of segments in a frame, $d(\cdot, \cdot)$ is the Euclidean distance between the two arguments, $|\cdot|$ denotes the amout of pixels contained by the argument at a frame, t indexes frames in a video (e.g. $\mathbf{g}^t$ denotes the set of all groundtruth segments on frame t), and i and j denote points on boundaries.

We also compute BRD for all cases in Fig. 3.1 and show the scores in Tab. 3.1. We note that BRD captures different aspects of an algorithm than UE3D and SA3D.

For example, among the imperfect segmentation cases (b)-(e) (which are typical situations), (c) has the best scores in terms of UE3D and SA3D, but worse in BRD than (d) which is poor in SA3D. Therefore, there is no single segmentation that has the best scores for all three metrics (except the perfect partition in (a)) in this toy example.

3.5.4 Label Consistency (LC)

LC is also proposed in [23], which provides a possible way to measure how well supervoxels track the parts of objects given annotated groundtruth flows. Define $\mathcal{F} = \{F^{t-1\to t} | t = 2, \ldots, T\}$ as the vectorized groundtruth forward flow field in a video, and $F^{t-1\to t}(s_i)$ as the operator that projects pixels contained in s_i at frame $t-1$ to pixels at frame t by the flow (subjected to the image boundary). The metric is defined as follows:

$$\mathrm{LC}(\mathbf{s}, \mathcal{F}) = \frac{\sum_{t=2}^{T} \sum_{i=1}^{n} |s_i^t \cap F^{t-1\to t}(s_i)|}{\sum_{t=2}^{T} \sum_{i=1}^{n} |F^{t-1\to t}(s_i)|} \ , \tag{3.6}$$

where s_i^t denotes the slice of supervoxel s_i at frame t, and the numerator measures the agreement of supervoxel labels and the projected labels by flow. We evaluate this metric on [115] and Middlebury Flow where the groundtruth flow annotation is available.

3.5.5 Human-Independent Metrics

The following are human-independent metrics; in other words, they are not susceptible to variation in annotator perception that would result in differences in the human annotations, unlike the previous metrics. They directly reflect basic properties of the supervoxel methods, such as the temporal extent of generated supervoxels.

3.5.5.1 Explained Variation (EV)

The metric is proposed in [133] and it considers the supervoxels as a compression method of a video (Sec. 3.2.2):

$$\mathrm{EV}(\mathbf{s}) = \frac{\sum_i (\mu_i - \mu)^2}{\sum_i (x_i - \mu)^2} \ , \tag{3.7}$$

where x_i is the color of the video voxel i, μ is the mean color of all voxels in a video and μ_i is the mean color of the supervoxel that contains voxel i. [43] observe a correlation between EV and the human-dependent metrics for a specific object tracking task.

3.5.5.2 Mean Size Variation (MSV)

Chang et al. [23] propose superpixel size variation that measures the size variation of all superpixels in a video (as a set of frames). Here, we extend their metric to measure the size variation of the 2D slices of a supervoxel. MSV is the average score of such variation defined by all supervoxels in a video:

$$\mathrm{MSV}(\mathbf{s}) = \frac{1}{n} \sum_{j=1}^{n} \sqrt{\frac{\sum_t \left(\left(|s_i^t| - |\hat{s}_i| \right)^2 \mathbb{1} \left[|s_i^t| > 0 \right] \right)}{\sum_t \mathbb{1}[|s_i^t| > 0] - 1}} \ , \tag{3.8}$$

where $|\hat{s}_i| = \frac{\sum_t |s_i^t|}{\sum_t \mathbb{1}[|s_i^t| > 0]}$ is the average size of 2D slices of a supervoxel. MSV favors the kind of supervoxels whose 2D sizes varies minimally over time.

3.5.5.3 Temporal Extent (TEX)

TEX measures the average temporal extent of all supervoxels in a video. The measure of supervoxel temporal extent is originally proposed in [221] as a way to compare different streaming video segmentation methods. Later, [23] extend the measure by normalizing over the number of frames contained in a video. We also use

	GB	GBH	streamGBH	SWA
Time (s)	115	1166	1000	934
Memory (GB)	6.9	9.4	1.6	19.9
	TSP	**MeanShift**	**NCut**	
Time (s)	1440	101	1198	
Memory (GB)	0.9	3.8	20.9	

Table 3.2: Computational cost.

it here for the evaluation. The metric is defined as follows:

$$\mathrm{TEX}(\mathbf{s}) = \frac{1}{nT} \sum_{i=1}^{n} \sum_{t=1}^{T} \mathbb{1}[|s_i^t| > 0] \ . \tag{3.9}$$

We note that using TEX alone is not sufficient to determine what method to use, and, in fact, it must be combined with another metric, such as UE3D, SA3D or LC. We illustrate this point using a combination of TEX and LC, since they are complementary. An ideal method, in terms of capturing the temporal movement of objects in video, is expected to have long TEX and good LC. A shortcoming in either of the metrics can hurt the performance. For example, having long TEX and bad LC means that the supervoxels are long in the time dimension but they do not track well the fine-grained motion of the movement.

3.5.6 Computational Cost

We report the computational cost of all methods for a typical video with $352 \times 288 \times 85$ voxels—we record the time and peak memory consumption on a laptop featured with Intel Core i7-3740QM @ 2.70GHz and 32GB RAM running Linux, see Table 3.2. All methods are implemented in C except NCut (Matlab) and TSP (Matlab with MEX). Furthermore, all methods are single threaded except NCut running with 8 threads with resized resolution to 240×160.

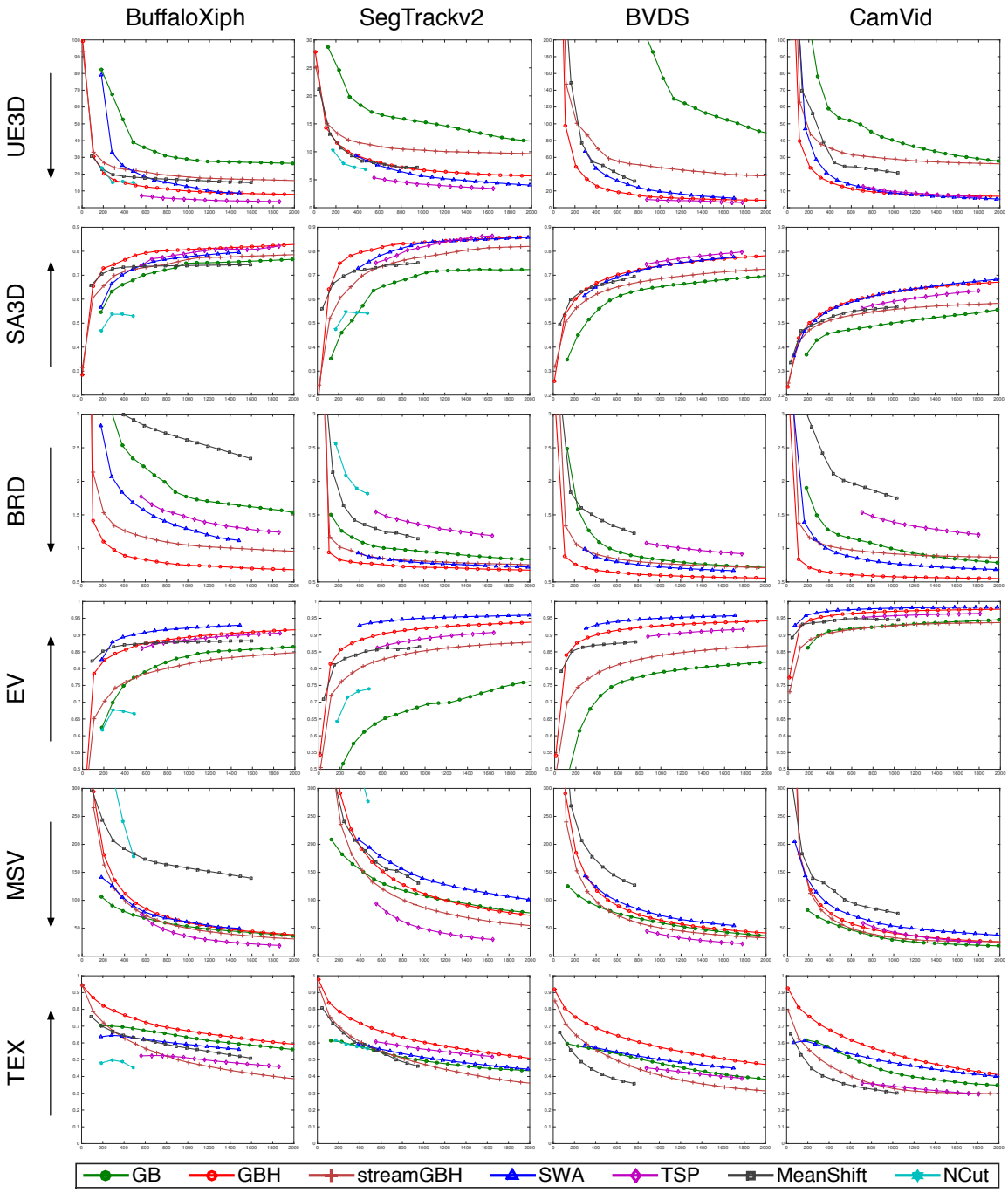


Figure 3.2: Graphs plot the number of supervoxels **per-video** (x-axis) against various metrics (y-axis). Datasets are organized by columns and metrics are organized by rows. Black arrows in each row are used to indicate the direction of better performance with regard to the metric. Plot ranges along the y-axis are aligned for all metrics except UE3D. Plotted dots are the average score of linear-interpolated values from all videos in a dataset at the same number of supervoxels per-video.

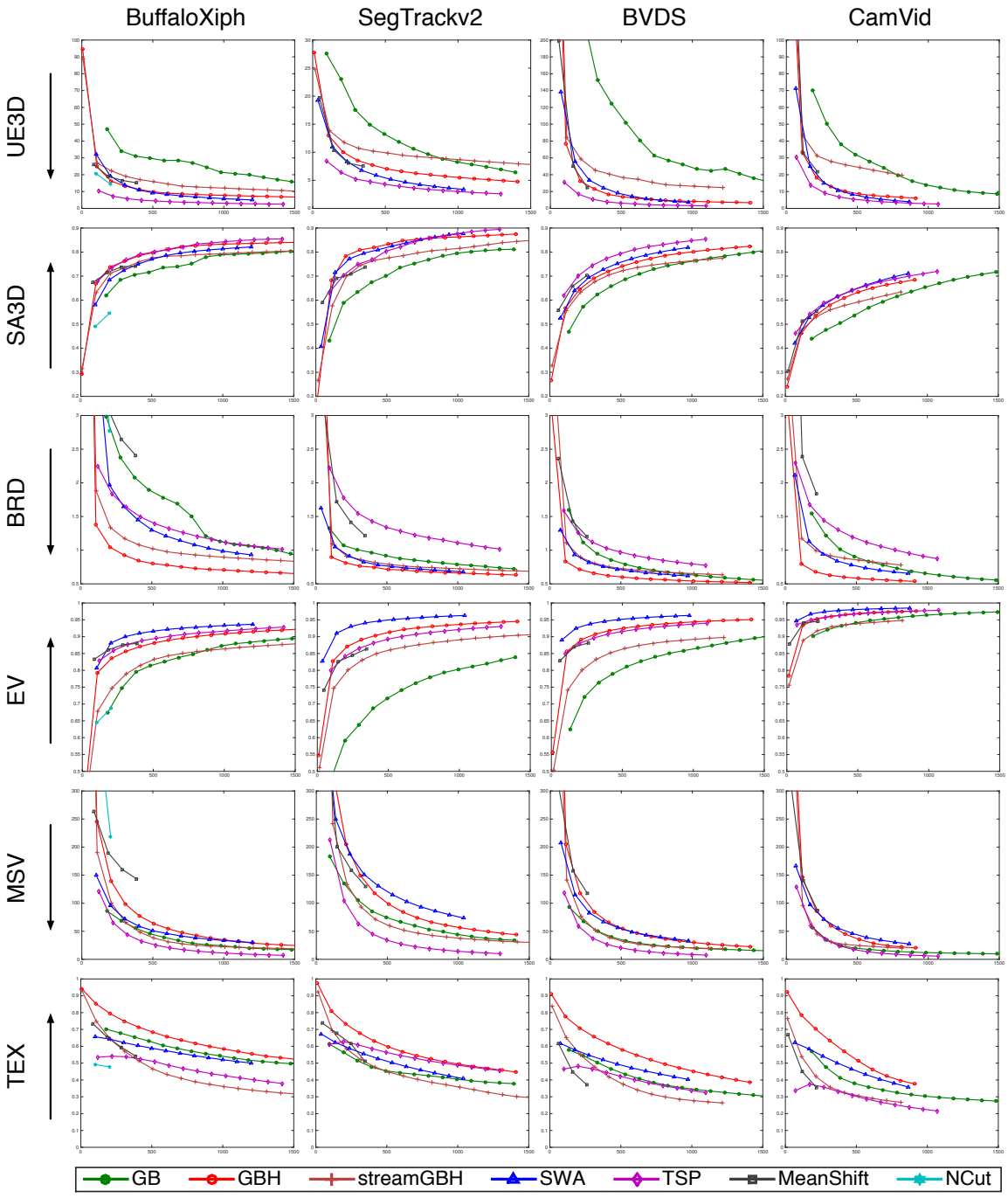


Figure 3.3: Graphs plot the number of supervoxels **per-frame** (x-axis) against various metrics (y-axis). Datasets are organized by columns and metrics are organized by rows. Black arrows in each row are used to indicate the direction of better performance with regard to the metric. Plot ranges along the y-axis are aligned for all metrics except UE3D. Plotted dots are the average score of linear-interpolated values from all videos in a dataset at the same number of supervoxels per-frame.

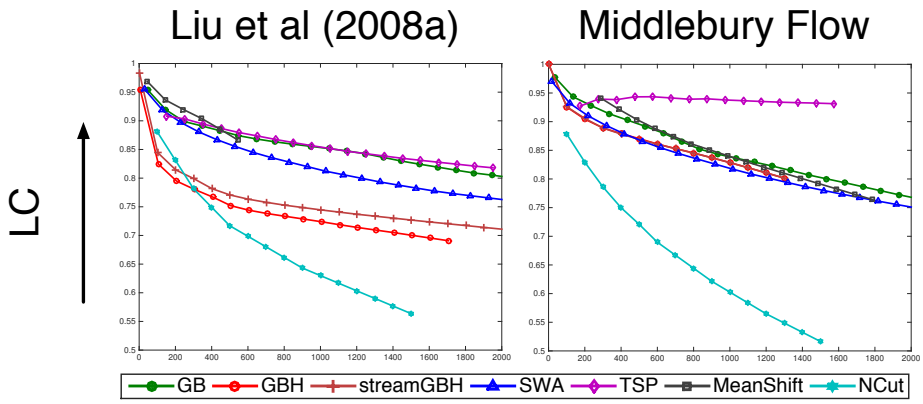


Figure 3.4: Plots for Label Consistency (LC) against the number of supervoxels **per-video** (x-axis). Black arrow indicates the direction of better performance. Plotted dots are the average score of linear-interpolated values from all videos in a dataset at the same number of supervoxels per-video.

3.5.7 Discussion

We evaluate seven methods over six datasets by the metrics defined above. We focus the evaluation in the range of 0 to 2000 supervoxels per-video (Fig. 3.2 and Fig. 3.4) as well as 0 to 1500 supervoxels per-frame (Fig. 3.3 and Fig. 3.5). We do the best to accommodate all methods in the above range, but not all methods can generate the full range of plots (e.g. Mean Shift requires huge memory to generate over 500 supervoxels per-frame for a typical video). The visualization of supervoxel segmentations can be found in Fig. 3.6. For the rest of this section, we first discuss the choice of two plot bases in Sec. 3.5.7.1, then conclude our findings in Sec. 3.5.7.2.

3.5.7.1 Plot Bases

We plot the results with two types of plot bases, namely the number of supervoxels per-video and per-frame. We summarize the rationale below, which basically distinguishes the two bases according to how space and time are treated.

Number of Supervoxels Per-Video (spv). In the earlier version of the work in this chapter [214], the number of supervoxels per-video is used for plotting figures of

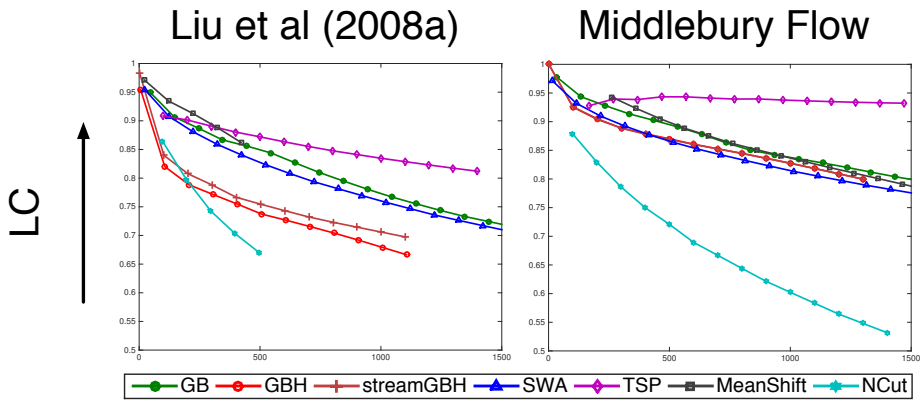


Figure 3.5: Plots for Label Consistency (LC) based on the number of supervoxels **per-frame** (x-axis). Black arrow indicates the direction of better performance. Plotted dots are the average score of linear-interpolated values from all videos in a dataset at the same number of supervoxels per-frame.

metric scores. Here, time is considered as an analogous, third dimension and treated accordingly, as in the definition of supervoxels. Hence, for example, one can consider this as a means of evaluating the compression-rate of a video as a whole. However, it may incorrectly relate videos of different lengths.

Number of Supervoxels Per-Frame (spf). Chang et al. [23] use the number of supervoxels per-frame (in this case, it is the same as the number of superpixels per-frame) in their evaluation. The mindset behind that differentiates the time dimension in a video from the spatial dimensions, such that the plot basis is not subject to different video lengths or motion. However, this approach fails to account for the temporal qualities of supervoxels—a good superpixel method can do well. For example, UE3D degenerates to UE2D for a superpixel method because it has perfect temporal boundaries.

Summary. We hence present plots against both bases and we discuss their comparisons.

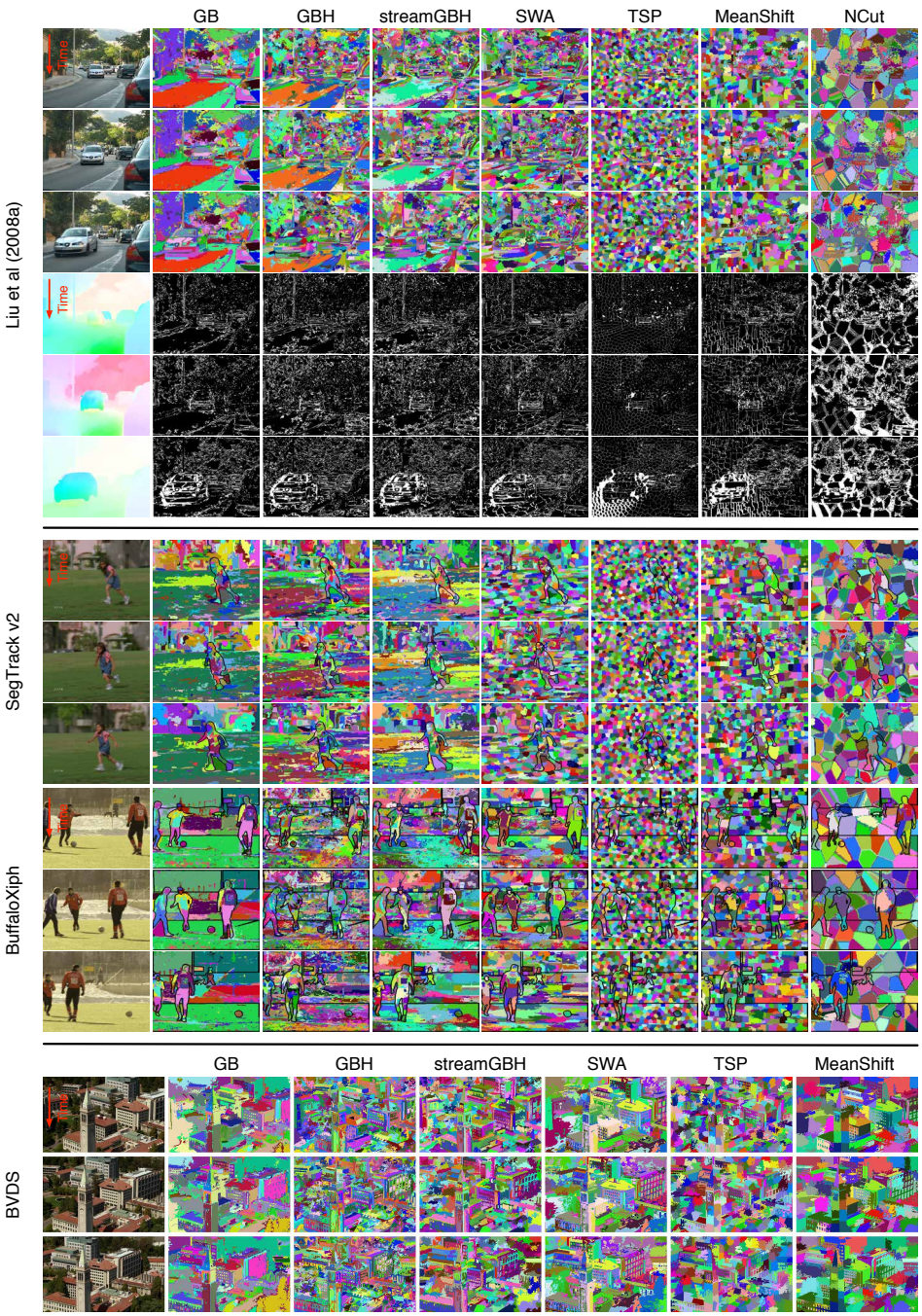


Figure 3.6: Visual comparative results of the seven methods on videos. Each supervoxel is rendered with its distinct color and these are maintained over time. We recommend viewing these images zoomed on an electronic display. In the top part, we show a video from [115] where label consistency is computed and shown in black and white (white pixels indicate inconsistency with respect to groundtruth flow). In the middle part, we show videos from SegTrack v2 and BuffaloXiph, where groundtruth object boundaries are drawn in black lines. We show a video from BVDS on the bottom.

3.5.7.2 Top Performing Methods

The metrics using human annotations, namely UE3D, SA3D, BRD and LC, reflect different preferences for supervoxels (see Sec. 3.2.2). A perfect segmentation can have all the best scores with respect to these metrics, while, often, a typical segmentation has its strengths in a subset of the metrics (recall the example in Fig. 3.1). Therefore, we organize our findings of the top performing methods by each metric and discuss the differences among datasets, if any. Recall that our choices of datasets in Sec. 3.4 represent many different types of video data (e.g. SegTrack v2 has only foreground object labels, and BuffaloXiph has pixel-level semantic class labels). Below we list the key results.

UE3D. For most cases, TSP has the best performance followed by SWA and GBH. The three methods perform similarly well on CamVid for spv in Fig. 3.2. However, TSP stands out when evaluating for spf in Fig. 3.3.

SA3D. GBH performs best on BuffaloXiph for spv, whereas TSP performs best when plotted by spf. GBH, SWA and TSP perform almost equally well on SegTrack v2. TSP performs best on BVDS, where annotators are instructed to label all objects on sampled frames of a video. SWA and GBH perform equally best on CamVid for spv, but when plotting by spf, SWA and TSP perform the best.

BRD. GBH is the clear winner method in this metric, and following that are streamGBH and SWA. GB has a faster trend to approach GBH than streamGBH on CamVid and BVDS for spf.

LC. TSP (the only method that uses optical flow in the implementations we use) has the best performance and there is a clear performance gap on Middlebury Flow, where videos only have two frames (Fig 3.4 and 3.5). Furthermore, unlike the other methods, the performance of TSP dose not dramatically decrease when spv and spf increase on Middlebury Flow.

EV. SWA has the overall best performance and followed by GBH and TSP. GBH

ranks better than TSP on BuffaloXiph for spv, but the ordering swapped when plotting against spf.

MSV. TSP has the best performance followed by streamGBH and GB except on CamVid for spv, where GB performs the best.

TEX. GBH has the longest temporal extent for both spv and spf within the range we plotted. We note [23] show that TSP has better performance than GBH in a different spectrum of spf on [115] and SegTrack [193].

Over all seven methods, GB and Mean Shift are the most efficient in time. Interestingly, neither GB nor Mean Shift performs best in any of the human annotation related quality measures—there is an obvious trade-off between the computational cost of the methods and the quality of their output (in terms of our metrics).

We have focused on the facts here. Although understanding *why* these various algorithms demonstrate this comparative performance is an ultimate goal of our work, it is beyond the scope of this chapter and would require a substantionally deeper understander of how space and time relate in video analysis. To overcome this limitation, we map these comparative performances onto specific problem-oriented needs in the Conclusion (Sec. 5.5).

3.6 Supervoxel Classification

In this section, we evaluate the supervoxel methods in a particular application: supervoxel semantic label classification. We use this application as a proxy to various high-level video analysis problems. For example, superpixel classification scores are frequently used as the unary term when building subsequent graphical model for scene understanding in images, e.g., [62]. We use the CamVid dataset for this task due to its widely use in semantic pixel labeling in videos. Recall that CamVid has videos over ten minutes and labeled frames with 11 semantic classes at 1Hz, such as building, tree, car and road. We follow the standard training/test split: two daytime and

one dusk sequence for training, and one daytime and one dusk sequence for testing. We process the videos into supervoxel segmentations as described in Sec. 3.4.1. We use all supervoxel methods except for the NCut method because of its high memory requirement for these CamVid data, which rendered the size of the supervoxels too large to train meaningful classifiers.

Supervoxel Features. [188] extract a set of low-level features on superpixels and use the supervoxels generated by [65] for their video parsing on CamVid dataset. We apply a similar set of features with some modifications to suit for our task. We first dilate the 2D slices of supervoxels by 3 pixels and then extract features histograms from supervoxel volumes. To be specific, we compute histogram of textons[4] and dense SIFT descriptors with 100 dimensions each. We also compute two types of color histograms, RGB and HSV, with 8 bins each channel. We describe the location of a supervoxel volume by averaging the distances of bounding boxes of its 2D slices to image boundaries. In addition to image features, we calculate dense optical flow and quantize flows in a supervoxel volume to 8 bins each according to vertical and horizontal velocity, and speed magnitude. Note that the way we extract the feature histograms is different than [188], where they use one representative superpixel of a supervoxel (the 2D slice with largest region). We think that the volume has better potential to represent the change of a supervoxel over time. We also note that more sophisticated video features can be added to supervoxel volumes such as dense trajectories [203] and HOG3D [85]. However, for a fair comparison of supervoxel methods, we stick to the dense image features and optical flow in order to prevent favoring one supervoxel method than another.

Supervoxel Labels. We assign a supervoxel with the most frequent groundtruth label occur in its volume and ignore supervoxels that fail to touch groundtruth frames (labeled at 1Hz on CamVid). We note that this step is distinct from most image

[4] `http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html`

superpixel classification work, e.g., [62], since videos are often sparsely labeled while images are densely labeled. Therefore, this step may introduce more noise in both training and testing than the image superpixel classification work, and it is closely related to two of our benchmark metrics—UE3D in Sec. 3.5.1 and SA3D in Sec. 3.5.2. We apply the pixel-level average per-class accuracy and global pixel accuracy to evaluate this *supervoxel label assignment* step and the top part in Fig. 3.7 shows the performance for all six methods in the experiment. Rather than using linear interpolated values as in Fig. 3.2 to 3.5, the plotted dots here map to actual segmentations generated by a single run of the algorithm over the dataset, and the plot basis is the number of supervoxels for every 100 frames.

Classification Performance. Finally, we use linear SVMs[5] on supervoxels to get the classification results on the test set. The output segmentations are for the entire video but we evaluate only on the labeled frames. We again show the performance in terms of pixel-level average per-class and global accuracy in the bottom part in Fig. 3.7 with the number of supervoxels ranging from less that 100 to more than 900 every 100 frames. To compare with pixel-based image segmentation method, we note that [17] report 53.0% average per-class and 69.1% global accuracy by using both appearance and geometric cues. The supervoxel-based methods with our setup in general achieve a better global pixel performance but a worse average per-class accuracy (e.g. 500 supervoxels in Fig. 3.7) with respect to the range of supervoxel numbers we sampled for the evaluation. We suspect that some classes with small regions, such as *sign symbol* and *bicyclist*, become too small to capture when we scale the videos down to a much lower resolution (a quarter of the original) to accommodate all six supervoxel methods.

Fig. 3.8 shows the pixel-level labeling accuracy for each class in the dataset. For large classes, such as *road* and *sky*, the performance of all methods is not largely af-

[5] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

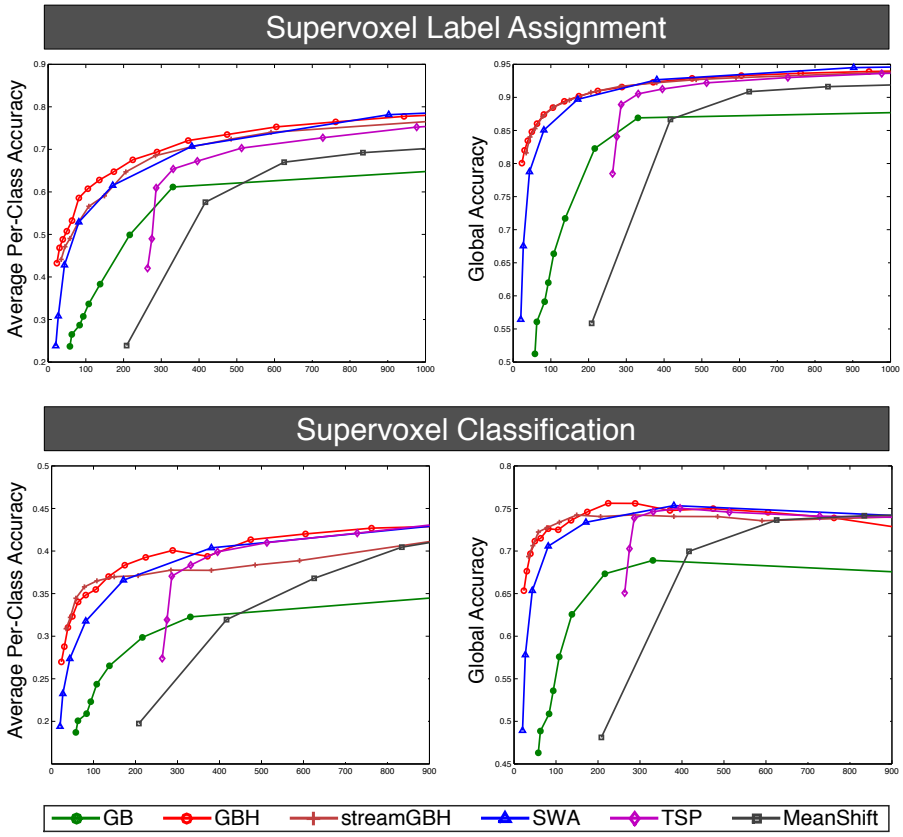


Figure 3.7: Plots on the top are the pixel-level average per-class accuracy (left) and global accuracy (right) for both training and testing sets when supervoxels directly take groundtruth labels (the most frequent ones in volumes). Plots on the bottom are the pixel-level classification performance on the test set with SVMs trained on supervoxels. We show the plots in the range of 100 to 900 supervoxels every 100 frames (x-axis). The plotted dots are from actual segmentations rather than interpolated values. We note that [17] report 53.0% average per-class and 69.1% global accuracy using random forests trained on pixels with both appearance and geometric cues, where we only use appearance cues with supervoxels.

fected by the number of supervoxels, except the performance for *building*, which rises then falls. This rise and fall results in a decrease in the overall global performance (see bottom right in Fig. 3.7). We explain this rise and fall behavior of the building class due to the overall scale-varying texture of buildings and the challenge to learn classifiers on them that perform equally well at different scales; for example, smaller supervoxels will cover small portions of buildings, say windows or bricks, which have

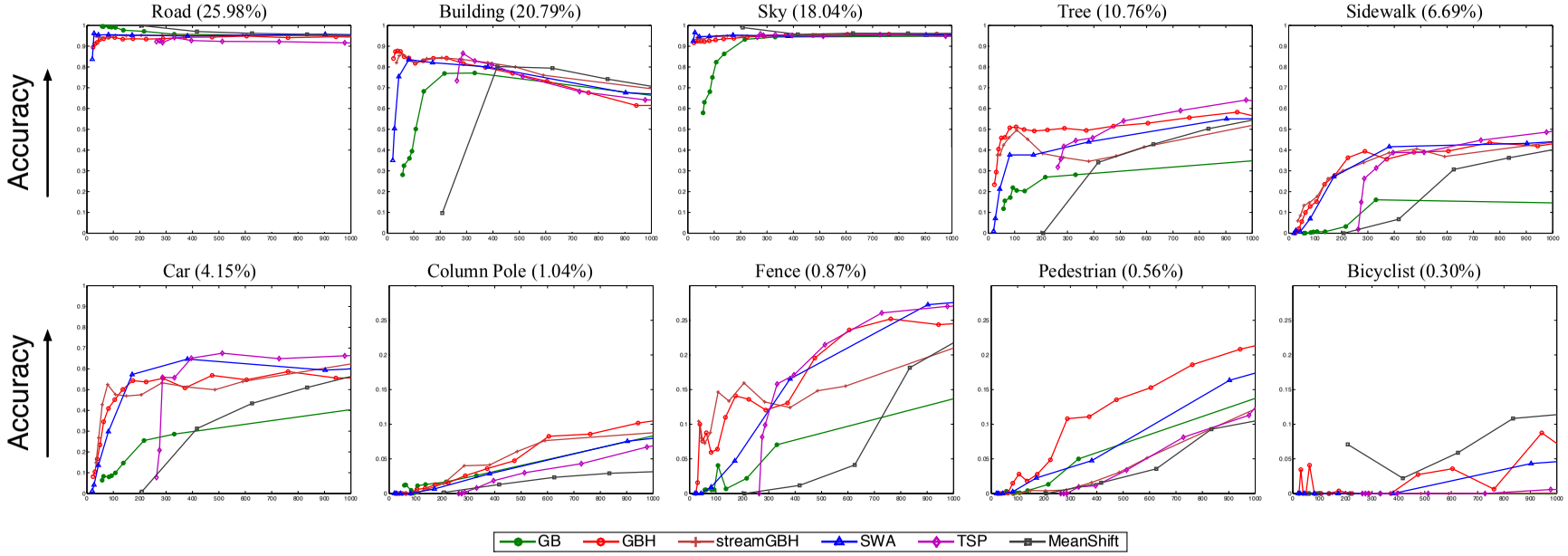


Figure 3.8: Pixel-level labeling accuracy for each semantic class in the CamVid dataset, where the percentages of total pixels for each class are shown on top. All plots are shown in the range of 0 to 1000 supervoxels every 100 frames (x-axis). The first six plots (horizontal) are plotted with an accuracy range from 0 to 1, and the other plots are from 0 to 0.3. We do not show the class *Sign Symbol (0.17%)* here due to its low accuracy for all methods.

distinct visual characteristics, yet a single classifier is to be learned (in our evaluation). For other classes the performance increases when adding more supervoxels, and different methods have distinct performance on different classes. For example, GBH leads the score on *pedestrian* while TSP and SWA are the methods of choice on *car*. Further investigation is needed to better understand these nuances.

Fig. 3.9 shows visual comparison of six methods on two clips from the daytime test video. Although GB and Mean Shift successfully segment the *sidewalk* in the supervoxel segmentation, they miss a large portion of the *sidewalk* in the labeling, while the other methods capture it well. The *tree* tends to be better labeled by GBH. All methods segment the moving cars well. However none of the method get the small *sign symbol* in the second clip. We also show the results during dusk in Fig. 3.10. GB works poorly here; the greedy algorithm of GB is highly sensitive to local color thus it easily produces large incorrect segments. TSP visually segments *bicyclist* well regardless the incorrect boundaries. We think this is due to the compact shape of supervoxels that TSP generated can better track the superpixels on the bicyclist and prevent easily merging with other large segments such as *sidewalk*, *tree* and *road*. However, it also brings more fragmented segments on large smooth regions, such as *road* and *sidewalk* and weak boundary accuracy.

Overall, GBH, SWA and TSP achieve equally strong performance in the supervoxel classification experiment (see Fig. 3.7), and, again, they are the top performing ones in terms of our benchmark evaluation in Sec. 3.5. Methods such as GB and MeanShift have poor classification performance also perform less well on the benchmark metrics. For the streaming methods, streamGBH achieves very similar performance to its full-video counterpart GBH.

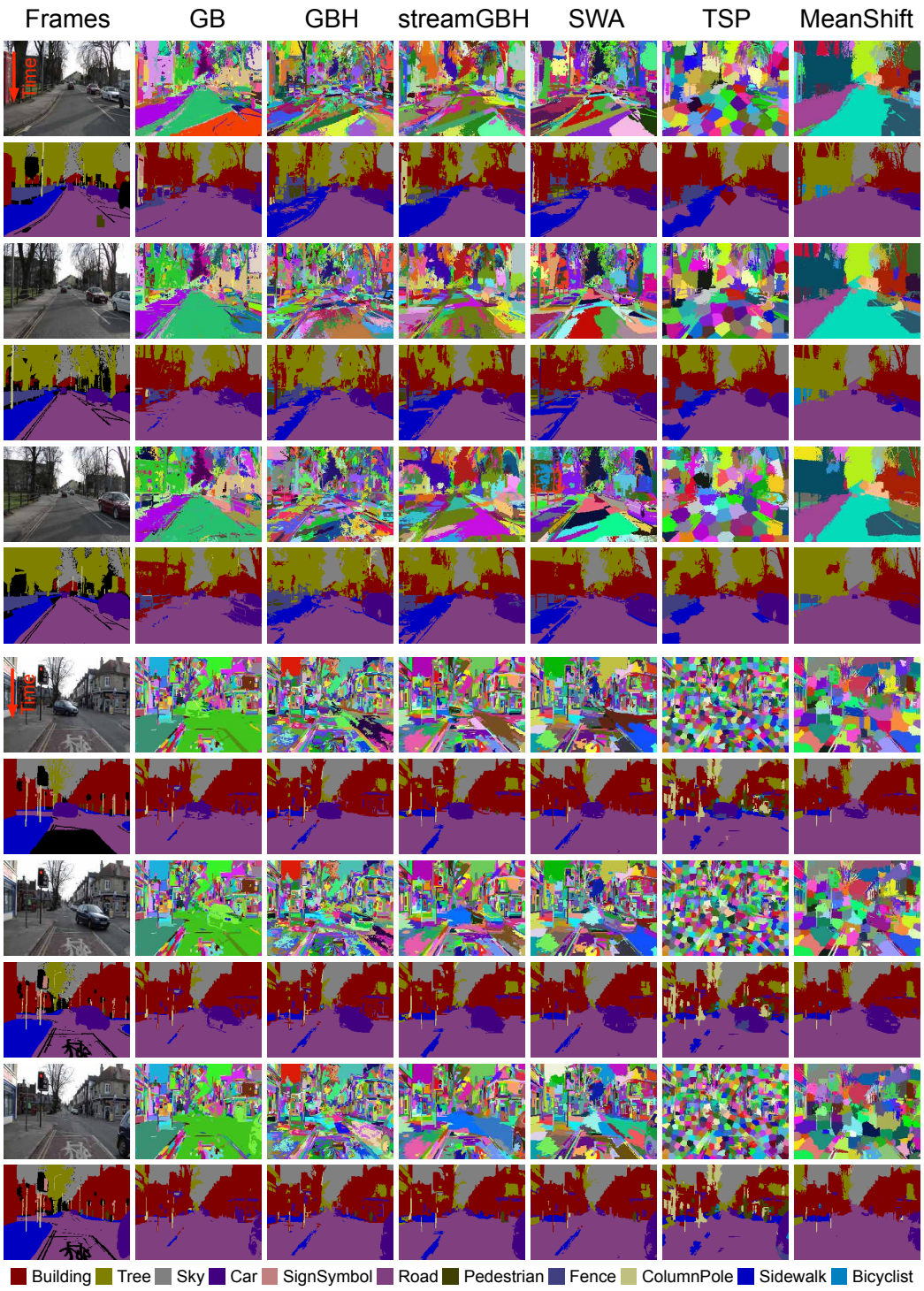


Figure 3.9: Example results on two short clips from the CamVid daytime test video. Images in the first column are video frames and groundtruth labels and the remaining columns are individual methods with supervoxel segmentation and semantic labeling on supervoxels.

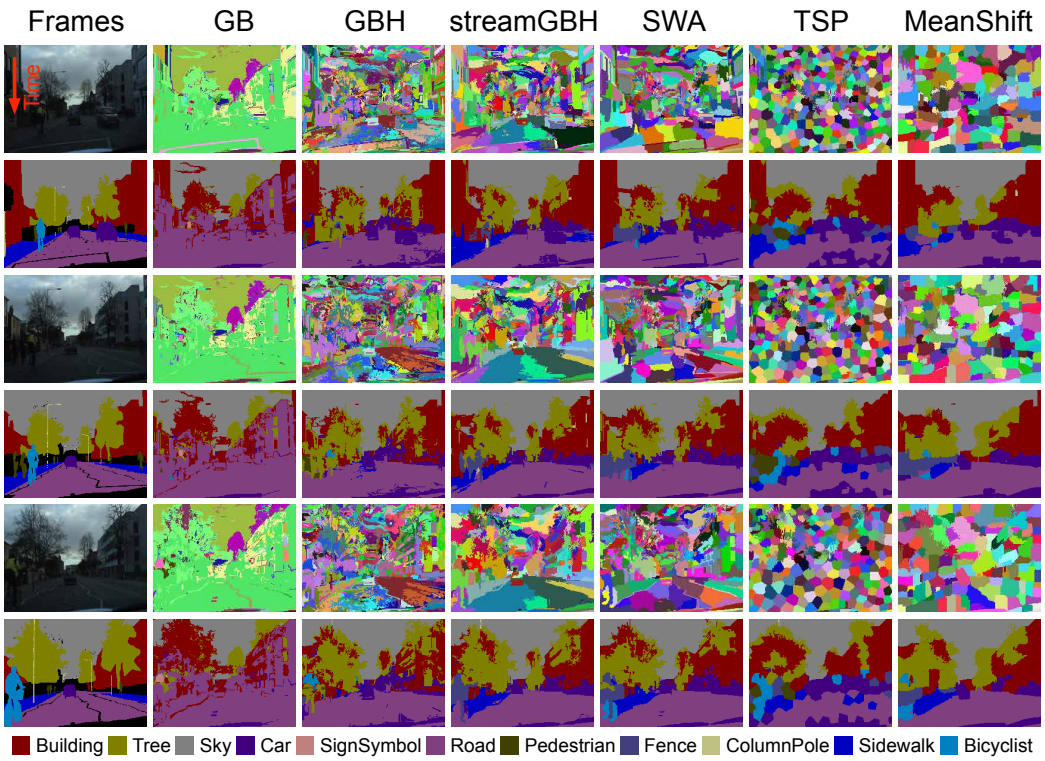


Figure 3.10: Example results on a clip from the CamVid dusk test video. Images in the first column are video frames and groundtruth labels and the remaining columns are individual methods with supervoxel segmentation and semantic labeling on supervoxels.

3.7 Conclusion

We have presented a thorough evaluation of seven supervoxel methods including both off-line and streaming methods on a set of seven benchmark metrics designed to evaluate supervoxel desiderata as well as the recognition performance on a particular application. Samples from the datasets segmented under all seven methods are shown in Fig. 3.6, Fig. 3.9, and Fig. 3.10. These visual results convey the overall findings we have observed in the quantitative experiments. GBH, SWA and TSP are the top-performers among the seven methods in both our benchmark evaluation and the classification task. They all share a common feature in that they perform well in terms of segmentation accuracy, but they comparatively vary in performance in regard to the other metrics. GBH captures object boundaries best making it well

suited for video analysis tasks when accurate boundaries are needed, such as robot manipulation. SWA has the best performance in the explained variation metric, which makes it most well-suited for compression applications. TSP follows object parts and achieves the best undersegmentation error making it well-suited for fine-grained activity analysis and other high-level video understanding problems. It seems evident that the main distinction behind the best offline methods, namely GBH and SWA, is the way in which they both compute the hierarchical segmentation. Although the details differ, the common feature among the two methods is that during the hierarchical computation, coarse-level aggregate features replace or modulate fine-level individual features. In contrast, TSP processes a video in a streaming fashion and also produces supervoxels that are the most compact and regular in shape. These differences suggest a complementarity that has the potential to be combined into a new method, which are currently investigating.

In this chapter, we have explicitly studied the general supervoxel desiderata regarding a set of proposed benchmark metrics including both human annotation dependent and independent ones. In addition, we compare the supervoxel methods in a particular application—supervoxel classification that evaluates methods in a recognition task, which we consider to be a proxy to various high-level video analysis tasks in which supervoxels could be used. A strong correlation is presented between the benchmark evaluation and the recognition task. Methods, such as GBH, SWA and TSP, that achieve the top performance in the benchmark evaluation also perform best in the recognition task. The obvious question to ask is how well will the findings translate to other application-specific ones, such as tracking and activity recognition. A related additional point that needs further exploration for supervoxel methods is the modeling of the relationship between spatial and temporal domains in a video. We plan to study these important questions in future work.

CHAPTER IV

Scale Generation II: A Streaming Hierarchical Framework

4.1 Introduction

The video analysis literature, despite a rich history of segmentation [130], has been dominated by feature-based methods, e.g., [98], over the last decade. More recently, a trend to incorporate an initial oversegmentation into supervoxels followed by various task-specific processes, such as hierarchical grouping [65] and long-range tracking [15, 106], and superpixel flow [198], has surfaced. Many driving applications in video, such as animation [206, 6], require a dense segmentation that feature-based methods cannot naturally provide.

Good progress has been reported on video segmentation in recent years with methods ranging from mean shift [146], region tracking [15], interactive matting [6] and graph-based processing [65]. To evaluate this progress and these methods' comparative suitability for early video oversegmentation into supervoxels, we have studied a set of methods in Chapter III, with varying characteristics. The methods have been evaluated for desirable video segment properties, such as spatiotemporal uniformity and coherence, explained variation, and spatiotemporal boundary extraction, and compared on human-labeled video benchmarks. The study conclusively reports

that the two hierarchical segmentation methods (SWA and GBH) generate space-time segments with the most desirable properties, paving the way for more principled use of video segmentation going forward.

However, there are two extant limitations in the state of practice in video segmentation. First, most methods build a representation on the entire video and process it at once (e.g. a large graph for whole video). Clearly, they exhaust memory resources on all typical and even some quite large machines (e.g., consider these loading a 320×240 video: 3 seconds of video requires 21MB of memory; 2 minutes: 840 MB; 30 minutes: 12.6 GB; 2h: 50GB, and so on). Thus, most existing video segmentation methods are unable to handle even medium-length videos. Those methods that do process the video online, such as streaming mean shift [145] perform comparatively worse on the recent evaluation [214]. Note, [65] presents a hierarchical graph-based algorithm that conclusively performs best in [214]; however, the clip-based version of their method is restricted to the voxel-layer and is not hierarchical, reducing it to graph-based segmentation [47], which performs comparatively worse on the benchmark. We also note that the flow-based work of [198] can also handle videos of arbitrary length, but outputs non-hierarchical segmentations. It is not included in the evaluation; we expect it to perform similarly to those other non-hierarchical methods in [214].

Second, in some applications, such as surveillance and interaction [172], it is impractical to expect the whole video to be available. In these cases, the video is continuously streaming in real time. These limitations seem to have stunted the use of video segmentation; e.g., contrast the relative lack of supervoxels in the video analysis literature with the dominance of superpixels in the image analysis literature [158, 47, 105, 199, 133, 134, 189]. Indeed, due to the lack of long-range, efficient supervoxel methods, some researchers resort directly to frame-by-frame superpixel methods, such as [102] who seek a video-based foreground object extraction.

To overcome these limitations, we present a principled approximation to hier-

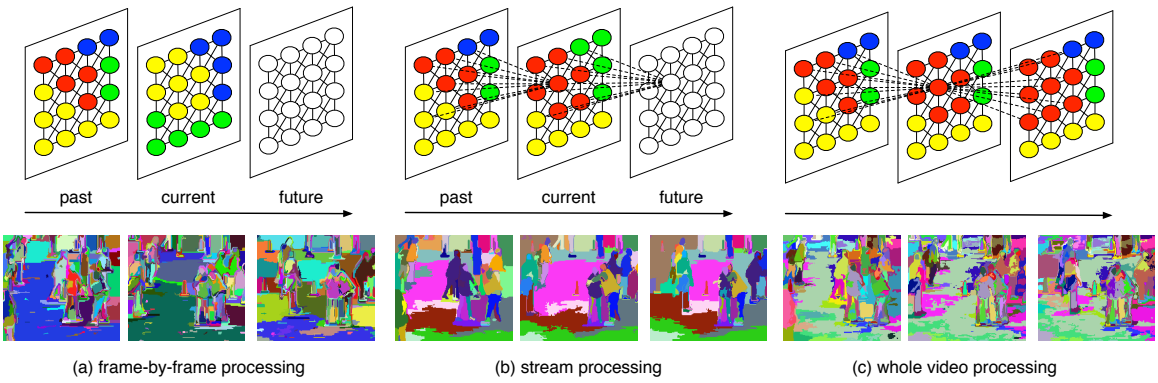


Figure 4.1: Three different processing paradigms for video segmentation. (a) Frame-by-frame processing such that each frame is independently segmented, but no temporal information is used. Even though it is fast, the results and temporal coherence are poor. (b) Stream processing segments the current frame only based on a few previously processed frames. It is forward-only online processing, and the results are good and efficient in terms of time and space complexity. (c) 3D volume processing that represents a model for the whole video. It is bidirectional multi-pass processing. The results are best, but the complexity is too high to process long and streaming videos.

archical video segmentation. Our key contributions are twofold: (1) we propose a streaming hierarchical video segmentation framework that leverages ideas from *data streams* [136] and enforces a Markovian assumption on the video stream to approximate full video segmentation and (2) we instantiate the streaming graph-based hierarchical video segmentation within this approximation framework. Our method is the first streaming hierarchical video segmentation method in the literature, to the best of our knowledge. Fig. 4.1 illustrates the differences between frame-by-frame, streaming, and full volume processing. Although frame-by-frame processing is efficient and can achieve high-performance in spatial respects, its temporal stability is limited.

We have implemented and evaluated our streaming hierarchical graph-based algorithm both on the existing benchmark [214] and on longer length videos. Our experiment results demonstrate the streaming approximation can perform almost as

well as the full-video hierarchical graph-based method and asymptotically approaches the full-video method as the size of the streaming-window increases. However, the streaming approximation maintains better performance of the hierarchical segmentation than full-video, non-hierarchical methods, and the proposed method outperforms other state of the art streaming methods on longer videos.

The remainder of this chapter is organized as follows: Sec. 4.2 presents our streaming hierarchical video segmentation approximation framework, Sec. 4.3 introduces the graph-based streaming hierarchical method, Sec. 6.4 discusses our experimental analysis, Sec. 5.5 concludes the chapter.

4.2 Streaming Hierarchical Video Segmentation—An Approximation for Hierarchical Video Segmentation

Problem Statement. For a given video $\mathcal{V}$, consider an objective function or criterion $E(\cdot|\cdot)$ to obtain the hierarchical segmentation result $\mathcal{S}$ by minimizing:

$$\mathcal{S}^* = \underset{\mathcal{S}}{\operatorname{argmin}}\, E(\mathcal{S}|\mathcal{V}) \ . \tag{4.1}$$

Concretely, let Λ^2 denote the 2D pixel lattice. We think of a video as a function on space-time lattice $\Gamma \doteq \Lambda^2 \times \mathbb{Z}$ mapping to color space $\mathbb{R}^3$, s.t. $\mathcal{V}\colon \Gamma \to \mathbb{R}^3$. The hierarchical segmentation results in h layers of individual segmentations $\mathcal{S} \doteq \{S^1, S^2, \ldots, S^h\}$ where each layer S^i is a set of segments $\{s_1, s_2, \ldots\}$ such that $s_j \subset \Gamma$, $\cup_j s_j = \Gamma$, and $s_i \cap s_j = \varnothing$ for all pairs of segments. Each layer in the hierarchy gives a full segmentation and the hierarchy itself defines a forest of trees (segments only have one parent). There are several methods, e.g., [65, 145], for pursuing this hierarchical segmentation. However, we are not aware of any method that has an explicit objective function to optimize; they are all based on some criteria to pursue hierarchical segmentation results. We consider them as greedy/gradient descent-like

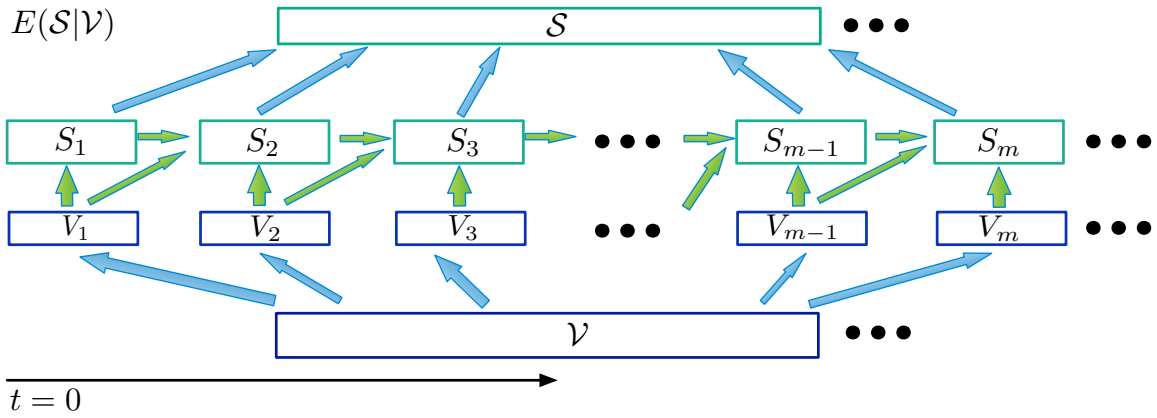


Figure 4.2: Framework of streaming hierarchical video segmentation.

methods to approximate the global or local minimal of an implicit objective function $E(\mathcal{S}|\mathcal{V})$.

Next, we introduce our streaming hierarchical video segmentation framework that approximates a solution for (4.1) by leveraging ideas from data streams [136].

4.2.1 Streaming Hierarchical Video Segmentation—An Approximation Framework for $E(\mathcal{S}|\mathcal{V})$

Consider a stream pointer t that indexes frames in a video $\mathcal{V}$ of arbitrary length. By definition, our streaming algorithm may touch each frame of video at most a constant number of times and it may not alter the previously computed results from frames $\hat{t} < t$. Without loss of generality, we can conceptualize this streaming video as a set of m non-overlapping subsequences, $\mathcal{V} = \{V_1, V_2, \cdots, V_m\}$ with k_i frames for subsequence V_i. Thus, hierarchical segmentation result $\mathcal{S}$ could approximately be decomposed into $\{S_1, S_2, \cdots, S_m\}$ as video $\mathcal{V}$ in Fig. 4.2, where S_i is hierarchical segmentation of subsequence V_i.

Then, to enforce the data stream properties, we make a Markov assumption. Assume hierarchical segmentation S_i of subsequence V_i is only conditioned on the subsequence V_{i-1} and its segmentation result S_{i-1}. Therefore we can obtain a directed

Bayesian-like network approximating the hierarchical video segmentation model, see Fig. 4.2, as:

$$\begin{aligned} E(\mathcal{S}|\mathcal{V}) &= E^1(S_1|V_1) + E^1(S_2|V_2, S_1, V_1) + \cdots + E^1(S_m|V_m, S_{m-1}, V_{m-1}) \\ &= E^1(S_1|V_1) + \sum_{i=2}^{m} E^1(S_i|V_i, S_{i-1}, V_{i-1}) \ , \end{aligned} \tag{4.2}$$

where, for convenience, we denote $E(\cdot|\cdot), E^1(\cdot|\cdot), E^2(\cdot|\cdot)$ as different (hierarchical) segmentation models for whole video, video subsequence and each layer of video subsequence, respectively. So,

$$\mathcal{S} = \{S_1, \cdots, S_m\} = \underset{S_1, S_2, \cdots, S_m}{\operatorname{argmin}} \left[E^1(S_1|V_1) + \sum_{i=2}^{m} E^1(S_i|V_i, S_{i-1}, V_{i-1}) \right] \ , \tag{4.3}$$

where $E^1(S_i|V_i, S_{i-1}, V_{i-1})$ is hierarchical segmentation model for each subsequence and $E^1(S_1|V_1)$ is a special case (no information from the previous subsequence).

Given an explicit objective function $E^1(\cdot|\cdot)$, Eq. (4.3) could be solved using dynamic programming, but because most hierarchical video segmentation methods do not provide an explicit energy function and the configuration space of S_i is too large, we can not use dynamic programming. For our streaming scenario—to obtain $\mathcal{S}$ efficiently—we again make the assumption that the hierarchical segmentation result S_i for the current subsequence V_i never influences segmentation results for previous subsequences, it only influences the next subsequence V_{i+1}. Therefore, Eq. 4.3 is approximated as:

$$\begin{aligned} \mathcal{S} = \{S_1, \cdots, S_m\} = \Big\{ &\underset{S_1}{\operatorname{argmin}} E^1(S_1|V_1), \cdots, \underset{S_i}{\operatorname{argmin}} E^1(S_i|V_i, S_{i-1}, V_{i-1}), \\ &\cdots, \underset{S_m}{\operatorname{argmin}} E^1(S_m|V_m, S_{m-1}, V_{m-1}), \Big\} \ , \end{aligned} \tag{4.4}$$

which can be solved greedily. Start from V_1, obtain S_1, then fix S_1 and obtain S_2

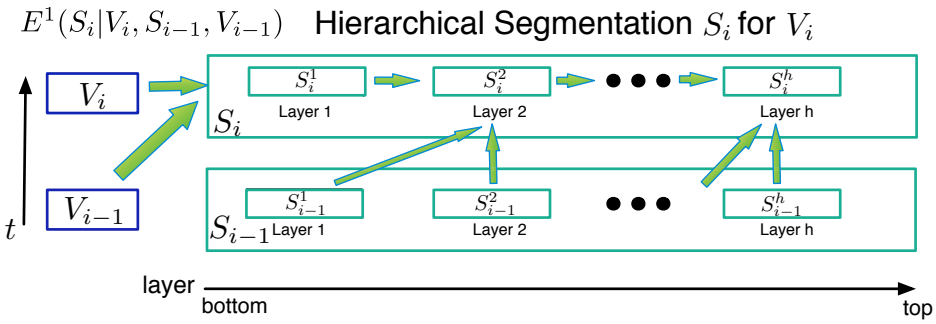


Figure 4.3: Sub-framework for hierarchical segmentation for single subsequence V_i.

based on S_1, V_1, and V_2. In the same way, obtain $S_3, S_4, \cdots, S_i$ until the whole video is completed.

Therefore, at any time, our approximation framework only needs to store two subsequences into memory, whatever the length of video is, and each subsequence is processed only once. The length k_i of a subsequence V_i is a parameter and can range from 1 to the full length of the video. According to the definition of data streams, we hence name this approximation framework with greedy solution as **streaming hierarchical video segmentation**. And our framework obviously can well handle the two proposed problems of current hierarchical video segmentation mentioned in the introduction: namely, we do not store the whole video at once and in some cases, the whole video may not be available.

4.2.2 Model For Estimating $S_i|(V_i, S_{i-1}, V_{i-1})$

We can hierarchically segment video of arbitrary length if we know how to solve:

$$S_i = \underset{S_i}{\operatorname{argmin}} E^1(S_i|V_i, S_{i-1}, V_{i-1}) \ , \tag{4.5}$$

where S_i is hierarchical segmentation of subsequence V_i, s.t. $S_i = \{S_i^1, S_i^2, \cdots, S_i^h\}$ with h layers. S_i^j is the segmentation result at the j_{th} layer and the first layer S_i^1 is the video itself, where each voxel is a segment. Continuing with a similar Markov

assumption, assume S_i^j only depends on S_i^{j-1}, S_{i-1}^{j-1}, S_{i-1}^j and the subsequence V_{i-1} and V_i, see Fig. 4.3. Then we can obtain:

$$\begin{aligned} S_i &= \underset{S_i}{\operatorname{argmin}} E^1(S_i|V_i, S_{i-1}, V_{i-1}) \\ &= \underset{S_i^2,\cdots,S_i^h}{\operatorname{argmin}} \sum_{j=2}^{h} E^2(S_i^j|V_i, S_i^{j-1}, S_{i-1}^{j-1}, S_{i-1}^j, V_{i-1}) \ . \end{aligned} \tag{4.6}$$

We can hence reformulate this using a greedy approach, as we did in Eq. 4.4:

$$\begin{aligned} S_i =& \underset{S_i}{\operatorname{argmin}} E^1(S_i|V_i, S_{i-1}, V_{i-1}) \\ =& \Big\{ \underset{S_i^2}{\operatorname{argmin}} E^2(S_i^2|V_i, S_i^1, S_{i-1}^1, S_{i-1}^2, V_{i-1}), \cdots , \\ & \underset{S_i^h}{\operatorname{argmin}} E^2(S_i^h|V_i, S_i^{h-1}, S_{i-1}^{h-1}, S_{i-1}^h, V_{i-1}) \Big\} \ , \end{aligned} \tag{4.7}$$

where, recall, $E^2(\cdot|\cdot)$ is the conditional segmentation model for a single layer of the subsequence. Since in the first layer, each voxel is a segment, S_i^1 is given. Eq. 4.7 can be solved by using the greedy solution for Eq. 4.4. In this way, we transfer complex conditional hierarchical segmentation $S_i|(V_i, S_{i-1}, V_{i-1})$ to a simple layer by layer conditional segmentation $S_i^j|(V_i, S_i^{j-1}, S_{i-1}^{j-1}, S_{i-1}^j, V_{i-1})$. Thus, we can obtain S_i by pursuing segmentation S_i^j layer by layer.

4.2.3 Semi-Supervised Grouping Method for Estimating $S_i^j|(V_i, S_i^{j-1}, S_{i-1}^{j-1}, S_{i-1}^j, V_{i-1})$

According to Eq. 4.7, now we propose a method to estimate S_i^j that:

$$S_i^j = \underset{S_i^j}{\operatorname{argmin}} E^2(S_i^j|V_i, S_i^{j-1}, S_{i-1}^{j-1}, S_{i-1}^j, V_{i-1}) \ . \tag{4.8}$$

First, we consider Eq. 4.8 to pose a semi-supervised grouping problem, given our streaming assumptions (not user input). Given joint segment set $S_{i-1,i}^{j-1} = S_{i-1}^{j-1} \cup S_i^{j-1}$,

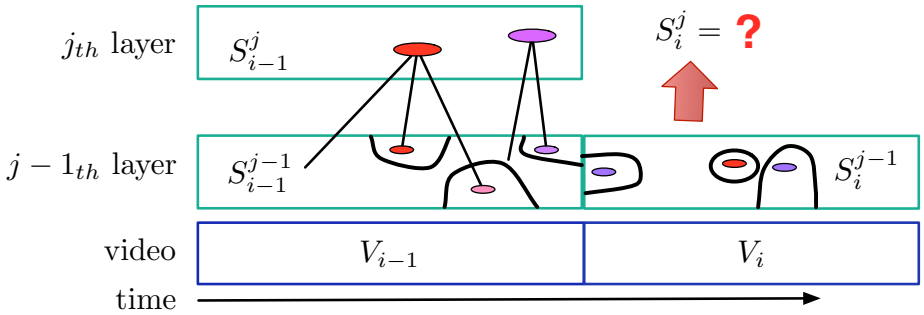


Figure 4.4: Illustration of posing Eq. 4.8 as a semi-supervised problem.

each small segment is described by a feature-vector on appearance from $V_i \cup V_{i-1}$. S_{i-1}^{j} is the segmentation result of the j_{th} layer for subsequence V_{i-1}; since it is above the $j-1_{th}$ layer, it is seen as supervised information for segments that belong to S_{i-1}^{j-1}, and these segments are hence supervised segments. The goal is to infer S_i^j: the labels of other unsupervised segments in S_i^{j-1}, with fixed labels for segments in S_{i-1}^{j-1}. Fig. 4.4 illustrates this idea.

Now we design a general semi-supervised algorithm $H_{semi-G}(S_i^{j-1})$ to infer S_i^j. Given any unsupervised grouping method $H_G(S_i^{j-1})$ that obtains segmentation result by merging small segments into large segments iteratively (see Sec. 4.3 for a graph-based example), we add one more step in its merging process to arrive at semi-supervised grouping algorithm $H_{semi-G}(S_i^{j-1})$. Assume there are two segments s_a and s_b in the current layer, they need to be merged based on the selected grouping method, this merging should be reevaluated according to three cases, which we call the **additional merging criteria** (see an example of each case in Fig. 4.5 (b-d)):

1. If s_a and s_b both are unsupervised segments, as in Fig. 4.5(b), then s_a and s_b can be merged.

2. If s_a is an unsupervised segment and s_b contains some supervised segments, as in Fig. 4.5(c), then s_a and s_b also can be merged, vice versa.

3. If s_a and s_b both contain some supervised segments, as in Fig. 4.5(d), if they

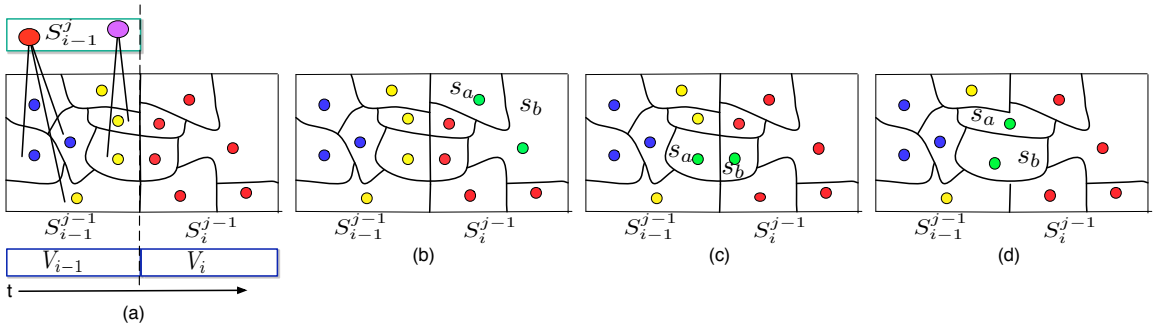


Figure 4.5: Example of the three cases in the additional merging criteria: (a), the initial status of semi-supervised grouping; (b-d) three different cases, when s_a and s_b needed to be merged: (b) $s_a \subset S_i^{j-1}$ and $s_b \subset S_i^{j-1}$; (c) $s_a \cap S_{i-1}^{j-1} \neq \emptyset$ and $s_b \subset S_i^{j-1}$; (d) $s_a \cap S_{i-1}^{j-1} \neq \emptyset$ and $s_b \cap S_{i-1}^{j-1} \neq \emptyset$

have the same parent, then they are merged, otherwise they are not merged.

Finally, when the merging process stops, we get large merged segments from S_i^{j-1}, they are considered as j_{th} layer segmentation result S_i^j of hierarchical segmentation for subsequence V_i. Then we assign unique indices to these large segments. If the large merged segment contains some supervised segments in S_{i-1}^{j-1}, its index is same as parent node's index in S_{i-1}^{j} of the supervised segment, or assign it a new index. This ensures consistent indices of same object in continuous subsequences and also proposes new indices for new objects.

There are two advantages of our semi-supervised grouping method. First, from low to high layers, more and more segments will merge with segments from the previous subsequence and the indices of these segments from the current subsequence are the same as the indices of the corresponding segments in the previous subsequence. Thus in a higher layer, the indices of the segments between neighboring subsequences are more consistent than those of a lower layer.

Second, in the third case of the additional merging criteria, we avoid merging s_a and s_b when they contain some supervised segments that connect to different parent nodes in S_{i-1}^{j}. If we do not handle this case, supervised segments in $s_a \cup s_b$

will be merged and connected to the same parent node thereby contradicting with hierarchical segmentation results S_{i-1} for previously processed subsequence V_{i-1}, and violating the streaming constraints.

4.3 Graph-based Streaming Hierarchical Video Segmentation

Felzenszwalb and Huttenlocher [47] propose a graph-based unsupervised grouping method for image segmentation (GB). Their objective is to group pixels that exhibit similar appearance. First represent the image as graph with nodes as pixels and edges connecting nodes according to their spatial relationship. The edge weight is calculated based on the similarity of two nodes. Then the method iteratively merges neighboring regions based on relaxed internal variation $RInt(\cdot)$ of regions, which we will define below. It allows regions to be merged if the weight of the edge connecting them is larger than their relaxed internal variations.

Although Grundmann et al. [65] have already extended the GB method to hierarchical form, we re-derive our version directly from [47] to put it into our streaming approximation framework; [65] do not provide a streaming version of GB. In order to adopt this graph-based grouping method into our streaming hierarchical segmentation framework, we construct a graph over the spatial-temporal video volume with a 26-neighborhood in 3D space-time. This graph is only constructed for the current two subsequences in process, V_i and V_{i-1}; this graph is the first layer of the hierarchy and edge weights are direct color similarity of voxels. Assume we have computed the streaming hierarchical segmentation result up to subsequence V_i using ideas from the previous section, i.e., given the hierarchical segmentation result S_{i-1} of V_{i-1}, we infer the hierarchical segmentation for V_i layer by layer.

Consider the j_{th} layer of V_i as an example. We build a graph $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$ based

on the $j-1_{th}$ layer of subsequences S_{i-1} and S_i, where each segment in the $j-1_{th}$ layer $\{S_i^{j-1}, S_{i-1}^{j-1}\}$ is a node x_a that is described by a histogram of Lab color of all voxels in the segment. If there is one edge between two segments/nodes $x_a \in \mathbf{V}$ and $x_b \in \mathbf{V}$ in the first layer, then there is edge $e_{ab} \in \mathbf{E}$ between segments, and its weight $w(e_{ab})$ is χ^2 distance of the corresponding two histograms. Since $S_{i-1}^{j} \in S_{i-1}$ is given, we consider S_{i-1}^{j} to be semi-supervised information for nodes in graph $\mathbf{G}$; thus, all segments/nodes in S_{i-1}^{j-1} are indexed nodes, and their index can not be changed by our semi-supervised grouping method.

According to [47], denote a region R as a subset of connected nodes in graph $\mathbf{G}$. The internal variation of region $RInt(R)$ is the maximum edge weight e_{max} of its Minimum Spanning Tree (MST) plus a relaxation term:

$$RInt(R) = Int(R) + \sigma(R), \text{ with } \quad \sigma(R) = \frac{\tau}{|R|} \tag{4.9}$$

$$\text{s.t.} \quad Int(R) = \max_{e \in MST(R)} w(e) \ , \tag{4.10}$$

where $|R|$ is the number of voxels in region R, and τ is a constant parameter that controls the number of segments to be computed. Our semi-supervised grouping method $H_{semi-G}(\cdot)$ simply combines the merging method in [47] by Eq. 4.10 and semi-supervised information from S_{i-1}^{j}. First, sort the edges in increasing order in the graph $\mathbf{G}$; and then traverse the sorted edges. At each edge, check whether the edge weight is smaller than the internal variation of both regions incident to the edge: if not, keep traversing; if so, then use the additional merging criteria to decide whether to merge or not. Therefore, based on our proposed semi-supervised $H_{semi-G}(\cdot)$, we finally obtain graph-based streaming hierarchical video segmentation. Fig. 4.6 shows some example layer-segmentations of the method on a long video (30 seconds) with an equal subsequence length $k = 10$ frames.

Figure 4.6: Example long term video StreamGBH output with $k = 10$. (a) the video with frame number on top-left, (b) the 5th layer, (c) the 10th layer, (d) the 14th layer segmentations. Faces are obscured for privacy concerns in the figure.

4.4 Experimental Evaluation

Implementation. We have implemented the graph-based streaming hierarchical video segmentation method (**StreamGBH**), which runs on arbitrarily long videos with low and constant memory consumption. Our implementation adaptively handles streaming video and can process bursts of k frames on the stream, where k can be as few as 1 or as many as the local memory on the machine can handle. When $k = all\ frames$, it is a traditional hierarchical graph-based video segmentation method (GBH) [65]. When the hierarchical level $h = 1$, it is a single layer graph-based streaming video segmentation method (**StreamGB**). When $k = all\ frames$ and $h = 1$, it is a graph-based video segmentation method (GB) by extending Felzenszwalb's graph-based image segmentation [47] to the 3D video volume. StreamGBH and all of these variants are available as part of our LIBSVX software library.

We analyze the following two aspects of StreamGBH in the context of the benchmark in Chapter III and provide details on the specific results in Sec. 4.4.1 and 4.4.2. We discuss space and time complexity in Sec. 4.4.3.

Comparison between different subsequence lengths: vary k. We explore the effectiveness of the streaming variate k in the performance of our StreamGB and StreamGBH methods. The algorithms are expected to reach their lower bound on performance when $k = 1$, and their upper bound on performance (GB/GBH) when $k = all\ frames$, respectively.

Comparison against state of the art streaming methods. We compare our StreamGB and StreamGBH methods against Grundmann's clip-based graph-based method (CGB) [65] and Paris's streaming mean shift method [145] with fixed $k = 10$. Since neither of their implementations is publicly available, we have implemented Grundmann's CGB method. For Paris's Streaming Mean Shift, we use the source code for whole video processing provided on the author's website [1] as an upper bound

[1] `http://people.csail.mit.edu/sparis/#code`

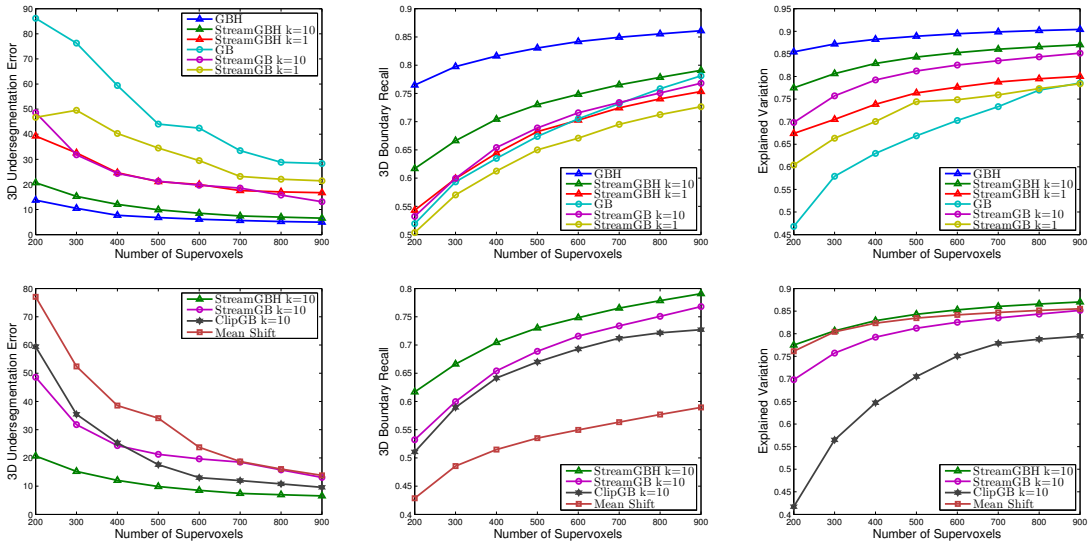


Figure 4.7: Quantitative experiments on the benchmark dataset. Left: 3D Undersegmentation Error. Middle: 3D Boundary Recall. Right: Explained Variation. Top Row: performance of Streaming GB/GBH with different k against full-video versions. Bottom Row: comparison against streaming methods.

for the streaming mean shift implementation.

4.4.1 Quantitative Performance: Benchmark Comparisons

Data. We use the benchmark dataset, BuffaloXiph [25], and a subset of evaluation metrics from Chapter III in the quantitative experiments. Specifically, the video dataset is a subset of the well-known `xiph.org` videos that have been supplemented with a 24-class semantic pixel labeling set (the same classes from the MSRC object-segmentation dataset [174]). The 8 videos in this set are densely labeled with semantic pixels and have a length of 85 frames each. This dataset allows us to evaluate the segmentation methods against human perception.

3D Undersegmentation Error. Fig. 4.7 (left column) shows the dependency of 3D Undersegmentation Error on the number of segments. The upper-left of Fig. 4.7 shows an interesting finding. The curves of StreamGBH are more smooth than StreamGB, and the performance of StreamGBH gradually converges to GBH as the

length k of incoming video subsequences increases, which agrees with our approximation framework in Sec. 4.2.1. The StreamGBH with $k = 10$ has comparative performance with GBH. However the GB method performs worst among all the methods. The reasons are twofold: (1) without a hierarchical structure, the GB method is more dependent on the input parameters for different videos which makes the curve less smooth and (2) without an over-segmentation stage, the GB method is partially dependent on an enforced minimum region size, which is not robust for videos of different lengths. According to reason (2), StreamGB method works more reasonably. The bottom-left of Fig. 4.7 shows our proposed StreamGBH method achieves the best among all the other streaming methods.

3D Boundary Recall. The 3D boundary is the shape boundary of a 3D object, composed by surfaces. It measures the spatiotemporal boundary detection: for each segment in the ground-truth and segmentations, we extract the within-frame and between-frame boundaries and measure recall using the standard formula. Fig. 4.7 (middle column) shows the dependency of 3D Boundary Recall on the number of segments. StreamGBH again performs best and StreamGB is also better than the other two in the bottom-middle of Fig. 4.7.

Explained Variation. Fig. 4.7 (right) shows the dependency of Explained Variation on the number of segments. The bottom-right of Fig. 4.7 again shows that StreamGBH, StreamGB and mean shift methods have comparative performance when the number of segments is larger than 500.

Mean Duration of Segments. When evaluating the performance among different streaming algorithms, we believe the mean duration of segments is a more important metric, as it measures the temporal coherence of a segmentation method more directly. Fig. 4.8 shows the dependency of mean duration of segments on number of segments. Our proposed StreamGBH and StreamGB both perform better than the other two. ClipGB performs worst and loses most of the temporal information.

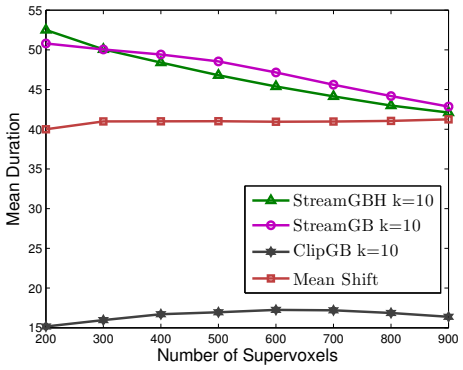


Figure 4.8: Mean duration of segments vs. number of segments.

4.4.2 Qualitative Performance on Long Videos

Here we show some qualitative results on long videos, which necessitate a streaming method. Recall, Fig. 4.6 shows a long term temporal coherence of StreamGBH. In the segmentation, objects have consistent indices along the time duration as long as the shot is not changed (see below for a shot change detection example). We see different details of an object in the layers of hierarchy. For example, one can find three papers in the violin case in 5th layer, and locate a single musician body pose in 14th layer. Note, however, the person walking behind the musician gets mixed into the musician's segment at frame 401 at higher layers in the hierarchy; this is a shortcoming of our method due to the streaming assumptions. Fig. 4.9 (top) is a shot cut from a long video with changes in illumination as a person enters a kitchen. StreamGB and StreamGBH have better temporal coherence than the others. However, the arm of the person is only able to be segmented out by StreamGBH; StreamGB has some small noise fragments and Clip-based GB completely loses the arm on Frame 513. We also show how StreamGBH captures a shot change in a video as Frame 931 to Frame 932 in Fig. 4.9 (bottom). Our proposed StreamGBH has a promising performance even without any motion cues in the above experiments.

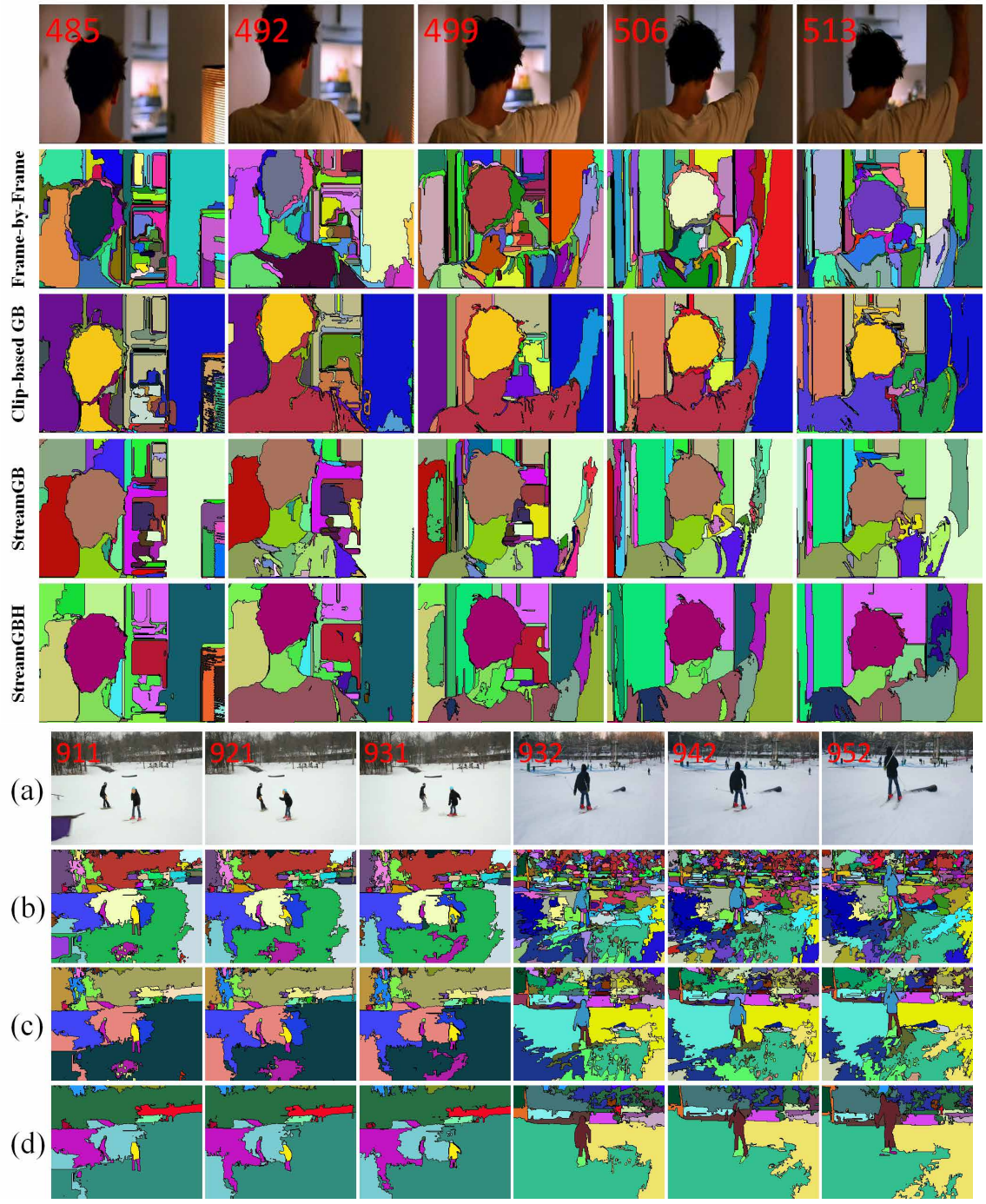


Figure 4.9: (top) Qualitative comparison of $k = 10$ for the streaming methods. (bottom) Shot change of StreamGBH with $k = 10$. (a) the video with frame number on top-left, (b) the 5th layer segmentation, (c) the 10th layer segmentation, (d) the 14th layer segmentation.

4.4.3 Space and Time Complexity

Since our method adopts streaming techniques, at any time, we only need constant memory to keep the previous and current video subsequences in the memory. This constant memory is $O(nk)$, n is the number of voxels per frame and k is the length of video subsequence window; thus it is independent of the video length. For a p-frame video, there are $n \times p$ voxels in total. The complexity of GBH is $O(np \log np)$. Since our StreamGBH considers only k frames at each time, its complexity is $O(np \log nk)$. For an arbitrarily long video, $p \gg k$ and $p \gg n$ where k and n are constant numbers, thus the complexity of StreamGBH is $O(p)$ whereas GBH is $O(p \log p)$. Therefore, StreamGBH is more efficient in both space and time complexity. For our experiments, the timing result on benchmark [214] for StreamGBH on average is 355 seconds for 85 frames (about 0.25 fps) generating 16 layer hierarchies; our method is not optimized or parallelized.

4.5 Conclusion, Limitations and Future Work

We have proposed a framework for streaming hierarchical video segmentation and a graph based streaming hierarchical approach (StreamGBH) based on our framework. To the best of our knowledge, it is the first solution for fully unsupervised online hierarchical video segmentation that can handle any arbitrarily long video (and our code is available). We evaluate this approach on a benchmark dataset under different metrics and qualitatively on some long videos. The results show that our approach outperforms other state of the art streaming methods in both quantitative and qualitative perspectives. Although our StreamGBH can process videos of arbitrary length, our implementation is not real time. In future, we plan to improve this implementation to leverage parallelization hardware and optimized data structures from data stream algorithms.

CHAPTER V

Semantic Retention in Supervoxel Segmentation

5.1 Introduction

We are drowning in video content—YouTube, for example, receives more than 400 hours of uploaded video content every minute. In many applications, there is so much video content that a sufficient supply of human observers to manually tag or annotate the videos is unavailable. Furthermore, it is widely known that the titles and tags on the social media sites like Flickr and YouTube are noisy and semantically ambiguous [184]. Automatic methods are needed to index and catalog the salient content in these videos in a manner that retains the semantics of the content to facilitate subsequent search and ontology learning applications.

However, despite recent advances in computer vision, such as the deformable parts model for object detection [46], the scalability as the semantic space grows remains a challenge. For example, the state of the art methods on the ImageNet Large Scale Visual Recognition Challenge [10] have accuracies near 20% [38] and a recent work achieves a mean average precision of 0.16 on a 100,000 class detection problem [35], which is the largest such multi-class detection model to date. To compound this difficulty, these advances are primarily on images and not videos. Methods in video analysis, in contrast, still primarily rely on low-level features [183], such as space-time interest points [98], histograms of oriented 3D gradients [85], or trajectories [202].

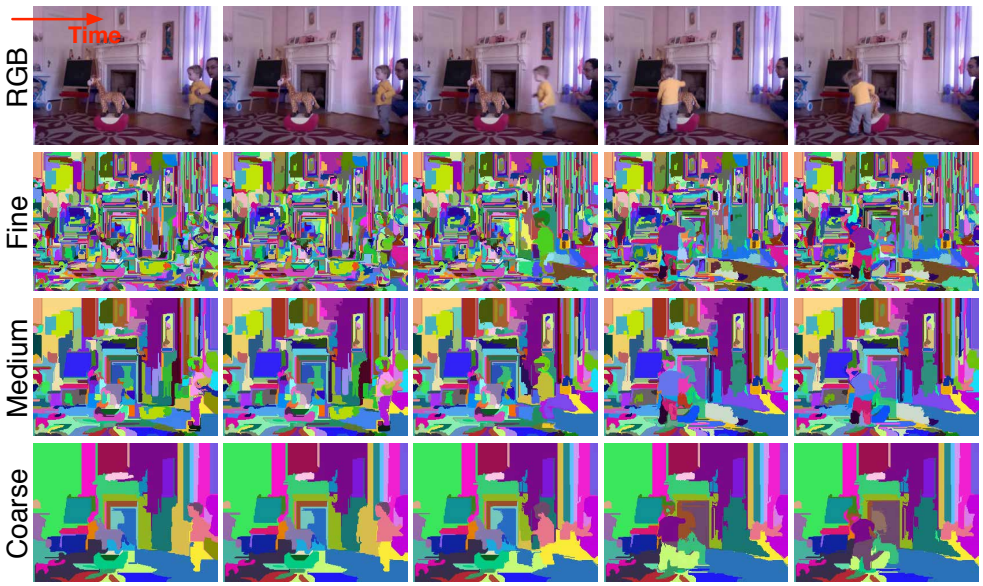


Figure 5.1: Example output of the streaming hierarchical supervoxel method in Chapter IV. From left to right columns are frames uniformly sampled from a video. From top to bottom rows are: the original RGB video, the fine segmentation (low level in the hierarchy), the medium segmentation (middle level in the hierarchy), and the coarse segmentation (high level in the hierarchy).

These low-level methods cannot guarantee retention of any semantic information and subsequent indices likewise may struggle to mirror human visual semantics. More recently, a high-level video feature, called Action Bank [163], explicitly represents a video by embedding it in a space spanned by a set, or bank, of different individual actions. Although some semantic transfer is plausible with this Action Bank, it is computationally intensive and struggles to scale with the size of the semantic space; it is also limited in its ability to deduce viewpoint invariant actions.

In contrast, segmenting the video into spatiotemporal regions with homogeneous character, such as supervoxels, has a strong potential to overcome these limitations. Supervoxels are significantly fewer in number than the original pixels and frequently surpass the low-level features as well, and yet they capture strong features such as motion and shape, which can be used in retention of the semantics of the underlying video content. Figure 5.1 shows an example supervoxel segmentation. Furthermore,

results in the visual psychophysics literature demonstrate that higher order processes in human perception rely on shape [4] and boundaries [64, 139, 142]. For instance, during image/video understanding, object boundaries are interpolated to account for occlusions [64] and deblurred during motion [142]. However, the degree to which the human semantics of the video content are retained in the final segmentation is unclear. Ultimately, a better understanding of semantic retention in video supervoxel segmentation could pave the way for the future of automatic video understanding methods.

To that end, we conduct a systematic study of how well the action and actor semantics in moving video are retained through various supervoxel segmentations. Concretely, we pose and answer the following five questions:

1. Do the segmentation hierarchies retain enough information for the human perceiver to discriminate actor and action?

2. How does the semantic retention vary with density of the supervoxels?

3. How does the semantic retention vary with actor?

4. How does the semantic retention vary with static versus moving background?

5. How does response time vary with action?

Our study presents novice human observers with supervoxel segmentation videos (i.e., not RGB color videos but supervoxel segmentation videos of RGB color videos) and asks them to, as quickly as possible, determine the actor (human or animal) and the action (one of eight everyday actions such as walking and eating). The system records these *perceptions* as well as the response time for them and then scores whether or not they match the ground truth perception; if so, then we consider that the semantics of the actor/action have been retained in the supervoxel segmentation. We systematically conduct the study with a cohort of 20 participants and 96 videos.

Ultimately, our results indicate that a significant amount of semantics have been retained in the supervoxel segmentation.

This chapter is organized as follows. Section 5.2 compares supervoxels to other video representations. Section 5.3 describes the details of the dataset acquisition and the experiment setup. Finally, Section 5.4 presents the results and our analysis thereof.

5.2 Supervoxels: Rich Decompositions of RGB Videos

Considering the example in Fig. 5.1, we observe that the hierarchy of the supervoxel segmentation captures different levels of semantics of the original RGB video. For example, one tends to recognize the humans easier from coarser levels in the hierarchy, since they are captured by fewer supervoxels; however, the coarser levels lose the detailed content in the video, such as the woman in the painting hanging on the wall, which is still captured at the medium level.

Comparing with other features, such as optical flow [178] and space-time interest points (STIP) [98], which are frequently used in video analysis [183], the supervoxel segmentation seems to retain more semantics of the RGB video (in this chapter we seek to quantify how many of these semantics are retained for one set of actions and actors). Figure 5.2 shows a visual comparison among those features. STIP uses the sampled points as the data representation—this is not the full STIP representation, which also measures gradient information—and optical flow is viewed as vectors from voxels.

By only watching the feature videos of STIP and optical flow, as shown in the bottom two rows of Fig. 5.2, it seems unlikely that humans could recover the content of a video, especially when there is little motion in a video. On the other hand, one can easily recover the content of a video by watching the supervoxel segmentation video, likely due to the fact that the supervoxel segmentation retains the shape of

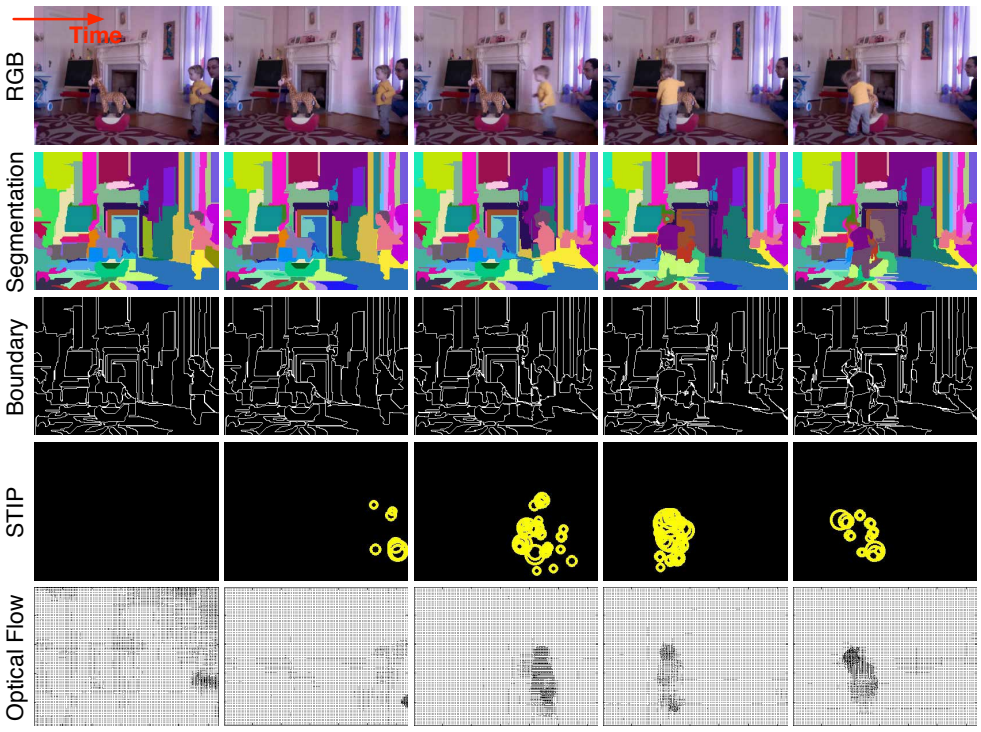


Figure 5.2: A comparison of different video feature representations. From top to bottom rows are: the RGB video, the supervoxel segmentation [221], extracted boundaries of supervoxel segmentation, space-time interest points [98], and optical flow [178].

the objects (boundaries of the supervoxel segmentation are also shown in the third row of Fig. 5.2). Zitnick and Parikh [232] show that the segmentation boundaries are in general better than classical edge detection methods, such as canny edge [20], for automatic image understanding, and they perform as well as humans using low-level cues. The precise goal of this chapter is to explore exactly how much semantic content, specifically the actor (human or animal) and the action, is retained in the supervoxel segmentation. We describe the experiment and results in the next two sections.

5.3 Experiment Setup

We have set up a systematic experiment to study actor and action semantics retention in the supervoxel segmentation. By actor we simply mean human or animal. For

Figure 5.3: A snapshot of the RGB videos in our dataset. The actors in the top two rows are humans and in the bottom two rows are animals. The dataset consists of two kinds of actors, eight actions and two types of background settings, resulting in a total of 32 videos.

action, we include a set of eight actions: climbing, crawling, eating, flying, jumping, running, spinning and walking. We have gathered a *complete* set of videos (Sec. 5.3.1) and processed them through the segmentation algorithm. Then, we show the segmentation videos to human observers and request them to make a forced-choice selection of actor and action (Sec. 5.3.3). Finally, we analyze the aggregate results over the full data cohort and quantify the retention of semantics (Sec. 5.4).

5.3.1 Dataset

Data collection. We have collected a dataset with two kinds of actors (humans and animals) performing eight different actions: climbing, crawling, eating, flying, jumping, running, spinning and walking. We only include animals that frequently appear in (North American) daily life, such as dogs, cats, birds, squirrels and horses. The backgrounds of the videos fall into two categories: static (relatively static objects such as ground and buildings with little camera changes) and moving (moving objects in the background, such as in a traffic or dramatic camera changes). A complete RGB

video dataset consists of 32 videos in total (2 actors $\times$ 8 actions $\times$ 2 background types = 32). Figure 5.3 shows a snapshot of the RGB videos we collected.

Each video is about four seconds long and the actor starts the action immediately after the video plays. We, however, show the videos at half-frame-rate when conducting the experiment to allow ample response time for the human participants. We have attempted to exclude those videos that have ambiguity with either the actors or the actions, and only use videos that have a major actor performing one single action. For example, a disqualified human jumping video usually contains the running before jumping. But, some ambiguity remains due to the general complexity of dynamic video. The dataset used in this chapter is a complete dataset having a single video for each actor, action and background type tuple. All videos were downloaded from public "wild" repositories, such as YouTube.

For each of the RGB videos, we use the method described in Chapter IV to obtain a supervoxel segmentation hierarchy. We first use ffmpeg to resize the videos to 320x240 maintaining the original aspect ratio. We sample three different levels from the hierarchy, similar as in Fig. 5.1. Therefore, the full set of data we used to run the semantic retention experiment is the 96 supervoxel segmentation videos. Note that the audio is disabled, so that the participants only have the vision perception of the supervoxel segmentation videos (and never the RGB videos).

Data split. We create a threeway split of the 96 videos into three sets: alpha, beta and gamma. Since each of the original 32 videos is represented in three levels of the hierarchy, it is imperative to make the threeway split and thereby avoid one participant seeing the same video twice but on two different supervoxel hierarchy levels. So, each of alpha, beta and gamma have the full 32 videos, but on different hierarchy levels (and uniformly varying over levels in each of the three splits). Based on the ordering in the database, alpha will start with one level (say coarse) in the hierarchy, then beta will have the next (medium) and gamma the third (fine) for one

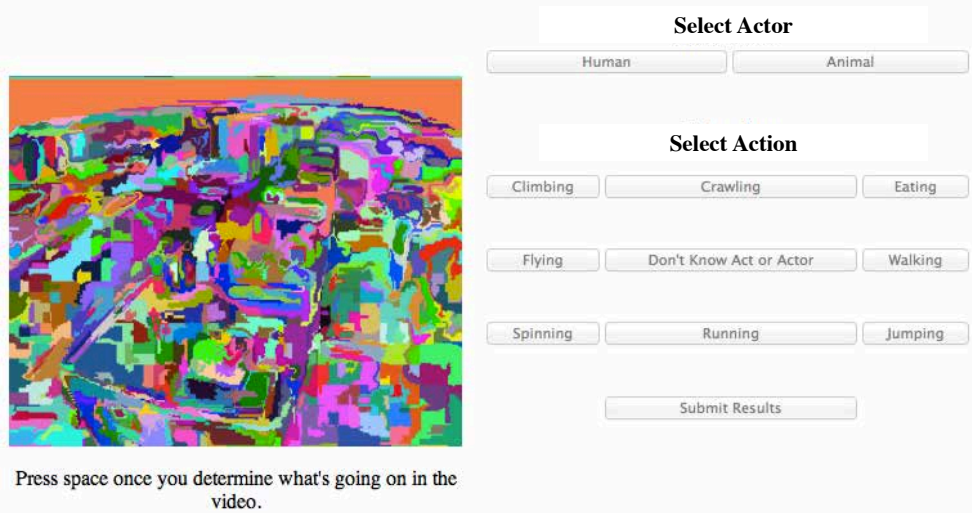


Figure 5.4: A snapshot of the user interface for the experiment.

original RGB video. For the next original RGB video, it will rotate, so that alpha has the next level (now medium), beta the next (fine) and gamma will wrap around to the first (coarse) again. This repeats through all 96 supervoxel videos. Before the videos are shown to the participant, the order of the videos is shuffled, so that the participant cannot deduce the contents based on an ordering of the videos (like human human human ... animal animal animal).

5.3.2 Study Cohort

The study cohort is 20 college-age participants. To ensure generality, we exclude those students who are studying video segmentation (and hence may have already developed an *eye* for semantic content in supervoxel segmentations). Each participant is shown one split of the videos (alpha, beta or gamma). And each participant sees a given video only once. Participants never see the input RGB videos.

5.3.3 Human User Interface and Instructions

The user interface is web-based. Figure 5.4 shows a snapshot of it. The left part of the participant's screen is the supervoxel segmentation video and the right part of

the participant's screen comprises two sets of buttons that allow the user to choose either human or animal as the actor, and to choose one of the eight actions. The participant has the option to choose *unknown* (the option "Don't know act or actor" is shown in the center of the select action area in Fig. 5.4). Such an *unknown* option prevents the participant from random selection.

Initially, when the participant starts the experiment, the left part of the screen is blank and buttons on the right side are deactivated (grayed out); once the next video in their split is downloaded locally, it prompts the user with a "ready" message. As soon as the participant presses the *space* key, it starts to show the supervoxel segmentation video and the interface triggers a timer that records the response time of the participant. The participant is required to respond by pressing the *space* key again as soon as he or she captured enough information to reach a decision (i.e., knows the actor and action in the supervoxel video). The amount of time between these two *space* key pressing is recorded as one's response time. After this second *space* key is hit, the buttons on the right side are activated and ready for the participant to select them. The participant can only watch the video once, which means once the video reaches the end, the participant is forced to make a decision (or choose unknown) without the option to watch it again. In this case, the whole video time is recorded as one's response time. This process is repeated for each video in the split (alpha, beta or gamma) until the end.

Before a participant begins, s/he is instructed briefly as to the nature of the experiment (trying to recognize the actor and action in a supervoxel video) and walked through the user interface. They are instructed that time is recorded and important; they should hence stop the video as soon as they know the answer. They are not shown any example supervoxel video before the experiment starts.

Table 5.1: Confusion matrix for actor discrimination.

	un	hu	an
unknown	0	0	0
human	0.11	0.86	0.03
animal	0.17	0.05	0.78

5.4 Results and Analysis

The response of a single video by one participant is defined as a *supervoxel perception*: $< actor, action, response\ time >$. In total, we have 640 supervoxel perceptions collected (32 videos in each split $\times$ 20 participants). The original RGB videos are used as the ground truth data to measure the *match* of the supervoxel perceptions. We also measure the response time of the participants for both *matched* perceptions and *unmatched* perceptions. Our analysis is organized systematically according to five key questions regarding semantic retention.

5.4.1 Do the segmentation hierarchies retain enough information for the human perceiver to discriminate actor and action?

Actor discrimination. Table 5.1 shows a confusion matrix of the actor discrimination. As high as 86% of the supervoxel perception correctly identifies the human actors, 78% for the animal actors, and in average 82% for actors in general. We also note that participants tend to choose the *unknown* option when they are less confident of the supervoxel segmentation. There is only a small portion of unmatched perceptions 3% and 5% that mistake human as animal or vice versa. This is hence strong evidence showing that the supervoxel segmentation indeed has the ability to retain the actor semantics from the original RGB videos. We suspect this binary discrimination performance is so high because the data cohort includes videos with one dominant actor and the human participant is able to localize this actor with the

Table 5.2: Confusion matrix for action discrimination.

	un	wl	sp	rn	jm	ea	cl	cr	fl
unknown	0	0	0	0	0	0	0	0	0
walking	0.11	0.57	0.12	0.12	0	0.01	0.01	0.04	0
spinning	0.15	0.06	0.65	0.03	0	0	0.01	0.04	0.06
running	0.01	0.07	0.07	0.79	0.04	0	0	0.01	0
jumping	0.19	0.01	0.04	0.09	0.57	0	0	0.01	0.09
eating	0.19	0	0	0	0	0.76	0.04	0	0.01
climbing	0.06	0.01	0	0	0.03	0	0.90	0	0
crawling	0.20	0.03	0	0.06	0.01	0	0.01	0.69	0
flying	0.19	0.03	0.01	0	0.01	0.01	0.03	0.03	0.70

supervoxel motion information and then use the supervoxel shape information to determine human or animal. We suspect the reason why the supervoxel perception of animal actors is less than that of human actors is because the animals in the dataset vary more broadly in relative location and orientation than the humans do.

Action discrimination. Table 5.2 shows a confusion matrix of the action discrimination. The top three scoring actions are climbing (90%), running (79%), and eating (76%), while the bottom three ones are walking (57%), jumping (57%), and spinning (65%). On average, 70.4% of supervoxel perceptions correctly match the actions. Of the lower performing actions, only walking has been easily confused with the other actions (12% to spinning and 12% to running, which may be due to semantic ambiguity—see the example of the human walking in the spinning wheel in Fig. 5.9); jumping and spinning have more *unknowns* (19% and 15% respectively) rather than being confused with other actions. An interesting point to observe is that running and climbing are perceived *unknown* significantly fewer than the other six actions.

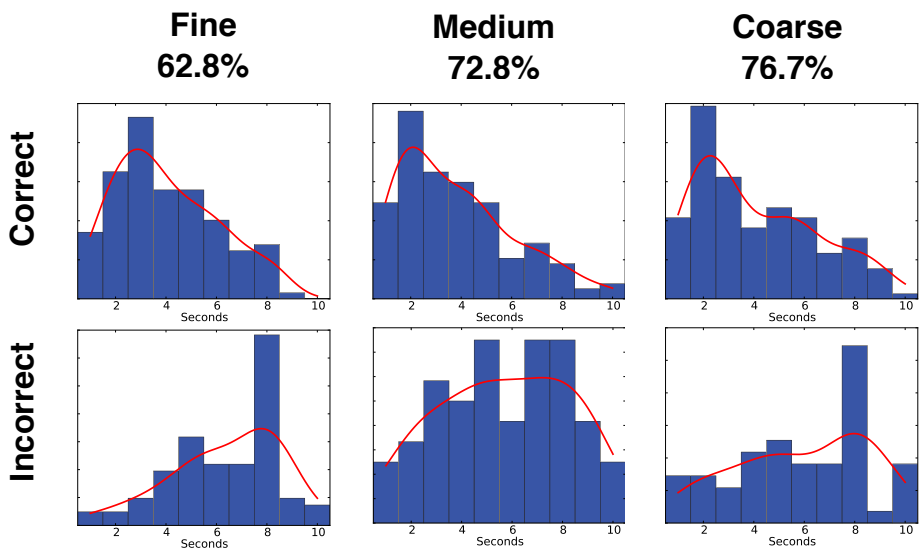


Figure 5.5: The performance of supervoxel semantic retention of actor and action on three levels from the supervoxel segmentation hierarchy: fine, medium and coarse. The percentages on top are computed when both the actor and action of supervoxel perception are correctly matched to ground truth. The middle and bottom rows are the response time figures when the supervoxel perception is correctly matched and incorrectly matched respectively.

We suspect this is due to the dominant *unidirectional* motion of these two actions. Overall, this is more strong evidence that suggests the supervoxel segmentation can well retain the action semantics from the original RGB videos.

5.4.2 How does the semantic retention vary with density of the supervoxels?

Following the discussion of supervoxel hierarchy in Sec. 5.2, we seek to understand how the supervoxel size influences retention of action and actor semantics. Recall that we sampled three levels from the supervoxel hierarchy to obtain fine, medium and coarse level supervoxel segmentations. Figure 5.5 shows the overall performance of the supervoxel perception on different levels. The percentage of correctly matched supervoxel perceptions increases when the size of supervoxels grows, suggesting that coarse segmentations more readily retain the semantics of the action and that even

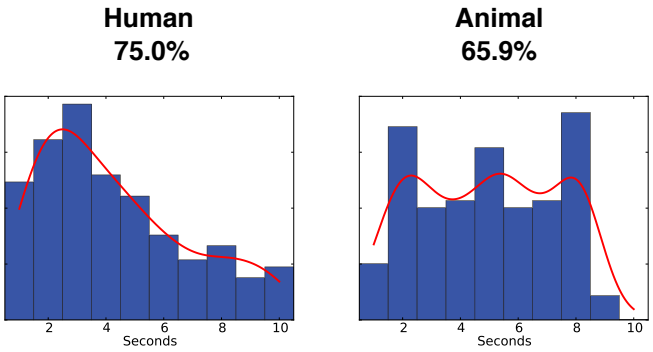


Figure 5.6: Performance comparison between human actors and animal actors. The percentages on top are computed when both the actor and action of supervoxel perception are correctly matched to ground truth. The response time plots include both correctly and incorrectly matched supervoxel perceptions.

coarser segmentations could perform better (i.e., the perfect segmentation of the actor performing the action would likely perform best). A second plausible explanation is that for actor and action discrimination the finer details in the other levels are unlikely to be needed.

We also show study of the response time in Fig. 5.5. Here, we plot the density of responses (horizontal axis is time, at half-frame-rate; vertical axis is density). The blue bars are a simple histogram and the red curve is a Gaussian kernel density estimate. For correct action matches, the response distributions are nearly equivalent, and are heavily weighted toward the shorter end of the plot, indicating that if the participant knows the answer then typically knows it quickly. However, for the incorrect matches, we see different patterns, the fine videos are peaked at about eight seconds, which is the maximum length for most videos, indicating the participant watched the whole video and still got the wrong action perception. For fine videos, one expects this due to the great number of supervoxels being perceived, which introduces more noise. The medium and coarse scales are more uniformly distributed (although the coarse scale also has a peak at eight seconds), indicating that sometimes the perception was simply wrong. This may either be due to intrinsic limitation of the supervoxels to

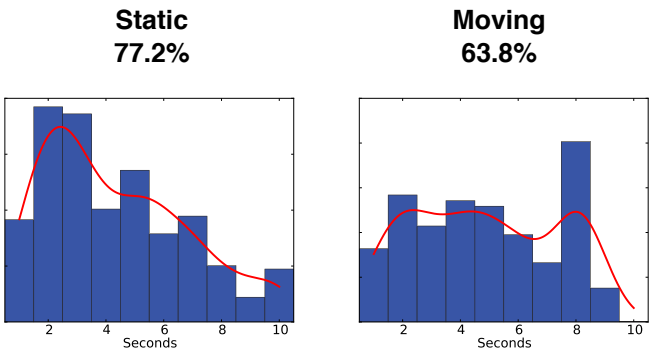


Figure 5.7: Performance comparison between static background and moving background. The percentages on top are computed when both the actor and action of supervoxel perception are correctly matched to ground truth. The response time plots include both correctly and incorrectly matched supervoxel perceptions.

retain some action semantics or due to the ambiguities of the specific videos in the dataset, which, although we did try to avoid, are present in some few cases. Further study on this point is needed to better understand the source of the error.

5.4.3 How does the semantic retention vary with actor?

We stratify the accuracy of the matches according to the actor performing the action. Figure 5.6 shows the overall performance by human actors and animal actors. In general, supervoxel perception of human actors has higher match than that of animal actors. For speed, the response time of human actors has only one peak at around three seconds, while that of animal actors has multiple peaks, which may be due to the greater variation in appearance of animals in the dataset than of humans. Moreover, human activity is easier to perceive than animal as studied by Pinto and Shiffrar [150]. Considering the result in Table 5.1, the result in Fig. 5.6 also suggests a correlation between knowing the actor and recognizing the action correctly.

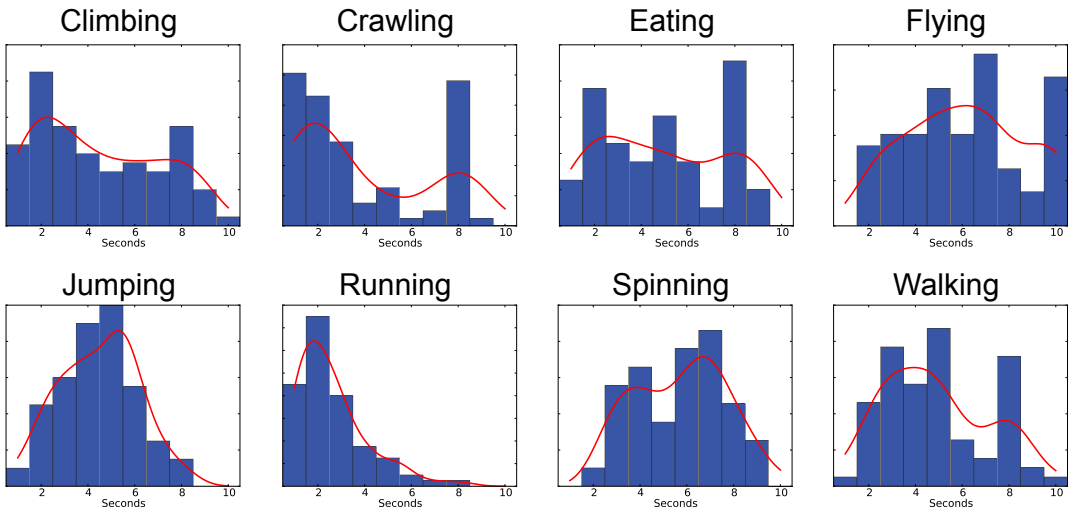


Figure 5.8: Response time of eight different actions for both correctly and incorrectly matched perceptions.

5.4.4 How does the semantic retention vary with static versus moving background?

Figure 5.7 shows the overall performance of the supervoxel perception match for static background and moving background. Supervoxel perception has higher match and shorter response time in the case of static background, as expected (the dominant actor is more easily picked out by the participant). The relatively "flat" curve in moving background suggests the response time for a single video highly depends on the specific background within that video.

5.4.5 How does response time vary with action?

Figure 5.8 shows the response time for the eight different actions. From the trend of the red curves in the figure, running and crawling get the shortest response time while the flying action takes longest. Bimodality in crawling is likely due to the very simple human baby crawling video (short response time) and very challenging cat preying video (long response time; see Fig. 5.9 for the example). The more

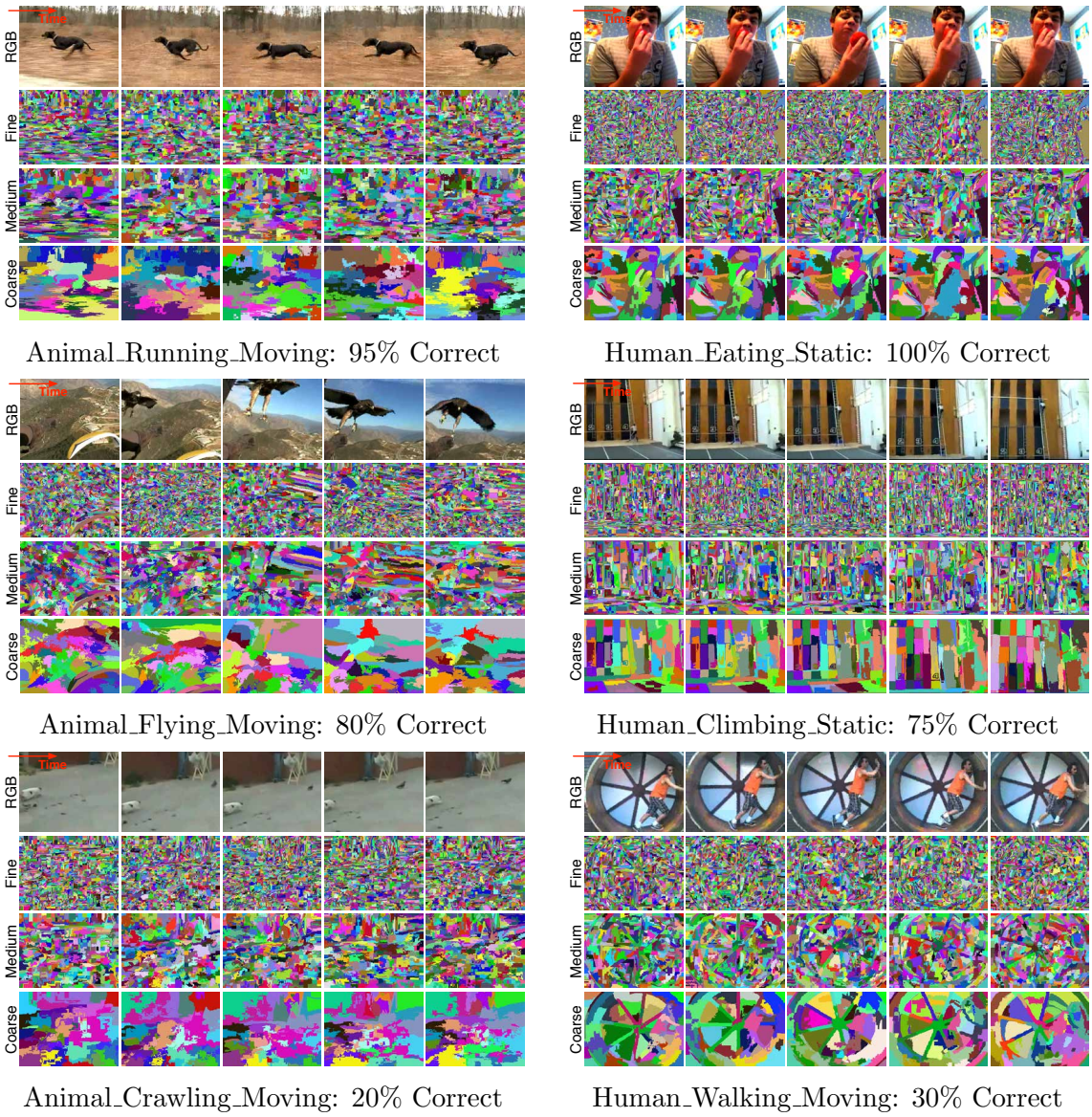


Figure 5.9: Visualization of videos with different levels of semantic retention. From top to bottom rows are videos picked from high, moderate, and low retention resepectively. Frames are uniformly sampled from each video. We notice that supervoxel motion plays an important role in helping human observers locate the actor in a supervoxel segmentation video, which is hard to see in the montages. Therefore, we encourage people to view those videos in our project website for a better visualization.

general messages behind these results are that those unusual actions such as human flying take more time to get a response, and that those actions whose semantics have been strongly retained (resulting in higher match statistics, Table 5.2) are generally

responded to more quickly than those whose semantics have less well been retained.

5.4.6 Easy, moderate and hard videos

Finally, in Fig. 5.9, we show montages of interesting videos, some with high action semantic retention and others with moderate or low retention. These top cases have distinct shape and motion properties that are readily transferred to the supervoxels; in the case of the running dog, the lateral motion is very strong. In the bottom left of retention examples, we see a cat crawling toward prey, but the cat is off-center from the camera and the participants likely dismiss this small off-center motion as noise for most of the video resulting in incorrect and slow responses. On the bottom right, we see a human walking in the spinning wheel. The human is walking; the wheel is spinning. There is likely semantic ambiguity here and further study is needed to understand the level and impact of the ambiguity.

5.5 Conclusion

In this chapter, we explore the degree to which actor and action semantics are retained in video supervoxel segmentation. We design and conduct a systematic study to answer a set of questions related to this semantic retention. Our experiment results indicate strong retention of actor and action semantics: supervoxel perception achieves 82% accuracy on actor and 70% on action. The overall finding suggests that supervoxel segmentation is a rich decomposition of the video content, compressing the signal significantly while retaining enough semantic information to remain discriminative.

The actor and action perception experiment we have reported is in a closed-world setting. In the future, we will explore the open world problem: how much semantic information is retained in supervoxel segmentation. We will let participants use lingual description to describe the supervoxel perception (such as one can recover

the woman in the painting hanging on the wall in Fig. 5.1). We will then compare this human perception to a machine perception using our recent video-to-text engine [33]. We will also collect a larger number of videos and conduct experiments with more participants to overcome the limitation of 20 participants and 32 input videos.

CHAPTER VI

Scale Selection I: Selection by Post Hoc Guidance

6.1 Introduction

In recent years, segmentation has emerged as a plausible first step in early processing of unconstrained videos, without needing to make an assumption of a static background as earlier methods have [32]. For example, the key segments work [102] proposes a method to take frame-by-frame superpixel segmentations and automatically segment the dominant moving actor in the video with category independence. Recent works in video segmentation generate spatiotemporally coherent segmentations relatively efficiently by methods like point trajectory grouping [18, 106, 53], superpixel tracking [15, 198, 209], probabilistic methods [2, 19, 101], supervoxels by minimum spanning trees [221, 65, 214], or compositing multiple constituent segmentations [151, 108].

These advances in video segmentation have also been thoroughly evaluated. Leveraging contributions in image segmentation evaluation [5] and criteria for good video segmentation [43], we have proposed the LIBSVX benchmark in Chapter III, which implements a suite of supervoxel algorithms and tests them in a set of evaluation metrics with multiple video datasets. Ultimately, it was determined that the two hierarchical agglomerative methods, Grundmann et al. [65] graph-based hierarchical method and Sharon et al. [170] segmentation by weighted aggregation, perform best

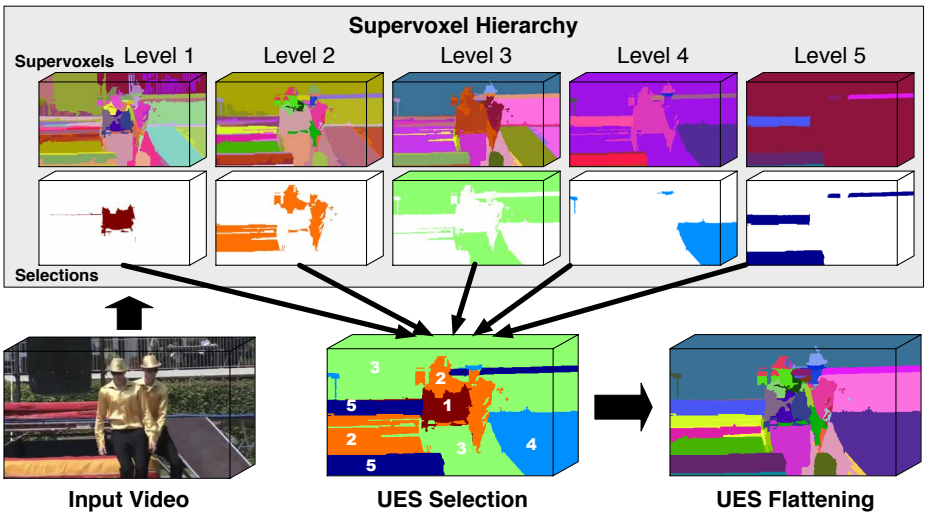


Figure 6.1: The uniform entropy slice (UES) selects supervoxels from multiple hierarchical levels in a principled way to balance the amount of information contributed by each selected supervoxel, according to some feature criterion (motion in this figure). UES Selection shows what levels are used and UES Flattening shows the final supervoxel output. Here, UES avoids oversegmentation of the background (present in Levels 1 and 2) and undersegmentation of the dancers (present in Levels 4 and 5); even just Level 3 joins the dancers' face with their shirts.

overall due to the way in which multiscale region similarity was reevaluated as the hierarchy was generated.

Despite these advances, hierarchical video segmentation has not yet been actively adopted. The hierarchies contain a rich multiscale decomposition of the video, but we are unaware of a principled approach to make use of this rich information by *flattening* it to a single non-trivial segmentation. Trivial flattenings, by arbitrarily taking a level, would carry over intrinsic limitations of the bottom-up supervoxel methods, as Fig. 6.1 illustrates. For example, taking a low level would mean very many supervoxels (oversegmentation), taking a high level would mean salient regions are missed (undersegmentation), but taking a middle level would oversegment in some regions but undersegment in others. We believe this is the key limitation to the adoption of supervoxels for early video analysis.

In this chapter, we propose the first principled solution to overcome this key lim-

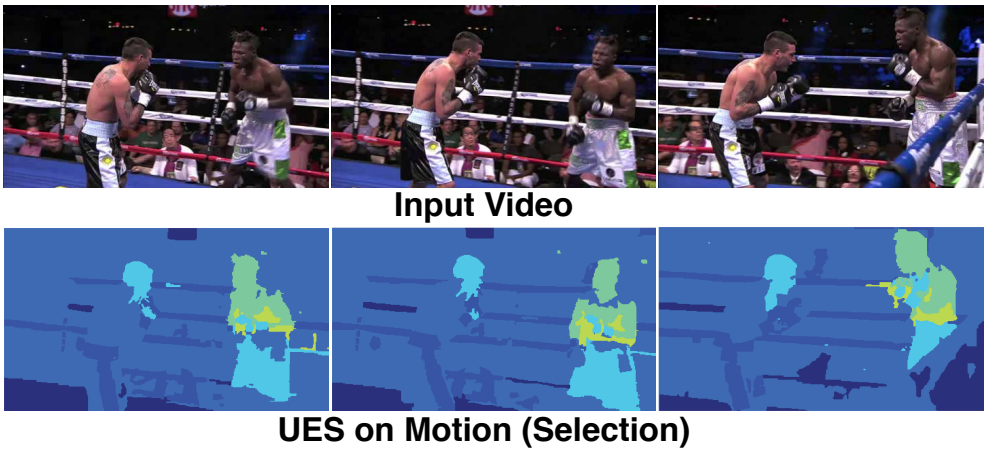


Figure 6.2: Example of supervoxel hierarchy selection by UES with a motion criterion on video *boxers*. The motion criterion drives the algorithm to select finer levels of the hierarchy (brighter regions on bottom row) on the dominant moving objects. The boxer on the right and the head of the boxer on the left are being selected from finer levels in the supervoxel hierarchy while the background segments are from coarser levels in the hierarchy. The boxer on the right (in an offensive posture) is moving much more than the boxer on the left.

itation of flattening a supervoxel hierarchy. Our emphasis is on video supervoxel hierarchies, but the core contribution is generally applicable to image and other segmentation hierarchies, given certain assumptions are met. Our approach includes a novel model—the uniform entropy slice (UES)—and a formulation for efficiently solving it via a binary quadratic program (QP). A *slice* through the hierarchy is a flattened supervoxel segmentation generally consisting of supervoxels from various levels of the hierarchy. The uniform entropy slice seeks to balance the amount of *information* in the selected supervoxels for a given feature criterion, such as motion, in which larger supervoxels from coarser-levels with less relative motion will be selected along with smaller supervoxels from finer-levels with more relative motion. Such a criterion enables us to pull out the most unique and dominant regions in a supervoxel hierarchy as shown in Figure 6.2.

The feature criterion, which drives the uniform entropy slice and hence the flattening, is independent of the supervoxel hierarchy itself. We explore four different

cases for the feature criterion underlying the uniform entropy slice: motion, object-ness, human-ness, and car-ness. Motion is an unsupervised criterion that emphasizes relatively unique motion of segments in the flattened hierarchy; the other three are supervised criteria with object-ness based on the category independent measure [3] and human- and car-ness based on trained deformable parts models [46] from PASCAL VOC [44]. The variability of these underlying feature criteria and our ultimate findings demonstrate the high degree of versatility in the proposed method: indeed it can take any form of a subsequent criterion and apply it to a previously computed supervoxel hierarchy.

We have implemented and tested the uniform entropy slice on top of the state of the art graph-based segmentation (GBH) [65] and segmentation by weighted aggregation (SWA) [170] methods. Our quantitative comparison on the SegTrack [193] dataset using the LIBSVX benchmark [214] systematically finds that UES outperforms the natural baseline of selecting a single level from the hierarchy as well as the state of the art method, SAS [108], which combines multiple segmentations. Our qualitative results demonstrate numerous clear cases in which the flattened supervoxels are precisely what is expected for various feature criteria, like human-ness.

Our code as well as the two-actor videos are available as part of the LIBSVX 3.0 software library.

6.2 Supervoxel Hierarchy Flattening Problem

Let $\mathcal{M}$ denote a given video and consider it as a mapping from the 3D lattice Λ^3 to the space of RGB colors. Each element of Λ^3 is a voxel. Based on some hierarchical supervoxel algorithm, consider an h level hierarchical oversegmentation of the video: $\mathcal{T} \doteq \{T^1, T^2, \ldots, T^h\}$ and V^i is the node set in supervoxel level T^i, $i \in [1, h]$. Individual nodes are denoted by subscripts V_s^i. The level superscript for V_s^i is dropped when the level is irrelevant. We let node V_0 be the root node of the

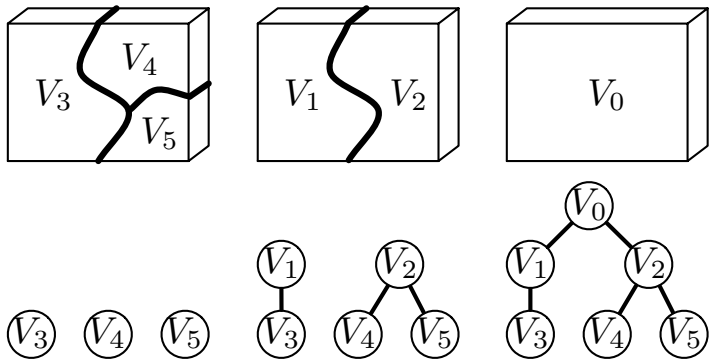


Figure 6.3: Illustration of the segmentation tree creation process. On the top of the figure, left, middle and right are bottom-up levels in a supervoxel hierarchy: T^1, T^2 and T^3 respectively. From left to middle, V_4 and V_5 are merged together, and V_3 remains itself as V_1 in the middle. From middle to right, V_1 and V_2 are merged together to a single top node V_0. The corresponding tree-graphs are in the bottom row.

supervoxel hierarchy $\mathcal{T}$, and V^1 is the set of leaf nodes in $\mathcal{T}$.

We consider only supervoxel hierarchies that are trees, i.e., each node has one and only one parent (other than the root) and each node has at least one child (other than the leaves). Figure 6.3 shows the general creation process of such a supervoxel tree; GBH generates such a tree. The algorithm initially builds a 26-connected voxel lattice over the whole video clip, then iteratively constructs a region graph over the obtained segmentation based on the minimum spanning tree merging criterion [47], and forms a bottom-up segmentation tree structure of the regions. The regions are described by their local texture histogram. The algorithm stops after a user-specified number of iterations. The algorithm tends to preserve the important region boundaries in the hierarchical merging process. We show results with both GBH and SWA, with a small modification of SWA to turn its general graph hierarchy into a tree.

Define a *tree slice* as a set of nodes from the hierarchy such that on each root-to-leaf path in the hierarchy, there is one and only one node in the slice set. Each such slice provides a plausible *flattened* hierarchy. If we combine all the nodes in the slice, then we can obtain a new composed segmentation of the original video from the

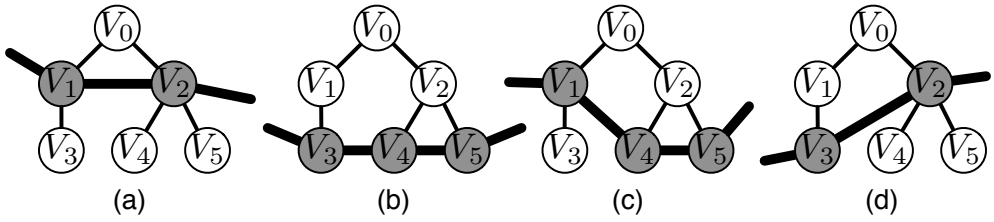
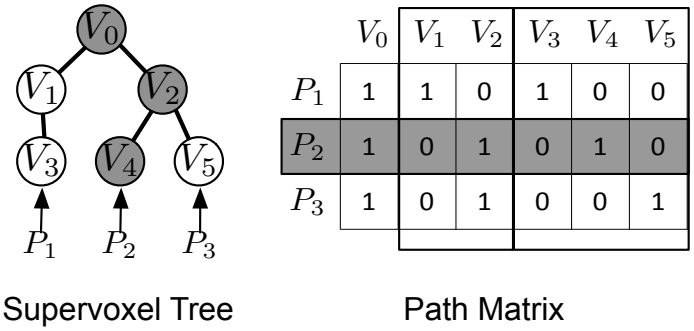


Figure 6.4: Slices in the example supervoxel tree. (a) - (d) list all 4 possible slices of the segmentation tree (excluding the root node). Each slice is highlighted as a thick black curve, and nodes on the slice are darkened.

hierarchical supervoxels. Fig. 6.4 shows example tree slices for the segmentation tree from the previous Fig. 6.3. The set of all tree slices includes both trivial (e.g., just nodes from one level) and non-trivial node selections. Note that we call this a tree slice rather than a *tree cut* to distinguish it from conventional use of the term *cut*, which generally indicates a set of edges and not nodes as we have in the slice.

More formally, consider a binary variable x_s for each node V_s in the tree $\mathcal{T}$. The binary variable x_s takes value 1 if node V_s is a part of the slice and value 0 otherwise. Denote the full set of these over the entire tree as $\mathbf{x}$. Any instance of $\mathbf{x}$ induces a selection of nodes in the tree $\mathcal{T}$, but not all instances of $\mathbf{x}$ are valid. For example, there are many instances of $\mathbf{x}$ that will select both a node and its ancestor. The trivial single-level selection is $\mathbf{x}(V^i) = 1$ and $\mathbf{x}(\mathcal{T} \setminus V^i) = 0$.

In a valid slice, each root-to-leaf path in the segmentation tree $\mathcal{T}$ has one and only node being selected. We formulate this constraint linearly. Let $\mathcal{P}$ denote a p by N binary matrix, where $p = |V^1|$ is the number of leaf nodes in $\mathcal{T}$, and $N = |\mathcal{T}|$ is the total number of nodes in $\mathcal{T}$. Each row of $\mathcal{P}$ encodes a root-to-leaf path by setting the corresponding columns for the nodes on the path as 1 and 0 otherwise. Such a matrix enumerates all possible root-to-leaf paths in $\mathcal{T}$. Fig. 6.5 shows the path matrix $\mathcal{P}$ for our example supervoxel tree from Fig. 6.3. There are three rows in the path matrix $\mathcal{P}$, which are the three root-to-leaf paths. The six columns of the path matrix $\mathcal{P}$ are the six nodes (including the root node V_0) in the segmentation tree $\mathcal{T}$. We use the



	V_0	V_1	V_2	V_3	V_4	V_5
P_1	1	1	0	1	0	0
P_2	1	0	1	0	1	0
P_3	1	0	1	0	0	1

Figure 6.5: Supervoxel tree $\mathcal{T}$ and the corresponding path matrix $\mathcal{P}$. The path P_2 is highlighted to illustrate the path matrix $\mathcal{P}$ in which each row specifies a root-to-leaf path through the tree.

path P_2 to illustrate the values on a row of $\mathcal{P}$—nodes $\{V_0, V_2, V_4\}$ are on path P_2.

For a given tree $\mathcal{T}$, we compute the path matrix $\mathcal{P}$ by a breadth-first search with complexity $O(ph)$. The size of $\mathcal{P}$ is tractable for typical videos: the number of rows is exactly the number of leaves in the tree, which is either the number of voxels or some number of supervoxels of the smallest scale maintained in the tree (in the case the full tree is not used); the number of columns is typically a factor of two on the number of rows due to the agglomerative nature of the supervoxel methods.

To ensure a tree slice is a valid, we have

$$\mathcal{P}\mathbf{x} = \mathbf{1}_p \ , \tag{6.1}$$

where $\mathbf{1}_p$ is an p-length vector of all ones. This linear constraint ensures that every root-to-leaf path (row of matrix $\mathcal{P}$) has one and only one node in the slice $\mathbf{x}$. If there is more than one node being selected in P_i, then $P_i\mathbf{x} > 1$. If there is no node being selected in P_i, then $P_i\mathbf{x} = 0$. The valid selection $\mathbf{x}$ is called a tree slice.

6.3 The Uniform Entropy Slice

The previous section presents the tree slice problem and a linear constraint to assess the validity of a slice; here we present a new model based on uniform entropy

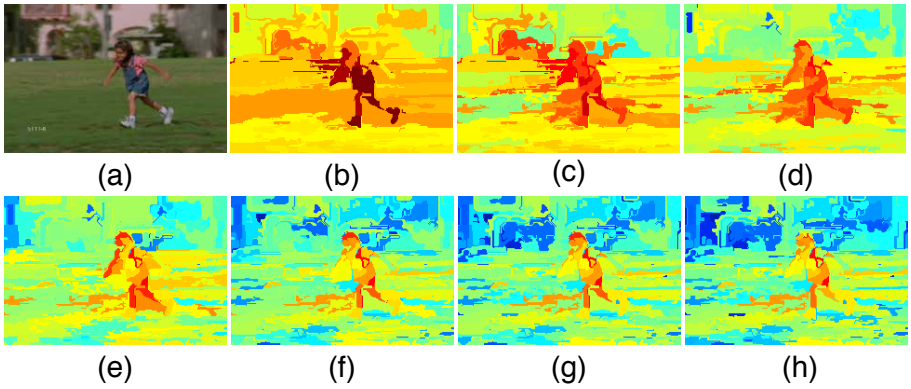


Figure 6.6: Example hierarchy node entropy for the motion feature criterion. (a) is the raw video *girl* from SegTrack, (b: coarse) – (h: fine) are node entropy from various levels in the hierarchy. The entropy color from dark blue to dark red maps entropy changing from low to high (using the *jet* colormap in Matlab). Notice how the entropy of the girls limbs is relatively higher than that of the background for corresponding hierarchy levels.

to quantify a slice. The intuitive idea behind the uniform entropy slice is that we want to select nodes in the tree that balance the information contribution to the overall slice. We are inspired by the Uniform Frequency Images work of Hunter and Cohen [74]. Their model is proposed for image compression; they automatically generate an invertible warping function that downshifts the image's highest spatial frequencies in exchange for upshifting some of its lowest spatial frequencies, producing a concentration of mid-range frequencies. In other words, the compressed image is able to focus more bits on the parts of the image that have a higher frequency signal than those with a lower frequency signal.

In our case for supervoxel hierarchies, one can relate finding the best slice in a hierarchy to such a compression problem; we want a slice that is able to retain the greatest amount of information relative to the number of selected supervoxels: select bigger supervoxels from coarse levels when there is little information content and conversely, select smaller supervoxels from fine levels when there is high information content.

Information content is specified relative to a certain feature criterion, such as

motion or human-ness. We specify four such feature criteria later in Sec. 6.3.2 and experiment with them in Sec. 6.4. For the current discussion, assume we have a feature criterion $\mathcal{F}(\cdot)$ that maps a node V_s to a discrete distribution over the feature range. For example, consider an unsupervised motion feature criterion in which we want the slice to focus on regions of the video that are moving uniquely relative to the rest of the video—e.g., a girl running leftward while the camera slowly pans as in Fig. 6.6. In this case, we compute optical flow over the video and then compute a bivariate discrete distribution over a set of flow magnitudes and flow directions for $\mathcal{F}$.

The information content of each node V_s in the hierarchy is computed by the entropy over $\mathcal{F}(\cdot)$:

$$E(V_s) \doteq -\sum_{\gamma} P_{\mathcal{F}(V_s)}(\gamma) \log P_{\mathcal{F}(V_s)}(\gamma) \ , \tag{6.2}$$

with γ varying over the bivariate discrete feature range.

We next propose the uniform entropy objective, which considers the node information content according to Eq. 6.2 and seeks a tree slice that balances the overall information content of the selected nodes. Again, consider a valid tree slice $\mathbf{x}$ which is a vector of binary variables with one x_s for each node V_s in the hierarchy taking value 1 if the node is on the slice and 0 otherwise. The uniform entropy objective hence seeks a valid tree slice that minimizes the difference in entropy of selected nodes:

$$\mathbf{x}^* = \text{argmin} \sum_{V_s, V_t \in \mathcal{T}} |E(V_s) - E(V_t)| x_s x_t \ . \tag{6.3}$$

where the minimization is over valid tree slices $\mathbf{x}$.

The intuition behind the uniform entropy objective is twofold. First, in a common case, the entropy of a supervoxel in coarser levels of the hierarchy drops down when the segment breaks up into smaller pieces at finer levels. Again consider Fig. 6.6, which shows the node entropy for a motion feature criterion on the video *girl* from

the SegTrack dataset [193]. It is clear that the node entropy generally decreases from coarser to finer levels, and those informative supervoxels (the girl in this case) have overall more motion entropy than the background. It is hence plausible the slice will select nodes around the girl at finer levels to match similar motion entropies to the background at coarser levels in the hierarchy. Second, regions of the video that are salient for the specified feature criterion tend to have higher entropy than non-salient regions because of articulation and variability of the features near the salient region boundaries. Hence, when selecting the supervoxels, our goal is to preserve the detail in the segmentation of these salient regions and less so in the non-salient regions.

6.3.1 Uniform Entropy Slice as a Binary QP

Directly minimizing Eq. 6.3 is complex because it requires enumerating all valid tree slices and includes a degenerate minimum which selects the root node only. We instead reformulate the objective as the following binary quadratic program, which we call the *uniform entropy slice*.

$$\begin{aligned} \text{minimize} \quad & \sum_s \alpha_s x_s + \sigma \sum_{s,t} \beta_{s,t} x_s x_t \\ \text{subject to} \quad & \mathcal{P}\mathbf{x} = \mathbf{1}_p \\ & \mathbf{x} = \{0,1\}^N \end{aligned} \tag{6.4}$$

where α_s forms a vector with length equal to N, $\beta_{s,t}$ is an entry in an N by N matrix, and σ controls the balance between the two terms. Note the $\mathcal{P}\mathbf{x} = \mathbf{1}_p$ slice validity constraint from Eq. 6.1. Furthermore, note that there is no explicit notion of neighborhood in the uniform entropy slice, but $\beta_{s,t}$ can be specified based on the neighborhood structure in the tree.

The linear term makes the slice prefer simpler segmentations when possible, i.e., prefer coarser levels in the hierarchy rather than finer levels in the hierarchy. The

following is the unary potential we set:

$$\alpha_s = |V^i| \quad \text{if } V_s \in V^i \ , \tag{6.5}$$

where $|V^i|$ means the total number of supervoxels in ith level of the tree. In typical supervoxel trees, there is a quadratic relationship between $|V^i|$ and $|V^{i+1}|$ due to algorithm construction.

The quadratic term implements the uniform entropy objective

$$\beta_{s,t} = |E(V_s) - E(V_t)||V_s||V_t| \tag{6.6}$$

where $|V_s|$ and $|V_t|$ denote the volume of the supervoxels V_s and V_t respectively. Although nodes in the coarser levels of the tree have relatively higher entropy than nodes in the finer levels, the number of coarser level nodes is dramatically less than those in the finer levels. By adding the volume factors, we push the selection down the hierarchy unless a uniform motion entropy has already been achieved. Indeed this behavior has generally been observed in our quantitative and qualitative experiments. See, for example, the level of the hierarchy selection for the video *girl* in Fig. 6.9 in Sec. 6.4.

We solve the QP using a standard solver (IBM CPLEX), but note that other approaches to solving it are plausible, such as spectral relaxation [104].

6.3.2 Feature Criteria

The uniform entropy slice operates directly on a supervoxel hierarchy that was computed by an unsupervised method such as GBH. However, the feature criteria, which drive the tree slice optimization, provide a doorway to apply situation-specific guidance post hoc. To illustrate this versatility, we describe four such feature criteria that span the spectrum of unsupervised to class-specific supervised. Each of these

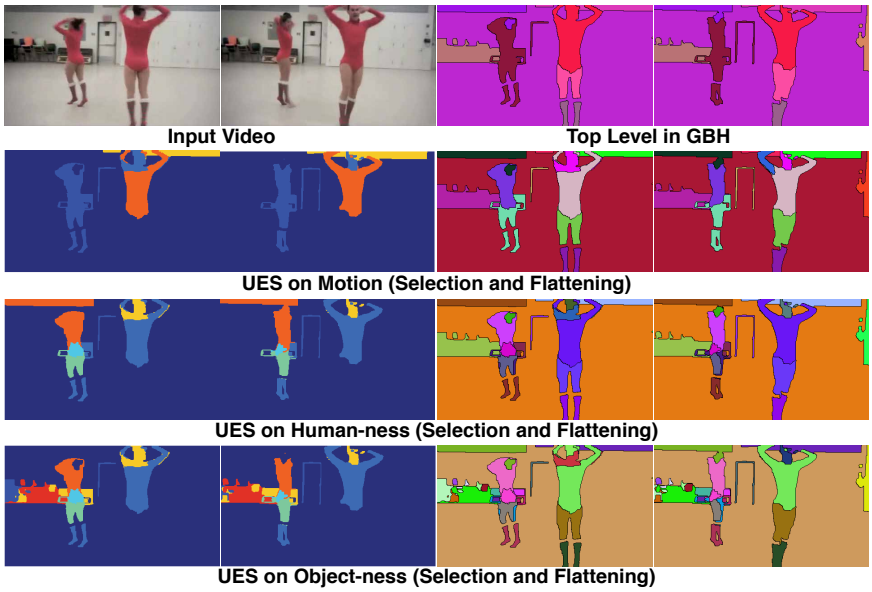


Figure 6.7: Different feature criteria focus on different parts of the video *dancers*. Here, the motion feature focuses mostly on the dominant man in front and some attention to the woman in the back. On the other hand, the human-ness criterion focuses on both dancers, while the object-ness also focuses on the chairs in the back. All these feature criteria try to avoid undersegmentation of interesting objects as shown in the top level in GBH (the woman merged with the door and bench in the back), and maintain a uniform clean background. In the UES Selection images (left two columns), the dark red to dark blue means the finer levels to coarser levels in the supervoxel hierarchy tree.

have been implemented and used in our experiments (Sec. 6.4). In Figure 6.7, we show the different feature criteria on one video and observe how different slices are computed with criterion-specific foci of attention.

Unsupervised: Motion. The motion feature criterion has been discussed as an example earlier and we hence do not describe it in detail here. For computing the feature we use the Liu [114] optical flow method and compute flow on each frame of the video. For the map $\mathcal{F}$ we discretize the range to four magnitude bins and eight angular bins.

Supervised, Category-Independent: Object-ness. This demonstrates a general category-independent, object-ness feature, as it is common in problems like video object segmentation [102]. We sample 1000 windows per frame using [3] according to

their probability of containing an object. Then we convert this measure to per-pixel probabilities by summing the object-ness score over all windows covering a pixel, and normalizing the result over the video, which is similar to [200]. We use six quantization levels for $\mathcal{F}$.

Supervised: Human-ness and Car-ness. The last two feature criteria are class-specific and demonstrate further post hoc flattening goals. We use the state of the art deformable part based model [46] with previously trained PASCAL VOC detectors to compute car-ness and human-ness. We use a low detection threshold to get more detection bounding boxes for each frame. Then, similar to object-ness, we count the per-pixel detection hits to obtain a detection hit map for each frame. We again set six quantization levels for $\mathcal{F}$.

6.4 Experiments

We evaluate the uniform entropy slice (UES) both quantitatively (Sec. 6.4.1) and qualitatively (Sec. 6.4.2) on various benchmark and new, challenging unconstrained videos. To explore the generality of UES, we apply it to supervoxel hierarchies generated by two different methods, GBH [65] as implemented in [214] and SWA [170] as implemented in [31]. For GBH, we construct a supervoxel tree directly from its output supervoxel hierarchy, since the method itself generates a tree structure. For SWA, we simplify the supervoxel hierarchy, which is a general directed acyclic graph, to a tree structure by taking the most dominant parent for each child node and denote this variant of SWA as SWA^T.

The most important parameter in UES is the ratio σ between the linear and quadratic terms. However, we have observed that, in practice, the relative hierarchy selection of supervoxels is not very sensitive to it. We L-1 normalize both of these terms and in our quantitative experiments, we empirically set $\sigma = 10$ for all the videos.

Video	SWA^T Flattening												GBH Flattening											
	3D ACCU			3D UE			3D BR			3D BP			3D ACCU			3D UE			3D BR			3D BP		
	BASE	SAS	UES	BASE	SAS	UES	BASE	SAS	UES	BASE	SAS	UES	BASE	SAS	UES	BASE	SAS	UES	BASE	SAS	UES	BASE	SAS	UES
birdfall2	9.0	0.0	**69.7**	36.8	38.3	**26.5**	82.1	81.9	**84.9**	0.66	0.65	**0.70**	1.8	0.0	**53.8**	26.9	27.1	**23.2**	74.3	74.0	**82.1**	0.83	0.83	**0.94**
cheetah	0.0	0.0	0.0	47.4	47.4	47.4	65.7	65.7	65.7	1.93	1.93	1.93	30.2	30.2	**39.4**	**31.7**	32.4	34.1	78.3	**79.3**	75.3	1.42	1.43	**1.60**
girl	**56.4**	55.9	56.1	7.8	8.2	**5.9**	56.6	56.5	**57.7**	3.36	**3.39**	3.31	41.9	**45.6**	41.9	11.2	**11.1**	13.7	54.4	54.1	**58.1**	2.90	2.91	**3.94**
monkeydog	0.0	0.0	0.0	52.0	52.2	**51.9**	84.9	**86.8**	86.7	3.32	3.12	**3.35**	71.9	**79.9**	**79.9**	37.1	**36.6**	43.2	90.7	90.9	**91.0**	2.55	2.47	**2.95**
parachute	83.7	85.5	**90.3**	23.6	24.4	**22.3**	93.2	93.0	**94.6**	1.66	1.69	**1.72**	89.4	89.4	89.4	38.6	38.6	38.6	87.4	87.4	87.4	10.0	10.0	10.0
penguin	**94.7**	94.4	94.4	**1.8**	1.9	**1.8**	**73.7**	72.3	71.0	1.36	**1.37**	1.27	84.7	83.1	**85.0**	2.2	1.9	1.8	**66.7**	65.4	65.5	**1.10**	0.96	0.88
AVERAGE	40.6	39.3	**51.8**	28.2	28.7	**26.0**	76.0	76.0	**76.8**	**2.05**	2.03	**2.05**	53.3	54.7	**64.9**	**24.6**	**24.6**	25.8	75.3	75.2	**76.6**	3.14	3.11	**3.39**

Table 6.1: Quantitative comparison of UES against the other two baseline methods on SegTrack dataset. We evaluate on two different hierarchical supervoxel methods: SWA^T and GBH. The leading scores of each metric per video are in bold font.

6.4.1 Quantitative Evaluation

Benchmark and Dataset. We use the recently published supervoxel benchmark LIBSVX [214] to evaluate the UES with GBH and SWA^T methods. The benchmark provides six supervoxel methods and a set of supervoxel evaluation metrics. We use the SegTrack dataset from Tsai et al. [193], which provides a set of human-labeled single-foreground objects with six videos stratified according to difficulty on color, motion and shape.

Baseline Methods. We compare with two baseline methods. The first is a simple trivial slice that takes a single level from the hierarchy, which we denote as "Base" in Table 6.1. Another method is a video extension of Segmentation by Aggregating Superpixels (SAS) [108], which composites multiple segmentations together based on bipartite graph matching. It achieves state-of-the-art performance on the image Berkeley Segmentation Database [128]. To the best of our knowledge, we are not aware of other video supervoxel selection algorithms. The number of supervoxels from the input hierarchy varies from less than 10 to about 800. For fair comparison, we feed SAS and UES with the unsupervised motion feature only. The scores in Table 6.1 are generated for the same number of supervoxels for all three methods per video. The scores of "Base" are generated by linear interpolation of nearby levels as in [214].

3D Segmentation Accuracy measures the average percentage area of the ground-truth segments being correctly segmented by the supervoxels. 3D Undersegmentation Error measures the fraction of voxels that go beyond the boundary of the ground-truth when mapping the supervoxels onto it. Along with 3D Boundary Recall, we add 3D Boundary Precision as a new metric. Overall, the proposed UES achieves better performance for both SWA^T and GBH supervoxel hierarchies than the other two baseline methods, and in some cases, such as 3D ACCU the improvement is significant for both methods. We note that neither the baseline one level selection

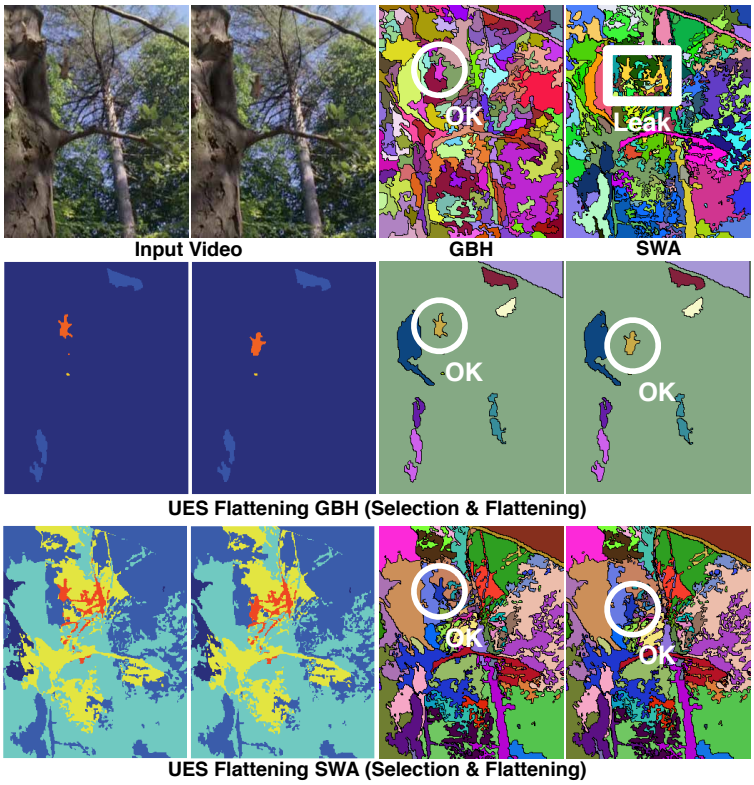


Figure 6.8: UES helps avoid foreground undersegmentation and background oversegmentation on video *birdfall2*. GBH and SWA on the top row show the middle levels from each hierarchy. A white circle means the bird has no segmentation leak, whereas a white rectangle means a segmentation leak with the surrounding tree branches. Here, we use the motion criterion.

nor the SAS can correctly segment the video "birdfall2" with only a small number of supervoxels. In some cases, such as the video "cheetah" using SWA^T, the scores are frequently the same for the three methods; this is a failure case of the overall supervoxel hierarchies, which we have observed to have little variation in supervoxels covered on the object at multiple levels in the hierarchy.

6.4.2 Qualitative Evaluation

UES on Motion. Figure 6.8 is an example video showing that UES can help avoid foreground undersegmentation and background oversegmentation. UES selects the coarse levels of the hierarchy for the background when doing so does not lead to foreground segments leaking, as in GBH. Similarly, UES pushes the foreground and

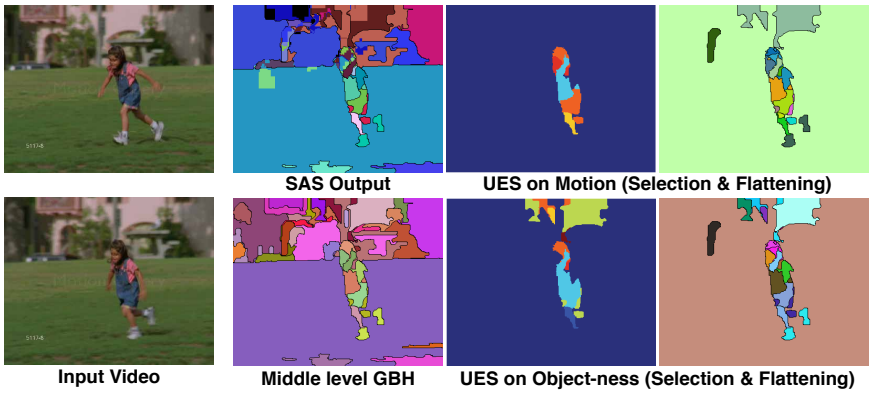


Figure 6.9: Comparison of UES against baseline methods on video *girl* from SegTrack. UES on Motion and SAS (based on motion) have identical number of supervoxels in their final outputs. We also show a simple selection of the middle level from GBH as well as UES on Object-ness for comparison.

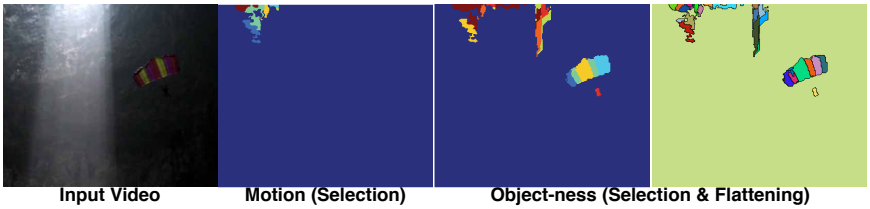


Figure 6.10: UES on Object-ness selects the *parachute* segments and the human, while UES on Motion fails.

the corresponding leaking parts of the video down to the finer levels of the SWA hierarchy, while it still keeps the other background regions in the coarser levels of hierarchy.

UES vs. Baselines. In Figure 6.9, the girl is running leftward, and the camera is also slowly moving leftward. The position of the girl in the video does not change much, but the pose changes drastically. The articulated pose generates more motion entropy over time than the surroundings do, which also allows UES to focus on the girl, as shown on the right half of the figure with both motion and object-ness criteria. In contrast, a simple selection of a middle level from GBH gives a quite fragmented background. If we take a coarser level of the hierarchy, then the girl is merged too much with the grass in background. SAS does merge the supervoxels, but it lacks a focus on selection.

Object-ness vs. Motion. Sometimes, the motion criterion fails when the rigid objects have same motion as the camera in a video, or in a video with chaotic motion. The object-ness can better handle the above situations. We show an example in Figure 6.10, where the motion completely fails to select the rigid object *parachute*, because the motion of it is uniform over the video (from left to right) with the camera. However, with the object-ness criteria, the algorithm can easily select it from the lower levels in the hierarchy. The supervoxels in the top part of the object-ness selection image may seem to be errors, but indeed, these are expected: the parachute moves from left to right across the light and these selected supervoxels touch it at an earlier frame when it was passing by.

Human-ness and Car-ness. Recall that Figure 6.7 shows an example of how different feature criteria drive the algorithm to focus on different parts of a video. The top level hierarchy in GBH mistakes the woman in the left with the door and bench in the background. With the motion criterion, UES selects the man in the front from a finer level than the woman in the back, since the man is the most dynamically moving part of the video. Interestingly, the human-ness focuses on the two dancers while the object-ness not only focuses on the two dancers but also on the chairs in the back. Figures 6.11 and 6.12 further demonstrate examples of the supervised feature criteria in comparison to the motion criterion; in both cases the unsupervised motion criterion slices as well as the trained feature criterion suggesting the unsupervised measure may be as useful as the trained ones, at least in cases of relatively static backgrounds.

6.5 Discussion and Conclusion

Summary. Supervoxel segmentation has gained potential as a first step in early video processing due to advances in hierarchical methods [65], streaming methods in Chapter IV and related evaluations in Chapter III. However, the high-performing

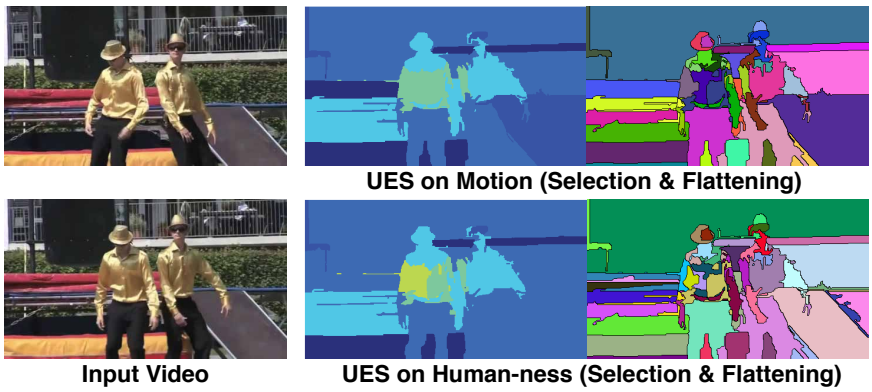


Figure 6.11: UES on Motion and Human-ness on video *danceduo*.

methods generate a hierarchy of supervoxels that often renders the user with more questions than at the outset due to the intrinsic limitations of unsupervised grouping. We have proposed the first principled method to flatten the hierarchy, called the uniform entropy slice (UES). Our method seeks to balance the level of information across the selected supervoxels: choose bigger supervoxels in *uninteresting* regions of the video and smaller ones in *interesting* regions of the video. A post hoc feature criterion is used to drive this information selection, and is independent of the original supervoxel process. Our experiments demonstrate strong qualitative and quantitative performance.

Generality. Although this chapter has strictly discussed video supervoxel hierarchies thus far, the proposed method is general and can directly be applied to other segmentation hierarchies, such as those on images [170] or even a hierarchical clustering on top of existing trajectories [18, 202], so long as two assumptions are met. First, the hierarchy must be a tree (or adequately transformed into one as we did for SWA in this chapter). Second, a feature criterion can be defined to drive the slice.

Implications to Related Video Problems. The proposed uniform entropy slice makes it plausible to provide an initial supervoxel map for further processing in problems like video object segmentation. In particular, every video object segmentation method we are aware of [102, 227, 126] begins with an oversegmentation (typically

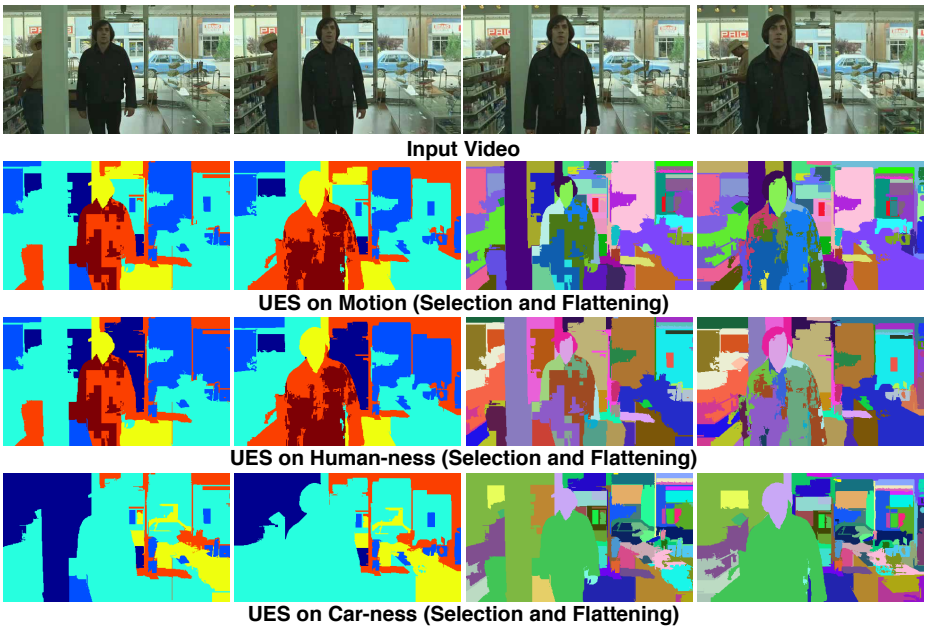


Figure 6.12: UES on Motion, Human-ness and Car-ness on video *nocountryforoldmen* from [65]. For Motion and Human-ness, the moving man is selected from the finer levels, while most others are from coarser levels. For car-ness, the car and nearby regions are selected from finer levels. The red selection around the window is to avoid leaks.

frame-level superpixels) and extracts a single moving foreground object. We expect our flattened output to provide a strong input for such methods as the community moves from single to multiple objects. Second, our method of using any feature criterion is more general than the existing strictly object-ness criterion that has thus far been used in video object segmentation. And, this has strong implications as the community begins to consider semantic video segmentation on unconstrained videos, which is a relatively new problem in video that has thus far focused on constrained videos [16, 24].

Limitations. The feature criterion is independent of the supervoxel method. In some respects, this fact is a clear benefit of the method, but it can also be considered a limitation: there is no guarantee that the uniform entropy slice is the optimal supervoxel segmentation for a given video and feature criterion. In other words, since the supervoxel hierarchy is computed independent of the feature criterion, its segments

may not coincide with the natural ones for a given criterion. Our experiments demonstrate that for typical feature criteria this limitation is not critical, but further work is needed to better understand the induced error for a feature criterion-hierarchical supervoxel method pair.

Future Work. In the future, we plan to extend UES into a streaming setting to handle longer videos [221]. A key hurdle to overcome will be the tractability of the subsequent NP-hard quadratic program; we plan to pursue adequate approximations in this streaming case.

CHAPTER VII

Scale Selection II: Joint Selection and Video Labeling

In this chapter, we first introduce a new video understanding task—the actor-action video understanding, and a set of methods to model the interplay of actors and actions in Sec. 7.1. Then we describe the novel method for joint scale selection and video labeling, called grouping process model, and show its superior performance on the actor-action dataset.

7.1 Actor-Action Video Understanding

Like verbs in language, action is the heart of video understanding. As such, it has received a significant amount of attention in the last decade. Our community has moved from small datasets of a handful of actions [61, 167] to large datasets with many dozens of actions [156, 90]; from constrained domains like sporting [161, 138] to videos in-the-wild [117, 156]. Notable methods have demonstrated that low-level features [203, 98, 204, 86], mid-level atoms [231], high-level exemplars [163], structured models [138, 187], and attributes [116] can be used for action recognition. Impressive methods have even pushed toward action recognition for multiple views [127], event recognition [76], group-based activities [96], and even human-object interactions [67,

148].

However, these many works emphasize a small subset of the broader action understanding problem. First, aside from Iwashita et al. [75] who study egocentric animal activities, these existing methods all assume the agent of the action, which we call the *actor*, is a human adult. Although *looking at people* is certainly a relevant application domain for computer vision, it is not the only one; consider recent advances in video-to-text [66, 9] that can be used for semantic indexing of large video databases [109], or advances in autonomous vehicles [59]. In these applications, understanding both the actor and the action are critical for success: e.g., the autonomous vehicle needs to distinguish between a child, a deer and a squirrel running into the road so it can accurately make an avoidance plan. Applications like these, e.g., robotic autonomy [186], are abundant and growing.

Second, these works largely focus on *action recognition*, which is posed as the classification of a pre-temporally trimmed clip into one of k action classes from a closed-world. The direct utility of results based on this problem formulation is limited. The community has indeed begun to move beyond this simplified problem into action detection [229, 187], action localization [79, 125], action segmentation [83, 81], and actionness ranking [27]. But, all of these works do so strictly in the context of human actors.

We overcome both of these narrow viewpoints and introduce a new level of generality to the action understanding problem by considering multiple different classes of actors undergoing multiple different classes of actions. To be exact, we consider seven actor classes (*adult*, *baby*, *ball*, *bird*, *car*, *cat*, and *dog*) and eight action classes (*climb*, *crawl*, *eat*, *fly*, *jump*, *roll*, *run*, and *walk*) not including the no-action class, which we also consider. We formulate a general actor-action understanding framework and implement it for three specific problems: actor-action recognition with single- and multiple-label, and actor-action semantic segmentation. These three problems cover

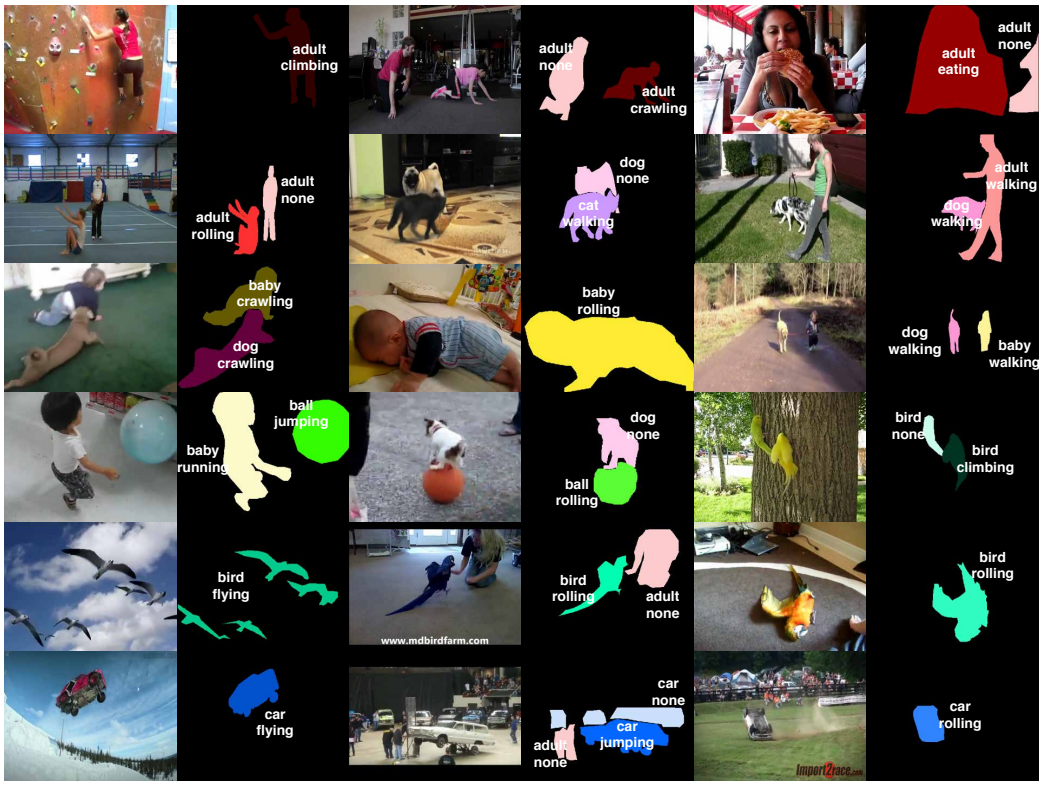


Figure 7.1: Montage of labeled videos in our new actor-action dataset, A2D. Examples of single actor-action instances as well as multiple actors doing different actions are present in this montage. Label colors are picked from the HSV color space, so that the same objects have the same *hue* (refer to Fig. 7.2 for the color-legend). Black is the background. **View zoomed and in color.**

different levels of modeling and hence allow us to analyze the new problem thoroughly. We further distinguish our work from multi-task learning [21] that focuses on getting a shared representation for training better classifiers, whereas we focus on modeling the relationship and interactions of the actor and action under a unified graphical model.

To support these new actor-action understanding problems, we have created a new dataset, which we call the Actor-Action Dataset or A2D (see Fig. 7.1), that is labeled at the pixel-level for actors and actions (densely in space over actors, sparsely in time). The A2D has 3782 videos with at least 99 instances per valid actor-action tuple (Sec. 7.1.1 and Fig. 7.2 have exact statistics). We thoroughly analyze empirical

performance of both state-of-the-art and baseline methods, including naïve Bayes (independent over actor and action), a joint product-space model (each actor-action pair is considered as one class), and a bilayer graphical model inspired by [95] that connects actor nodes with action nodes.

Our experiments demonstrate that inference jointly over actors and actions outperforms inference independently over them, and hence, supports the explicit consideration of various actors in comprehensive action understanding. In other words, although a *bird* and an *adult* can both *eat*, the space-time appearance of a *bird eating* and an *adult eating* are different in significant ways. Furthermore, the various mannerisms of the way *birds eat* and *adults eat* mutually reinforces inference over the constituent parts. This result is analogous to Sadeghi and Farhadi's visual phrases work [164] in which it is demonstrated that joint detection over small groups of objects in images is more robust than separate detection over each object followed a merging process and to Gupta et al.'s [67] work on human object-interactions in which considering specific objects while modeling human actions leads to better inferences for both parts.

7.1.1 A2D—The Actor-Action Dataset

We have collected a new dataset consisting of 3782 videos from YouTube; these videos are hence unconstrained "in-the-wild" videos with varying characteristics. Figure 7.1 has single-frame examples of the videos. We select seven classes of actors performing eight different actions. Our choice of actors covers articulated ones, such as *adult*, *baby*, *bird*, *cat* and *dog*, as well as rigid ones, such as *ball* and *car*. The eight actions are *climbing*, *crawling*, *eating*, *flying*, *jumping*, *rolling*, *running*, and *walking*. A single action-class can be performed by various actors, but none of the actors can perform all eight actions. For example, we do not consider *adult-flying* or *ball-running* in the dataset. In some cases, we have pushed the semantics of the

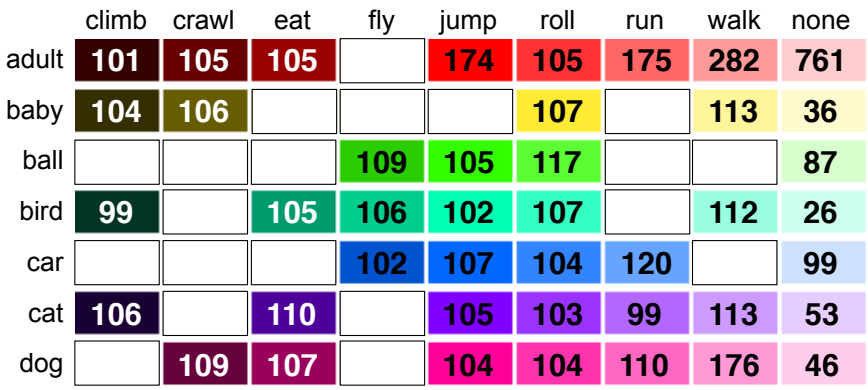


Figure 7.2: Statistics of label counts in the new A2D dataset. We show the number of videos in our dataset in which a given [actor, action] label occurs. Empty entries are joint-labels that are not in the dataset either because they are invalid (a *ball* cannot *eat*) or were in insufficient supply, such as for the case *dog-climb*. The background color in each cell depicts the color we use throughout the chapter; we vary hue for actor and saturation for action.

given action term to maintain a small set of actions: e.g., *car-running* means the car is moving and *ball-jumping* means the ball is bouncing. One additional action label *none* is added to account for actions other than the eight listed ones as well as actors in the background that are not performing an action. Therefore, we have in total 43 valid actor-action tuples.

To query the YouTube database, we use various text-searches generated from actor-action tuples. Resulting videos were then manually verified to contain an instance of the primary actor-action tuple, and subsequently temporally trimmed to contain that actor-action instance. The trimmed videos have an average length of 136 frames, with a minimum of 24 frames and a maximum of 332 frames. We split the dataset into 3036 training videos and 746 testing videos divided evenly over all actor-action tuples. Figure 7.2 shows the statistics for each actor-action tuple. One-third of the videos in A2D have more than one actor performing different actions, which further distinguishes our dataset from most action classification datasets. Figure 7.3 shows exact counts for these cases with multiple actors and actions.

To support the broader set of action understanding problems in consideration, we

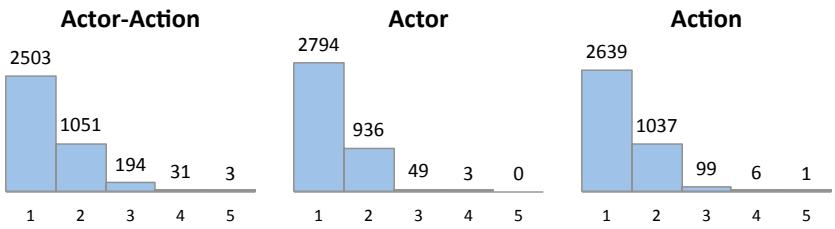


Figure 7.3: Histograms of counts of joint actor-actions, and individual actors and actions per video in A2D; roughly one-third of the videos have more than one actor and/or action.

label three to five frames for each video in the dataset with both dense pixel-level actor and action annotations (Fig. 7.1 has labeling examples). The selected frames are evenly distributed over a video. We start by collecting crowd-sourced annotations from MTurk using the LabelMe toolbox [162], then we manually filter each video to ensure the labeling quality as well as the temporal coherence of labels. Video-level labels are computed directly from these pixel-level labels for the recognition tasks. To the best of our knowledge, this dataset is the first video dataset that contains both actor and action pixel-level labels.

7.1.2 Problem Formulation

Without loss of generality, let $\mathcal{V} = \{v_1, \ldots, v_n\}$ denote a video with n voxels in space-time lattice Λ^3 or n supervoxels in a video segmentation [214, 221, 23] represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the neighborhood structure of the graph is given by the supervoxel segmentation method; when necessary we write $\mathcal{E}(v)$ where $v \in \mathcal{V}$ to denote the subset of $\mathcal{V}$ that are neighbors with v. We use $\mathcal{X}$ to denote the set of actor labels: {*adult*, *baby*, *ball*, *bird*, *car*, *cat*, *dog*}, and we use $\mathcal{Y}$ to denote the set of action labels: {*climbing*, *crawling*, *eating*, *flying*, *jumping*, *rolling*, *running*, *walking*, *none*[1]}.

Consider a set of random variables $\mathbf{x}$ for actor and another $\mathbf{y}$ for action; the

[1]The *none* action means either there is no action present or the action is not one of those we have considered.

specific dimensionality of $\mathbf{x}$ and $\mathbf{y}$ will be defined later. Then, the general actor-action understanding problem is specified as a posterior maximization:

$$(\mathbf{x}^*, \mathbf{y}^*) = \underset{\mathbf{x},\mathbf{y}}{\operatorname{argmax}} P(\mathbf{x}, \mathbf{y}|\mathcal{V}) \ . \tag{7.1}$$

Specific instantiations of this optimization problem give rise to various actor-action understanding problems, which we specify next, and specific models for a given instantiation will vary the underlying relationship between $\mathbf{x}$ and $\mathbf{y}$ allowing us to deeply understand their interplay.

7.1.2.1 Single-Label Actor-Action Recognition

This is the coarsest level of granularity we consider and it instantiates the standard action recognition problem [98]. Here, $\mathbf{x}$ and $\mathbf{y}$ are simply scalars x and y, respectively, depicting the single actor and action label to be specified for a given video $\mathcal{V}$. We consider three models for this case:

Naïve Bayes: Assume independence across actions and actors, and then train a set of classifiers over actor space $\mathcal{X}$ and a separate set of classifiers over action space $\mathcal{Y}$. This is the simplest approach and is not able to enforce actor-action tuple existence: e.g., it may infer *adult-fly* for a test video.

Joint Product Space: Create a new label space $\mathcal{Z}$ that is the joint product space of actors and actions: $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Directly learn a classifier for each actor-action tuple in this joint product space. Clearly, this approach enforces actor-action tuple existence, and we expect it to be able to exploit cross-actor-action features to learn more discriminative classifiers. However, it may not be able to exploit the commonality across different actors or actions, such as the similar manner in which a *dog* and a *cat walk*.

Trilayer: The trilayer model unifies the naïve Bayes and the joint product space

models. It learns classifiers over the actor space $\mathcal{X}$, the action space $\mathcal{Y}$ and the joint actor-action space $\mathcal{Z}$. During inference, it separately infers the naïve Bayes terms and the joint product space terms and then takes a linear combination of them to yield the final score. It models not only the cross-actor-action but also the common characteristics among the same actor performing different actions as well as the different actors performing the same action.

In all cases, we extract local features (see Sec. 7.1.3.1 for details) and train a set of one-vs-all classifiers, as is standard in contemporary action recognition methods, and although not strictly probabilistic, can be interpreted as such to implement Eq. 7.1.

7.1.2.2 Multi-Label Actor-Action Recognition

As noted in Fig. 7.3, about one-third of the videos in A2D have more than one actor and/or action present in a given video. In many realistic video understanding applications, we find such multiple-label cases. We address this explicitly by instantiating Eq. 7.1 for the multi-label case. Here, $\mathbf{x}$ and $\mathbf{y}$ are binary vectors of dimension $|\mathcal{X}|$ and $|\mathcal{Y}|$ respectively. x_i takes value 1 if the ith actor-type is present in the video and zero otherwise. We define $\mathbf{y}$ similarly. This general definition, which does not tie specific elements of $\mathbf{x}$ to those in $\mathbf{y}$, is necessary to allow us to compare independent multi-label performance over actors and actions with that of the actor-action tuples. We again consider a naïve Bayes pair of multi-label actor and action classifiers, multi-label actor-action classifiers over the joint product space, as well as the trilayer model that unifies the above classifiers.

7.1.2.3 Actor-Action Semantic Segmentation

Semantic segmentation is the most fine-grained instantiation of actor-action understanding that we consider, and it subsumes other coarser problems like detection and localization. Here, we seek a label for actor and action per-voxel over the entire

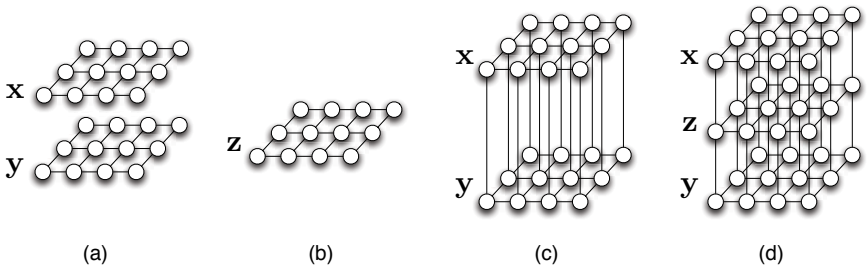


Figure 7.4: Visualization of different graphical models to solve Eq. 7.1. The figure here is for simple illustration and the actual voxel or supervoxel graph is built for a video volume.

video. Define the two sets of random variables $\mathbf{x} = \{x_1, \ldots, x_n\}$ and $\mathbf{y} = \{y_1, \ldots, y_n\}$ to have dimensionality in the number of voxels or supervoxels, and assign each $x_i \in \mathcal{X}$ and each $y_i \in \mathcal{Y}$. The objective function in Eq. 7.1 remains the same, but the way we define the graphical model implementing $P(\mathbf{x}, \mathbf{y}|\mathcal{V})$ leads to acutely different assumptions on the relationship between actor and action variables.

We explore this relationship in the remainder of this section. We start by again introducing a naïve Bayes-based model that treats the two classes of labels separately, and a joint product space model that considers actors and actions together in a tuple $[\mathbf{x}, \mathbf{y}]$. We then explore a bilayer model, inspired by Ladický et al. [95], that considers the inter-set relationship between actor and action variables. Finally, we introduce a new trilayer model that considers both intra- and inter-set relationships. Figure 7.4 illustrates these various graphical models. We then evaluate the performance of all models in terms of joint actor and action labeling in Sec. 7.1.3.

Naïve Bayes-based Model First, let us consider a naïve Bayes-based model, similar to the one used for actor-action recognition earlier:

$$
\begin{aligned}
P(\mathbf{x}, \mathbf{y}|\mathcal{V}) &= P(\mathbf{x}|\mathcal{V})P(\mathbf{y}|\mathcal{V}) \\
&= \prod_{i \in \mathcal{V}} P(x_i)P(y_i) \prod_{i \in \mathcal{V}} \prod_{j \in \mathcal{E}(i)} P(x_i, x_j)P(y_i, y_j) \\
&\propto \prod_{i \in \mathcal{V}} \phi_i(x_i)\psi_i(y_i) \prod_{i \in \mathcal{V}} \prod_{j \in \mathcal{E}(i)} \phi_{ij}(x_i, x_j)\psi_{ij}(y_i, y_j)
\end{aligned} \tag{7.2}
$$

where ϕ_i and ψ_i encode the separate potential functions defined on actor and action nodes alone, respectively, and ϕ_{ij} and ψ_{ij} are the pairwise potential functions within sets of actor nodes and sets of action nodes, respectively.

We train classifiers $\{f_c|c \in \mathcal{X}\}$ over actors and $\{g_c|c \in \mathcal{Y}\}$ on sets of actions using features described in Sec. 7.1.3.3, and ϕ_i and ψ_i are the classification scores for supervoxel i. The pairwise edge potentials have the form of a contrast-sensitive Potts model [13]:

$$
\phi_{ij} = \begin{cases} 1 & \text{if } x_i = x_j \\ \exp(-\theta/(1 + \chi^2_{ij})) & \text{otherwise,} \end{cases} \tag{7.3}
$$

where χ^2_{ij} is the χ^2 distance between feature histograms of nodes i and j, θ is a parameter to be learned from the training data. ψ_{ij} is defined analogously. Actor-action semantic segmentation is obtained by solving these two *flat* CRFs independently.

Joint Product Space We consider a new set of random variables $\mathbf{z} = \{z_1, \ldots, z_n\}$ defined again on all supervoxels in a video and take labels from the actor-action product space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. This formulation jointly captures the actor-action tuples as unique entities but cannot model the common actor and action behaviors among

different tuples as later models below do; we hence have a single-layer graphical model:

$$
\begin{aligned}
P(\mathbf{x}, \mathbf{y}|\mathcal{V}) \doteq P(\mathbf{z}|\mathcal{V}) &= \prod_{i\in\mathcal{V}} P(\mathbf{z}_i) \prod_{i\in\mathcal{V}} \prod_{j\in\mathcal{E}(i)} P(z_i, z_j) \\
&\propto \prod_{i\in\mathcal{V}} \varphi_i(z_i) \prod_{i\in\mathcal{V}} \prod_{j\in\mathcal{E}(i)} \varphi_{ij}(z_i, z_j) \\
&= \prod_{i\in\mathcal{V}} \varphi_i([x_i, y_i]) \prod_{i\in\mathcal{V}} \prod_{j\in\mathcal{E}(i)} \varphi_{ij}([x_i, y_i], [x_j, y_j]) \ ,
\end{aligned} \tag{7.4}
$$

where φ_i is the potential function for joint actor-action product space label, and φ_{ij} is the inter-node potential function between nodes with the tuple $[\mathbf{x}, \mathbf{y}]$. To be specific, φ_i contains the classification scores on the node i from running trained actor-action classifiers $\{h_c|c \in \mathcal{Z}\}$, and φ_{ij} has the same form as Eq. 7.3. Fig. 7.4 (b) illustrates this model as a one layer CRF defined on the actor-action product space.

Bilayer Model Given the actor nodes $\mathbf{x}$ and action nodes $\mathbf{y}$, the bilayer model connects each pair of random variables $\{(x_i, y_i)\}_{i=1}^n$ with an edge that encodes the potential function for the tuple $[x_i, y_i]$, directly capturing the *covariance* across the actor and action labels. We have

$$
\begin{aligned}
P(\mathbf{x}, \mathbf{y}|\mathcal{V}) = &\prod_{i\in\mathcal{V}} P(x_i, y_i) \prod_{i\in\mathcal{V}} \prod_{j\in\mathcal{E}(i)} P(x_i, x_j) P(y_i, y_j) \\
\propto &\prod_{i\in\mathcal{V}} \phi_i(x_i) \psi_i(y_i) \xi_i(x_i, y_i) \cdot \\
&\prod_{i\in\mathcal{V}} \prod_{j\in\mathcal{E}(i)} \phi_{ij}(x_i, x_j) \psi_{ij}(y_i, y_j) \ ,
\end{aligned} \tag{7.5}
$$

where $\phi.$ and $\psi.$ are defined as earlier, $\xi_i(x_i, y_i)$ is a learned potential function over the product space of labels, which can be exactly the same as φ_i in Eq. 7.4 above or a compatibility term like the contrast sensitive Potts model, Eq. 7.3 above. We choose the former. Fig. 7.4 (c) illustrates this model. We note that additional links

can be constructed by connecting corresponding edges between neighboring nodes across layers and encoding the occurrence among the bilayer edges, such as the joint object class segmentation and dense stereo reconstruction model in Ladický et al. [95]. However, their model is not directly suitable here.

Trilayer Model So far we have introduced three baseline formulations in Eq. 7.1 for semantic actor-action segmentation that relate the actor and action terms in different ways. The naïve Bayes model (Eq. 7.2) does not consider any relationship between actor $\mathbf{x}$ and action $\mathbf{y}$ variables. The joint product space model (Eq. 7.4) combines features across actors and actions as well as inter-node interactions in the neighborhood of an actor-action node. The bilayer model (Eq. 7.5) adds actor-action interactions among separate actor and action nodes, but it does not consider how these interactions vary spatiotemporally.

Therefore, we introduce a new trilayer model that explicitly models such variations (see Fig. 7.4d) by combining nodes $\mathbf{x}$ and $\mathbf{y}$ with the joint product space nodes $\mathbf{z}$:

$$\begin{aligned} P(\mathbf{x},\mathbf{y},\mathbf{z}|\mathcal{V}) &= P(\mathbf{x}|\mathcal{V})P(\mathbf{y}|\mathcal{V})P(\mathbf{z}|\mathcal{V})\prod_{i\in\mathcal{V}}P(x_i,z_i)P(y_i,z_i) \\ &\propto \prod_{i\in\mathcal{V}}\phi_i(x_i)\psi_i(y_i)\varphi_i(z_i)\mu_i(x_i,z_i)\nu_i(y_i,z_i)\cdot \\ &\quad \prod_{i\in\mathcal{V}}\prod_{j\in\mathcal{E}(i)}\phi_{ij}(x_i,x_j)\psi_{ij}(y_i,y_j)\varphi_{ij}(z_i,z_j) \ , \end{aligned} \tag{7.6}$$

where we define

$$\mu_i(x_i,z_i) = \begin{cases} w(y_i'|x_i) & \text{if } x_i = x_i' \text{ for } z_i = [x_i', y_i'] \\ 0 & \text{otherwise} \end{cases} \tag{7.7}$$

$$\nu_i(y_i,z_i) = \begin{cases} w(x_i'|y_i) & \text{if } y_i = y_i' \text{ for } z_i = [x_i', y_i'] \\ 0 & \text{otherwise} \end{cases} .$$

Terms $w(y_i'|x_i)$ and $w(x_i'|y_i)$ are classification scores of conditional classifiers, which are explicitly trained for this trilayer model. These conditional classifiers are the main reason for the increased performance found in this method: separate classifiers for the same action conditioned on the type of actor are able to exploit the characteristics unique to that actor-action tuple. For example, when we train a conditional classifier for action *eating* given actor *adult*, we use all other actions performed by *adult* as negative training samples. Therefore our trilayer model considers all relationships in the individual actor and action spaces as well as the joint product space. In other words, the previous three baseline models are all special cases of the trilayer model. It can be shown that the solution $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$ maximizing Eq. 7.6 also maximizes Eq. 7.1 (see Appendix).

7.1.3 Experiments

We thoroughly study each of the instantiations of the actor-action understanding problem with the overarching goal of assessing if the joint modeling of actor and action improves performance over modeling each of them independently, despite the large space. We follow the training and testing splits discussed in Sec. 7.1.1; for assigning a single-label to a video for the single-label actor-action recognition, we choose the label associated with the query for which we searched and selected that video from YouTube.

7.1.3.1 Single-Label Actor-Action Recognition

Following the typical action recognition setup, e.g., [98], we use the state-of-the-art dense trajectory features (trajectories, HoG, HoF, MBHx and MBHy) [203] and train a set of 1-versus-all SVM models (with RBF-χ^2 kernels from LIBSVM [22]) for the label sets of actors, actions and joint actor-action labels. Specifically, when training the *eating* classifier, the other seven actions are negative examples; when we train the

	Single-Label			Multiple-Label		
	Classification Accuracy			Mean Average Precision		
Model	Actor	Action	<A, A>	Actor	Action	<A, A>
Naïve Bayes	70.51	74.40	56.17	76.85	78.29	60.13
JointPS	72.25	72.65	61.66	76.81	76.75	63.87
Trilayer	**75.47**	**75.74**	**64.88**	**78.42**	**79.27**	**66.86**

Table 7.1: Single-label and multiple-label actor-action recognition in the three settings: independent actor and action models (naïve Bayes), joint actor-action models in a product-space and the trilayer model. The scores are not comparable along the columns (e.g., the space of independent actors and actions is significantly smaller than that or actor-action tuples); the point of comparison is along the rows where we find the joint model to outperform the independent models when considering both actors and actions. <A, A> denotes evaluating in the joint actor-action product-space.

bird-eating classifier, we use the 35 other actor-action labels as negative examples.

Table 7.1-left shows the classification accuracy of the naïve Bayes, joint product space and trilayer models, in terms of classifying actor, action and actor-action labels. To evaluate the joint actor-action (the <A, A> columns) for the naïve Bayes models, we train the actor and action classifiers independently, apply them to the test videos independently and then score them together (i.e., a video is correct if and only if actor and action are correct). We observe that the independent model for action outperforms the joint product space model for action; this can be explained by the regularity across different actors for the same action that can be exploited in the naïve Bayes model, but that results in more inter-class overlap in the joint product space. For example, a *cat-running* and a *dog-running* have similar signatures in space-time: the naïve Bayes model does not need to distinguish between these two, but the joint product space does. However, we find that when we consider both the actor and action in evaluation, it is clearly beneficial to jointly model them. This phenomenon occurs in all of our experiments. Finally, the trilayer model outperforms the other two models in terms of both individual actor or action tasks as well as the joint actor-action task. The reason is that the trilayer model incorporates both types of relationships that are separately modeled in the naïve Bayes and joint product space

models.

7.1.3.2 Multiple-Label Actor-Action Recognition

For the multiple-label case, we use the same dense trajectory features as in Sec. 7.1.3.1, and we train 1-versus-all SVM models again for the label sets of actor, action and actor-action pairs, but with different training regimen to capture the multiple-label setting. For example, when training the *adult* classifier, we use all videos containing any actor *adult* as positive examples no matter the other actors that coexist in the positive videos, and we use the rest of videos as negative examples. For evaluation, we adapt the approach from HOHA2 [127]. We treat multiple-label actor-action recognition as a retrieval problem and compute mean average precision (mAP) given the classifier scores. Table 7.1-right shows the performance of the three methods on this task. Again, we observe that the joint product space has higher mAP than naïve Bayes for the joint actor-action evaluation. We also observe the trilayer model further improves the scores following the same trend as in the single-label case.

However, we also note that large improvement in the both individual tasks from the trilayer model. This implies that the "side" information of the actor when doing action recognition (and vice versa) provides useful information to improve the inference task.

7.1.3.3 Actor-Action Semantic Segmentation

State-of-the-Art Pixel-Based Segmentation. We first apply the state-of-the-art robust P^N model [92] at the pixel level; we apply their supplied code off-the-shelf as a baseline. The average-per-class performance is 13.74% for the joint actor-action task, 47.2% for actor and 34.49% for action. We suspect that the modeling at pixel and superpixel level can not well capture the motion changes of actions, which explains why the actor score is high but the other scores are comparatively lower.

		bird							cat							dog						
Model	BK	climb	eat	fly	jump	roll	walk	none	climb	eat	jump	roll	run	walk	none	crawl	eat	jump	roll	run	walk	none
Naïve Bayes	79.5	21.0	6.2	28.7	17.3	28.3	2.8	**29.3**	28.2	24.3	1.6	38.2	43.6	1.0	**4.4**	6.1	13.2	**5.3**	21.9	**35.9**	25.8	**4.3**
JointPS	75.1	23.0	15.5	36.0	19.2	26.6	7.5	0.0	19.4	24.6	4.1	32.4	28.5	7.5	0.5	**10.9**	24.2	2.1	21.1	21.2	38.2	0.0
Conditional	79.5	23.2	8.4	40.7	**25.4**	30.5	7.5	0.0	26.0	30.5	8.0	31.7	**53.3**	**9.1**	0.0	7.4	16.2	3.1	24.6	29.3	**53.6**	0.0
Bilayer	**79.7**	24.5	13.3	40.8	13.0	35.4	7.0	0.0	32.7	**32.9**	1.1	38.0	37.0	7.5	0.1	2.5	22.8	2.4	**35.9**	27.0	29.6	0.0
Trilayer	78.5	**28.1**	**18.2**	**55.3**	20.3	**42.5**	**9.0**	0.0	**33.1**	27.2	**6.1**	**49.8**	48.5	6.6	0.0	9.9	**31.0**	2.0	27.6	23.6	39.4	0.0

	adult								baby					ball				car				
Model	climb	crawl	eat	jump	roll	run	walk	none	climb	crawl	roll	walk	none	fly	jump	roll	none	fly	jump	roll	run	none
Naïve Bayes	21.5	30.4	21.5	11.3	5.0	18.1	11.5	**25.8**	21.6	23.5	20.5	8.6	**7.4**	2.9	13.6	6.6	**8.6**	10.0	71.2	22.2	5.5	**13.7**
JointPS	23.1	59.3	44.0	17.5	17.6	34.6	28.4	21.4	18.3	24.0	28.1	17.2	0.6	0.0	6.5	4.7	2.8	13.2	74.7	43.9	30.5	8.1
Conditional	18.5	43.1	36.3	**25.4**	17.4	31.8	30.7	12.1	**26.5**	20.4	36.7	13.9	5.6	**3.7**	**16.2**	**21.4**	9.0	**27.7**	**77.6**	43.5	37.2	1.7
Bilayer	27.2	49.6	**51.6**	25.1	**28.4**	27.9	**39.2**	0.6	13.2	**25.4**	**44.0**	24.0	0.0	0.3	10.3	6.0	0.0	20.9	76.8	37.2	39.6	0.5
Trilayer	**33.1**	**59.8**	49.8	19.9	27.6	**40.2**	31.7	24.6	20.4	21.7	39.3	**25.3**	0.0	1.0	11.9	6.1	0.0	24.4	75.9	**44.3**	**48.3**	2.4

Unary Term Only

	Average Per Class Accuracy		
Model	**Actor**	**Action**	**<A, A>**
Naïve Bayes	43.02	40.08	16.35
JointPS	40.89	38.50	20.61
Conditional	43.02	41.19	22.55
Bilayer	43.02	40.08	16.35
Trilayer	43.08	41.61	22.59

Full Model

	Average Per Class Accuracy		
Model	**Actor**	**Action**	**<A, A>**
Naïve Bayes	44.78	42.59	19.28
JointPS	41.96	40.09	21.73
Conditional	44.78	41.88	24.19
Bilayer	44.46	43.62	23.43
Trilayer	**45.70**	**46.96**	**26.46**

Table 7.2: Average per-class semantic segmentation accuracy in percentage of joint actor-action labels for all models (for individual classes, left, and in summary, right). The leading scores of each label are displayed in bold font. The summary scores on the right and indicate that the trilayer model, which considers the action and actor models alone as well as the actor-action product-space, performs best.

The P^N model could be generalized to fit within our framework, which we leave for future work. We use supervoxel segmentation and extract spatiotemporal features for assessing the various models posed for actor-action semantic segmentation.

Supervoxel Segmentation and Features. We use TSP [23] to obtain supervoxel segmentations due to its strong performance on the supervoxel benchmark [214]. In our experiments, we set $k = 400$ yielding about 400 supervoxels touching each frame. We compute histograms of textons and dense SIFT descriptors over each supervoxel volume, dilated by 10 pixels. We also compute color histograms in both RGB and HSV color spaces and dense optical flow histograms. We extract feature histograms from the entire supervoxel 3D volume, rather than a single representative superpixel [190]. Furthermore, we inject the dense trajectory features [203] to supervoxels by assigning each trajectory to the supervoxels it intersects in the video.

Frames in A2D are sparsely labeled; to obtain a supervoxel's groundtruth label, we look at all labeled frames in a video and take a majority vote over intersecting labeled pixels. We train sets of 1-versus-all SVM classifiers (linear kernels) for actor, action, and actor-action as well as conditional classifiers separately. The parameters of the graphical model are tuned by empirical search, and loopy belief propagation is used for inference. The inference output is a dense labeling of video voxels in space-time, but, as our dataset is sparsely labeled in time, we compute the average per-class segmentation accuracy only against those frames for which we have groundtruth labels. We choose average per-class accuracy over global accuracy because our goal is to compares actor and action rather than full video labeling.

Evaluation. Table 7.2–right shows the overall performance of the different methods. The upper part is results with only the unary terms and the lower part is the full model performance. We not only evaluate the actor-action pairs but also individual actor and action tasks. The conditional model is a variation of bilayer model with different aggregation—we infer the actor label first then the action label conditioned

on the actor. Note that the bilayer model has the same unary scores as the naïve Bayes model (using actor ϕ_i and action ψ_i outputs independently) and the actor unary of the conditional model is the same as that of the naïve Bayes model (followed by the conditional classifier for action).

Over all models, the naïve Bayes model performs worst, which is expected as it does not encode any interactions between the two label sets. We observe that the conditional model has better action unary and actor-action scores, which indicates that knowing actors can help with action inference. We also observe that the bilayer model has a poor unary performance of 16.35% (actor-action) that is the same as naïve Bayes but for the full model it improves dramatically to 23.43%, which suggests that the performance boost again comes from the interaction of actor and action nodes in the full bilayer model. We also observe that the full trilayer model has not only much better performance in the joint actor-action task, but also better scores for actor and action individual tasks in the full model, as it is the only model considered that incorporates classifiers in both individual actor and action tasks and also in the joint space.

Table 7.2–left shows the comparison of quantitative performance for specific actors and actions. We observe that the trilayer model has leading scores for more actor-action tuples than the other models. The trilayer model has significant improvement on labels such as *bird-flying*, *adult-running* and *cat-rolling*. We note the systematic increase in performance as more complex actor-action variable interactions are included. We also note that the tuples with *none* action are sampled with greater variation than the action classes (Fig. 7.2), which contributes to the poor performance of *none* over all actors. Interestingly, the naïve Bayes model has relatively better performance on the *none* action classes. We suspect that the label-variation for *none* leads to high-entropy over its classifier density and hence when joint modeling, the actor inference pushes the action variable away from the *none* action class.

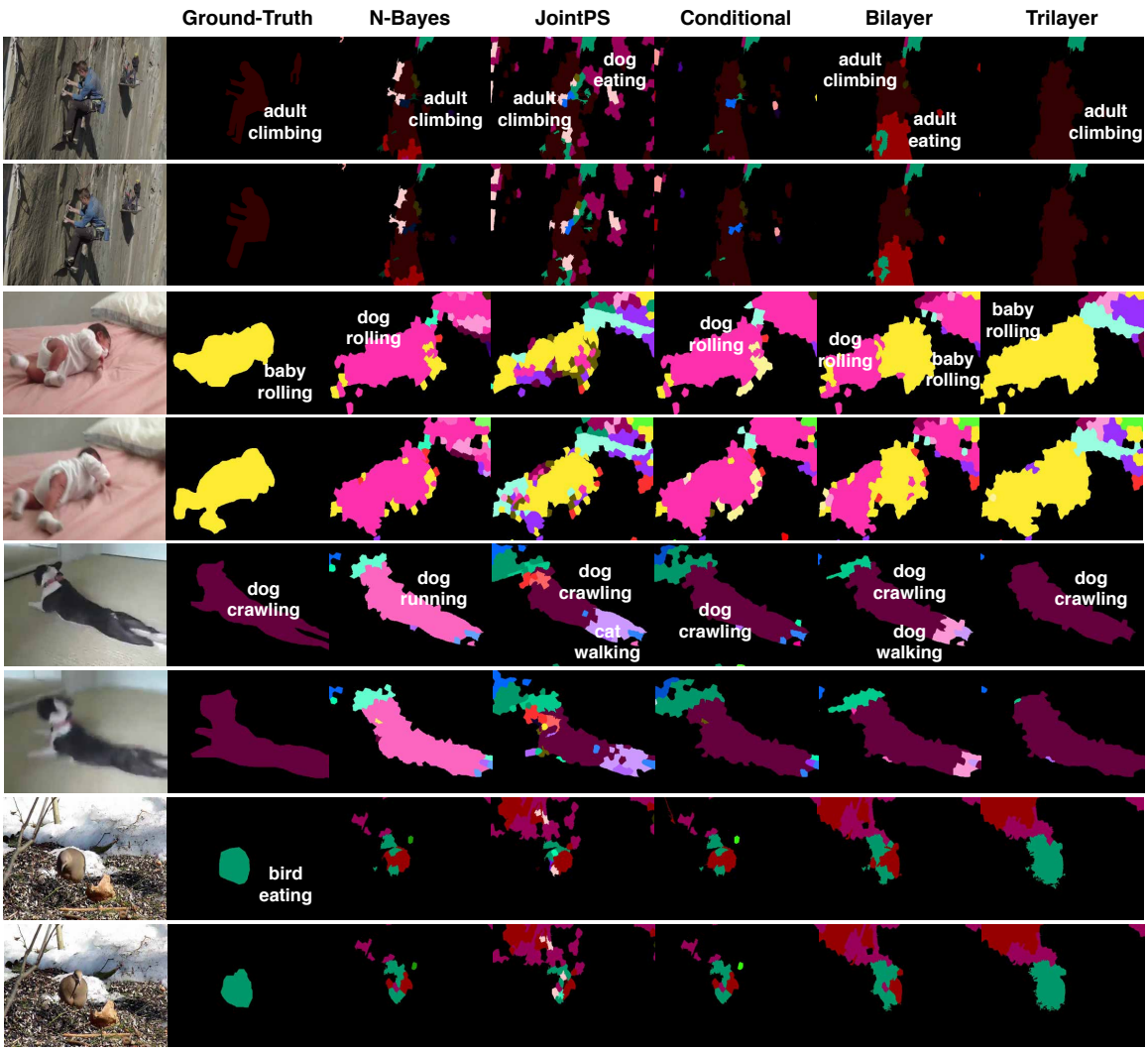


Figure 7.5: Comparative example of semantic segmentation results. These sample only two frames from the each dense video outputs.

Fig. 7.5 shows example segmentations. Recall that the naïve Bayes model considers the actor and action labeling problem independent of each other. Therefore, the *baby-rolling* in the second video get assigned with actor label *dog* and action label *rolling* when there is no consideration between actor and action. The bilayer model partially recovers the *baby* label, whereas the trilayer model successfully recovers the *baby-rolling* label, due to the modeling of inter-node relationship in the joint actor-action space of the trilayer model. We also visualize more example outputs of the

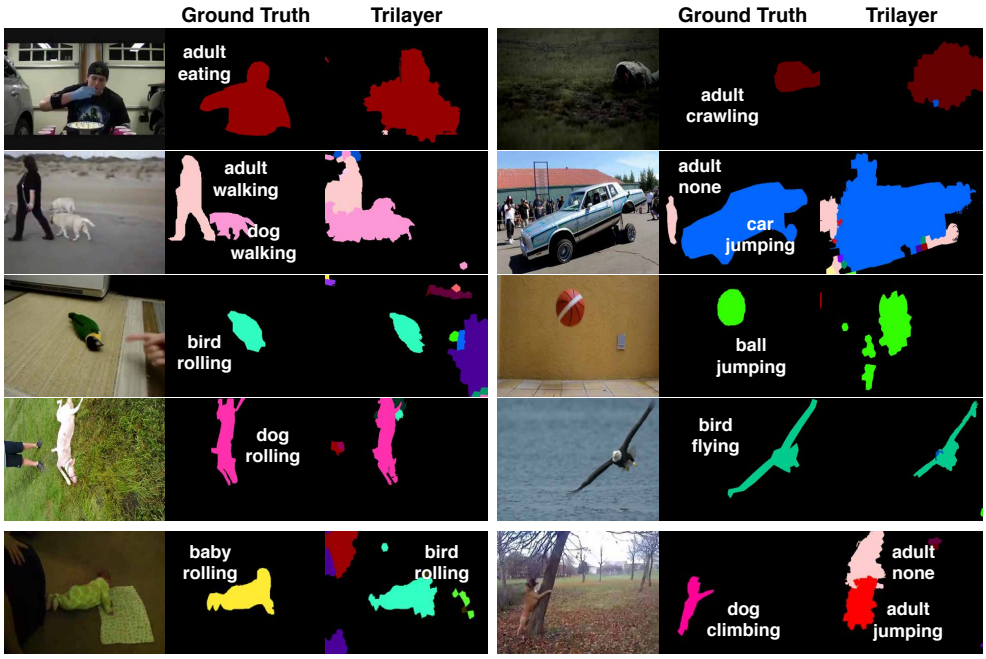


Figure 7.6: Example results from the trilayer model (upper are good, lower are failure cases).

trilayer model in Fig. 7.6. Note that the fragmented segmentation in the *ball* video is due to poor supervoxel segmentation algorithm in the pre-processing step. We also show trilayer failure cases in the bottom row of Fig. 7.6, which are due to weak cross-class visual evidence.

7.1.4 Discussion and Contributions

Our thorough assessment of all instantiations of the actor-action understanding problem at both coarse video-recognition level and fine semantic segmentation level provides strong evidence that the joint modeling of actor and action improves performance over modeling each of them independently. We find that for both individual actor and action understanding and joint actor-action understanding, it is beneficial to jointly consider actor and action. A proper modeling of the interactions between actor and action results in dramatic improvement over the baseline models of the naïve Bayes and joint product space models, as we observe from the bilayer and

trilayer models.

We set out with two goals: first, we sought to motivate and develop a new, more challenging, and more relevant actor-action understanding problem, and second, we sought to assess whether joint modeling of actors and actions improved performance for this new problem. We achieved these goals through the three contributions:

1. New actor-action understanding problem and dataset.
2. Thorough evaluation of actor-action recognition and semantic segmentation problems using state-of-the-art features and models. The experiments unilaterally demonstrate a benefit for jointly modeling actors and actions.
3. A new trilayer approach to recognition and semantic segmentation that combines both the independent actor and action variations and product-space interactions.

7.2 Grouping Process Models

The task of actor-action semantic segmentation is challenging—the benchmarked leading method, the *trilayer* model, only achieves a 26.46% per-class accuracy for the joint actor-action video labeling. The method builds a large three-layer CRF on video supervoxels, where random variables are defined for sets of actor, actor-action, and action labels, respectively. It connects layers with potential functions that capture conditional probabilities (e.g. conditional distribution of actions given a specific actor class). Although the model accounts for the interplay of actors and actions, the interactions are restricted to the local CRF neighborhoods, which, based on the low absolute performance, is insufficient to solve this unique actor-action problem for three reasons.

First, we believe the pixel-level model must be married to a secondary process that captures instance-level or video-level global information in order to properly model the actors performing actions. Lessons learned from images strongly supports this argument—the performance of semantic image segmentation on the MSRC dataset

seems to hit a plateau [173] until information from secondary processes, such as context [93, 135], object detectors [94] and a holistic scene model [223], are added. However, to the best of our knowledge, there is no method in video semantic segmentation that directly leverages the recent successes in action recognition.

Second, the two sets of labels, actors and actions, exist at different granularities. For example, we want to label *adult-clapping* in a video. The actor, *adult*, can probably be recognized by looking only at the lower human body, e.g. legs. However, in order to recognize the *clapping* action, we have to either locate the acting parts of the human body or simply look at the whole actor body.

Third, actors and actions have different emphases on space and time in a video. Actors are more space-oriented—they can be fairly well labeled using only still images, as in semantic image segmentation [174, 223], whereas actions are space- and time-oriented. Although one can possibly identify actions by still images alone [222], there are strong distinctions between actions in time. For example *running* is faster and thus may results more repeated motion patterns than *walking* for a common time duration; and *walking* performed by a *baby* is very different compared to an *adult*, despite the two actor classes may easily confuse a spatially trained detector.

Here, we overcome the above limitations in two ways: (1) we propose a novel grouping process model (GPM) that adaptively adds long-ranging interactions to the labeling CRF; and (2) we incorporate the video-level recognition into segment-level labeling by the means of global labeling cost and the GPM. The GPM models a dynamic and continuous process of information exchange of a labeling CRF and a supervoxel hierarchy. The supervoxel hierarchy provides a rich multi-scale decomposition of video content, where object parts, identities, deformations and actions are retained in space-time supervoxels across various levels in the hierarchy [220, 79, 141]. Rather than using object and action proposals as separate processes, we directly locate the actor and action groupings in the supervoxel hierarchy by the labeling CRF.

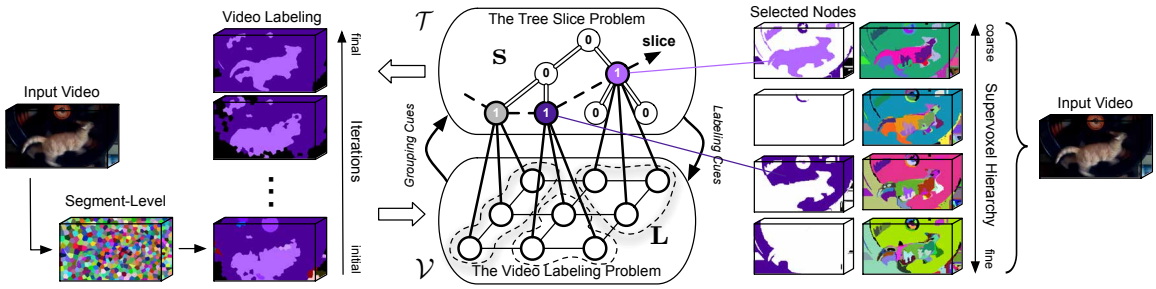


Figure 7.7: An overview of the grouping process model. The left side shows an input video and its segment-level segmentation. The right side shows the same video being segmented into a supervoxel hierarchy. During inference, the CRF defined on the segment-level starts with a coarse video labeling. It influences what supervoxels are active in the hierarchy. The active supervoxels, in turn, affect the connectivities in the CRF. This process is dynamic and continuous, where the video labeling is being iteratively refined.

During inference, the labeling CRF influences what supervoxels in a hierarchy are *active*, and these active supervoxels, in turn, influence the connectivities in the CRF, thus refining the labeling.

Directly solving the joint energy function of GPM is hard. However, it can be efficiently solved by decomposing it into two subproblems, a video labeling problem and a tree slice problem, such as Chapter VI, where the former one can be solved by graph cuts and the latter one can be rewritten into a binary linear program. Therefore, the inference of GPM is dynamic and iterative as shown in Fig. 7.7. Throughout the entire process, information is being exchanged at various levels in the supervoxel hierarchy, thus the multiscale space-time representation is explicitly explored in our model.

We conduct thorough experiments on the large-scale actor-action video dataset (A2D). We compare the proposed method to the previous benchmarked leading method, the trilayer model, as well as two leading semantic segmentation methods [93, 88] that we have extended to the actor-action problem. The experimental results show that our proposed method outperforms the second best method by a

large margin of 17% per-class accuracy (60% relative improvement) and over 10% global pixel accuracy, which demonstrates the effectiveness of our modeling.

7.3 The Modeling of GPM

In this section, we give the general form of GPM, and Fig. 7.7 shows an overview. We define the detailed potentials adapted to the actor-action problem in Sec. 7.5.

Segment-Level. Without loss of generality, we define $\mathcal{V} = \{q_1, q_2, \ldots, q_N\}$ as a video with N voxels or a video segmentation with N segments. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined over the entire video, where the neighborhood structure $\mathcal{E}(\cdot)$ is induced by the connectivities in the voxel lattice Λ^3 or the segmentation graph over space-time in a video. We define a set of random variables $\mathbf{L} = \{l_1, l_2, \ldots, l_N\}$ where the subscript corresponds to a certain node in $\mathcal{V}$ and each l_i takes some label from a label set $\mathcal{L}$. The GPM is inherently a labeling CRF, but it leverages a supervoxel hierarchy to dynamically adjust its non-local grouping structure.

Supervoxel Hierarchy. Given a supervoxel hierarchy generated by a hierarchical video segmentation method, such as GBH [65], we extract a supervoxel tree[2], denoted as $\mathcal{T} = \{T_1, T_2, \ldots, T_S\}$ with S total supervoxels in the tree, by ensuring that each supervoxel at a finer level segmentation has one and only one parent at its coarser level (Sec. 7.6 details the tree extraction process in the general case). We define a set of random variables $\mathbf{s} = \{s_1, s_2, \ldots, s_S\}$ on the tree supervoxels, where $s_t \in \{0, 1\}$ takes a binary label to indicate whether the tth supervoxel is active or not. Each supervoxel in the hierarchy connects to a set of nodes in the segment-level according to their overlap in voxel lattice Λ^3. Thus we have s_t, which is connected to a set of random variables at the segment-level CRF, denoted as $\mathbf{L}_t \subset \mathbf{L}$. Intuitively, when s_t is active, the fully-connected clique containing all nodes in $\mathbf{L}_t$ is considered in

[2]We add one virtual node as root to make it a tree if the segmentation at the coarsest level contains more than one supervoxel.

the labeling CRF; otherwise, when s_t is inactive, that fully-connected clique is not evaluated.

Supervoxel hierarchies, such as [65, 31], are built by iteratively recomputing and merging finer supervoxels into coarser ones based on appearance and motion features, where the body parts of an actor and its local motion are contained at the finer levels and the identity of the actor and its long-ranging action are contained at the coarser levels. However, choosing an arbitrary level in a hierarchy can be risky—going too coarse will cause overmerging and going too fine will lose the meaningful actions. It is challenging to locate the supervoxels in a hierarchy that best describe the actor and its action. Here, the GPM uses the evidence directly from the segment-level CRF to locate supervoxels across various scales that are best supported by the labeling $\mathbf{L}$. Once the supervoxels $\mathbf{s}$ are selected, they provide strong labeling cues to the segment-level CRF—the CRF nodes connected to the same supervoxel are encouraged to have the same label.

The objective of GPM is to find the best labeling $\mathbf{L}^*$ and the best selection $\mathbf{s}^*$ that minimize the following energy:

$$
\begin{aligned}
(\mathbf{L}^*, \mathbf{s}^*) &= \operatorname*{argmin}_{\mathbf{L},\mathbf{s}} E(\mathbf{L}, \mathbf{s}|\mathcal{V}, \mathcal{T}) \\
E(\mathbf{L}, \mathbf{s}|\mathcal{V}, \mathcal{T}) &= E^v(\mathbf{L}|\mathcal{V}) + E^h(\mathbf{s}|\mathcal{T}) \\
&\quad + \sum_{t \in \mathcal{T}} (E^h(\mathbf{L}_t|s_t) + E^h(s_t|\mathbf{L}_t)) \ ,
\end{aligned}
\tag{7.8}
$$

where $E^v(\mathbf{L}|\mathcal{V})$ and $E^h(\mathbf{s}|\mathcal{T})$ encode the energies at the segment-level and in the supervoxel hierarchy, respectively; $E^h(\mathbf{L}_t|s_t)$ and $E^h(s_t|\mathbf{L}_t)$ are conditional energy functions defined as directional edges in Fig. 7.7. To keep the discussion general, we do not define the specific form of $E^v(\mathbf{L}|\mathcal{V})$ here—it can be any labeling CRF, such as [93, 88, 173]. We define the other terms next.

7.3.1 Labeling Cues from Supervoxel Hierarchy

Given an active node s_t in the supervoxel hierarchy, we use it as a cue to refine the segment-level labeling $\mathbf{L}_t$ and we define the energy of this process as:

$$E^h(\mathbf{L}_t|s_t) = \begin{cases} \sum_{i\in\mathbf{L}_t}\sum_{j\neq i, j\in\mathbf{L}_t}\psi^h_{ij}(l_i,l_j) & \text{if } s_t = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{7.9}$$

Here, $\psi^h_{ij}(\cdot)$ has the form:

$$\psi^h_{ij}(l_i,l_j) = \begin{cases} \theta_t & \text{if } l_i \neq l_j \\ 0 & \text{otherwise,} \end{cases} \tag{7.10}$$

where θ_t is a parameter to be tuned. $\psi^h_{ij}(l_i,l_j)$ penalizes any two nodes in the field $\mathbf{L}_t$ that contain different labels. Eq. 7.9 changes the graph structure in $\mathbf{L}_t$ by fully connecting the nodes inside, and has clear semantic meaning—-this set of nodes in $\mathbf{L_t}$ at the segment-level are linked to the same supervoxel node s_t and hence expected to be from the same object, taking evidences from the appearance and motion features used in a typical supervoxel segmentation method.

7.3.2 Grouping Cues from Segment Labeling

If the selected supervoxels are too fine, they are subject to losing object identity and long-ranging actions; if they are too coarse, they are subject to overmerging with the background. Therefore, we set the selected supervoxels to best reflect the segment-level labeling while also respecting a selection prior. Given a video labeling $\mathbf{L}$ at the segment-level, we select the nodes in the supervoxel hierarchy that best correspond to the current labeling:

$$E^h(s_t|\mathbf{L}_t) = (\mathcal{H}(\mathbf{L}_t)|\mathbf{L}_t| + \theta_h)s_t \ , \tag{7.11}$$

where $|\cdot|$ denotes the number of video voxels and θ_h is a parameter to be tuned that encodes a prior of the node selection in the hierarchy. $\mathcal{H}(\cdot)$ is defined as the entropy of the labeling field connected to s_t:

$$\mathcal{H}(\mathbf{L}_t) = -\sum_{\gamma \in \mathcal{L}} P(\gamma; \mathbf{L}_t) \log P(\gamma; \mathbf{L}_t) \ , \tag{7.12}$$

where $P(\gamma; \mathbf{L}_t) = \frac{\sum_{i \in \mathbf{L}_t} \delta(l_i = \gamma)}{|\mathbf{L}_t|}$ and $\delta(\cdot)$ is an indicator function. Intuitively, the first term in Eq. 7.11 pushes down the selection of nodes in the hierarchy such that they only include the labeling field that has the most consistent labels, and the second term pulls up the node selection, giving penalties for going down the hierarchy.

7.3.3 Tree Slice Constraint

The active nodes in $\mathbf{s}$ define what groups of segments the GPM will enforce during labeling; hence the name grouping process model. However, not all instances of $\mathbf{s}$ are permissible: since we seek a single labeling over the video, we enforce that each segment in $\mathcal{V}$ is associated with one and only one active node in $\mathbf{s}$. This notion was introduced in Chapter VI by a way of *tree slice*: for every root-to-leaf path in $\mathcal{T}$, there is one and only one node being active.

We follow Chapter VI to define a matrix $\mathcal{P}$ that encodes all root-to-leaf paths in $\mathcal{T}$. $\mathbf{P}_p$ is one row in $\mathcal{P}$, and it encodes the path from the root to pth leaf with 1s for nodes on the path and 0s otherwise. We define the energy to regulate $\mathbf{s}$ as:

$$E^h(\mathbf{s}|\mathcal{T}) = \sum_{p=1}^{P} \delta(\mathbf{P}_p^\intercal \mathbf{s} \neq 1)\theta_\tau \ , \tag{7.13}$$

where P is the total number of leaves (also the number of such root-to-leaf paths) and θ_τ is a large constant to penalize an invalid tree slice. The tree slice selects supervoxel nodes to form a new video representation that has a one-to-one mapping to the 3D

video lattice Λ^3.

7.4 Iterative Inference for GPM

Directly solving the objective function defined in Eq. 7.8 is hard. Here, we show that we can use an iterative inference schema to efficiently solve it—given the segment-level labeling, we find the best supervoxels in the hierarchy; and given the selected supervoxels in the hierarchy, we refine the segment-level labeling.

The Video Labeling Problem. Given a tree slice $\mathbf{s}$, we would like to find the best $\mathbf{L}^*$ such that:

$$\begin{aligned} \mathbf{L}^* &= \operatorname*{argmin}_{\mathbf{L}} E(\mathbf{L}|\mathbf{s}, \mathcal{V}, \mathcal{T}) \\ &= \operatorname*{argmin}_{\mathbf{L}} E^v(\mathbf{L}|\mathcal{V}) + \sum_{t \in \mathcal{T}} E^h(\mathbf{L}_t|s_t) \ . \end{aligned} \tag{7.14}$$

The above can have a standard CRF form depending on how $E^v(\mathbf{L}|\mathcal{V})$ is defined. The second energy term $E^h(\mathbf{L}_t|s_t)$ can be decomposed to a locally fully connected CRF, and its range is constrained by s_t such that the inference is feasible even without Gaussian kernels [88].

The Tree Slice Problem. Given the current labeling $\mathbf{L}$, we would like to find the best $\mathbf{s}^*$ such that:

$$\begin{aligned} \mathbf{s}^* &= \operatorname*{argmin}_{\mathbf{s}} E(\mathbf{s}|\mathbf{L}, \mathcal{V}, \mathcal{T}) \\ &= \operatorname*{argmin}_{\mathbf{s}} E^h(\mathbf{s}|\mathcal{T}) + \sum_{t \in \mathcal{T}} E^h(s_t|\mathbf{L}_t) \ . \end{aligned} \tag{7.15}$$

The above equation can be rewritten as a binary linear program of the following form:

$$\min \sum_{t \in \mathcal{T}} \alpha_t s_t \quad \text{s.t. } \mathcal{P}\mathbf{s} = \mathbf{1}_P \ \text{ and } \ \mathbf{s} \in \{0,1\}^S \ , \tag{7.16}$$

where $\alpha_t = \mathcal{H}(\mathbf{L}_t)|\mathbf{L}_t| + \theta_h$. Note that this optimization is different than that proposed by the original tree slice in Chapter VI, which incorporated quadratic terms in a binary quadratic program. We use a standard solver (IBM CPLEX) to solve the binary linear programming problem.

Iterative Inference. The inference of the above two subproblems is iteratively carried out, as depicted in Fig. 7.7. To be specific, we initialize a coarse labeling $\mathbf{L}$ by solving Eq. 7.14 without the second term, then we solve Eq. 7.16 and 7.14 in an iterative fashion. Each round of the tree slice problem enacts an updated set of grouped segments, which are then encouraged to be assigned the same label during the subsequent labeling process. We notice that the solution converges after a few rounds.

Relation to AHRF. The associative hierarchical random field (AHRF) [93] performs inference exhaustively from finer levels to coarser levels in the segmentation tree $\mathcal{T}$, whereas the GPM explicitly models the best set of active supervoxels by the means of a tree slice. AHRF defines a full multi-label random field on the hierarchy; our model leverages the hierarchy to adaptively modify the labeling field. Our model is hence more scalable to videos. Furthermore, the GPM assumes that the best representations of the video content exist in a tree slice rather than enforcing the agreement across different levels as in AHRF. For example, a video of *long jumping* often contains *running* in the beginning. The running action exists and has a strong classifier signal at a fine-level in a supervoxel hierarchy, but it quickly diminishes when one goes to higher levels in the hierarchy where supervoxels capture longer range in the video and would then favor the *jumping* action.

Relation to FCRF. The fully-connected CRF (FCRF) in [88] imposes Gaussian mixture kernels to regularize the pairwise interactions. Although our model fully connects the nodes in each $\mathbf{L}_t$ for a given iteration of inference, we explicitly take the evidence from the supervoxel groupings. Equation 7.11 restricts the selected

supervoxels to avoid overmerging. Although a more complex process, in practice, our inference is efficient (see Sec. 7.6 for running time).

7.5 The Actor-Action Problem Modeling

We train segment-level classifiers to capture the local appearance and motion of the actors' body parts. They have some ability to localize the actor-action, but the predictions are noisy; they use no context, for example. In contrary, video-level recognition, as a secondary process, captures the global information of actors performing actions and have good prediction performance at the video-level. However, it is not able to tell where the action is happening. These two streams of information are captured at the segment-level and at the video-level, and hence are complementary to each other. In this section, we fuse them together in a single model, leveraging the grouping process model as a means of marrying the two.

Let us first define notation, extending that from Sec. 7.3 where possible. We use $\mathcal{X}$ to denote the set of actor labels (e.g. *adult*, *baby* and *dog*) and $\mathcal{Y}$ to denote the set of action labels (e.g. *eating*, *walking* and *running*). The segment-level random field $\mathbf{L}$ now takes two sets of labels—for the ith segment, $l_i^{\mathcal{X}} \in \mathcal{X}$ takes a label from the actors and $l_i^{\mathcal{Y}} \in \mathcal{Y}$ from the actions. We denote $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ as the joint product space of the actor-action labels. We define a set of binary random variables $\mathbf{v} = \{v_1, v_2, \ldots, v_{|\mathcal{Z}|}\}$ on the video-level, where $v_z = 1$ denotes the zth actor-action label is active at the video-level. They represent the video-level multi-label recognition problem. Again, we have the set of binary random variables $\mathbf{s}$ defined on the supervoxel hierarchy as in Sec. 7.3.

Therefore, we have the total energy function of the actor-action semantic segmen-

tation defined as:

$$
\begin{aligned}
(\mathbf{L}^*, \mathbf{s}^*, \mathbf{v}^*) &= \underset{\mathbf{L},\mathbf{s},\mathbf{v}}{\operatorname{argmin}} E(\mathbf{L}, \mathbf{s}, \mathbf{v}|\mathcal{V}, \mathcal{T}) \\
E(\mathbf{L}, \mathbf{s}, \mathbf{v}|\mathcal{V}, \mathcal{T}) &= E^v(\mathbf{L}|\mathcal{V}) + \sum_{z\in\mathcal{Z}} E^{\mathcal{V}}(v_z|\mathcal{V}) + E^{\mathcal{V}}(\mathbf{L}, \mathbf{v}) \\
&\quad + E^h(\mathbf{s}|\mathcal{T}) + \sum_{t\in\mathcal{T}} (E^h(\mathbf{L}_t, \mathbf{v}|s_t) + E^h(s_t|\mathbf{L}_t)) \ , \qquad (7.17)
\end{aligned}
$$

where the term $E^h(\mathbf{L}_t, \mathbf{v}|s_t)$ now models the joint potentials of the segment-level labeling field $\mathbf{L}_t$ and the video-level label $\mathbf{v}$, which is slightly different from its form in Eq. 7.9. We have two new terms, $E^{\mathcal{V}}(v_z|\mathcal{V})$ and $E^{\mathcal{V}}(\mathbf{L}, \mathbf{v})$, from the video-level, where v_z is the zth coordinate in $\mathbf{v}$. We explain these new terms next.

7.5.1 Segment-Level CRF E^v

At the segment-level, we use the same bilayer actor-action CRF model from Sec. 7.1 to capture the local pairwise interactions of the two sets of labels:

$$
\begin{aligned}
E^v(\mathbf{L}|\mathcal{V}) &= \sum_{i\in\mathcal{V}} \psi_i^v(l_i^{\mathcal{X}}) + \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{E}(i)} \psi_{ij}^v(l_i^{\mathcal{X}}, l_j^{\mathcal{X}}) \qquad (7.18) \\
&\quad + \sum_{i\in\mathcal{V}} \phi_i^v(l_i^{\mathcal{Y}}) + \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{E}(i)} \phi_{ij}^v(l_i^{\mathcal{Y}}, l_j^{\mathcal{Y}}) + \sum_{i\in\mathcal{V}} \varphi_i^v(l_i^{\mathcal{X}}, l_i^{\mathcal{Y}}) \ ,
\end{aligned}
$$

where ψ_i^v and ϕ_i^v encode separate potentials for random variables $l_i^{\mathcal{X}}$ and $l_i^{\mathcal{Y}}$ to take the actor and action labels, respectively. φ_i^v is a potential to measure the compatibility of the actor-action tuples on segment i, and ψ_{ij}^v and ϕ_{ij}^v capture the pairwise interactions between segments, which have the form of a contrast sensitive Potts model [13, 173].

7.5.2 Video-Level Potentials $E^{\mathcal{V}}$

Rather than a uniform penalty over all labels [36], we use the video-level recognition signals as global multi-label labeling costs to impact the segment-level labeling.

We define the unary energy at the video-level as:

$$E^{\mathcal{V}}(v_z|\mathcal{V}) = -(\xi^{\mathcal{V}}(z) - \theta_T)\theta_B v_z \ , \tag{7.19}$$

where $\xi^{\mathcal{V}}(\cdot)$ is the video-level classification response for a particular actor-action label, and Sec. 7.6 describes its training process. Here, θ_T is a parameter to control response threshold, and θ_B is a large constant parameter. In other words, to minimize Eq. 7.19, the label $v_z = 1$ only when the classifier response $\xi^{\mathcal{V}}(z) > \theta_T$.

We define the interactions between the video-level and the segment-level:

$$E^{\mathcal{V}}(\mathbf{L}, \mathbf{v}) = \sum_{x \in \mathcal{X}} \delta_x(\mathbf{L}) h_x(\mathbf{v}) \theta_{\mathcal{V}} + \sum_{y \in \mathcal{Y}} \delta_y(\mathbf{L}) h_y(\mathbf{v}) \theta_{\mathcal{V}} \ , \tag{7.20}$$

where $\delta_x(\cdot)$ is an indicator function to determine whether the current labeling $\mathbf{L}$ at the segment-level contains a particular label $x \in \mathcal{X}$ or not:

$$\delta_x(\mathbf{L}) = \begin{cases} 1 & \text{if } \exists i \in \mathcal{V} : l_i^{\mathcal{X}} = x \\ 0 & \text{otherwise.} \end{cases} \tag{7.21}$$

Similarly, $h_x(\cdot)$ is another indicator function to determine whether a particular label x is supported at the video-level or not:

$$h_x(\mathbf{v}) = \begin{cases} 0 & \text{if } \exists z \in \mathcal{Z} : v_z = 1 \wedge g(z) = x \\ 1 & \text{otherwise,} \end{cases} \tag{7.22}$$

where $g(\cdot)$ maps a label in the joint actor-action space to the actor space. $\theta_{\mathcal{V}}$ is a constant cost for any label that exists in $\mathbf{L}$ but not supported at the video-level. We define $\delta_y(\cdot)$ and $h_y(\cdot)$ similarly. To make the cost meaningful, we set $\theta_B > 2\theta_{\mathcal{V}}$. In practice, we observe that these labeling costs from video-level recognition help the segment-level labeling to achieve a more parsimonious-in-labels result that enforces

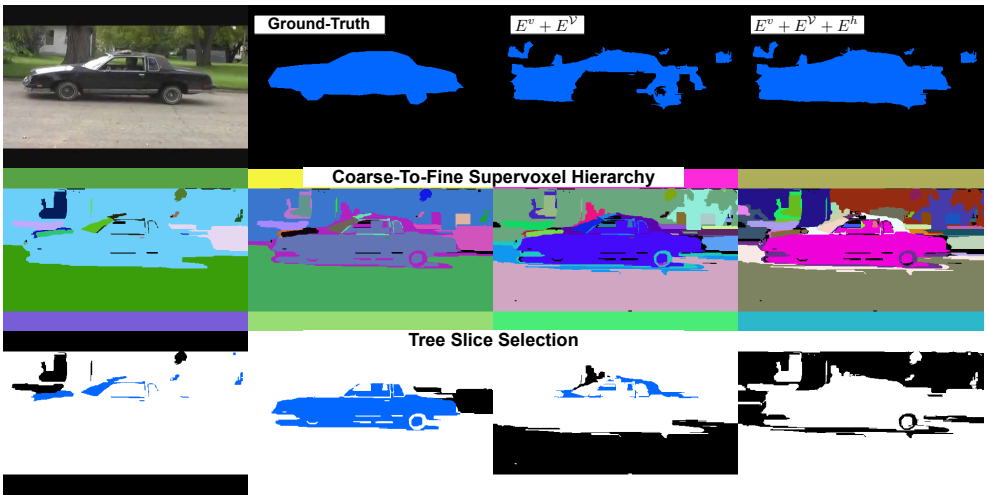


Figure 7.8: The video labeling of actor-action is refined by GPM. First row shows a test video *car-jumping* with its labelings. The second row shows a supervoxel hierarchy and the third row shows the active nodes in the hierarchy with their dominant labels.

more global information than using local segments alone (see results in Table 7.3).

7.5.3 The GPM Potentials E^h

The energy terms $E^h(\mathbf{s}|\mathcal{T})$ and $E^h(s_t|\mathbf{L}_t)$ involved in the tree slice problem are defined the same as in Sec. 7.3. Now, we define the new labeling term:

$$E^h(\mathbf{L}_t, \mathbf{v}|s_t) = \tag{7.23}$$
$$\begin{cases} \sum_{i\in\mathbf{L}_t}\sum_{j\neq i, j\in\mathbf{L}_t}\psi^h_{ij}(l^{\mathcal{X}}_i, l^{\mathcal{X}}_j, \mathbf{v}) \\ \quad + \sum_{i\in\mathbf{L}_t}\sum_{j\neq i, j\in\mathbf{L}_t}\phi^h_{ij}(l^{\mathcal{Y}}_i, l^{\mathcal{Y}}_j, \mathbf{v}) & \text{if } s_t = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Here, $\psi^h_{ij}(\cdot)$ has the form:

$$\psi^h_{ij}(l^{\mathcal{X}}_i, l^{\mathcal{X}}_j, \mathbf{v}) = \tag{7.24}$$
$$\begin{cases} \theta_t & \text{if } l^{\mathcal{X}}_i \neq l^{\mathcal{X}}_j, \exists z \in \mathcal{Z} : v_z = 1 \wedge g(z) = f(s_t) \\ 0 & \text{otherwise,} \end{cases}$$

where $f(\cdot)$ denotes the dominant actor label in the segment-level labeling field $\mathbf{L}_t$ that connected to s_t, and we define $\psi^h_{ij}(l^{\mathcal{Y}}_i, l^{\mathcal{Y}}_j, \mathbf{v})$ similarly. This new term selectively refines the segmentation where the majority of the segment-level labelings agree with the video-level multi-label labeling.

We show in Fig. 7.8 how this GPM process helps to refine the actor's shape (the car) in the labeling process. The initial labelings from $E^v + E^{\mathcal{V}}$ propose a rough region of interest, but they do not capture the accurate boundaries or shape. After two iterations of inference, the tree slice selects the best set of supervoxels in the GBH hierarchy that represents the actor (the car), and they regroup the segment-level labelings such that the labelings can better capture the actor shape. Notice that the car body in the third column merges with the background, but our full model (fourth column) overcomes the limitation by selecting different parts from the hierarchy to yield the final labeling.

7.5.4 Inference

The inference of the actor-action problem defined in Eq. 7.17 follows the iterative inference described in Sec. 7.4. The tree slice problem is efficiently solved by binary linear programming. Although we could solve the video labeling problem with loopy belief propagation, it would be expensive due to the two sets of labels over which the CRF is defined. Here, we derive a way to solve it efficiently using graph cuts inference with label costs [11, 12, 36]. We show this conceptually in Fig. 7.9 and rewrite Eq. 7.18 as:

$$E^v(\mathbf{L}|\mathcal{V}) = \sum_{i \in \mathcal{V}} \xi^v_i(l_i) + \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{E}(i)} \xi^v_{ij}(l_i, l_j) \ , \tag{7.25}$$

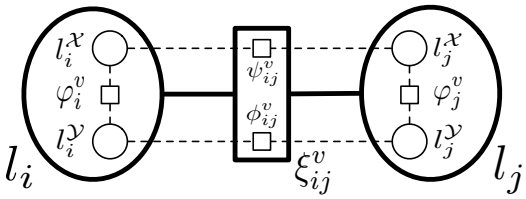


Figure 7.9: Visualization of two nodes of the bilayer model in our efficient inference.

where we define the new unary as:

$$\xi_i^v(l_i) = \psi_i^v(l_i^{\mathcal{X}}) + \phi_i^v(l_i^{\mathcal{Y}}) + \varphi_i^v(l_i^{\mathcal{X}}, l_j^{\mathcal{Y}}) \ , \tag{7.26}$$

and the pairwise interactions as:

$$\xi_{ij}^v(l_i, l_j) = \tag{7.27}$$
$$\begin{cases} \psi_{ij}^v(l_i^{\mathcal{X}}, l_j^{\mathcal{X}}) & \text{if } l_i^{\mathcal{X}} \neq l_j^{\mathcal{X}} \wedge l_i^{\mathcal{Y}} = l_j^{\mathcal{Y}} \\ \phi_{ij}^v(l_i^{\mathcal{Y}}, l_j^{\mathcal{Y}}) & \text{if } l_i^{\mathcal{X}} = l_j^{\mathcal{X}} \wedge l_i^{\mathcal{Y}} \neq l_j^{\mathcal{Y}} \\ \psi_{ij}^v(l_i^{\mathcal{X}}, l_j^{\mathcal{X}}) + \phi_{ij}^v(l_i^{\mathcal{Y}}, l_j^{\mathcal{Y}}) & \text{if } l_i^{\mathcal{X}} \neq l_j^{\mathcal{X}} \wedge l_i^{\mathcal{Y}} \neq l_j^{\mathcal{Y}} \\ 0 & \text{if } l_i^{\mathcal{X}} = l_j^{\mathcal{X}} \wedge l_i^{\mathcal{Y}} = l_j^{\mathcal{Y}}. \end{cases}$$

We can rewrite Eq. 7.23 in a similar way, and they satisfy the submodular property according to the triangle inequality [87]. The label costs can be solved as in [36].

Parameters. We manually explore the parameter space based on the pixel-level accuracy in a heuristic fashion. We first tune the parameters involved in the video-level recognition, then those involved in the segment-level labeling, and finally, those involved in GPM by running the iterative inference as in Sec. 7.4.

7.6 Experiments

We evaluate our method on the A2D dataset and evaluate the performance. We compare with the top-performing trilayer model, and two strong semantic image

	All Test Videos						Single Actor-Action Videos						Multiple Actor-Action Videos					
	Actor		Action		<A, A>		Actor		Action		<A, A>		Actor		Action		<A, A>	
Model	Ave.	Glo.	Ave.	Glo.	Ave.	Glo.	Ave.	Glo.	Ave.	Glo.	Ave.	Glo.	Ave.	Glo.	Ave.	Glo.	Ave.	Glo.
E^v	45.9	76.9	47.2	76.8	24.8	75.0	46.7	76.5	50.0	76.9	31.5	74.8	41.7	77.7	42.1	76.5	18.2	75.4
$E^v + E^{\mathcal{V}}$	57.3	85.7	59.4	85.9	42.4	84.8	60.4	86.0	67.0	86.5	55.4	85.4	50.6	85.1	55.1	84.4	33.3	83.6
AHRF	38.0	64.9	29.0	63.9	13.9	63.0	38.1	66.6	29.7	65.8	16.6	64.8	37.0	60.6	28.3	59.3	11.3	58.5
FCRF	44.8	77.9	45.5	77.6	25.4	76.2	45.9	77.6	47.4	77.7	32.1	76.1	40.2	78.8	42.2	77.5	19.4	76.5
Trilayer	45.7	74.6	47.0	74.6	26.5	72.9	47.0	74.1	50.3	74.6	33.9	72.7	41.0	75.6	42.3	74.5	20.4	73.4
GPM (TSP)	58.3	85.2	60.5	85.3	43.3	84.2	61.5	85.4	68.2	86.0	56.5	84.8	51.7	84.5	56.2	83.8	33.9	83.0
GPM (GBH)	61.2	84.9	59.4	84.8	43.9	83.8	63.1	85.1	69.3	85.7	57.6	84.5	51.7	84.1	56.3	83.3	33.9	82.5

Table 7.3: The overall performance on the A2D dataset. The top two rows are intermediate results of the full model. The middle three rows are comparison methods. The bottom two rows are our full models with different supervoxel hierarchies for the grouping process.

segmentation methods, AHRF [93] and FCRF [88]. For AHRF, we use the publicly available code from [93] as it contains a complete pipeline from training classifiers to learning and inference. For FCRF, we extend it to use the same features as our method.

Data Processing. We experiment with two distinct supervoxel trees: one is extracted from the hierarchical supervoxel segmentations generated by GBH [65], where supervoxels across multiple levels natively form a tree structure hierarchy, and the other one is extracted from multiple runs of a generic non-hierarchical supervoxel segmentation by TSP [23]. To extract a tree structure from the non-hierarchical video segmentations, we first sort the segmentations by the number of supervoxels they contain. Then we enforce the supervoxels in the finer level segmentation to have one and only one parent supervoxel in the coarser level segmentation, such that the two supervoxels have the maximal overlap of the video pixels. We use four levels from a GBH hierarchy, where the number of supervoxels varies from a few hundred to less than one hundred. We also use four different runs of TSP to construct another segmentation tree where the final number of nodes contained in the tree varies from 500 to 1500 at the fine level, and from 50 to 150 at the coarse level.

We also use TSP to generate the segments for the base labeling CRF. We extract the same set of appearance and motion features as in Sec. 7.1.3 and train one-versus-

all linear SVM classifiers on the segments for three sets of labels: actor, action, and actor-action pair, separately. At the video-level, we extract improved dense trajectories [204], and use Fisher vectors [149] to train linear SVM classifiers at the video-level for the actor-action pair. We use the inference schema described in Sec. 7.4 and Sec. 7.5.4, and follow the train/test splits used in Sec. 7.1.1. The output of our system is a full video pixel labeling. We evaluate the performance on sampled frames where the ground-truth is labeled.

Results and Comparisons. We evaluate performance for joint actor-action and separate individual tasks. Table 7.3 shows the overall results of all methods in three different calculations: when all test videos are used; when only videos containing single-label actor-action are used; and when only videos containing multiple actor-action labels are used. Roughly one-third of the videos in the A2D dataset have multiple actor-action labels. Overall, we observe that our methods (both GPM-TSP and GPM-GBH) outperform the next best one, the trilayer method, by a large margin of 17% average per-class accuracy and more than 10% global pixel accuracy over all test videos. The improvement of global pixel accuracy is consistent over the two subdivisions of test videos, and the improvement of average per-class accuracy is larger on videos that only contain single-label actor-action. We suspect that videos containing multiple-label actor-action are more likely to confuse the video-level classifiers.

We also observe that the added grouping process in GPM-TSP and GPM-GBH consistently improves the average per-class accuracy over the intermediate result ($E^v + E^{\mathcal{V}}$) on both single-label and multiple-label actor-action videos. There is a slight decrease on the global pixel accuracy. We suspect the decrease mainly comes from the background class, which contributes a large portion of the total pixels in evaluation. To verify that, we also show the individual actor-action class performance in Tab. 7.4 when all test videos are used. We observe that GPM-GBH has the best performance on majority classes and improves $E^v + E^{\mathcal{V}}$ on all classes except *dog-crawling*,

		adult								cat							dog						
Model	**BK**	**climb**	**crawl**	**eat**	**jump**	**roll**	**run**	**walk**	**none**	**climb**	**eat**	**jump**	**roll**	**run**	**walk**	**none**	**crawl**	**eat**	**jump**	**roll**	**run**	**walk**	**none**
E^v	81.0	22.1	60.4	45.2	20.0	18.9	32.3	26.8	31.5	25.3	29.8	4.4	29.5	45.2	6.5	0.0	17.0	26.6	1.1	38.1	**29.8**	38.7	0.0
$E^v + E^{\mathcal{V}}$	**89.9**	73.3	77.6	68.0	47.1	49.4	49.8	39.8	0.0	41.9	48.0	31.0	69.8	48.0	18.7	0.0	**45.8**	58.9	30.7	61.4	25.1	72.4	0.0
AHRF	69.2	0.0	56.0	6.1	1.1	0.0	0.0	15.3	10.9	18.3	38.8	0.0	8.8	0.0	9.3	0.0	13.2	16.4	0.0	0.0	0.0	0.0	0.0
FCRF	82.2	21.6	64.5	46.3	25.3	12.0	**50.9**	26.9	**33.8**	25.3	33.6	2.5	33.9	**48.9**	**21.5**	**0.8**	11.7	35.7	2.2	31.9	25.2	40.2	0.0
Trilayer	78.5	33.1	59.8	49.8	19.9	27.6	40.2	31.7	24.6	33.1	27.2	6.1	49.8	48.5	6.6	0.0	9.9	31.0	2.0	27.6	23.6	39.4	0.0
GPM (TSP)	89.1	74.6	79.8	70.7	**49.3**	51.5	50.6	40.4	0.0	42.5	49.3	31.9	71.1	46.4	18.8	0.0	45.3	60.2	31.3	62.5	25.8	74.0	0.0
GPM (GBH)	88.4	**74.8**	**81.0**	**76.4**	**49.3**	**52.4**	50.4	**41.0**	0.0	**42.8**	**52.3**	**33.7**	**71.7**	48.0	19.1	0.0	44.1	**61.5**	**31.4**	**62.6**	25.7	**74.2**	0.0

	baby					ball				bird							car						
Model	**climb**	**crawl**	**roll**	**walk**	**none**	**fly**	**jump**	**roll**	**none**	**climb**	**eat**	**fly**	**jump**	**roll**	**walk**	**none**	**fly**	**jump**	**roll**	**run**	**none**	**Ave.**	**Glo.**
E^v	13.8	32.8	38.3	20.0	0.0	3.8	10.4	4.5	0.0	28.1	14.1	51.6	18.2	33.1	7.2	0.0	25.7	78.0	35.7	45.9	1.8	24.8	75.0
$E^v + E^{\mathcal{V}}$	63.6	64.0	55.4	60.6	0.0	**11.3**	26.7	20.5	0.0	58.7	35.4	65.8	17.2	44.2	41.1	0.0	40.8	83.4	67.3	63.7	0.0	42.4	**84.8**
AHRF	21.3	5.5	39.8	13.5	0.0	3.2	2.3	13.6	**1.5**	14.6	11.4	19.9	5.0	29.6	7.5	0.0	18.1	68.0	13.6	47.9	**12.2**	13.9	63.0
FCRF	3.4	23.4	41.0	17.8	0.0	3.7	0.3	1.0	0.0	25.9	16.1	57.3	17.1	35.0	7.4	0.0	13.7	78.4	55.4	43.7	1.8	25.4	76.2
Trilayer	20.4	21.7	39.3	25.3	0.0	1.0	11.9	6.1	0.0	28.1	18.2	55.3	**20.3**	42.5	9.0	0.0	24.4	75.9	44.3	48.3	2.4	26.5	72.9
GPM (TSP)	65.3	64.7	57.2	60.5	0.0	**11.3**	27.0	20.8	0.0	**62.2**	37.1	**66.6**	17.4	45.4	42.2	0.0	**42.9**	84.5	69.2	64.8	0.0	43.3	84.2
GPM (GBH)	**65.4**	**65.0**	**58.4**	**61.5**	0.0	**11.3**	**28.3**	**21.1**	0.0	60.6	**38.8**	66.5	17.5	**45.9**	**47.9**	0.0	41.2	**86.3**	**70.9**	**65.9**	0.0	**43.9**	83.8

Table 7.4: The performance on individual actor-action labels using all test videos. The leading scores for each label are in bold font.

Figure 7.10: Visual example of the actor-action video labelings for all methods. (a) - (c) are videos where most methods get correct labelings; (d) - (g) are videos where only GPM models get the correct labelings; (h) - (g) are difficult videos in the dataset where the GPM models get partially correct labelings. Colors used are from the A2D benchmark.

which further shows the effectiveness of the grouping process. The performance of our method using the GBH hierarchy is slightly better than our method using the TSP hierarchy. We suspect that this is due to the GBH method's greedy merging process that complements the Gaussian process in TSP, such that the resulting segmentation complements the segment-level TSP segmentation we used.

Figure 7.10 shows the visual comparison of video labelings for all methods, where (a)-(c) show cases where methods output correct labels and (d)-(g) show cases where our proposed method outperforms other methods. We also show failure cases in (h)

and (i) where videos contain complex actors and actions. For example, our method correctly labels the *ball-rolling* but confuses the label *adult-running* as *adult-walking* in (h); we correctly label *adult-crawling* but miss the label *adult-none* in (i).

Inference Speed. We empirically set the stopping criteria by observing a balance between the performance gain and the running time. We set two iterations for all experiments. For all the test videos, GPM-GBH has an average inference speed of 8.6 seconds-per-video (spv) faster than 26.7 spv of GPM-TSP. Both of them are faster than 142 spv of the trilayer model in Sec. 7.1. The experiments are conducted with a Linux server with AMD Opteron 6380 2.5GHz CPU.

7.7 Conclusion

Our thorough experiments on the A2D dataset show that when the segment-level labeling is combined with secondary processes, such as our grouping process models and video-level recognition signals, the semantic segmentation performance increases dramatically. For example, GPM-GBH improves almost every class of actor-action labels compared to the intermediate result without the supervoxel hierarchy, i.e., without the dynamic grouping of CRF labeling variables. This finding strongly supports our motivating argument that the two sets of labels, actors and actions, are best modeled at different levels of granularities and that they have different emphases on space and time in a video.

Besides the new actor-action video understanding problem, we make the following contributions to the actor-action semantic segmentation problem:

1. A novel model that dynamically combines segment-level labeling with a hierarchical grouping process that influences connectivities of the labeling variables.
2. An efficient inference method that iteratively solves the two conditional tasks by graph cuts for labeling and binary linear programming for grouping allowing for continuous exchange of information.

3. A new framework that uses video-level recognition signals as cues for segment-level labeling thru global labeling costs and the grouping process model.
4. Our proposed method significantly improves performance (60% relative improvement over the next best method) on the recently released large-scale actor-action semantic video dataset.

Future Work. We set two directions for our future work. First, although our model is able to improve the labeling performance dramatically, the opportunity of this joint modeling to improve video-level recognition is yet to be explored. Second, our grouping process does not incorporate semantics in the supervoxel hierarchy; we believe this would further improve results.

CHAPTER VIII

Conclusion

Understanding the structure of the multiscale decompositions in visual data is of core importance to computer vision. Recall that no single level in the supervoxel hierarchy contains all desired structures. To that end, we have presented a set of tools to manipulate scales in supervoxel hierarchies including both scale generation and scale selection methods; and we have demonstrated that we can improve the performance on post hoc segmentation and video labeling tasks by adaptively choosing supervoxels from various levels in a hierarchy.

Scale generation. In the first part of scale generation, we evaluate a set of seven supervoxel methods in the context of what we consider to be a good supervoxel for video representation: namely, spatiotemporal uniformity, object/region boundary detection, region compression and parsimony. In addition to the basic properties, we evaluate the methods in a supervoxel classification task as a proxy for subsequent high-level uses of supervoxels in video analysis. The metrics and benchmark we developed have become the de facto standard evaluation for supervoxel methods. In the second part of scale generation, we address a key limitation that has traditionally prevented the supervoxel scale generation on long videos. We do so by proposing an approximation framework for streaming hierarchical scale generation motivated by the data stream idea: each frame is processed only once and does not change the

segmentation of previous frames. Our method represents the first and remains the only one, as of the time of writing this dissertation, that is able to generate multiscale decompositions for arbitrarily-long videos using constant memory.

Scale selection. We have subsequently presented two scale selection methods that are able to adaptively choose the scales for given applications. The first scale selection method flattens an entire supervoxel hierarchy into a single segmentation that overcomes the limitation induced by trivial selection of a single scale level—undersegmentation at coarser levels and oversegmentation at finer levels. Our method, called uniform entropy slice, selects supervoxels from various scales by balancing the relative level of information entropy in the field. We show that the selection can be driven by different post hoc feature criteria, such as motion-ness and object-ness, and that they result in different selection attentions. The second scale selection method combines the supervoxel hierarchy with a CRF for the task of video labeling of actors and actions. We formulate the scale selection problem and the video labeling problem in a joint framework. It defines a dynamic and continuous process of information exchange: the CRF influences which supervoxels in the hierarchy are active, and these active supervoxels, in turn, affect the connectivity in the CRF. We show that it is beneficial to adaptively choose the scales in the context of actor-action video labeling.

Aside from the computational methods, we present a visual psychophysical study of semantic retention in supervoxel hierarchies, where we conduct a systematic study of how well the actor and action semantics are retained in video supervoxel segmentation. The ultimate findings suggest that some semantics are well-retained in the video supervoxel hierarchies and can be used for further video analysis.

Therefore, this dissertation has laid down a foundation and provided a set of tools for further exploring the abilities of using supervoxels as a type of video representation for subsequent video analysis.

Many research problems have been revealed by this dissertation. We are interested

in exploring them in the future. The space-time of video can not only be modeled as a 3D volume, but also as 2D space plus 1D time, where time is treated differently from space. In Chapter II, we show that methods are proposed by both ideas across almost every type of video representation: interest points versus trajectories; two-stream CNN versus 3D convolution; and 3D supervoxel versus temporal superpixels. We believe the modeling of space and time is tightly tied with application needs. Therefore, we are interested in exploring when and where each spatiotemporal modeling in appropriate.

From fine-grained actions to activities and events, there are different granularities in video analysis. We explore scale generation and selection to the point that coarse action classes can be distinguished, such as bird-flying and adult-jumping. However, we know that seagulls fly differently than hummingbirds do, and that the same person can either jump far or jump near. We are interested in exploring whether supervoxels are able to model these fine-grained actions. Furthermore, their ability to model interactions and activities is also yet to be explored.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[2] Chad Aeschliman, Johnny Park, and Avinash C Kak. A probabilistic framework for joint segmentation and tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[3] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. What is an object? In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[4] Ori Amir, Irving Biederman, and Kenneth J. Hayworth. Sensitivity to nonaccidental properties across various shape dimensions. *Vision Research*, 62(0):35 – 43, 2012.

[5] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[6] Xue Bai and Guillermo Sapiro. Geodesic matting: A framework for fast interactive image and video segmentation and matting. *International Journal of Computer Vision*, 82(2):113–132, 2009.

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

[8] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[9] Andrei Barbu, Alexander Bridge, Zachary Burchill, Dan Coroian, Sven J. Dickinson, Sanja Fidler, Aaron Michaux, Sam Mussman, Siddharth Narayanaswamy, Dhaval Salvi, Lara Schmidt, Jiangnan Shangguan, Jeffrey Mark Siskind, Jarrell W. Waggoner, Song Wang, Jinlian Wei, Yifan Yin, and Zhiqi Zhang. Video in sentences out. In *Conference on Uncertainty in Artificial Intelligence*, 2012.

[10] A. Berg, J. Deng, S. Satheesh, H. Su, and L. Fei-Fei. Imagenet large scale visual recognition challenge, `http://www.image-net.org/challenges/LSVRC/2011/`, 2011.

[11] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[12] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[13] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary region segmentation of objects in nd images. In *IEEE International Conference on Computer Vision*, 2001.

[14] Matteo Bregonzio, Shaogang Gong, and Tao Xiang. Recognising action as clouds of space-time interest points. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[15] William Brendel and Sinisa Todorovic. Video object segmentation by tracking regions. In *IEEE International Conference on Computer Vision*, 2009.

[16] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.

[17] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *European Conference on Computer Vision*, 2008.

[18] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *European Conference on Computer Vision*, 2010.

[19] Ignas Budvytis, Vijay Badrinarayanan, and Roberto Cipolla. Semi-supervised video segmentation using tree structured graphical models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[20] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

[21] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[22] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

[23] Jason Chang, Donglai Wei, and John W. Fisher III. A video representation using temporal superpixels. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[24] A. Y. C. Chen and J. J. Corso. Temporally consistent multi-class video-object segmentation with the video graph-shifts algorithm. In *WMVC*, 2011.

[25] A. Y. C. Chen and Jason J. Corso. Propagating multi-class pixel labels throughout video frames. In *Proceedings of Western New York Image Processing Workshop*, 2010.

[26] Sheng Chen, Alan Fern, and Sinisa Todorovic. Multi-object tracking via constrained sequential labeling. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[27] W. Chen, C. Xiong, R. Xu, and J. J. Corso. Actionness ranking with lattice conditional ordinal random fields. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[28] Prakash Chockalingam, Nalin Pradeep, and Stan Birchfield. Adaptive fragments-based tracking of non-rigid objects using level sets. In *IEEE International Conference on Computer Vision*, 2009.

[29] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[30] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[31] Jason J Corso, Eitan Sharon, Shishir Dube, Suzie El-Saden, Usha Sinha, and Alan Yuille. Efficient multilevel brain tumor segmentation with integrated bayesian model classification. *Medical Imaging, IEEE Transactions on*, 27(5):629–640, 2008.

[32] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. Bilayer segmentation of live video. In *CVPR*, 2006.

[33] Pradipto Das, Chenliang Xu, Richard Doell, and Jason J. Corso. A thousand frames in just a few words: lingual description of videos through latent topics and sparse object stitching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[34] Kleber Jacques Ferreira de Souza, Arnaldo de Albuquerque Araújo, et al. Graph-based hierarchical video segmentation based on a simple dissimilarity measure. *Pattern Recognition Letters*, 2014.

[35] Thomas Dean, Mark A Ruzon, Mark Segal, Jonathon Shlens, Sudheendra Vijayanarasimhan, and Jay Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[36] Andrew Delong, Anton Osokin, Hossam N Isack, and Yuri Boykov. Fast approximate energy minimization with label costs. *International journal of computer vision*, 96(1):1–27, 2012.

[37] Daniel DeMenthon and Remi Megret. Spatio-temporal segmentation of video by hierarchical mean shift analysis. In *Statistical Methods in Video Processing Workshop*, 2002.

[38] J. Deng, A. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *European Conference on Computer Vision*, 2010.

[39] Konstantinos G Derpanis, Mikhail Sizintsev, Kevin Cannons, and Richard P Wildes. Efficient action spotting based on a spacetime oriented structure representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[40] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.

[41] Fabio Drucker and John MacCormick. Fast superpixels for video analysis. In *IEEE Workshop on Motion and Video Computing*, 2009.

[42] Ali Elqursh and Ahmed Elgammal. Online motion segmentation using dynamic label propagation. In *IEEE International Conference on Computer Vision*, 2013.

[43] Çigdem Eroglu Erdem, Bülent Sankur, and A Murat Tekalp. Performance measures for video object segmentation and tracking. *IEEE Transactions on Image Processing*, 13(7):937–951, 2004.

[44] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[45] Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1915–1929, 2013.

[46] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[47] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[48] Georgios Floros and Bastian Leibe. Joint 2d-3d temporally consistent semantic segmentation of street scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[49] Wolfgang Förstner and Eberhard Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *Intercommission Workshop of Int. Soc. for Photogrammetry and Remote Sensing*, 1987.

[50] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[51] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

[52] Charless Fowlkes, Serge Belongie, and Jitendra Malik. Efficient spatiotemporal grouping using the nystrom method. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[53] Katerina Fragkiadaki, Geng Zhang, and Jianbo Shi. Video segmentation by tracing discontinuities in a trajectory embedding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[54] Huazhu Fu, Dong Xu, Bao Zhang, and Stephen Lin. Object-based multiple foreground video co-segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[55] Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975.

[56] Fabio Galasso, Roberto Cipolla, and Bernt Schiele. Video segmentation with superpixels. In *Asian Conference on Computer Vision*, 2012.

[57] Fabio Galasso, Margret Keuper, Thomas Brox, and Bernt Schiele. Spectral graph reduction for efficient image and streaming video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[58] Fabio Galasso, Naveen Shankar Nagaraja, Tatiana Jimenez Cardenas, Thomas Brox, and Bernt Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *IEEE International Conference on Computer Vision*, 2013.

[59] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[60] Daniela Giordano, Francesca Murabito, Simone Palazzo, and Concetto Spampinato. Superpixel-based video object segmentation using perceptual organization and location prior. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[61] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, 2007.

[62] Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *IEEE International Conference on Computer Vision*, 2009.

[63] Hayit Greenspan, Jacob Goldberger, and Arnaldo Mayer. Probabilistic space-time video modeling via piecewise gmm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):384–396, 2004.

[64] Kalanit Grill-Spector. The neural basis of object perception. *Current opinion in neurobiology*, 13(2):159–166, 2003.

[65] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient hierarchical graph-based video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[66] Sergio Guadarrama, Niveda Krishnamoorthy, Girish Malkarnenkar, Subhashini Venugopalan, Raymond Mooney, Trevor Darrell, and Kate Saenko. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *IEEE International Conference on Computer Vision*, 2013.

[67] Abhinav Gupta, Aniruddha Kembhavi, and Larry S Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1775–1789, 2009.

[68] Allan Hanbury. How do superpixels affect image segmentation? In *Progress in Pattern Recognition, Image Analysis and Applications*, pages 178–186. Springer, 2008.

[69] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

[70] Glenn Hartmann, Matthias Grundmann, Judy Hoffman, David Tsai, Vivek Kwatra, Omid Madani, Sudheendra Vijayanarasimhan, Irfan Essa, James Rehg, and Rahul Sukthankar. Weakly supervised learning of object segmentations from web-scale video. In *European Conference on Computer Vision Workshops*, pages 198–208. Springer, 2012.

[71] Xuming He, Richard S Zemel, and Debajyoti Ray. Learning and incorporating top-down cues in image segmentation. In *European Conference on Computer Vision*, 2006.

[72] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[73] Derek Hoiem, Alexei A Efros, and Martial Hebert. Automatic photo pop-up. In *ACM Transactions on Graphics*, volume 24, pages 577–584. ACM, 2005.

[74] Adam Hunter and Jonathan D Cohen. Uniform frequency images: adding geometry to images to produce space-efficient textures. In *Visualization 2000. Proceedings*, pages 243–250. IEEE, 2000.

[75] Y. Iwashita, A. Takamine, R. Kurazume, and M. S. Ryoo. First-person animal activity recognition from egocentric videos. In *IEEE International Conference on Pattern Recognition*, 2014.

[76] Hamid Izadinia and Mubarak Shah. Recognizing complex events using large margin joint low-level event model. In *European Conference on Computer Vision*, 2012.

[77] Aastha Jain, Shuanak Chatterjee, and Rene Vidal. Coarse-to-fine semantic video segmentation using supervoxel trees. In *IEEE International Conference on Computer Vision*, 2013.

[78] Arpit Jain, Abhinav Gupta, Mikel Rodriguez, and Larry S Davis. Representing videos using mid-level discriminative patches. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[79] Mihir Jain, Jan Van Gemert, Hervé Jégou, Patrick Bouthemy, Cees Snoek, et al. Action localization with tubelets from motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[80] Suyog Dutt Jain and Kristen Grauman. Supervoxel-consistent foreground propagation in video. In *European Conference on Computer Vision*, 2014.

[81] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J. Black. Towards understanding action recognition. In *IEEE International Conference on Computer Vision*, 2013.

[82] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013.

[83] Yan Ke, Rahul Sukthankar, and Martial Hebert. Spatio-temporal shape and flow correlation for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[84] Sohaib Khan and Mubarak Shah. Object based segmentation of video using color, motion and spatial information. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[85] Alexander Kläser, Marcin Marszałek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *British Machine Vision Conference*, 2008.

[86] Orit Kliper-Gross, Yaron Gurovich, Tal Hassner, and Lior Wolf. Motion interchange patterns for action recognition in unconstrained videos. In *European Conference on Computer Vision*, 2012.

[87] Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, 2004.

[88] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in Neural Information Processing Systems*, 2011.

[89] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[90] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *IEEE International Conference on Computer Vision*, 2011.

[91] Abhijit Kundu, Yin Li, Frank Daellert, Fuxin Li, and James M. Rehg. Joint semantic segmentation and 3d reconstruction from monocular video. In *European Conference on Computer Vision*, 2014.

[92] Lubor Ladicky, Christopher Russell, Pushmeet Kohli, and Philip HS Torr. Associative hierarchical crfs for object class image segmentation. In *IEEE International Conference on Computer Vision*, 2009.

[93] Lubor Ladicky, Craig Russell, Pushmeet Kohli, and Philip HS Torr. Associative hierarchical random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6):1056–1077, 2014.

[94] L'ubor Ladickỳ, Paul Sturgess, Karteek Alahari, Chris Russell, and Philip HS Torr. What, where and how many? combining object detectors and crfs. In *European Conference on Computer Vision*, 2010.

[95] Lubor Ladickỳ, Paul Sturgess, Chris Russell, Sunando Sengupta, Yalin Bastanlar, William Clocksin, and Philip HS Torr. Joint optimization for object class segmentation and dense stereo reconstruction. *International Journal of Computer Vision*, 100(2):122–133, 2012.

[96] Tian Lan, Yang Wang, Weilong Yang, and Greg Mori. Beyond actions: Discriminative models for contextual group activities. In *Advances in Neural Information Processing Systems*, 2010.

[97] Tian Lan, Yuke Zhu, Amir Roshan Zamir, and Silvio Savarese. Action recognition by hierarchical mid-level action elements. In *IEEE International Conference on Computer Vision*, 2015.

[98] Ivan Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2):107–123, 2005.

[99] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[100] Juho Lee and Seungjin Choi. Incremental tree-based inference with dependent normalized random measures. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 558–566, 2014.

[101] Juho Lee, Suha Kwak, Bohyung Han, and Seungjin Choi. Online video segmentation by bayesian split-merge clustering. In *European Conference on Computer Vision*, 2012.

[102] Yong Jae Lee, Jaechul Kim, and Kristen Grauman. Key-segments for video object segmentation. In *IEEE International Conference on Computer Vision*, 2011.

[103] Victor Lempitsky, Andrea Vedaldi, and Andrew Zisserman. A pylon model for semantic segmentation. In *Advances in Neural Information Processing Systems*, 2011.

[104] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2005.

[105] Alex Levinshtein, Adrian Stere, Kiriakos N Kutulakos, David J Fleet, Sven J Dickinson, and Kaleem Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2290–2297, 2009.

[106] José Lezama, Karteek Alahari, Josef Sivic, and Ivan Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[107] Fuxin Li, Taeyoung Kim, Ahmad Humayun, David Tsai, and James M. Rehg. Video segmentation by tracking many figure-ground segments. In *IEEE International Conference on Computer Vision*, 2013.

[108] Zhenguo Li, Xiao-Ming Wu, and Shih-Fu Chang. Segmentation using superpixels: A bipartite graph partitioning approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[109] Dahua Lin, Sanja Fidler, Chen Kong, and Raquel Urtasun. Visual semantic search: Retrieving videos via complex textual queries. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[110] Tony Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of Applied Statistics*, 21(1-2):225–270, 1994.

[111] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, 1998.

[112] Buyu Liu and Xuming He. Multiclass semantic video segmentation with object-level active inference. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[113] Buyu Liu, Xuming He, and Stephen Gould. Joint semantic and geometric segmentation of videos with a stage model. In *IEEE Winter Conference on Applications of Computer Vision*, 2014.

[114] Ce Liu. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, Massachusetts Institute of Technology, 2009.

[115] Ce Liu, William T Freeman, Edward H Adelson, and Yair Weiss. Human-assisted motion annotation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[116] Jingen Liu, Benjamin Kuipers, and Silvio Savarese. Recognizing human actions by attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[117] Jingen Liu, Jiebo Luo, and Mubarak Shah. Recognizing realistic actions from videos "in the wild". In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[118] Ming-Yu Liu, Oncel Tuzel, Srikumar Ramalingam, and Rama Chellappa. Entropy rate superpixel segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[119] Siying Liu, Guo Dong, Chye Hwang Yan, and Sim Heng Ong. Video segmentation: Propagation, validation and aggregation of a preceding graph. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[120] Xiao Liu, Dacheng Tao, Mingli Song, Ying Ruan, Chun Chen, and Jiajun Bu. Weakly supervised multiclass video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[121] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[122] Jiasen Lu, Ran Xu, and Jason J. Corso. Human action segmentation with hierarchical supervoxel consistency. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[123] Anette Lundvall. Ericsson mobility report. Technical report, Ericsson, 2016.

[124] Shugao Ma, Leonid Sigal, and Stan Sclaroff. Space-time tree ensemble for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[125] Shugao Ma, Jianming Zhang, Nazli Ikizler-Cinbis, and Stan Sclaroff. Action recognition and localization by hierarchical space-time segments. In *IEEE International Conference on Computer Vision*, 2013.

[126] Tianyang Ma and Longin Jan Latecki. Maximum weight cliques with mutex constraints for video object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[127] Marcin Marszalek, Ivan Laptev, and Cordelia Schmid. Actions in context. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[128] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE International Conference on Computer Vision*, 2001.

[129] Pyry Matikainen, Martial Hebert, and Rahul Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *IEEE International Conference on Computer Vision Workshops*, 2009.

[130] Remi Megret and Daniel DeMenthon. A survey of spatio-temporal grouping techniques. Technical report, UMD, 2002.

[131] Ross Messing, Chris Pal, and Henry Kautz. Activity recognition using the velocity histories of tracked keypoints. In *IEEE International Conference on Computer Vision*, 2009.

[132] Ondrej Miksik, Daniel Munoz, J Andrew Bagnell, and Martial Hebert. Efficient temporal consistency for streaming video scene analysis. In *IEEE International Conference on Robotics and Automation*, 2013.

[133] Alastair P Moore, Simon Prince, Jonathan Warrell, Umar Mohammed, and Graham Jones. Superpixel lattices. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[134] Greg Mori, Xiaofeng Ren, Alexei A Efros, and Jitendra Malik. Recovering human body configurations: Combining segmentation and recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[135] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[136] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.

[137] Manjunath Narayana, Allen Hanson, and Erik Learned-Miller. Coherent motion segmentation in moving camera videos using optical flow orientations. In *IEEE International Conference on Computer Vision*, 2013.

[138] Juan Carlos Niebles, Chih-Wei Chen, and Li Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *European Conference on Computer Vision*, 2010.

[139] J Farley Norman, Flip Phillips, and Heather E Ross. Information concentration along the boundary contours of naturally shaped solid objects. *Perception*, 30(11):1285–1294, 2001.

[140] Peter Ochs and Thomas Brox. Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions. In *IEEE International Conference on Computer Vision*, 2011.

[141] Dan Oneata, Jérôme Revaud, Jakob Verbeek, and Cordelia Schmid. Spatio-temporal object detection proposals. In *European Conference on Computer Vision*, 2014.

[142] Haluk Öğmen. A theory of moving form perception: Synergy between masking, perceptual grouping, and motion computation in retinotopic and non-retinotopic representations. *Advances in Cognitive Psychology*, 3(1):67–84, 2007.

[143] G. Palou and P. Salembier. Hierarchical video representation with trajectory binary partition tree. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[144] Anestis Papazoglou and Vittorio Ferrari. Fast object segmentation in unconstrained video. In *IEEE International Conference on Computer Vision*, 2013.

[145] Sylvain Paris. Edge-preserving smoothing and mean-shift segmentation of video streams. In *European Conference on Computer Vision*, 2008.

[146] Sylvain Paris and Frédo Durand. A topological approach to hierarchical segmentation using mean shift. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[147] Nilesh V Patel and Ishwar K Sethi. Video shot detection and characterization for video databases. *Pattern Recognition*, 30(4):583–592, 1997.

[148] Mingtao Pei, Yunde Jia, and Song-Chun Zhu. Parsing video events with goal inference and intent prediction. In *IEEE International Conference on Computer Vision*, 2011.

[149] Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng. Action recognition with stacked fisher vectors. In *European Conference on Computer Vision*, 2014.

[150] Jeannine Pinto and Maggie Shiffrar. The visual perception of human and animal motion in point-light displays. *Social Neuroscience*, 4(4):332–346, 2009.

[151] Jordi Pont-Tuset and Ferran Marques. Supervised assessment of segmentation hierarchies. In *European Conference on Computer Vision*, 2012.

[152] Brian L Price, Bryan S Morse, and Scott Cohen. Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *IEEE International Conference on Computer Vision*, 2009.

[153] S Avinash Ramakanth and R Venkatesh Babu. Seamseg: Video object segmentation using patch seams. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[154] Michalis Raptis and Stefano Soatto. Tracklet descriptors for action modeling and video analysis. In *European Conference on Computer Vision*, 2010.

[155] S Hussain Raza, Matthias Grundmann, and Irfan Essa. Geometric context from videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[156] Kishore K Reddy and Mubarak Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications Journal*, 2012.

[157] Xiaofeng Ren, Liefeng Bo, and D Fox. Rgb-(d) scene labeling: Features and algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[158] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *IEEE International Conference on Computer Vision*, 2003.

[159] Xiaofeng Ren and Jitendra Malik. Tracking as repeated figure/ground segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[160] Matthias Reso, Jörn Jachalsky, Bodo Rosenhahn, and Jörn Ostermann. Temporally consistent superpixels. In *IEEE International Conference on Computer Vision*, 2013.

[161] M.D. Rodriguez, J. Ahmed, and M. Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[162] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173, 2008.

[163] Sreemanananth Sadanand and Jason J Corso. Action bank: A high-level representation of activity in video. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[164] Mohammad Amin Sadeghi and Ali Farhadi. Recognition using visual phrases. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[165] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–535, 1997.

[166] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

[167] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *IEEE International Conference on Pattern Recognition*, 2004.

[168] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM International Conference on Multimedia*, 2007.

[169] Eitan Sharon, Achi Brandt, and Ronen Basri. Fast multiscale image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[170] Eitan Sharon, Meirav Galun, Dahlia Sharon, Ronen Basri, and Achi Brandt. Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442(7104):810–813, 2006.

[171] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[172] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from a single depth image. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[173] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision*, 2006.

[174] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23, 2009.

[175] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, 2014.

[176] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[177] Paul Sturgess, Karteek Alahari, Lubor Ladicky, and Philip Torr. Combining appearance and structure from motion features for road scene understanding. In *British Machine Vision Conference*, 2009.

[178] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[179] Ju Sun, Yadong Mu, Shuicheng Yan, and Loong-Fah Cheong. Activity recognition using dense long-duration trajectories. In *IEEE International Conference on Multimedia and Expo*, 2010.

[180] Ju Sun, Xiao Wu, Shuicheng Yan, Loong-Fah Cheong, Tat-Seng Chua, and Jintao Li. Hierarchical spatio-temporal context modeling for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[181] Patrik Sundberg, Thomas Brox, Michael Maire, Pablo Arbeláez, and Jitendra Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[182] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[183] A. Tamrakar, S. Ali, Q. Yu, J. Liu, O. Javed, A. Divakaran, H. Cheng, and H. Sawhney. Evaluation of low-level features and their combinations for complex event detection in open source videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[184] J. Tang, S. Yan, R. Hong, G.-J. Qi, and T.-S. Chua. Inferring semantic concepts from community-contributed images and noisy tags. In *ACM International Conference on Multimedia*, 2009.

[185] Kevin Tang, Rahul Sukthankar, Jay Yagnik, and Li Fei-Fei. Discriminative segment annotation in weakly labeled video. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[186] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth J Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI Conference on Artificial Intelligence*, 2011.

[187] Yicong Tian, Rahul Sukthankar, and Mubarak Shah. Spatiotemporal deformable part models for action detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[188] Joseph Tighe and Svetlana Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. *International Journal of Computer Vision*, 2010.

[189] Joseph Tighe and Svetlana Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. In *European Conference on Computer Vision*, 2010.

[190] Joseph Tighe and Svetlana Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. *International Journal of Computer Vision*, 2012.

[191] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *IEEE International Conference on Computer Vision*, 2015.

[192] Subarna Tripathi, Youngbae Hwang, Serge Belongie, and Truong Nguyen. Improving streaming video segmentation with early and mid-level visual processing. In *IEEE Winter Conference on Applications of Computer Vision*, 2014.

[193] David Tsai, Matthew Flagg, and James M Rehg. Motion coherent tracking with multi-label mrf optimization. In *British Machine Vision Conference*, 2010.

[194] Tinne Tuytelaars and Luc J Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *British Machine Vision Conference*, 2000.

[195] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[196] Michael Van den Bergh, Gemma Roig, Xavier Boix, Santiago Manen, and Luc Van Gool. Online video seeds for temporal window objectness. In *IEEE International Conference on Computer Vision*, 2013.

[197] David Varas and Ferran Marques. Region-based particle filter for video object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[198] Amelio Vazquez-Reina, Shai Avidan, Hanspeter Pfister, and Eric Miller. Multiple hypothesis video segmentation from superpixel flows. In *European Conference on Computer Vision*, 2010.

[199] Olga Veksler, Yuri Boykov, and Paria Mehrani. Superpixels and supervoxels in an energy optimization framework. In *European Conference on Computer Vision*, 2010.

[200] Alexander Vezhnevets, Vittorio Ferrari, and Joachim M Buhmann. Weakly supervised semantic segmentation with a multi-image model. In *IEEE International Conference on Computer Vision*, 2011.

[201] Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[202] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[203] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 2013.

[204] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *IEEE International Conference on Computer Vision*, 2013.

[205] Jue Wang, Bo Thiesson, Yingqing Xu, and Michael Cohen. Image and video segmentation by anisotropic kernel mean shift. In *European Conference on Computer Vision*, 2004.

[206] Jue Wang, Yingqing Xu, Heung-Yeung Shum, and Michael F Cohen. Video tooning. In *ACM SIGGRAPH*, 2004.

[207] Le Wang, Gang Hua, Rahul Sukthankar, Jianru Xue, and Nanning Zheng. Video object discovery and co-segmentation with extremely weak supervision. In *European Conference on Computer Vision*, 2014.

[208] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[209] Shu Wang, Huchuan Lu, Fan Yang, and Ming-Hsuan Yang. Superpixel tracking. In *IEEE International Conference on Computer Vision*, 2011.

[210] Geert Willems, Tinne Tuytelaars, and Luc Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *European Conference on Computer Vision*, 2008.

[211] Andrew P. Witkin. Scale-space filtering. In *International Joint Conference on Artificial Intelligence*, 1983.

[212] Christian Wojek and Bernt Schiele. A dynamic conditional random field model for joint labeling of object and scene classes. In *European Conference on Computer Vision*, 2008.

[213] Shu-Fai Wong and Roberto Cipolla. Extracting spatiotemporal interest points using global information. In *IEEE International Conference on Computer Vision*, 2007.

[214] Chenliang Xu and Jason J Corso. Evaluation of super-voxel methods for early video processing. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[215] Chenliang Xu and Jason J. Corso. Actor-action semantic segmentation with grouping process models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[216] Chenliang Xu and Jason J. Corso. Libsvx: A supervoxel library and benchmark for early video processing. *International Journal of Computer Vision*, 2016.

[217] Chenliang Xu, Richard F. Doell, Stephen José Hanson, Catherine Hanson, and Jason J. Corso. Are actor and action semantics retained in video supervoxel segmentation? In *IEEE International Conference on Semantic Computing*, 2013.

[218] Chenliang Xu, Richard F. Doell, Stephen José Hanson, Catherine Hanson, and Jason J. Corso. A study of actor and action semantic retention in video supervoxel segmentation. *International Journal of Semantic Computing*, 07(04):353–375, 2013.

[219] Chenliang Xu, Shao-Hang Hsieh, Caiming Xiong, and Jason J. Corso. Can humans fly? action understanding with multiple classes of actors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[220] Chenliang Xu, Spencer Whitt, and Jason J. Corso. Flattening supervoxel hierarchies by the uniform entropy slice. In *IEEE International Conference on Computer Vision*, 2013.

[221] Chenliang Xu, Caiming Xiong, and Jason J. Corso. Streaming hierarchical video segmentation. In *European Conference on Computer Vision*, 2012.

[222] Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas Guibas, and Li Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *IEEE International Conference on Computer Vision*, 2011.

[223] Jian Yao, Sanja Fidler, and Raquel Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[224] Lahav Yeffet and Lior Wolf. Local trinary patterns for human action recognition. In *IEEE International Conference on Computer Vision*, 2009.

[225] Jenny Yuen, Bryan Russell, Ce Liu, and Antonio Torralba. Labelme video: Building a video database with human annotations. In *IEEE International Conference on Computer Vision*. IEEE, 2009.

[226] Gang Zeng, Peng Wang, Jingdong Wang, Rui Gan, and Hongbin Zha. Structure-sensitive superpixels via geodesic distance. In *IEEE International Conference on Computer Vision*, 2011.

[227] Dong Zhang, Omar Javed, and Mubarak Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[228] Dong Zhang, Omar Javed, and Mubarak Shah. Video object co-segmentation by regulated maximum weight cliques. In *European Conference on Computer Vision*, 2014.

[229] Weiyu Zhang, Menglong Zhu, and Konstantinos G. Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. In *IEEE International Conference on Computer Vision*, 2013.

[230] Yu Zhang, Xiaowu Chen, Jia Li, Chen Wang, and Changqun Xia. Semantic object segmentation via detection in weakly labeled video. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[231] Jun Zhu, Baoyuan Wang, Xiaokang Yang, Wenjun Zhang, and Zhuowen Tu. Action recognition with actons. In *IEEE International Conference on Computer Vision*, 2013.

[232] C Lawrence Zitnick and Devi Parikh. The role of image understanding in contour detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.