Feature Quantization and Pooling for Videos

Ekaterina Taralova

CMU-CS-14-135
May 2014

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Professor Martial Hebert, Co-chair
Professor Fernando De la Torre, Co-chair
Professor Kayvon Fatahalian
Professor Silvio Savarese, Stanford University

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science.

This research was made possible through the generous support of the National Science Foundation under Grant No. EEEC-0540865 and Graduate Research Fellowship No. DGE0750271, the Air Force Office of Scientific Research under Grant No. FA95500810356, and the PIA Consortium. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Videos, motion features, action classification and retrieval, quantization, pooling.

Abstract

Building video representations typically involves four steps: feature extraction, quantization, encoding, and pooling. While there have been large advances in feature extraction and encoding, the questions of how to quantize video features and what kinds of regions to pool them over have been relatively unexplored. To tackle the challenges present in video data, it is necessary to develop robust quantization and pooling methods.

The first contribution of this thesis, Source Constrained Clustering, quantizes features into a codebook that generalizes better across actions. The main insight is to incorporate readily available labels of the sources generating the data. Sources can be the people who performed each cooking recipe, the directors who made each movie, or the YouTube users who shared their videos.

In the pooling step, it is common to pool feature vectors over local regions. The regions of choice include the entire video, coarse spatio-temporal pyramids, or cuboids of pre-determined fixed size. A consequence of using indiscriminately chosen cuboids is that widely dissimilar features may be pooled together if they are in nearby locations. It is natural to consider pooling video features over supervoxels, for example, obtained from a video segmentation. However, since videos can have a different number of supervoxels, this produces a video representation of variable size. The second contribution of this thesis is a new, fixed size video representation, Motion Words, where we pool features over video segments.

The ultimate goal of video segmentation is to recover object boundaries, often grouping pixels from regions of very different motion. However, in the context of Motion Words, it is important that regions preserve motion boundaries. The third contribution of this thesis is a supervoxel segmentation, Globally Consistent Supervoxels, which respects motion boundaries and provides better spatio-temporal support for Motion Words.

Finally, it is essential that the representations we build are interpretable, that is, we want to be able to visualize and understand *why* videos are similar. Motion Words enable localization of common regions across videos in both space and time. This gives us the power to understand which regions make videos similar.

Acknowledgments

To Martial Hebert and Fernando De la Torre, my advisors at Carnegie Mellon University, to Kobus Barnard, my advisor at University of Arizona, and to my dear friends: your support, encouragement and the fantastic hours spent in your presence has fueled and inspired this thesis. Kobus, for introducing me to computer vision, and for seamlessly convincing me that my future path should include a PhD. Martial and Fernando, your energy is exceptional! Its influence can be seen in the burning passion I have my work. At the end of every meeting, no matter how I had felt before, I always left with a renewed enthusiasm, determined to tackle on the most challenging research problems.

I am very grateful to the members of my dissertation committee, Kayvon Fatahalian and Silvio Savarese, who have generously given their time and expertise to better my work. Thank you for your contribution and support.

To the Computer Science Department, the Robotics Institute and the Machine Learning department, for promoting the development of exceptional researchers. To Catherine Copetas and Deborah Cavlovich: knowing I can always depend on you has been instrumental during my journey as a graduate student. To Lynnetta Miller, who can do anything in no time, always perfectly. To Ed Walter, who was always there with immediate support and solutions to server problems at all times of the day and the week. To the members of the Computer Vision group for the insightful and inspiring conversations.

To my lab mates, especially Scott Satkin, Daniel Munoz, Santosh Divvala, Edward Hsiao: your life advice, research insights and persistent encouragement helped me tackle this journey with ease. To Brendan Meeder, Dafna Shahaf, Dana and Yair Movshovitz-Attias, Justine Kasznica, Vladimir Ivanov, Victor Hwang: your endless support gave me energy to keep going and kept me on track.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Goal

This thesis addresses the question of constructing video representations that are robust to the challenges presented by videos collected in unconstrained environments. These challenges arise from camera motion, illumination changes, poor video quality, variability in appearance, and different styles of executing actions. Building the video representations typically involves four steps: feature extraction, quantization, encoding, and pooling. While there have been large advances in feature extraction and encoding, the questions of how to quantize video features and what kinds of regions to pool them over have been relatively unexplored. To tackle the challenges present in video data, it is necessary to develop robust quantization and pooling methods. Furthermore, it is essential that the representations we build are interpretable, that is, we want to be able to visualize and understand *why* videos are similar.

1.2 Context

Video has become a very popular media for communication, entertainment, and science. Videos are widely used in educational websites like Khan Academy, Video Lectures, WatchKnow, and free online courses by MIT, Stanford, Harvard, Berkeley. The YouTube website has more than 1 trillion views, more than 1 billion unique users each month, 100 hours of video are uploaded every minute[1]. Fields outside of computer science rely on video analysis to model and understand human behavior to address questions in healthy living and quality of life. Medical imaging

[1] http://www.youtube.com/yt/press/statistics.html

uses video understanding to better diagnose and understand health problems. Other important application areas are video surveillance, manufacturing, autonomous vehicles, wearable sensors, robotics, human computer interaction systems, gaming devices, movies. Video is important to these applications because it captures visual information about a real-world event over a period of time, which can be used for later analysis.

The main challenge in these applications is how to build technologies which can find and present useful information to the user. Due to the vast amounts of video data, doing so manually is infeasible. We need powerful automatic tools to analyze visual content. It is important to be able to build compact and robust models for visual data, which we can be used to quickly interpret the data and to predict future events.

The focus of this thesis is on action classification, which is an important part of understanding visual data. Computer vision has made tremendous progress in the areas directly related to this task: features for robust video representation, compact models, efficient classifier methods, object detection, meaningful video segmentation. Nevertheless, low performance on challenging benchmark datasets, e.g. [22, 47], demonstrate the need for methods that are more robust to every day video data.

We aim to classify generic actions performed in videos collected in uncontrolled conditions, e.g., YouTube videos [47, 57] and first person videos [22, 86]. Our goal is to design robust representations for a broad range of action categories that generalizes well across different styles of execution and that can provide interpretability of the classification results.

1.3 Contributions

This thesis considers building robust video representations for the task of recognizing actions performed in unconstrained environments. The steps involved in constructing video representations, namely, feature extraction, quantization, encoding, and pooling, are also the key components of the popular Bag of Words framework [20, 64, 85, 109]. We address two fundamental aspects of this system and validate the proposed methods in the context of the Bag of Words framework applied to activity classification: 1) quantizing features to enable generalization across different execution styles, and 2) providing a richer spatio-temporal support for feature pooling to enable robust encoding of the variable action layouts. In addition, we provide a way of understanding which features the classifier uses to assign a particular class label to each video, i.e., we enable visualization of the "words" the classifier has discovered as important for the action model.

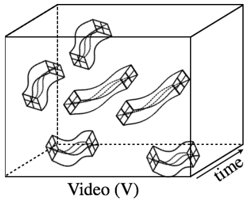
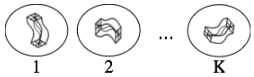
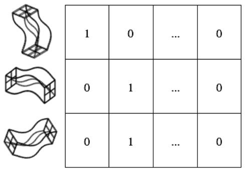
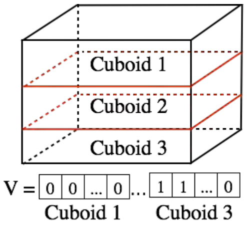
1. **Feature extraction**	2. **Quantization**	3. Encoding	4. **Pooling**
			 Cuboid 1 Cuboid 2 Cuboid 3 V = 0 0 ... 0 ... 1 1 ... 0 Cuboid 1 Cuboid 3
Interest point [28, 52, 62, 67, 78, 97], patch-based [5, 42, 100, 110], global [68, 92], semantic [56].	Unsupervised clustering [4, 7, 23, 27, 51, 59, 96, 106].	Hard, soft, locality-constrained linear (LLC) [14].	Aggregation strategy: sum or max pooling [6, 14]. Regions: global [28, 78], cuboids [52, 56, 62, 100], spatio-temporal pyramids (STP) [97], Fisher Vectors [14].

Figure 1.1: The standard Bag of Words (BoW) typically includes four steps, which can be executed using various methods. While there have been large advances in feature extraction and encoding, the questions of how to quantize video features and what kinds of regions to pool them over have been relatively unexplored. This thesis builds upon and makes contributions to how we quantize, pool and represent videos.

We build upon the Bag of Words framework which has been successfully applied to videos as a bag of local spatio-temporal features [28, 52, 62, 67, 78, 97]. To address the challenge of creating robust codebooks for videos from unconstrained environments, we augment the quantization step with readily available source labels. For example, sources can be subjects performing actions with different styles, movies with particular genre bias, videos collected from different news sources, etc. We propose an extension of the K-means [59] algorithm, which is one of the most widely used techniques for clustering and learning codebooks, due to its simplicity and good performance. We build on existing work on unsupervised discriminative clustering [4, 23, 27, 106] and distance metric learning algorithms [82, 103] by adding constraints which enforce clusters to group samples from multiple sources.

Next, to provide more robust spatio-temporal support for modeling the layouts of actions, we extend the standard approach of globally pooling local features over the entire video [28, 97]. We build on works which propose pooling features over spatio-temporal pyramids [97], and pooling features over fixed cuboids to create learn mid-level representations [52, 56]. Inspired by superpixels [65] and video segmentation algorithms [8, 18, 37], we propose a new intermediate representation for video, Motion Words, where we pool features over coherent spatio-temporal regions. Motion Words encode global cues and enable interpretability of the classification results.

> **Thesis statement:** This thesis contributes a new quantization method and more robust spatio-temporal support for pooling features to build robust video representations for the task of action recognition. The proposed methods enable better generalization across activity styles, add flexibility to modeling the spatio-temporal layout of unconstrained actions, and add interpretability and localization of the regions important during the classification step.

First, we present **Source Constrained Clustering (SCC)** - a modification to k-means which addresses the problem of feature quantization in datasets generated by diverse sources (e.g. subjects performing actions with different styles). Our experiments show improvement in action classification performance across several tasks and features compared to standard k-means. Furthermore, we show that improvement in performance is correlated with the variability of the sources.

Second, we propose **Motion Words (MWs)** - a new video representation which adds flexibility in modeling actions with varying spatio-temporal layouts and adds interpretability to the BoW framework. Rather than pool features over rigid cuboids or spatio-temporal pyramids (e.g. [62, 97, 100]), in Motion Words we pool features over coherent spatio-temporal regions, e.g. regions generated by a video segmentation. Importantly, since video segmentation algorithms produce meaningful spatio-temporal regions, Motion Words enable localization and interpretation of the "words" used at classification time. This approach differs from action localization methods (e.g. [80]), which attempt to locate the action region and which require additional supervision. Instead, we build Motion Words in an unsupervised manner and simultaneously make it possible to explain why the classifier assigns the final action label in each video.

Third, in the context of Motion Words, we propose a novel supervoxel segmentation method which encodes *global* motion properties. We propose **Globally Consistent Supervoxels (GCSs)**: groups of spatially neighboring and temporally associated spatio-temporal regions with the same instantaneous motion which preserve global motion relationships. The grouping is done in an unsupervised manner, without the assumption of a specific motion model. The resulting supervoxels encode both local appearance and global motion cues.

We present extensive experiments on several datasets and show that the proposed video representation achieves better action classification performance, giving us the power to understand *why* videos are classified as similar.

The remainder of this document is structured as follows. In Chapter 2 we present and validate the proposed new quantization technique, Source Constrained Clustering. In Chapter 3 we propose and quantitatively evaluate the Motion Words representation for video. In Chapter 4 we examine existing desirable properties for supervoxels in the context of Motion Words and propose new properties, encoded by a segmentation into Globally Consistent Supervoxels. We present quantitive and qualitative evaluation of the supervoxels as both stand-alone video segmentation and for activity classification using Motion Words. In the final chapter we summarize our contributions and discuss extensions and future work.

Chapter 2

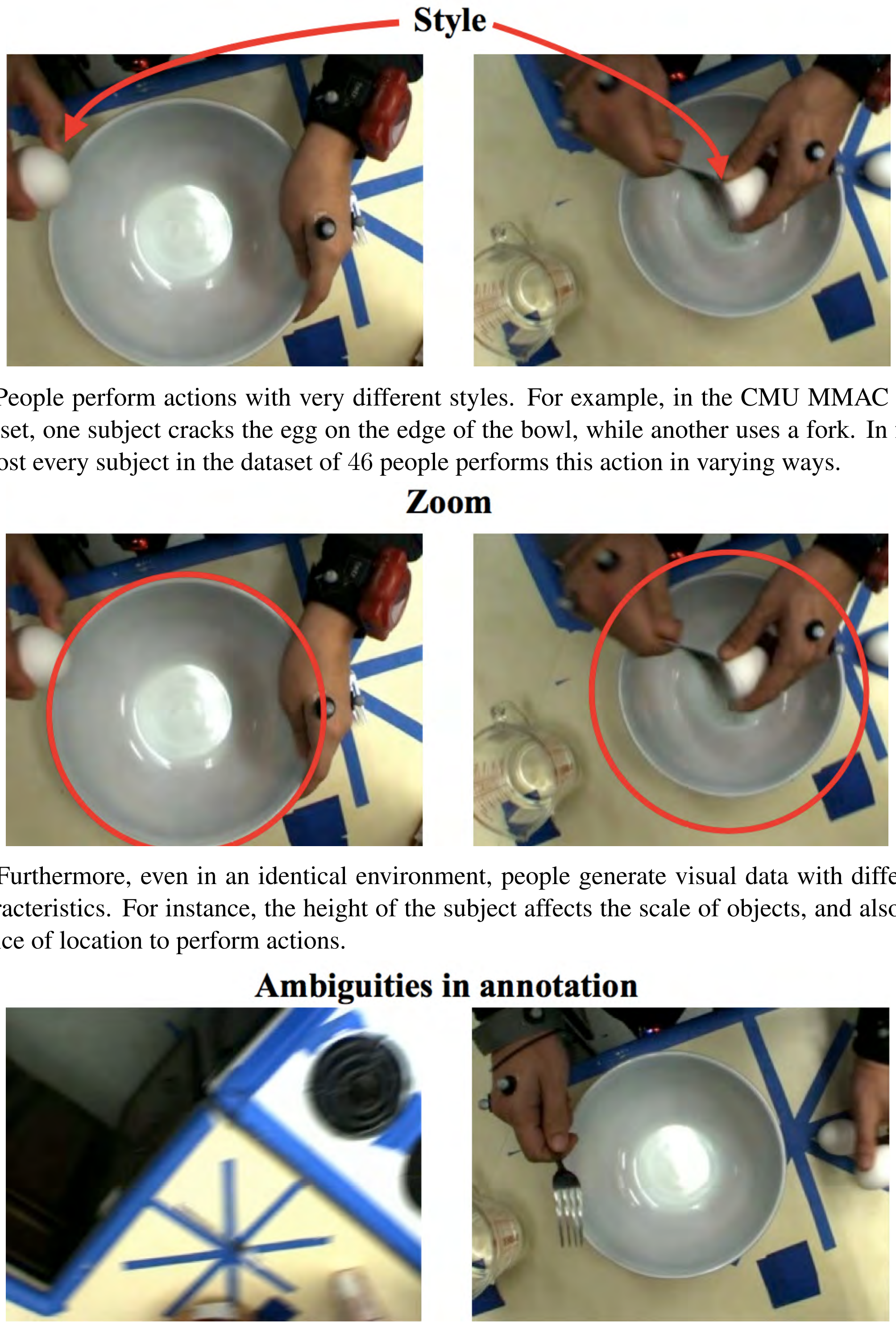
Source Constrained Clustering

2.1 Introduction

We explore the problem of generating a codebook by clustering features from data with large within-class variability. In particular, we study the problem of data quantization via K-means in the context of a BoW model, which has demonstrated state of the art performance in a wide range of computer vision tasks [20, 64, 85, 109]. The BoW framework relies heavily on the assumption that the clustering step produces a grouping of the samples which is meaningful for the classification task. However, when data comes from disparate sources the resulting clusters might not be suitable for distinguishing between the desired semantic classes. Example scenarios which use data from disparate sources include: activity recognition tasks where multiple subjects perform actions with varied styles in unconstrained environments; object recognition tasks in which the images come from environments with dissimilar characteristics [91]; data from web sources with different presentation style and bias, etc.

It is often the case that data collected in such settings exhibits a large within-class variability for a range of semantic categories. For example, in Figure 2.1a we show how people perform the action "crack an egg" with very different styles, how the data generated by different people exhibits different scale and illumination even when recorded in the same environment (Figure 2.1b), and how the annotation task is not trivial in such unconstrained scenarios (Figure 2.1c).

This large within-class variability presents a challenge when learning codebooks via unsupervised clustering algorithms, which tend to group samples from the same source, rather than cluster samples from the same semantic class. For example, in Figure 2.2, features from four video segments are discretized according to a codebook learned via K-means. Due to the differ-



(a) People perform actions with very different styles. For example, in the CMU MMAC [86] dataset, one subject cracks the egg on the edge of the bowl, while another uses a fork. In fact, almost every subject in the dataset of 46 people performs this action in varying ways.

(b) Furthermore, even in an identical environment, people generate visual data with different characteristics. For instance, the height of the subject affects the scale of objects, and also the choice of location to perform actions.

(c) Standardizing manual labeling of such actions is a challenge - when does an action start and end? For example, does "cracking the egg" action begin when the person lifts the egg, or once the shell is broken?

Figure 2.1: Challenges in first-person videos.

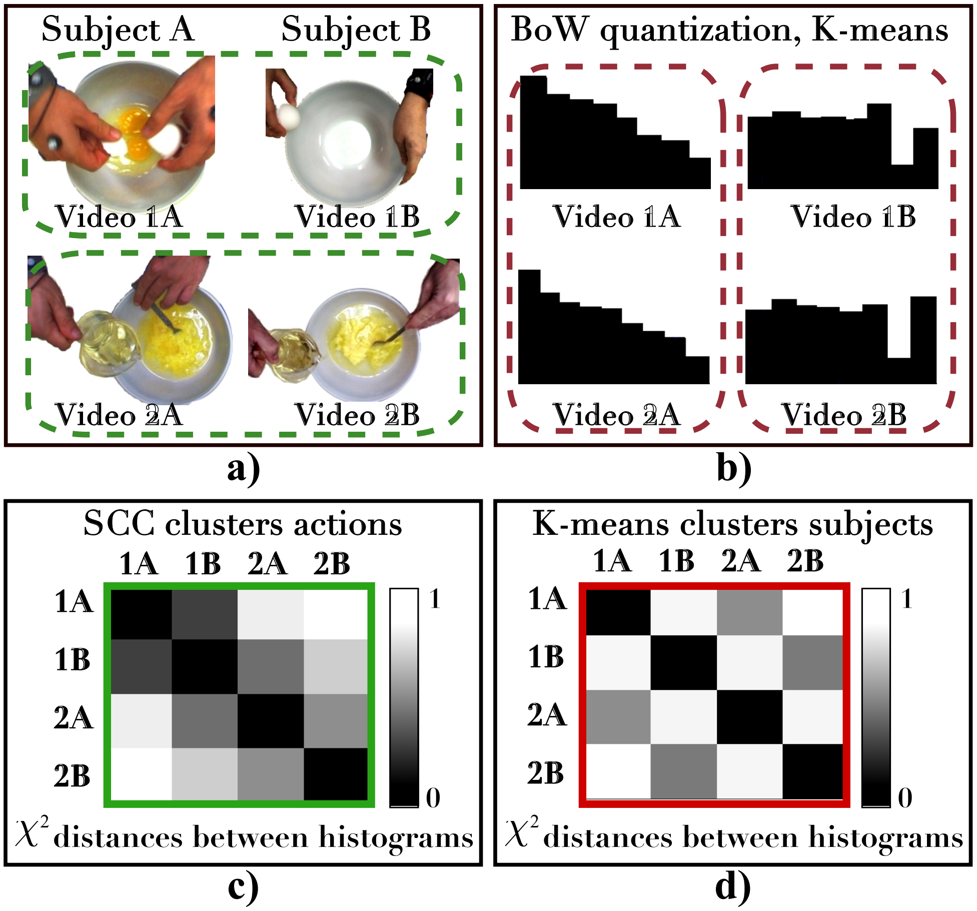


Figure 2.2: We want to cluster actions performed by different subjects **(a)**. In a BoW framework with K-means, a quick look at the discretized videos **(b)** reveals clustering of subjects, which is due to the distinctive styles of execution. SCC uses source information and clusters actions, as seen from the χ^2 distances between the discretized videos **(c)** - 1A is closer to 1B, as desired, while with K-means **(d)** video 1A is closer to 2A (same subject).

ent execution styles, the discretized samples from the same source are closer to each other, than to the sample in the same action class. This makes the codebook less suitable for distinguishing between the two action classes.

We propose a quantization technique which accounts for the problem of source clustering. We evaluate the method in the context of activity recognition of subjects performing actions in an unconstrained environment. In addition to a synthetic data set example, we present results on two realistic data sets. First, we analyze the Hollywood 2 data set [61], comprised of 69 different movies with 12 manually labeled action classes. We consider each movie as a source that generates examples in a particular, often unique, style, which is due to producer and genre bias. For example, "driving" could be a clip of a car moving on a street, or a person behind the wheel. In the second data set, the CMU Multimodal database [22], multiple subjects were asked to prepare different recipes. The data set contains tremendous variability within action classes, and even simple tasks such as cracking an egg or opening a package are performed in a multitude of ways and styles.

To address the large variability across sources, we propose Source Constrained Clustering (SCC), which imposes the constraint that each cluster includes samples from several sources. We ground this idea in the widely used setting of learning a codebook for a bag-of-words (BoW) model. This framework represents an image or video sequence as an orderless collection of local features. The standard algorithm proceeds by clustering all the training features, discretizing the original data using the learned cluster centers, and finally training a model for classification of new examples. The usual K-means clustering step in a BoW framework is replaced by a new optimization step which incorporates these source constraints. The new constraints require source information for each training data sample, which is generally readily available, or otherwise easy to annotate (e.g., subject id or movie id, etc.).

Our hypothesis is that 1) we can incorporate source constraints in a K-means formulation; 2) source information produces better quantization of the data according to semantic classes. We evaluate this hypothesis across three data sets and three types of features and show improvement in classification performance when source information is used to learn a codebook in a BoW setting.

2.2 Related work

One example of prior work on activity recognition where data comes from disparate sources is the Hollywood 2 data set [61]. Video clips in the same action class but extracted from different movies vary greatly in their visual characteristics. Marszalek *et al.* [61] report classification results of average precision 0.326 using a BoW framework and STIP features, clearly showing the difficulty of the problem. The same approach applied to action classification from YouTube videos of sport events shows that BoW approaches on real world data sets need further improvement [67]. Similarly, prior work on clustering features extracted from video sequences from the CMU Multimodal Database [22] shows that several algorithms cluster samples from the same subjects, rather than discriminate samples across action classes. This type of data sets (and similarly [63, 89]) presents a challenge to standard codebook learning algorithms because subjects perform the same actions in very different ways (e.g. cracking an egg using a fork, the rim of a bowl, finger, or on the counter surface). The bias of data from such unconstrained environments complicates the creation of robust codebooks, because the learned clusters represent styles, rather than contain representative samples from multiple sources.

The solution we propose is an extension of the K-means [59] algorithm – one of the most widely used techniques for clustering and learning codebooks, due to its simplicity and good performance. This work is inspired by the constrained K-means clustering method proposed by Bradley *et al.* [7], in which new constraints ensure that each cluster contains a minimum number of data samples. Our algorithm differs from this work by imposing a different set of constraints – each cluster should contain data from multiple sources. Among many other K-means extensions, the work of Wagstaff *et al.* [96] is closely related to ours. The authors extend K-means with must-link and cannot-link constraints specified directly on the features using domain knowledge. This algorithm and its many extensions, like soft-constrained clustering [51], have shown excellent results. However, these methods are not suitable in scenarios with interest samples or aggregate features statistics, where we do not have knowledge of how to constrain individual data samples.

This work builds on existing work on discriminative clustering [4, 23, 27, 106]. Generative methods for clustering such as K-means and spectral clustering do not provide a feature selection or feature weighting mechanism to remove irrelevant features for clustering. In our scenario, we are interested in weighting the features such that each cluster contains multiple disparate sources. Discriminative clustering algorithms combine distance metric learning [82, 103] with clustering algorithms. Typically, discriminative clustering algorithms compute a low dimensional projection that also encourages cluster separability. Unlike previous discriminative clustering algorithms [4, 23, 27, 106] which are unsupervised, we propose to weakly guide clustering and metric learning using source information.

2.3 Background

2.3.1 Regular clustering (RC): K-means

Consider the problem of clustering N data samples into K clusters. Let $\mathbf{D}$ be a $D \times N$ real matrix of samples where D is the data dimension and N is the number of samples. K-means clustering splits a set of N samples into K groups by minimizing the within-cluster variation. That is, K-means finds a grouping of the data that is a local optimum of the following energy function [23, 26, 108]:

$$\text{minimize}_{\mathbf{G},\mathbf{M}} \ \|\mathbf{D} - \mathbf{M}\mathbf{G}^T\|_F^2 \tag{2.1a}$$

$$\text{subject to: } \sum_{c=1}^{K} g_{ic} = 1, \ \forall i \in [1, N] \tag{2.1b}$$

$$\mathbf{G} : \text{Binary} \tag{2.1c}$$

where $\mathbf{G}$ is a $N \times K$ binary indicator matrix with elements g_{ic} specifying if point i belongs to cluster c, and $\mathbf{M}$ is the $D \times K$ matrix of data means (see notation[1]).

The K-means algorithm performs coordinate descent in (2.1a). Given an initial value for $\mathbf{M}$, the algorithm iterates between optimizing for $\mathbf{G}$ and recomputing $\mathbf{M}$, until the change in objective is small, or a maximum number of iterations is reached. The constraint (2.1b) enforces that each data point belongs to only one cluster.

2.3.2 Linear Discriminant Analysis (LDA)

LDA is a supervised algorithm which finds a projection of the data onto a subspace where the distance between clusters is maximized, while the distance within each cluster $c \in [1, K]$, is minimized. The formulation we use is [33]:

$$\text{minimize}_{\mathbf{B}} \ \text{tr}\left(\mathbf{B}^T\mathbf{S}_w\mathbf{B}\right) \tag{2.2a}$$

$$\text{subject to: } \text{tr}\left(\mathbf{B}^T\mathbf{S}_t\mathbf{B}\right) \geq 1 \tag{2.2b}$$

where $\mathbf{B}$ is a $D \times (K-1)$ projection matrix, and the covariance matrices are defined as:

$$\mathbf{S}_w = \frac{1}{N-1}\sum_{c=1}^{k}\sum_{\mathbf{d}_i \in c}(\mathbf{d}_i - \mathbf{m}_c)(\mathbf{d}_i - \mathbf{m}_c)^T \tag{2.3a}$$

$$\mathbf{S}_t = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{d}_i - \mathbf{m})(\mathbf{d}_i - \mathbf{m})^T \tag{2.3b}$$

where the data mean is denoted by $\mathbf{m}$ and $\mathbf{m}_c$ is the mean for cluster c.

[1] Bold capital letters denote a matrix $\mathbf{X}$, bold lower-case letters a column vector $\mathbf{x}$. $\mathbf{x}_i$ represents the i^{th} column of the matrix $\mathbf{X}$ and x_{ij} denotes the scalar in the i^{th} row and j^{th} column. $\mathbf{I}_N$ is the $N \times N$ identity matrix, $\mathbf{1}_N$ is a column vector of ones. $\|\mathbf{X}\|_F^2 = \text{tr}(\mathbf{X}^T\mathbf{X}) = \text{tr}(\mathbf{X}\mathbf{X}^T)$ is the Frobenious norm of $\mathbf{X}$, and $\|\mathbf{X}\|_{\mathbf{B}}^2 = \text{tr}(\mathbf{X}\mathbf{B}\mathbf{B}^T\mathbf{X}^T)$. The Kronecker product $\mathbf{A}_{m\times n} \otimes \mathbf{B}_{p\times q}$ produces an $mp \times nq$ block matrix.

2.4 Regular Clustering (RC) with LDA

Following existing work in discriminative clustering [4, 23, 27, 106], we formulate an energy function for joint clustering and metric learning with dimensionality reduction. A key observation is that we can re-write $\mathbf{S}_w$ in terms of the cluster assignment matrix $\mathbf{G}$:

$$\mathbf{S}_w = \frac{1}{N-1}(\mathbf{D} - \mathbf{M}\mathbf{G}^T)(\mathbf{D} - \mathbf{M}\mathbf{G}^T)^T,$$

and thus the objective in (2.2a) becomes:

$$\begin{aligned} \operatorname{tr}\left(\mathbf{B}^T\mathbf{S}_w\mathbf{B}\right) &\propto \operatorname{tr}\left(\mathbf{B}^T(\mathbf{D} - \mathbf{M}\mathbf{G}^T)(\mathbf{D} - \mathbf{M}\mathbf{G}^T)^T\mathbf{B}\right) && \text{(2.4a)} \\ &= \|\mathbf{D} - \mathbf{M}\mathbf{G}^T\|^2_{\mathbf{B}}, && \text{(2.4b)} \end{aligned}$$

which is the K-means objective from (2.1a), weighted by a projection matrix $\mathbf{B}$. We optimize the following error function, which combines metric learning (dimensionality reduction) and K-means clustering, and which we term RC LDA:

$$\begin{aligned} \text{minimize }_{\mathbf{G},\mathbf{M},\mathbf{B}} \; & \|\mathbf{D} - \mathbf{M}\mathbf{G}^T\|^2_{\mathbf{B}} && \text{(2.5a)} \\ \text{subject to: } & \sum_{c=1}^{K} g_{ic} = 1, \forall i \in [1, N] && \text{(2.5b)} \\ & \operatorname{tr}\left(\mathbf{B}^T\mathbf{S}_t\mathbf{B}\right) \geq 1 && \text{(2.5c)} \\ & \mathbf{G} : \text{Binary} && \text{(2.5d)} \end{aligned}$$

We perform coordinate descent and alternate between clustering with K-means in the projected space and computing LDA. That is, we iterate between: 1) for a fixed $\mathbf{B}$, solve the standard K-means problem specified in (2.1); 2) for fixed $\mathbf{G}$ and $\mathbf{M}$, solve for the projection matrix, $\mathbf{B}$, via the generalized eigenvalue problem $\mathbf{S}_t\mathbf{B} = \mathbf{S}_w\mathbf{B}\Lambda$. We stop when the change in objective is small or until a maximum number of iterations has been reached. Unlike [27], we use the total covariance matrix $\mathbf{S}_t$, which is more suitable for expressing the source constraints in the next section. We use K-means, which differs from the work of [23], which uses a continuous formulation to estimate $\mathbf{G}$.

2.5 SCC formulation

When quantizing data from disparate sources, we want to discover clusters which 1) group similar samples **and** 2) are representative of the sources. For example, in activity recognition, K-means might cluster the particular styles of subjects performing actions. However, we seek a discretization that generalizes the styles, i.e. clusters which are representative of the subjects. While it is impossible to know the optimal number of sources that should be represented in each cluster, SCC approximates this problem by constraining that each cluster contains a minimum number of sources. Formally, we consider the problem of clustering N samples from R sources into K clusters, such that each cluster contains data from at least a fixed number of sources, h. For each data point $i \in [1, N]$, we are given the source id, $s \in [1, R]$, which generated this point. We can add source constraints to K-means by counting the number of sources represented in each cluster.

2.5.1 RC LDA reformulation

We denote by vec$(\mathbf{G})$ the column vector produced by concatenating the columns of the matrix $\mathbf{G}$, so that the first N entries, $g_{11} \dots g_{N1}$, contain a 1 for samples in cluster 1, the next N entries, $g_{12} \dots g_{N2}$, contain a 1 for samples in cluster 2, and so forth:

$$\text{vec}(\mathbf{G}) = \begin{bmatrix} g_{11} \dots g_{N1} & \dots & g_{1K} \dots g_{NK} \end{bmatrix}^T_{NK}. \tag{2.6}$$

The constraints in (2.5b) which ensure that a point belongs to only one cluster can be expressed as:

$$\mathbf{U}\text{vec}(\mathbf{G}) = \mathbf{1}_N, \tag{2.7}$$

where $\mathbf{U}_{N \times NK} = \mathbf{1}_N^T \otimes \mathbf{I}_N$.

The objective function (2.5a) can be expressed using $\text{vec}(\mathbf{G})$ and a column vector $\mathbf{f}_{NK}$ which contains the squared distances from each data point to each cluster center:

$$\mathbf{f}^T\text{vec}(\mathbf{G}) = \begin{bmatrix} \begin{pmatrix} (\mathbf{d}_1 - \mathbf{m}_1)^T\mathbf{B}\mathbf{B}^T(\mathbf{d}_1 - \mathbf{m}_1) \\ \cdots \\ (\mathbf{d}_N - \mathbf{m}_1)^T\mathbf{B}\mathbf{B}^T(\mathbf{d}_N - \mathbf{m}_1) \end{pmatrix} \\ \vdots \\ \begin{pmatrix} (\mathbf{d}_1 - \mathbf{m}_k)^T\mathbf{B}\mathbf{B}^T(\mathbf{d}_1 - \mathbf{m}_k) \\ \cdots \\ (\mathbf{d}_N - \mathbf{m}_k)^T\mathbf{B}\mathbf{B}^T(\mathbf{d}_N - \mathbf{m}_k) \end{pmatrix} \end{bmatrix}_{NK}^T \text{vec}(\mathbf{G}). \tag{2.8}$$

The RC LDA optimization problem (2.5) can be written as:

$$\text{minimize}_{\mathbf{G},\mathbf{M},\mathbf{B}}\ \mathbf{f}^T\text{vec}(\mathbf{G}) \tag{2.9a}$$

$$\text{subject to: } \mathbf{U}\text{vec}(\mathbf{G}) = \mathbf{1}_N \tag{2.9b}$$

$$\text{tr}\left(\mathbf{B}^T\mathbf{S}_t\mathbf{B}\right) \geq 1 \tag{2.9c}$$

$$\mathbf{G} : \text{Binary} \tag{2.9d}$$

2.5.2 Source constraints

Sketch. First, we describe the source constraints for a cluster c, then we construct the matrices used by the final linear integer program (see Table 2.1). We enforce each cluster c to have samples from at least h sources by constructing the sum over all sources in the cluster and thresholding it by h. If x_{cs} is a binary variable equal to 1 when source s has at least one data point in cluster c, we can express this as:

$$\sum_{s=1}^{R} x_{cs} \geq h. \tag{2.10}$$

Let $w_{cs} \geq 0$ be the number of samples source s has in cluster c, as assigned by $\mathbf{G}$. Since $x_{cs} \in \{0, 1\}$, the inequality

$$x_{cs}\text{: Binary and } x_{cs} \leq w_{cs} \tag{2.11}$$

ensures that x_{cs} is set to 0 if source s does not have any points in cluster c.

LIP formulation. We now write the source constraints in matrix form for all clusters. Let $\mathbf{X}$ be the $K \times R$ binary matrix whose entries are x_{cs}. Then (2.10) can be written as:

$$\mathbf{V}\text{vec}(\mathbf{X}) = \left(\mathbf{1}_R^T \otimes \mathbf{I}_K\right)\text{vec}(\mathbf{X}) \geq h\mathbf{1}_K. \tag{2.12}$$

$\mathbf{G}_{N\times K}$	-	cluster assignments for each point (binary)
$\mathbf{U}_{N\times NK}$	-	hard cluster assignment K-means (binary)
$\mathbf{X}_{K\times R}$	-	source assignments per cluster (binary)
$\mathbf{V}_{R\times K}$	-	sum unique sources in a cluster (binary)
$\mathbf{Q}_{N\times R}$	-	source assignments for each point (binary)
$\mathbf{P}_{RK\times NK}$	-	select sources represented in each cluster (binary)
$\mathbf{R}_{RK\times NK}$	-	select total number of samples in a cluster (binary)

Table 2.1: Matrices used in the SCC integer program formulation.

We can write (2.11) for all clusters and all sources as:

$$\mathrm{vec}(\mathbf{X}) \leq \mathbf{P}\mathrm{vec}(\mathbf{G}), \tag{2.13}$$

where the $RK \times NK$ binary matrix $\mathbf{P}$ selects the samples each source has in the cluster specified by $\mathbf{G}$. To construct $\mathbf{P}$, we represent the source information in a $N \times R$ binary matrix $\mathbf{Q}$, with elements $q_{is} = 1$ if point i comes from source s, and 0 otherwise. To get the number of samples source 1 has in each cluster, take $\mathbf{q}_1$, the first column of $\mathbf{Q}$, and duplicate it K times:

$$\mathbf{w}_s = \widetilde{\mathbf{Q}}_1\mathrm{vec}(\mathbf{G}) = (\mathbf{I}_K \otimes \mathbf{q}_1^T)\mathrm{vec}(\mathbf{G}), \tag{2.14}$$

so that the vector $\mathbf{w}_s$ of length K contains the number of samples s has in each cluster, and $\widetilde{\mathbf{Q}}_1$ is a $K \times NK$ binary matrix. Repeating for each source, the per cluster source information matrix $\mathbf{P}$ is given by:

$$\mathbf{P} = \begin{bmatrix} \widetilde{\mathbf{Q}}_1 \\ \vdots \\ \widetilde{\mathbf{Q}}_R \end{bmatrix}_{RK\times NK} . \tag{2.15}$$

The optimization problem (2.9) with source constraints is:

$$\text{minimize}_{\,\mathbf{G},\mathbf{M},\mathbf{B},\mathbf{X}}\ \mathbf{f}^T\text{vec}(\mathbf{G}) \tag{2.16a}$$

$$\text{subject to: }\ \mathbf{U}\text{vec}(\mathbf{G}) = \mathbf{1}_N \tag{2.16b}$$

$$\text{vec}(\mathbf{X}) - \mathbf{P}\text{vec}(\mathbf{G}) \leq 0 \tag{2.16c}$$

$$\mathbf{V}\text{vec}(\mathbf{X}) \geq h\mathbf{1}_K \tag{2.16d}$$

$$\text{tr}\left(\mathbf{B}^T\mathbf{S}_t\mathbf{B}\right) \geq 1 \tag{2.16e}$$

$$\mathbf{G}, \mathbf{X}:\ \text{Binary} \tag{2.16f}$$

2.5.3 Regularization

Constraints (2.16c) and (2.16d) ensure that each cluster contains samples from at least h sources. However, this does not guarantee that the sources contribute a meaningful number of samples in participating clusters. Indeed, an undesirable solution would be to assign to cluster c some samples from source $s = 1$, and only one data sample from each of the remaining $s = 2 \dots h$ sources. To account for this, we add a regularizing constraint to ensure each source contributes a non-trivial amount in every participating cluster.

Sketch. Let t_c be the total number of samples in cluster c and assume every cluster has at least h sources. We approximate the fraction of samples sources should have in each cluster by a fraction of $\frac{h+R}{2}$ samples of t_c. We constrain w_{cs}, the number of samples s has in c, to be within $\tilde{\theta}n_s$ samples of this quantity:

$$w_{sc} \geq \gamma t_c - \tilde{\theta}n_s, \tag{2.17}$$

where $\gamma = \frac{2}{h+R}$, and n_s is the total number of training samples generated by source s. The slack of $\tilde{\theta}n_s$ samples account for the case where the number of training samples per source can differ greatly, and we cannot expect sources to contribute the same number of samples. First, we set $\tilde{\theta} = 0$, which approximates a uniform source distribution. If the problem is infeasible, we relax it and try a range of thresholds that take into account the number of samples per source. We vary $\tilde{\theta}$ from 0.05 to 0.4 in increments of 0.05, and define $\theta_s = \tilde{\theta}n_s$. Furthermore, (2.17) should hold only if the solution assigns non-zero number of samples from s to c, i.e. when $x_{cs} = 1$, otherwise the constraint should be inactive:

$$\gamma t_c - w_{sc} \leq \theta_s x_{cs} + N(1 - x_{cs}), \tag{2.18}$$

where the right-hand side is θ_s if s has at least one point in c ($x_{cs} = 1$), or it evaluates to N otherwise. In the latter case, the constraint is trivially satisfied, since $x_{cs} = 0$ implies $w_{sc} = 0$, and it is always the case that $\gamma t_c \leq N$.

LIP formulation. The total number of samples in $c = 1$ are obtained by summing up the first N entries of $\text{vec}(\mathbf{G})$:

$$t_1 = [\mathbf{1}_N^T \ \mathbf{0}_N^T \ldots \mathbf{0}_N^T]\text{vec}(\mathbf{G}). \tag{2.19}$$

For all clusters we can write:

$$\widetilde{\mathbf{R}}_{K \times NK} = (\mathbf{I}_K \otimes \mathbf{1}_N^T), \tag{2.20}$$

and for all sources we construct the binary selector matrix:

$$\mathbf{R}_{RK \times NK} = \mathbf{1}_R \otimes \widetilde{\mathbf{R}}. \tag{2.21}$$

We re-write (2.18) as:

$$\gamma t_c - w_{sc} + (N - \theta_s)x_{cs} \leq N. \tag{2.22}$$

Using $\mathbf{P}$ and $\mathbf{X}$ defined in the previous section, constraint (2.22) for all clusters and all sources becomes:

$$\gamma \mathbf{R}\text{vec}(\mathbf{G}) - \mathbf{P}\text{vec}(\mathbf{G}) + (N\mathbf{1}_{RK}^T - \theta^T)\text{vec}(\mathbf{X}) \leq N\mathbf{1}_{RK}, \tag{2.23}$$

where $\theta = \mathbf{1}_K \otimes [\theta_1 \ldots \theta_R]^T$, a vector of length RK.

2.5.4 SCC algorithm

The final SCC optimization problem is:

$$\text{minimize}_{\mathbf{G},\mathbf{M},\mathbf{X},\mathbf{B}} \ \mathbf{f}^T \text{vec}(\mathbf{G}) \tag{2.24a}$$

$$\text{subject to:} \quad \mathbf{U}\text{vec}(\mathbf{G}) = \mathbf{1}_N \tag{2.24b}$$

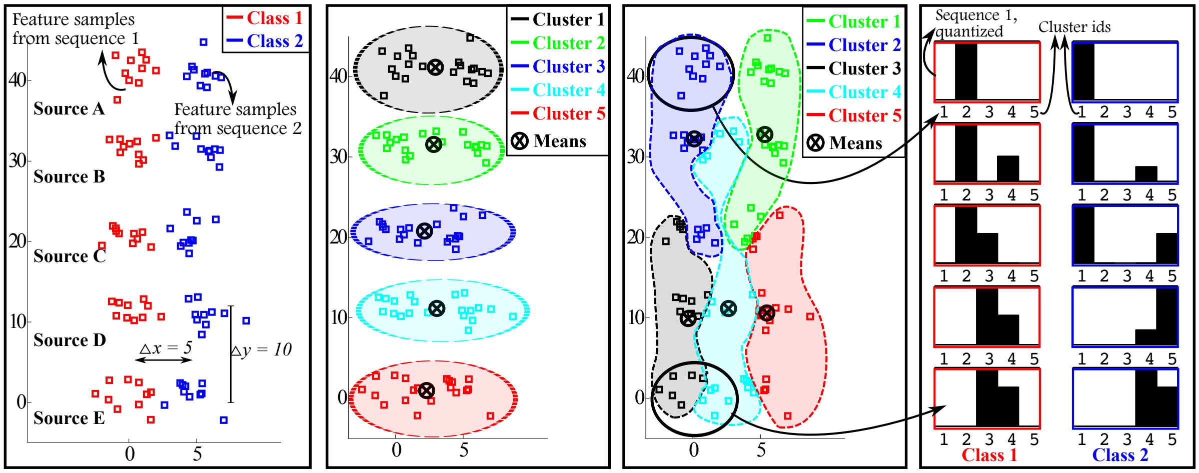
$$\text{vec}(\mathbf{X}) - \mathbf{P}\text{vec}(\mathbf{G}) \leq 0 \tag{2.24c}$$

$$\mathbf{V}\text{vec}(\mathbf{X}) \geq h\mathbf{1}_K \tag{2.24d}$$

$$(\gamma\mathbf{R} - \mathbf{P})\text{vec}(\mathbf{G}) + (N\mathbf{1}_{RK}^T - \theta^T)\text{vec}(\mathbf{X}) \leq N\mathbf{1}_{RK} \tag{2.24e}$$

$$\text{tr}\left(\mathbf{B}^T\mathbf{S}_t\mathbf{B}\right) \geq 1 \tag{2.24f}$$

$$\mathbf{G}, \mathbf{X}: \ \text{Binary} \tag{2.24g}$$



a) Input b) K-means clusters c) SCC clusters d) SCC quantization

Figure 2.3: Synthetic data where samples generated by the same source are closer than the samples in the same class (see Section 2.6). K-means groups sources **(b)**, resulting in a non-discriminative feature quantization. Our algorithm, SCC, with $h = 3$, produces a more meaningful clustering, as shown in **(c)** with rough cluster outlines. The quantized data is clearly separated into the two classes **(d)**.

We approximately solve (2.24) as in [23, 27] by initializing $\mathbf{M}$ to K samples at random, setting $\mathbf{B} = \mathbf{I}$, and iterating between: 1) solving for $\mathbf{G}$ and $\mathbf{M}$ in the standard K-means setting, with $\mathbf{B}$ fixed; and 2) solve for $\mathbf{B}$, using $\mathbf{G}$ and $\mathbf{M}$ found in the previous step (see *Algorithm* 1). To solve the LIP problem, we use the ILOG CPLEX [39] software.

Algorithm 1

```
function $\mathrm{SCC}((\mathbf{D}, \mathbf{M}, h))$
    for t = 0... do // iterative clustering and metric learning
        for s = 0... do // iterative SCC
            $\mathrm{f}_s \leftarrow$ ComputeObjective$(\mathbf{D}_t, \mathbf{M}_s)$
            $(\mathbf{G}^*_{s+1}, \mathbf{X}^*_{s+1}) =$ SolveLIP$(\mathbf{f}_s, h)$ // Eq. (2.24)
            $\mathbf{M}_{s+1} \leftarrow$ ComputeMeans$(\mathbf{D}_t, \mathbf{G}^*_{s+1})$
        end for
        $\mathbf{M}_{t+1} \leftarrow \mathbf{M}_s, \;\; \mathbf{G}_{t+1} \leftarrow \mathbf{G}^*_s$
        $\mathbf{B}_t \leftarrow$ LDA$(\mathbf{G}_{t+1}, \mathbf{M}_{t+1})$ // metric learning
        $\mathbf{D}_{t+1} \leftarrow \mathbf{B}_t^T \mathbf{D}_t; \;\; \mathbf{B}_{\text{total}} \leftarrow \mathbf{B}_t^T \mathbf{B}_{\text{total}}$
    end for
    return $\mathbf{G}_t, \mathbf{B}_{\text{total}}$
end function
```

2.6 Qualitative analysis

We show that using source information when data comes from disparate sources improves classification in a BoW codebook task. Furthermore, we confirm that the improvement in performance is correlated with the source variance. To show that the approach can be applied to a broad class of vision tasks, we perform experiments on three data sets and three types of features in BoW classification tasks. First, we illustrate SCC on a simple synthetic data set, then we compare to previously reported results on the Hollywood 2 action data set [61], and finally we report results on action classification on the CMU-MMAC data set [22].

Given a data set of action sequences with class labels we follow the standard procedure for codebook creation in BoW frameworks. This includes extracting features (in our case, STIP [50] and GIST [92]); generating disjoint training, validation and testing sets; clustering training features to learn a codebook; discretizing the features using the learned cluster centers; and training a discriminative classifier. We train a one-versus-all χ^2 SVM classifier [77] and report average precision (AP) as in [61].

We evaluate three codebook creation schemes: regular k-means clustering (RC), k-means with LDA (RC LDA), and SCC Eq. (2.24) in two unsupervised tasks and two supervised tasks. The former setting is the standard codebook creation scheme which clusters all the features in an unsupervised way. In the latter setting we use the training labels to cluster samples within each class, creating one codebook per class. This is useful in practice when large amounts of data limit the size of the optimization problem that can be solved efficiently.

We experiment with $K = 30, 40, 60, 100$ clusters. First, we validate the method works even for a fixed value of the parameter $h = 5$ for all experiments - a reasonable choice, since $h = 0$ is RC, while $h = R$ would imply all sources should participate in all clusters, which is unrealistic in these data sets. Second, we confirm SCC is not sensitive to the choice of h by setting $h = 2, 3, 5, 7, 10$ and noting a maximum change in overall AP of around 2%. For comparison, we also report results with varying values of h per class, based on manual inspection of the per-class performance. A complete validation system for discovering the optimal h for each class would require a large amount of validation data, currently unavailable for these data sets.

Synthetic experiment

We build a synthetic data set of 5 sources generating a sequence in each of two classes, 1 and 2. The sequences are bags of ten 2D samples. The first component of the features, x, is correlated with the class id, while the second, y, is correlated with the source id. The samples are drawn from Gaussian distributions with parameters chosen to simulate disparate sources. For every source y_i, the distance between the samples in the bag from class 1 and the bag from class 2 is smaller than the distance to the correct class bag from other sources ($\Delta x < \Delta y$), as shown in Figure 2.3.

Figure 2.3 shows one representative clustering for $K = 5$. RC clusters the samples from the same source – the discretized sequences from class 1 are nearly identical to the discretized sequences from class 2, making it impossible to train a classifier to distinguish the two classes. On the other hand, SCC produces a quantization of the sequences which clearly discriminates the two classes. For ease of visualization, Figure 2.3 (d) shows the rough boundaries of the SCC clusters in the original feature space (not LDA).

Hollywood movie data set

We use the clean dataset provided by the authors, which contains 1707 action samples divided into a training set (823 sequences) and a disjoint test set (884 sequences). Following [61], we subsample the STIPs at random, retaining 10% of the features for training. To further reduce the computational complexity, we learn a codebook per training action class. As shown in Table 2.2, we verify that performance of per class codebooks is comparable to published results [61], and thus the method is suitable for comparison. Table 2.2 shows the results with $K = 100$ for the three clustering algorithms along with the previously published results of Marszalek *et al.* [61].

We see improvement in performance for 10 out of 12 actions compared to both RC and RC LDA. The classes for which SCC performs better than RC have a larger number of training

	Marszalek	RC	RC LDA	SCC	SCC	
	et al.			$h = 5$	varied h	
AnswerPhone	*0.088*	0.103	0.098	**0.112**	**0.116**	4
DriveCar	*0.749*	0.797	0.794	**0.824**	**0.841**	7
Eat	*0.263*	0.381	**0.465**	0.382	**0.493**	3
FightPerson	*0.675*	0.564	0.584	**0.620**	**0.630**	7
GetOutCar	*0.090*	**0.195**	0.162	0.174	0.174	5
HandShake	*0.116*	0.100	0.112	**0.166**	**0.172**	7
HugPerson	*0.135*	0.188	0.154	**0.195**	**0.212**	3
Kiss	*0.496*	**0.442**	0.433	0.420	0.437	2
Run	*0.537*	0.422	**0.494**	0.489	0.476	2
SitDown	*0.316*	0.331	0.351	**0.372**	**0.399**	2
SitUp	*0.072*	0.099	0.093	**0.131**	**0.131**	5
StandUp	*0.350*	0.342	0.360	**0.449**	**0.449**	4
Mean AP	*0.324*	0.330	0.342	**0.361**	**0.378**	

Table 2.2: Comparison of clustering methods for learning per class codebooks from HoG+HoF features on the Hollywood 2 [61] data set. SCC improves mean AP by 3.1%compared to RC, and 1.9% compared to RC LDA for a fixed $h = 5$. Class specific value of h gives a further 1.7% increase. As a reference to learning a global codebook, the results from [61] are shown in the left column.

sources and exhibit stronger source clustering. In these scenarios, using source information helps build more robust codebooks.

CMU kitchen data set

The Carnegie Mellon University Multimodal Activity database (CMU-MMAC) [22] contains multimodal measurements of subjects performing different recipes with no prior instructions. The actions vary greatly in time span, repetitiveness, and manner of execution. From the 35 manually annotated action classes from [86] we merge semantically similar ones (e.g. “take from cupboard left” and “take from cupboard right” are combined, etc.), and we ignore actions which have a very small number of instances. In total we have 15 classes listed in Table 2.3. We use the videos from the wearable camera and extract STIP and GIST features from every fifth frame. 14 subjects are used for training and two for testing. We average results over 4 disjoint sets of withheld subjects, chosen at random.

Per class codebook using STIP features

Following the procedure of [61], we subsample the STIPs at random, retaining 20% of the training data for learning a codebook. Again, to allow for more training samples to be used, we cluster samples in each class, learning one codebook and distance metric transformation per class. In Table 2.3 we report results for $h = 5$ and $K = 40$.

	STIP			GIST		
	RC	RC LDA	SCC	RC	RC LDA	SCC
crack-egg	0.775	**0.787**	0.733	0.119	0.125	**0.289**
open-bag	0.683	0.640	**0.707**	**0.296**	0.191	0.240
fridge-door	0.891	0.906	**0.922**	**0.669**	0.515	0.594
pour-oil	**0.567**	0.198	0.563	0.030	**0.050**	0.031
pour-bowl	0.660	0.685	**0.724**	0.302	**0.530**	0.528
put-obj-lower	0.397	**0.402**	0.386	0.641	**0.727**	0.717
spray-pam	**0.889**	0.595	0.778	0.534	0.493	**0.725**
stir-egg	**1.000**	**1.000**	**1.000**	0.071	0.088	**0.207**
read-switch	**0.863**	0.807	0.844	0.416	0.515	**0.595**
take-fridge	0.633	**0.676**	0.611	0.309	0.283	**0.490**
take-drawer	0.569	0.503	**0.710**	0.705	0.485	**0.736**
take-top	**0.891**	0.733	0.872	0.817	**0.846**	0.772
take-bottom	0.394	0.567	**0.654**	**0.561**	0.502	0.521
twist-cap	0.547	**0.560**	0.533	0.091	**0.295**	0.206
walk	0.966	0.951	**0.969**	**0.563**	0.340	0.524
Mean AP	0.715	0.667	**0.734**	0.408	0.398	**0.478**

Table 2.3: Classification performance on the CMU-MMAC [22] data set using STIP and GIST features. Using the source information increases the mean AP by 2.5% for STIPs, and by 7% for GIST, compared to RC.

A quick look at the videos shows large variability in styles for the classes which show improvement, especially for the “taking from drawer” and “taking from bottom cupboard.” On the other hand, for classes with low or no improvement, we observe less style variability. For example, the action “walk” from the counter to the fridge was performed without much variance, and likewise for “stirring egg,” and regular clustering has no problem in classifying such actions nearly perfectly. The 2.5% average increase in AP for this task also shows the benefits of using source information.

Global codebook using GIST features

Prior work on the CMU-MMAC data set reports source clustering when using GIST features [86]. These features encode the style of execution more strongly and we see a more pronounced source clustering problem compared to using STIPs (several of the RC clusters contain samples only from one subject). Our hypothesis is that using source information in this case will have an even stronger impact on performance. Indeed, the results in Table 2.3 show a 7% improvement in AP when learning a global codebook in an unsupervised manner using features from 11 training subjects, and testing on 5 subjects., with $K = 40$ and $h = 5$. There is a clear improvement in performance for 9 out of 15 classes. This additional experiment verifies that the concept of SCC is beneficial across different types of features.

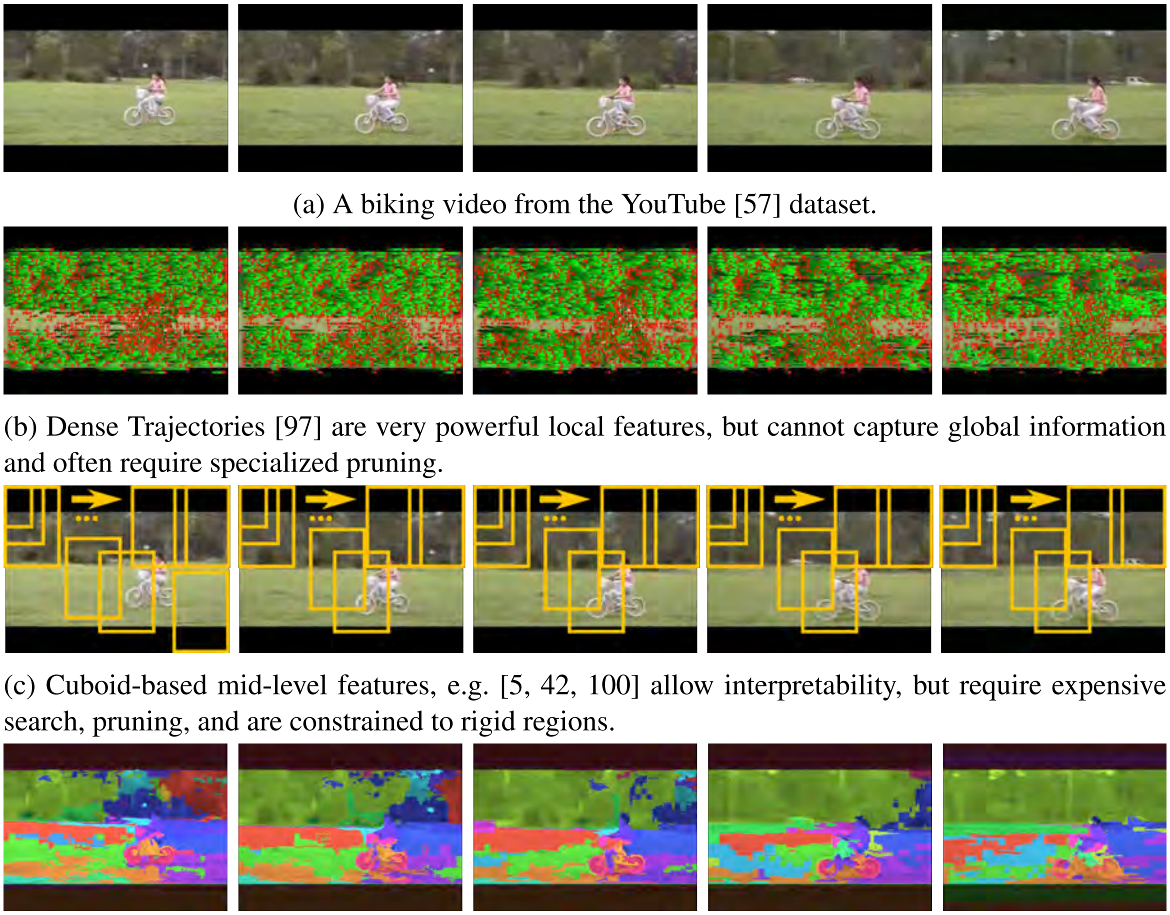
2.7 Conclusion

In this section we presented SCC – a novel improvement to k-means which addresses the problem of feature quantization in data sets generated by diverse sources. Our experiments show improvement in classification performance across several tasks and features compared to standard k-means. Furthermore, we show that improvement in performance is correlated with the variability of the sources.

However, the low average precision performance on both datasets, along with preliminary results with a random k-means codebook (which incurs a drop of only 1-2%), indicate the need to 1) have a way to interpret why the classifier is assigning the class labels, and 2) explore new video representations which are robust to the challenges of real-world videos. In the following chapters, we propose a video representation that is both robust for the tasks of action classification and retrieval, and enables interpretability of the features common between videos.

Chapter 3

Motion Words for Video



(a) A biking video from the YouTube [57] dataset.

(b) Dense Trajectories [97] are very powerful local features, but cannot capture global information and often require specialized pruning.

(c) Cuboid-based mid-level features, e.g. [5, 42, 100] allow interpretability, but require expensive search, pruning, and are constrained to rigid regions.

(d) Motion Words provide flexible spatio-temporal support, and enable interpretability.

Figure 3.1: Examples of video representations.

3.1 Introduction

In the task of activity recognition in videos, computing the video representation often involves pooling feature vectors over spatially local neighborhoods. The pooling is done over the entire video, over coarse spatio-temporal pyramids, or over pre-determined rigid cuboids (Figure 1.1). However, pooling is not necessarily local in the feature vector space and widely dissimilar features may be pooled together if they are in nearby locations [6]. Similarly to pooling image features over superpixels in the image analysis community [36, 90], it is natural to consider pooling spatio-temporal features over a video segmentation. However, since videos have different number of segments, this produces a video representation of variable size. We propose Motion Words - a new, fixed size video representation, where we pool features over supervoxels (Figure 3.2).

We rigorously analyze each component of the proposed method - the types of low-level features, the types of supervoxels and the quantization parameters. To segment the video into supervoxels, we explore two recent video segmentation algorithms. First, we evaluate a streaming graph-based method (GBH) [105] which generates a hierarchical segmentation. Second, we explore a method that incorporates user-defined objectives, Uniform Entropy Slice (UES) [104]. In particular, we explore supervoxels that preserve motion boundaries by optimizing for "motionness" across hierarchy levels. Both methods perform joint appearance and motion optimization on a per-pixel basis. In Chapter 4 we propose a heuristic way of generating supervoxels, Globally Consistent Supervoxels, that operates on a superpixel level.

Evaluation on classification and retrieval tasks on the YouTube and the HMDB datasets shows that Motion Words achieves state-of-the-art performance. We show that pooling state-of-the-art Dense Trajectory features [97] over non-coarse supervoxels (i.e., fine, or a combination of fine and coarse), performs better than pooling features that do not capture temporal information, or pooling over very large supervoxels. Furthermore, we obtain a large improvement in retrieval performance, where we compare videos directly, without the use of a classifier. This is important as it shows the promise of the proposed representation independently of the type of classifier or the parameters used to tune it.

Moreover, the representation enables localization of common regions across videos in both space and time. A key aspect of the representation is that, since the video segments are meaningful regions, we can interpret the proposed features and obtain a better understanding of *why* two videos are similar.

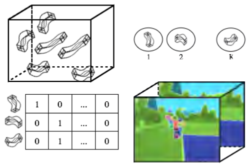
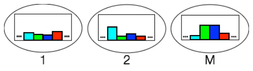
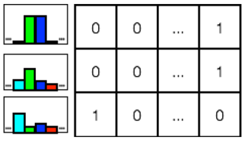
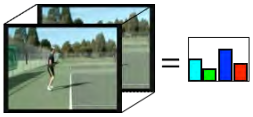
1. Feature extraction	2. Quantization	3. Encoding	4. Pooling
			
1.1. BoW (Figure 1.1) 1.2. Supervoxels	2.1 K-means	3.1. Hard 3.2. Soft 3.3. LLC	4.1. Global

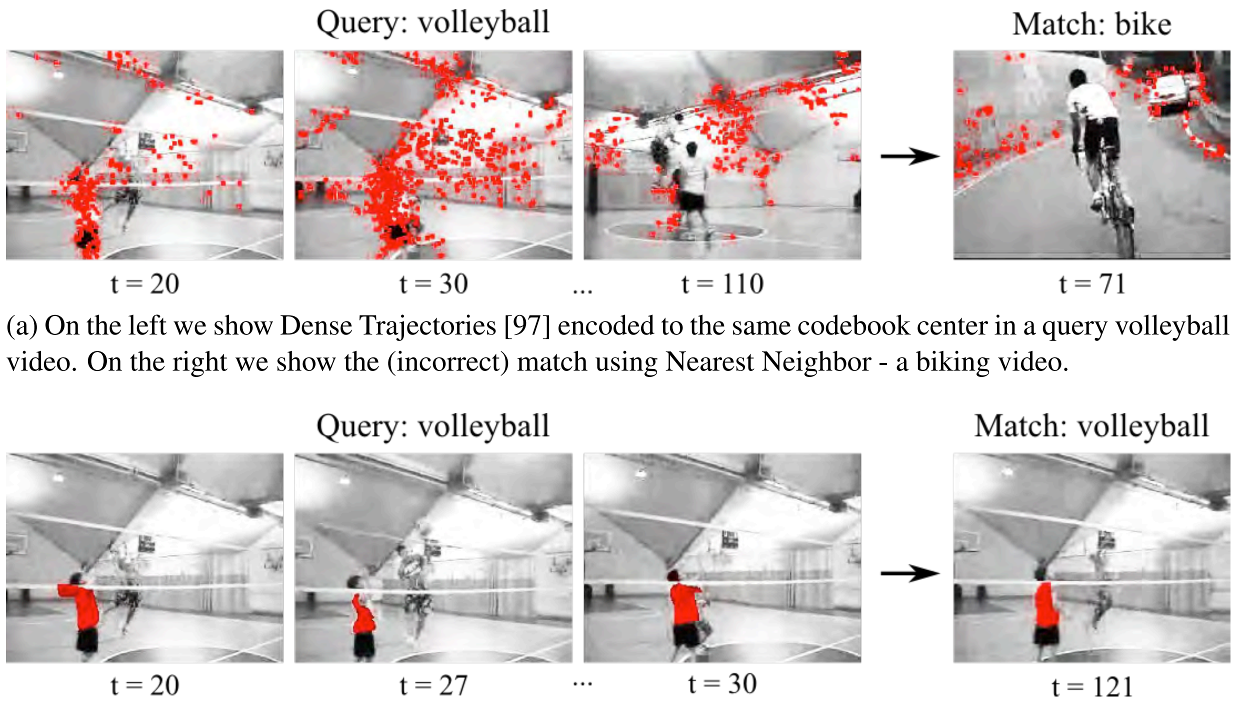
Figure 3.2: In the first step of the proposed Bag of Motion Words (BMW) framework, BoW is applied to low-level features (e.g., Dense Trajectories [97]), which are then pooled over supervoxels. The resulting motion features are histograms of size k (number of clusters used in BoW; see Figure 3.8). The subsequent steps (quantization, encoding and pooling) are identical to a standard BoW framework applied to these motion features.

3.2 Motivation

Features for video classification and retrieval include low-level interest point features [10, 28, 97], mid-level patch-based features [5, 42, 84, 100, 110], and higher level, semantic features [56]. Even though low-level features are limited either in temporal scale [28, 97], or in density [10], they robustly capture local information, and in fact obtain state-of-the-art classification performance on several datasets [47, 57]. However, if we want to know why two videos are classified as similar, visualizing the low-level features does not allow us to interpret the results. On the other hand, mid-level video patches also perform well for activity classification [13, 42, 100], and we can visualize the cuboids learned as important for classification. These representations have been limited to cuboids of predetermined spatial and temporal sizes [42], or rectangles from object/foreground detectors [112]. High-level features provide semantic interpretation at the expense of additional annotations or training [12, 32, 44, 56, 75, 87], or rely on short tracklets [35].

Ultimately, we seek a video representation that captures both low-level and region-based statistics. Furthermore, it is important that the representation enables interpretability. That is, when visualized, we want features that give us the power to understand which regions make two videos similar. We propose a video representation where we pool low-level features over regions defined by a video segmentation.

It is a natural idea to pool features over coherent spatio-temporal regions, e.g. supervoxels. In fact, work in image analysis shows that descriptors computed over segmentation regions (e.g. superpixels) provide more robust image representations [6]. Nevertheless, in video analysis, pooling is currently done either over the entire video [28, 97], over coarse spatio-temporal



(a) On the left we show Dense Trajectories [97] encoded to the same codebook center in a query volleyball video. On the right we show the (incorrect) match using Nearest Neighbor - a biking video.

(b) In the proposed representation, Motion Words, features pooled over supervoxels match similar regions and enable interpretability.

Figure 3.3: A volleyball query video and the retrieved nearest neighbor match using the YouTube dataset. In (a), Dense Trajectories encoded to the same codebook center originate from regions with very different motion and appearance (the players, the volleyball net, the ceiling and walls) and it is not clear why the biking video is more similar to the query, provided that the dataset contains a very similar volleyball video. In we show that pooling features over supervoxels eliminates spurious matches because we now compare groups of features. Furthermore, when using this proposed representation, we successfully retrieve the expected volleyball video.

pyramids [97], or over pre-determined rigid cuboids [29, 52, 66]. Even though cuboids define a local neighborhood, this pooling is not necessarily local in the feature vector space and widely dissimilar features may be pooled together if they are in nearby locations [6].

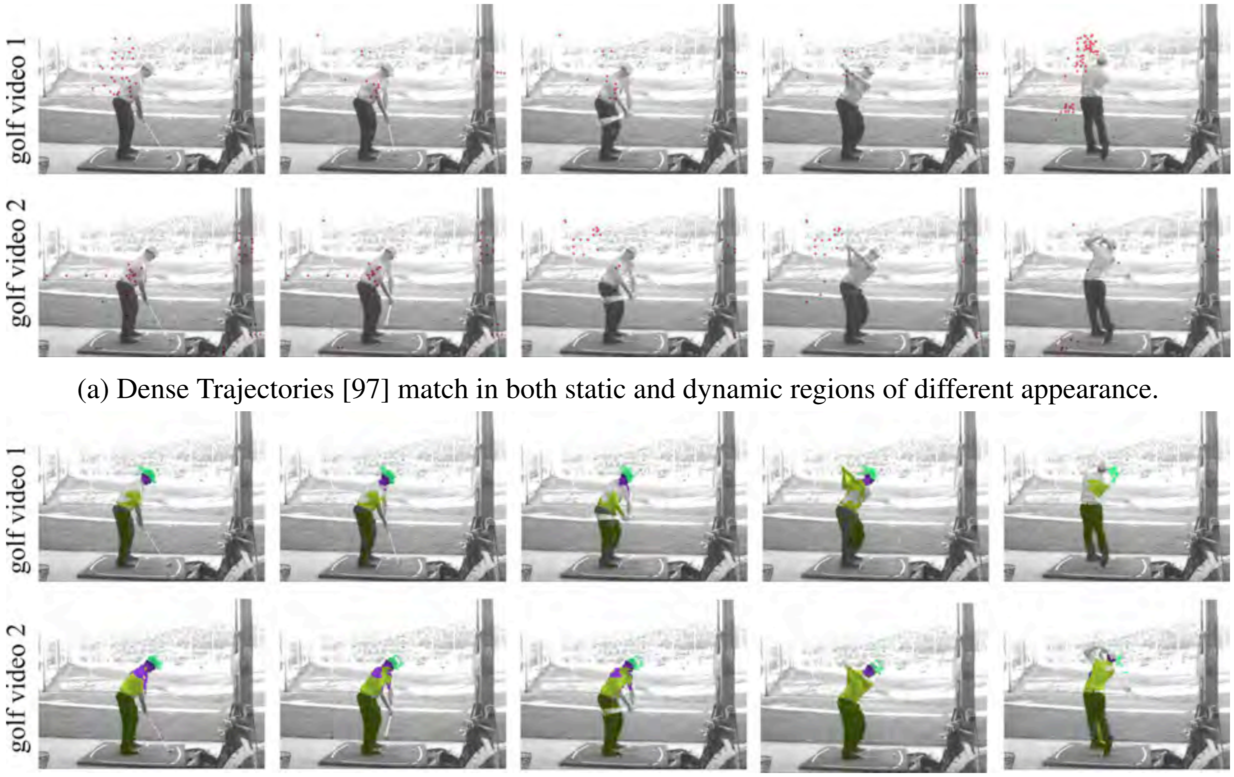
We visualize such a scenario in Figure 3.3a, where volleyball players and their interactions with the ball occur at various spatio-temporal locations in a video from the Youtube [57] action dataset. We take this video as a query and ask for the nearest neighbor from the dataset using the standard Bag of Words framework. The dataset contains a very similar video of the same players, in the same environment, performing a different trial. We are thus puzzled when the system retrieves an incorrect result, an outdoors biking video. The two videos have largely different motions, with only a few similarities. In particular, the characteristic motion spanning a large number of frames in the biking video is the camera following the biker moving to the right.

The spiking video has a small number of frames in which the camera moves to the right. One of the characteristic motions in the volleyball video, spanning several frames, is when one of the players, the setter, moves to the left after setting the ball for the spiker. We wish to visualize all the features that are similar to those capturing the setter's motion in all video frames from both videos.

In this example, we chose Dense Trajectories [97] as the low-level descriptors, since they have been successfully used for activity classification. The Dense Trajectories from the spatio-temporal region corresponding to the setter moving to the left encode the player's motion very well for several frames (Figure 3.3a). We pick the BoW codebook centers that these descriptors encode to. We can visualize all the descriptors encoded to the same center in both the query and the match videos. We find that descriptors that encode to the same codebook center originate from regions corresponding to the person moving to the left, the ceiling, the volleyball net, and the walls (Figure 3.3a). Furthermore, many Dense Trajectories from the biking video, originating from the sidewalk and cars, which have distinctively different motion and appearance, also encode to the same codebook centers as the trajectories in the volleyball video. While we can peek into BoW in this manner, this visualization does not provide any intuition as to what makes the videos similar.

On the other hand, in the proposed representation, Motion Words, features pooled over supervoxels match to features in supervoxels with a similar distribution of low-level descriptors. In Figure 3.3b we show in red all the supervoxels which encode to the same codebook center as the region corresponding to the setter moving to the left. When using Motion Words, the retrieved video is indeed the expected volleyball video. Furthermore, the supervoxels corresponding to the setter moving to the left in both videos are quantized to the same codebook center. What is more, since Motion Words are build on top of meaningful regions, we can more easily examine such matches and interpret the results.

While low-level features are well suited for local comparison operations, when compared (pooled) globally, descriptors from heterogenous regions are matched together and it is difficult to interpret the resulting matches. There has been extensive work that explores mid-level representations for more robust action classification and that allows visualization and interpretation. Representations that allow interpretation include mid-level features, e.g. [5, 42, 100, 110], which are computed over cuboids. We hypothesize that video segments provide more flexible spatio-temporal support over cuboids of manually chosen spatial and temporal sizes. For example, in the scenario discussed above, when we pool features over supervoxels, we obtain a much better match, including matches at the region level. However, when pooling over supervoxels, each



(a) Dense Trajectories [97] match in both static and dynamic regions of different appearance.

(b) Motion Words constrain the low-level features to regions of similar motion and appearance.

Figure 3.4: Two golf videos taken in the same environment from a static camera, showing the same person performing two golf swing trials.

video can have a different number of regions, resulting in video representations of variable sizes. We propose a simple way of constructing a fixed-size representation by using the popular Bag of Words (BoW) framework. Rather than constrain all videos to have the same number of regions, or develop specialized distance metrics, we treat each video segment as a feature vector and cluster the segments from training videos to learn Motion Words. Each video is then represented as a Bag of Motion Words (BMW), as shown in Figure 3.8. Furthermore, the proposed Motion Words representation enables localization and interpretation. Since segmentation algorithms produce meaningful spatio-temporal regions, we can visualize and interpret the "words" that are common to both videos (Figure 3.3b).

The above volleyball example includes camera motion and actions performed over a very short period of time. Perhaps the Dense Trajectories do in fact describe similar regions of motion and appearance, but we are unable to tell due to the dynamic environment. To test this hypothesis, we explore a second, more constrained example. Consider two nearly identical golf videos, Figure 3.4, where the camera is stationary and the environment is the same - the same person

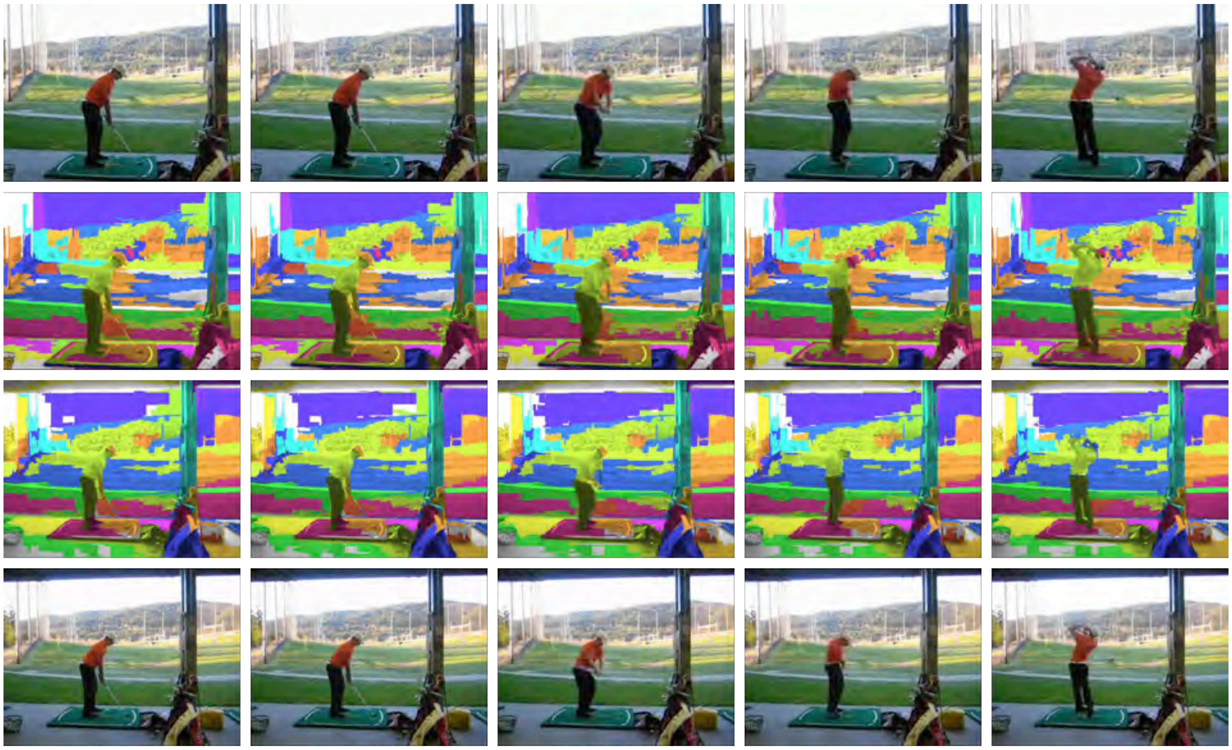


Figure 3.5: Two golf swing videos, performed by the same person. The camera in the second video is positioned further away and thus the scale is different. We take the Motion Words that are common between both videos and color the supervoxels based on their codebook center. The regions corresponding to both the ground, the sky and the person are well matched between the two videos.

performing two different trials of a golf swing. We wish to visualize all the Dense Trajectories which are similar to the ones on the person's torso, since that region exhibits the characteristic golf swinging motion. To do so, we pick the HOG and MBHX channels of the Dense Trajectory descriptors, since they capture appearance and horizontal motion information. We examine the descriptors from the torso region from the time the person initiates the swing. These descriptors quantize to two different codebook centers. In Figure 3.4a we show all the Dense Trajectories [97] which quantize to the same two codebook centers. Even though many of the descriptors come from regions with distinctively different motion and appearance (e.g. the static background versus the swinging torso), they are all encoded to the same codebook center.

On the other hand, if we pool Dense Trajectories over supervoxels, which are regions defined by pixels with similar motion and appearance, these region-based descriptors match to other regions with similar low-level descriptor distributions. In Figure 3.4b we show all the supervoxels that encode to the same codebook centers (Motion Words) as the ones corresponding to the

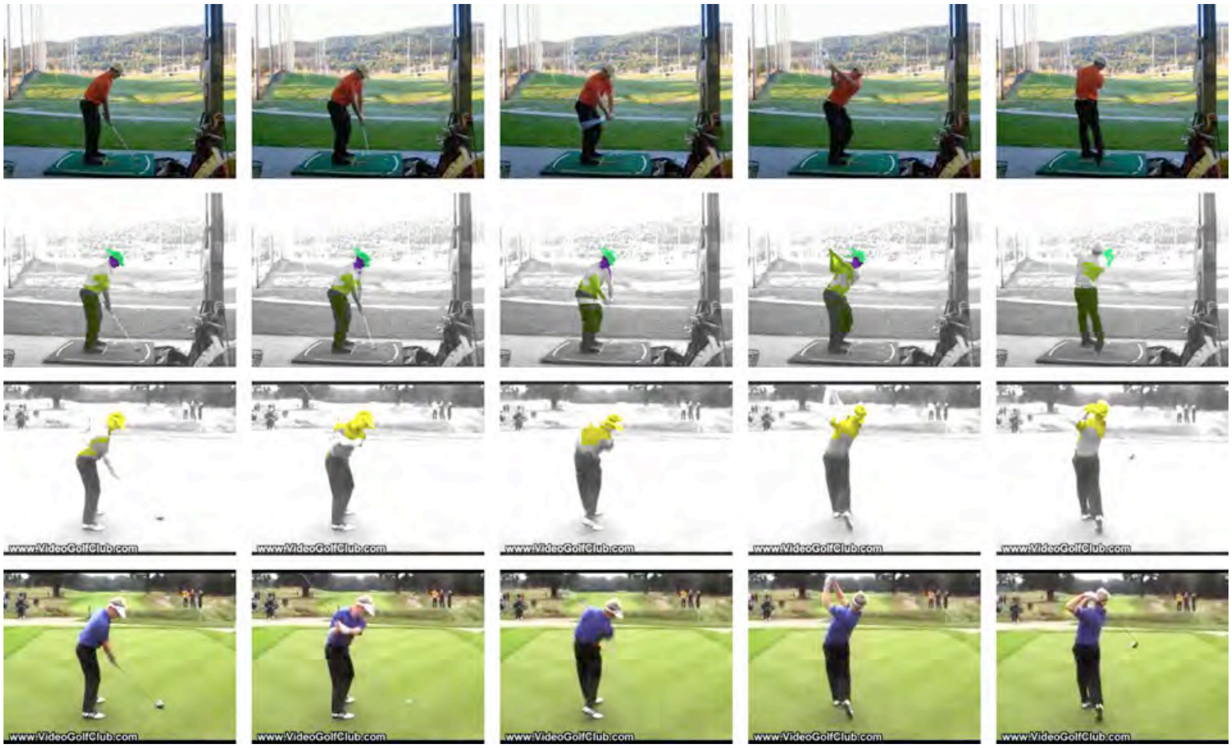
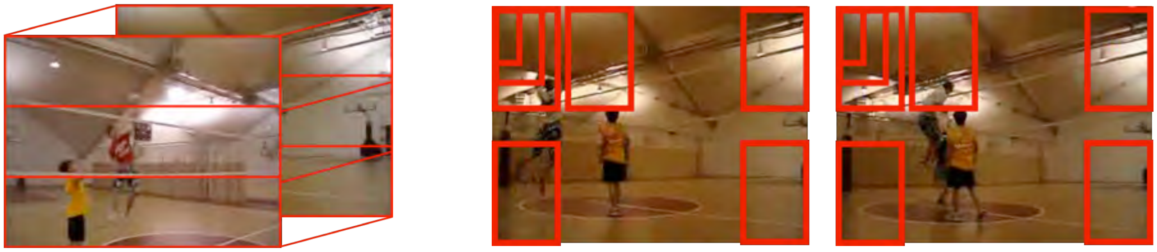


Figure 3.6: Supervoxels from videos with different appearance still encode to the same Motion Word if the low-level descriptor distribution is similar. We display the supervoxels in the second video which encode to the same Motion Word as the supervoxel corresponding to the first golfer's upper body.

person's upper body from the first frame of the swinging motion. Since we use the HOG and MBHX descriptors from Dense Trajectories, and since the camera is stationary, there are no other regions that have similar HOG and MBHX descriptors as the golfer. Furthermore, pooling over regions is robust to scale changes and enables finding correspondences between two videos. For example, in Figure 3.5 we show the same golfer (top), and another trial, but this time the camera is positioned further away, thus the second video has a different scale. We color the supervoxels which encode to Motion Words common to both videos. We see that the regions we expect to match, namely, those of the ground, the sky and the person, do indeed correspond based on their Motion Word encoding. An additional key question which we explore in detail in the evaluation of Motion Words, is whether the representation is too restrictive when matching regions and thus does not generalize. That is, whether we find matches only among regions that are exactly the same. We observe that in fact region matching is general - in Figure 3.6 we show the supervoxels in another golf video that encode to the Motion Word of the golfer's body we used in Figure 3.4b. This video was shot in a different environment, and the swing is performed by a different person.



(a) Spatio-Temporal Pyramid (STP).

(b) Cuboids.

Figure 3.7: Cuboids are widely used to pool low-level features. Spatio-temporal pyramids [97] split the video into several large grid segments in space and in time. Smaller cuboids are extracted at various spatial and temporal scales.

Supervoxels are encoded to the same Motion Word even though the appearance and motion of the supervoxels in the two videos is not exactly the same.

3.3 Related work

Pooling is a key step in computing numerous types of video representations. For example, when applied to videos, the Bag of Words representation is computed either by pooling features in an unstructured way over the entire video [28, 97], over a coarse spatio-temporal pyramid [97], or over predetermined cuboids chosen for convenience or computational reasons [29, 52, 66] (see Figure 3.7). Pooling low-level features over cuboids is also a key step to many methods that learn mid-level representations [13, 42, 56, 100]. For instance, Wang *et al.* [97] split each video into several non-overlapping coarse cuboids to form a spatio-temporal pyramid. The cuboids are quantized to a multi-channel codebook and then concatenated to produce the final video representation. Le *et al.* [52] automatically learn features from video data over predetermined cuboids, which are also used at pooling time. Tang *et al.* [88] use a variable-length discriminative HMM model which infers latent sub-actions together with a non-parametric duration distribution. Izadinia *et al.* [40] use a tree-structured CRF to model co-occurrence relations among sub-events and complex event categories, but require additional labeling of the sub-events. Oneata *et al.* [68] evaluate the use of Fisher vectors as an alternative to Bag of Words histograms to aggregate MBH and SIFT low-level descriptors. Patch-based descriptors [13, 42, 84, 100] also pool features over rectangular regions. However, since the spatial and temporal scales for these cuboids are unknown, works that extract cuboids over several sizes have to prune many of them for computational reasons. Recent works use cuboids defined by users' gaze [62, 79], or consider foreground cuboids, e.g. by detecting regions of interest [112]. Many methods attempt to prune regions corresponding to the background, but prior work has shown that using all regions can be helpful [62]. To enable more robust spatio-temporal support, we propose to use an initial oversegmentation into coherent spatio-temporal regions, which is similar to using superpixel segmentation as the pre-processing step for image analysis [65].

The localization method of Shapovalova *et al.* [80], attempts to locate the action region given training videos annotated only with action class labels. They propose Similarity Constrained Latent Support Vector Machine, a model that relies on rectangular regions of interest extracted by applying an “objectness” operator and mean shift clustering. Other structured prediction models for action recognition and localization include sub-volume search methods [25, 45, 74, 107], more flexible part-based models computed over multiple sub-volumes [45], and path search methods [93]. However, the search procedure required has high computational complexity. Therefore, to formulate the search more efficiently, these methods constrain the action model to an axis aligned rectangular 3D volume search, which is suitable for relatively restricted action scenarios. Furthermore, to initialize the sub-volumes, many methods rely on sophisticated pre-processing [71], person detectors [49], pose annotations [72], bounding box annotations [16, 49], or manual segmentation [45].

In their recent work, Ma *et al.* [84] develop space-time segments which are built in an unsupervised manner and applied for action recognition and localization. A drawback of the representation is the need to prune background regions and to track the foreground regions across frames, which is not robust to noise and to non-professional video quality. Instead, in Motion Words we use all regions obtained from a video segmentation, capturing both foreground and important background information. Even though we do not model foreground regions explicitly, we can still localize and interpret the regions used in classifying videos, which in many cases includes background regions.

The idea of using an initial over-segmentation has been explored in the image analysis community. For example, Gould *et al.* [36] use superpixels as the basic data layer for decomposing a scene into geometric and semantically consistent regions. Similarly, Tighe [90] propose non-parametric image parsing with superpixels. Other works restrict pooling to inputs close in input space [43, 111]. Image processing and de-noising works consider similar inputs to smooth noisy data over a homogeneous sample without throwing out the signal [11, 19, 60]. We hypothesize that in videos it is important to pool features local in space and time, for example, supervoxels, which are regions coherent in motion and appearance. Indeed, the properties considered important in some supervoxel segmentations, e.g. [105], are inspired by properties described as desirable for superpixels by Moore *et al.* [65].

In video analysis, works that first compute supervoxels followed by various task-specific processes include hierarchical grouping [37], long-range tracking [8, 55], superpixel flow [95] and mid-level features [29]. On the other hand, Chen *et al.* [110] model combinations or co-occurances of low-level features, Essa *et al.* [5] use n-grams and regular expressions to encode long-term motion information. Zhang *et al.* [110] propose mid-level features that rely on the definition of a correspondence transform to compare videos with variable number of regions. Rather than develop new metrics to compare videos with a variable size representation, we simply perform a second clustering step to learn a codebook of the region-based features. This provides a fixed size representation for videos which can be used in standard classification methods.

3.4 Motion Words (MWs)

While pooling over rigid cuboids provides computational efficiency, it is natural to consider pooling features over more flexible spatio-temporal regions. Supervoxel segmentation algorithms provide excellent spatio-temporal support for feature pooling. Supervoxels are regions coherent in both appearance and motion over time, e.g. the streaming GBH segmentation [105] and the UES [104] methods shown in Figure 3.9. We propose a new video representation, Motion Words, where we pool low-level features over such coherent spatio-temporal regions. One way to represent a supervoxel would be to average the low-level descriptors within the supervoxel. However, averaging descriptors like HOG, HOF, MBH, or STIPs with their neighbors results in the loss of a considerable amount of information [6]. Instead, we first encode the low-level descriptors to a standard Bag of Words codebook and average the codes within each supervoxel.

We construct the Motion Words representation in four steps (see Figure 3.8):

1. Compute a standard Bag of Words codebook from low-level descriptors;
2. Compute a supervoxel segmentation;
3. Pool encoded low-level descriptors in each supervoxel to compute supervoxel-based feature vectors.
4. Cluster the supervoxel-based vectors to learn a codebook of Motion Words.

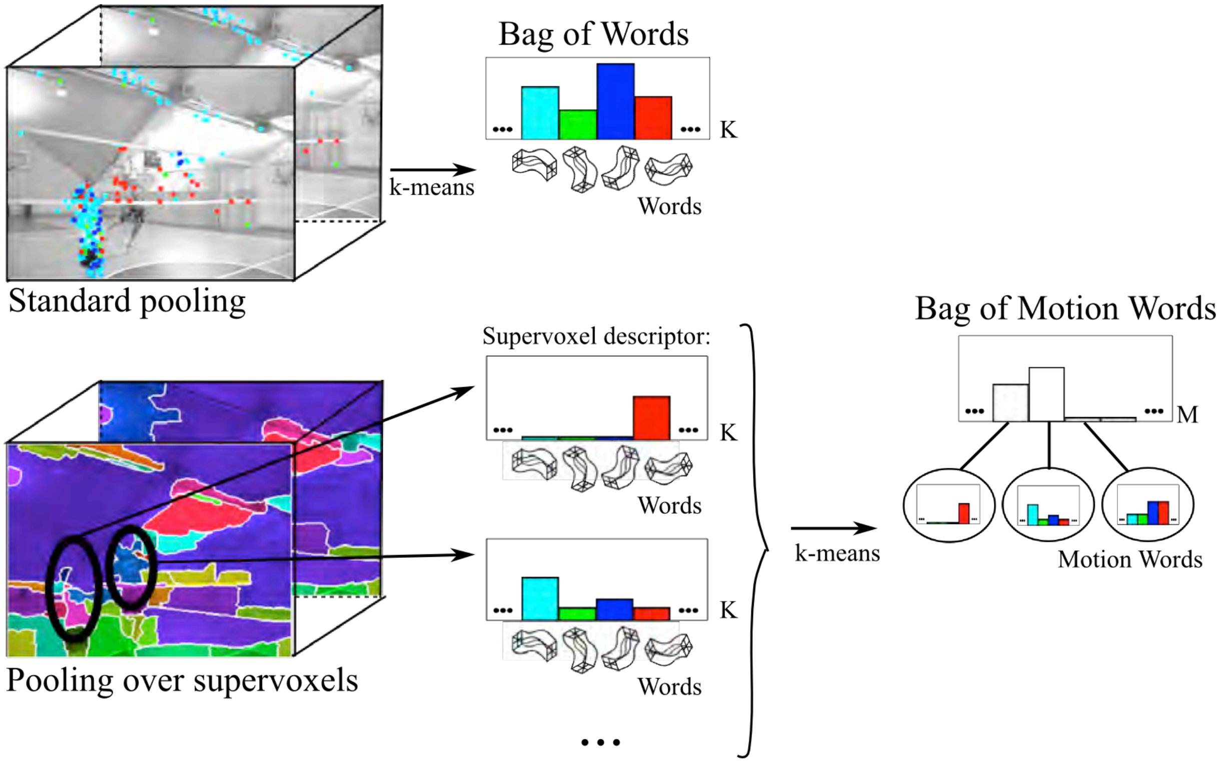
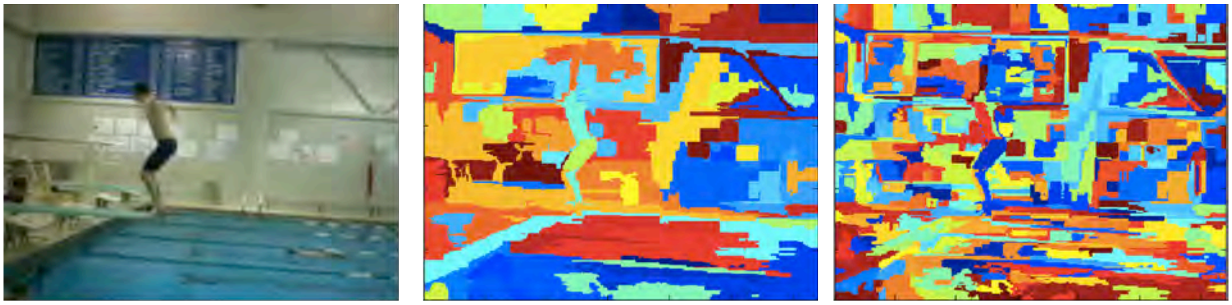


Figure 3.8: In the standard Bag of Words framework, visual Words are learned using a clustering method, e.g. k-means. Then, features from all video frames are pooled (usually via average or max pooling) and encoded to the learned visual Words. The video is represented as a histogram of Word counts (top). Instead, we propose to pool the BoW encoded feature vectors over supervoxels. That is, *each supervoxel* is represented as histogram of Word counts. We cluster the supervoxel histograms to learn a codebook of Motion Words and represent the video as a histogram of Motion Word counts.

3.5 Bag of Motion Words (BMWs)

Each video is an unordered collection of a variable number of supervoxel-based feature vectors of the same dimension (Figure 3.10). This representation does not allow for direct comparison of two videos because it does not provide a unique mapping of supervoxel features. There are several ways we can proceed to construct the video representation. Zhang *et al.* [110] develop an approach to handle a variable number of features per video by defining a correspondence transform for comparing videos. Wang *et al.* [100] define a similarity metric on subregions to compare cuboids of different dimensions. Other methods cluster trajectories but rely heavily on finding a good distance metric, which is not straightforward [10, 73].

Other works constrain the number of features to be the same for each video and then concate-



Diving action

Coarse supervoxels

Fine supervoxels

Figure 3.9: Example supervoxels from the coarse and fine segmentation hierarchies of the streaming GBH algorithm [105].

nate the descriptors [80]. Instead, we propose to use all available information from all regions and deploy the statistical power of the Bag of Words framework a second time (Figure 3.10). Given a collection $\mathcal{M}$ of Motion Words from a training set of videos, we cluster them via k-means to learn a Motion Words codebook $\mathcal{K}$ of size K. Thus, the video representation is a Bag of Motion Words (BMW), i.e. a normalized histogram of Motion Word counts. Finally, a classifier can be trained using the proposed representation, for instance, using the setup of [52, 97] where we train a one-vs-all SVM classifier to learn a model for each action class.

Formally, let κ be a codebook of size k learned over a set of features in a standard BoW framework (e.g., dense trajectories [97], or features learned directly from video data [52]). Let $\Gamma^v = \{\gamma_1^v, \ldots, \gamma_n^v\}$ be a segmentation of video v into n spatio-temporal regions (e.g., obtained using the hierarchical method of Grundmann and Essa [37], or the streaming method of Xu *et al.* [105]). We encode the low-level features in each spatio-temporal region γ_i^v according to the codebook κ, and average them within each region γ_i^v. That is, each supervoxel γ_i^v is represented as a histogram of size k. We cluster the supervoxel histograms to learn a codebook of Motion Words of size M. Finally, the supervoxel histograms are encoded to the Motion Words codebook and each video v is represented as a histogram $\mathcal{M} = \{\mu_1, \ldots, \mu_M\}$ of Motion Word counts (see Figure 3.10). We visualize some of the learned Motion Words in Section 3.8.

3.6 Motion Words analysis

We evaluate the design choices involved in building the Motion Words representation. We discuss and evaluate the choice of low-level features, segmentation, quantization, and classifier methods. It is important to know if the BoW codebook used for Motion Words provides better

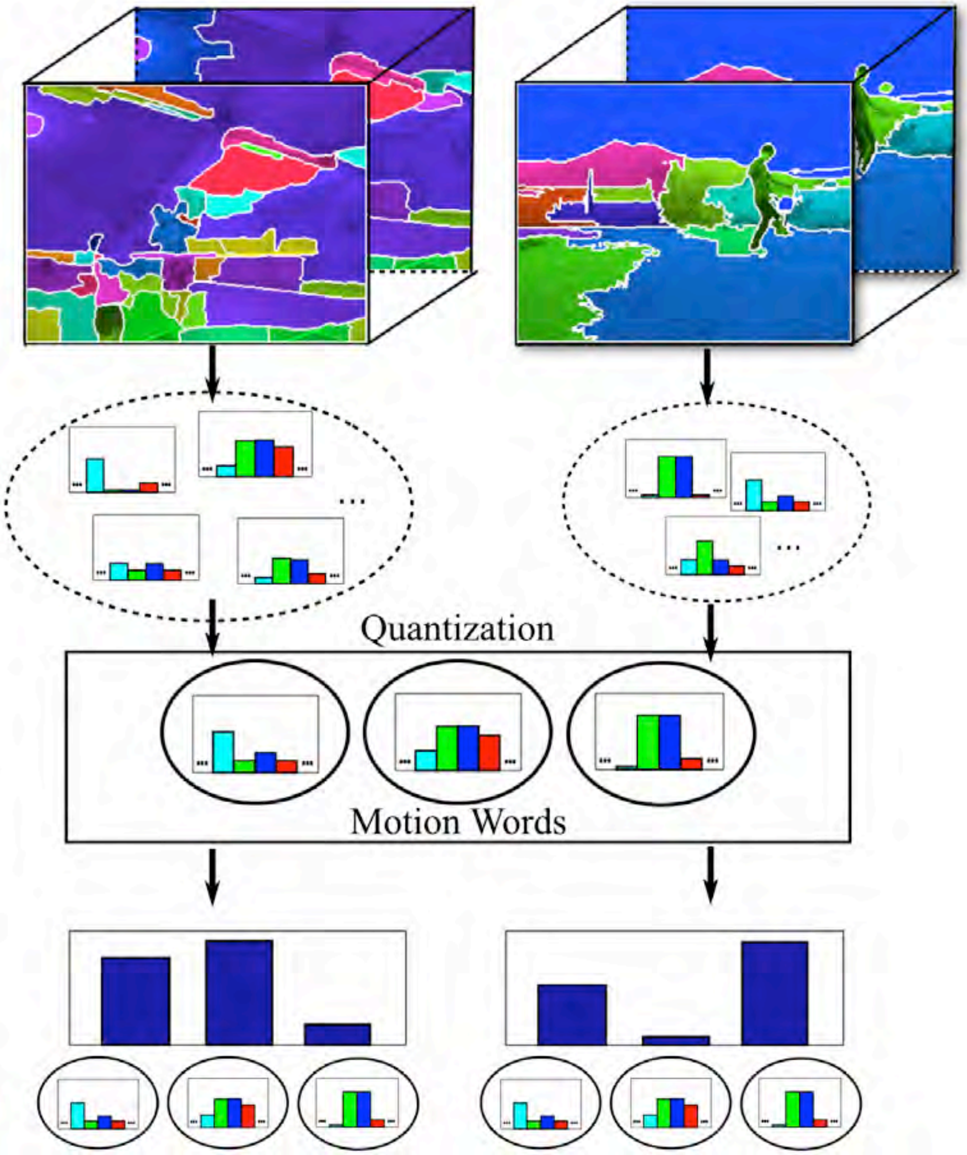


Figure 3.10: We compute a Bag of Motion Words representation using the supervoxel-based descriptors as the input to a standard Bag of Words framework. Each video generates a number of supervoxel-based descriptors. The descriptors from all the training videos are clustered to learn a codebook, i.e., Motion Words. Each video is then represented as a histogram of Motion Words counts.

support when learned from dense features, or if sparser, interest point based features are able to capture sufficient low-level information? We analyze a variety of feature modalities and their combinations, including a manually constructed codebook. Furthermore, a key question is that of the choice of supervoxels - is pooling over regions only useful if the supervoxels strictly respect object boundaries, or is the method suitable even with supervoxels that approximate object and motion boundaries? We also answer the question of how sensitive the representation is to the sizes of the low-level Bag of Words codebook and of the Motion Words codebook. Additionally, we test sensitivity to the size of the BoW codebook and explore if dimensionality reduction at this step is beneficial. Finally, we examine whether an over segmentation or an under segmentation into spatio-temporal regions is more useful for this representation. In the following sections we present detailed analysis of the Motion Words representation in several quantitative and qualitative tasks. We report quantitative results in Section 3.7 and qualitative analysis in Section 3.8.

3.6.1 Low-level feature representation

Many kinds of low-level features and their combinations can be pooled over supervoxels to to built Motion Words. We can easily pool descriptors extracted on a dense grid by simply counting those that fall within each supervoxel. Examples of dense features include HOG3D [46], Motion Boundary Histograms (MBH) [97], STIPs [50]. On the other hand, features that span several frames, e.g. Dense Trajectories [97], with default length of 15 frames, can be pooled by defining a minimum temporal overlap threshold with a supervoxel to determine which trajectories should be counted. Similarly, we can pool cuboid-based features by defining a minimum volume overlap threshold with a supervoxel. Examples of descriptors computed over cuboids include the automatically learned features via subspace analysis, ISA [52], and mid-level features [5, 42, 75, 100, 110]. While we can also use long trajectories (e.g. Brox and Malik [10]), or features extracted only at interest points, they are sparse and many supervoxels will be empty. Nevertheless, such sparse features could be used to complement dense features.

To evaluate the usefulness of Motion Words in the experimental section we consider three types of features that have been successfully used in activity classification: state-of-the-art Dense Trajectories (*DT*) [97], automatically learned features through independent subspace analysis (*ISA*) [52], and dense HOG, HOF and MBH descriptors [97]. We extract features using code provided by the authors. Since we want to test pooling of different types of descriptors, we use the default settings without performing any parameter tuning.

3.6.2 Supervoxels

For the choice of spatio-temporal regions, we consider state-of-the-art video segmentation methods. The streaming graph-based algorithm of Xu *et al.* [105] is well suited for Motion Words because it encodes properties such as spatio-temporal uniformity and coherence, and boundary detection. We find that the default parameters suggested by the authors are a good trade-off between size of supervoxels and motion boundary preservation. We extract three hierarchy levels and compare pooling over supervoxels from the coarsest level (GBH coarse), the finest level (GBH fine), and the union of all three levels (GBH coarse + fine). Furthermore, we evaluate the Bag of Motion Words framework using the Uniform Entropy Slice segmentation algorithm [104], choosing to optimize for "motion-ness." Each video has in the range of $300 - 1000$ supervoxels generated from the streaming GBH method at the finest level, $20 - 100$ generated at the coarsest level, and $20 - 30$ supervoxels generated by the UES method. We randomly sample $100,000$ of the training supervoxels to learn a BMW codebook using k-means and Euclidean distance. We count trajectories as part of a supervoxel if at least half of the trajectory is contained in the supervoxel. In the case of ISA features, we count only those that overlap a supervoxel by at least 30%.

3.6.3 Feature pooling

There are three aspects when computing the Bag of Words and the Bag of Motion Words representations: 1) when do we count a descriptor as part of a supervoxel, i.e., inclusion rules; 2) what type of pooling we apply, i.e., pooling strategy; and 3) how we encode the features.

Inclusion rules. Depending on the types of low-level features used to learn the BoW codebook, there are several ways to perform the pooling over spatio-temporal regions by varying the minimum overlap threshold τ which determines whether to include a feature data point in the Motion Word. For the *Dense Descriptors* modality pooling is straightforward - descriptors that fall in each spatio-temporal segment μ_i^v are discretized to the BoW codebook κ learned from clustering the *DDs*. For the *Dense Trajectories* modality, where the suggested length of the descriptors is 15 frames [97], we consider the overlap of the trajectories with the spatio-temporal regions. First, we report results when quantizing the descriptors that intersect each spatio-temporal region, μ_i^v, for at least $\tau = 7$ frames (i.e., we require that at least half of the trajectory be covered by the spatio-temporal region to be considered as part of the Motion Word μ_i^v). We will perform further analysis for lower values of τ.

For the *ISA* modality, we use the same BoW codebook as the authors, computed by clustering features computed over cuboids in a two-layer neural network. Each cuboid is a vector of responses from the first layer concatenated with the PCA-reduced features from the second layer. A cuboid c_i^f, which is closest to codebook center κ_j, contributes to the final histogram representation by the sum of responses from the first neural network layer. However, rather than pool the neural network responses from the first layer over cuboids, we consider only the response values over each spatio-temporal region. Let $\Sigma^v = \{\gamma_1^v, \ldots, \gamma_s^v\}$ denote the set of cuboids into which video v is decomposed, for a total of s cuboids. For the spatio-temporal region μ_i^v, denote the overlapping cuboids as the set $\sigma_i^v = \{c_1^v, \ldots, c_o^v\}$, where for each i, $c_i^v \in \Sigma^v$. Suppose that each of the cuboids in σ_i^v is closest in Euclidean distance to the codebook centers $\{\kappa_1, \ldots, \kappa_o\}$. We compute the intersection of the spatio-temporal region μ_i^v with each overlapping cuboid and sum the neural network responses in each intersection: $\{r_1^v, \ldots, r_o^v\}$. Thus, the spatio-temporal region μ_i^v is represented by a histogram which has the weights $\{r_1^v, \ldots, r_o^v\}$ in the corresponding codebook centers $\{\kappa_1, \ldots, \kappa_o\}$.

Pooling strategy. When computing the encoding for each region, the features can be pooled in one of two ways: sum pooling, in which case the encodings of features in a given region are combined additively; or max pooling, in which case each bin in the encoding is assigned a value equal to the maximum across feature encodings in that region [6, 14]. We evaluate both strategies during pooling to learn the standard Words, and during pooling to learn the proposed Motion Words.

Encoding methods. The two most common encoding methods for Bag of Words are hard quantization (histogram encoding) and soft quantization [14]. Other successful encoding methods include locality-constrained linear encoding (LLC), Fisher vector encoding [68, 76, 101], and super vector encoding [14]. In the case of Motion Words, especially when pooling over small supervoxels, it makes sense to use a soft quantization method, since each supervoxel will have a small number of low-level descriptors, making the supervoxel histograms very sparse. Based on the detailed evaluation of these methods on images presented in Chatfield *et al.* [14], we analyze learning Motion Words via standard hard quantization and via locality-constrained linear (LLC) pooling. We evaluate several combinations during both encoding steps: learning the low-level feature Words and learning the Motion Words.

3.6.4 Quantization

For computing the low-level feature codebook, we follow the setup of Wang *et al.* [97] by randomly sampling $100,000$ data points per feature channel, and clustering with k-means, where

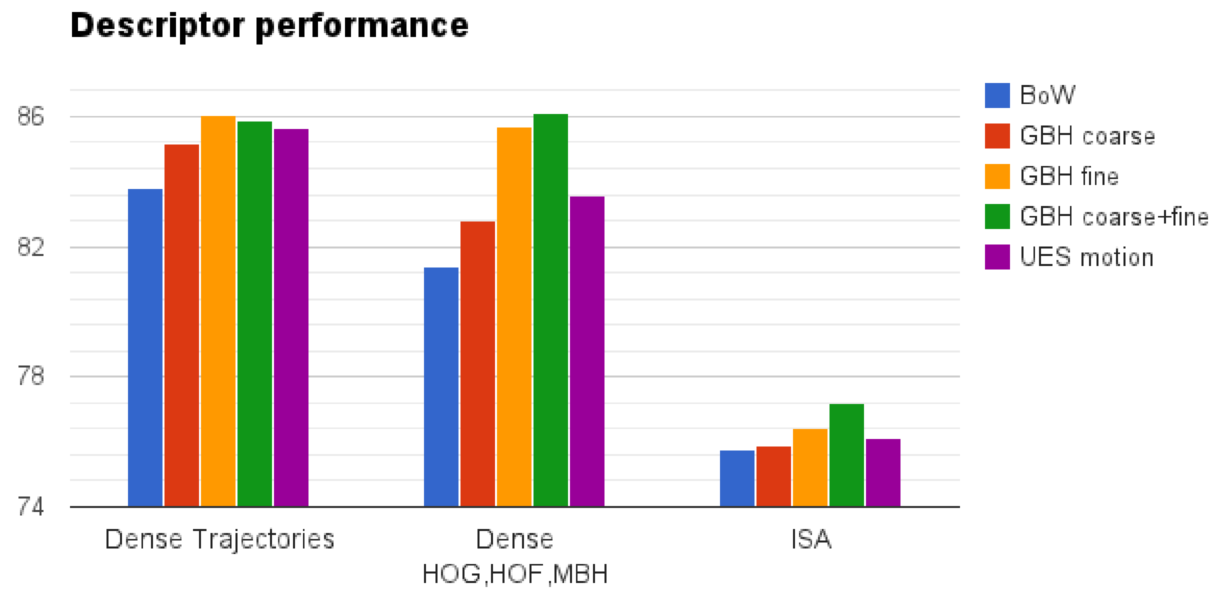


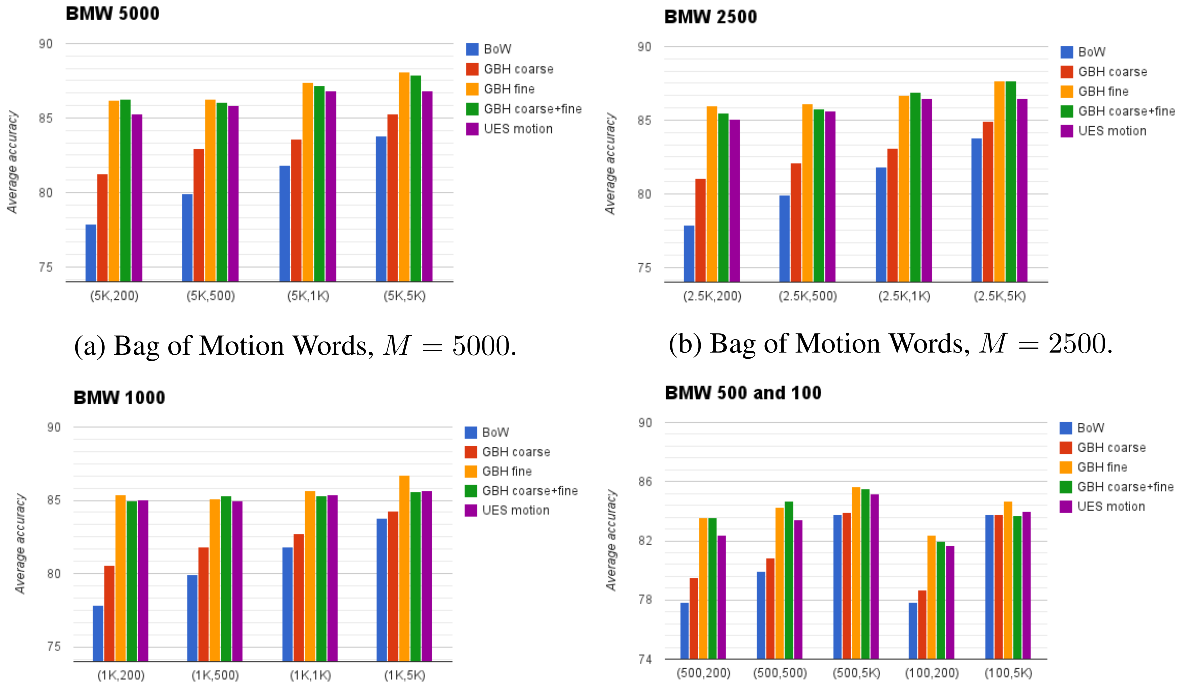
Figure 3.11: Classification accuracy on the YouTube dataset using standard BoW representation and the proposed Motion Words representation for different types of video descriptors. Pooling over non-coarse supervoxels always improves performance. Dense Trajectories are more robust for action classification since they capture information over several frames (the default is 15 frames).

$k = 5000$. Additionally, to analyze the sensitivity of Motion Words on the size of this codebook, we evaluate codebooks of sizes $1000, 500$, and 200. Finally, for the Bag of Motion Words we consider k-means with $5000, 2500$ and 1000 clusters using Euclidean distance. Since the number of supervoxels per video can be very small, soft quantization is better suited when encoding the Motion Words.

3.7 Quantitative evaluation

3.7.1 Datasets

We evaluate the framework on two datasets: the YouTube dataset [57], and the HMDB [47] dataset. The former dataset contains 11 action categories with a total of $1,168$ sequences, with roughly 44 test videos per split. It is a challenging dataset due to large variations in camera motion and viewpoint, object appearance, pose, and scale. The HMDB dataset consists of 6849 clips divided into 51 action categories, with 1530 test videos per split. For both datasets, we use the train/test splits provided by the authors.



(a) Bag of Motion Words, $M = 5000$.

(b) Bag of Motion Words, $M = 2500$.

(c) Bag of Motion Words, $M = 1000$.

(d) Bag of Motion Words, $M = 500, 100$.

Figure 3.12: We report classification accuracy on the YouTube dataset using Dense Trajectories and different sizes of codebooks. When pooling features over non-coarse supervoxels encoded to sparse codebooks (e.g., 5000 codebook centers), we obtain better classification performance compared to BoW and STP.

3.7.2 Action classification

We learn a one-vs-all SVM [38] classifier with a χ^2 kernel and find the parameters via 5-fold cross validation on the training set. Similarly to Wang *et al.* [97], we combine the feature channels and report average accuracy[1]. We evaluate the key components of BMW, namely, the supervoxel settings, the low-level features, and the size of the representation.

Low-level features. In Figure 3.11 we show classification accuracy on the YouTube dataset using the proposed Motion Words representation where we pool different types of low-level descriptors: Dense Trajectories [97], which capture HOG, HOF, and MBH over 15 frames by tracking interest points; Dense Descriptors [97] which do not track points; and automatically learned ISA features computed over cuboids. Compared to the standard BoW representation, pooling over supervoxels always performs better. The Dense Trajectories capture temporal information better than the Dense Descriptors and obtain higher performance. We find that the ISA

[1] We compute average classification accuracy by taking the label of the most confident classification among the one-vs-all SVM classifiers for each test video.

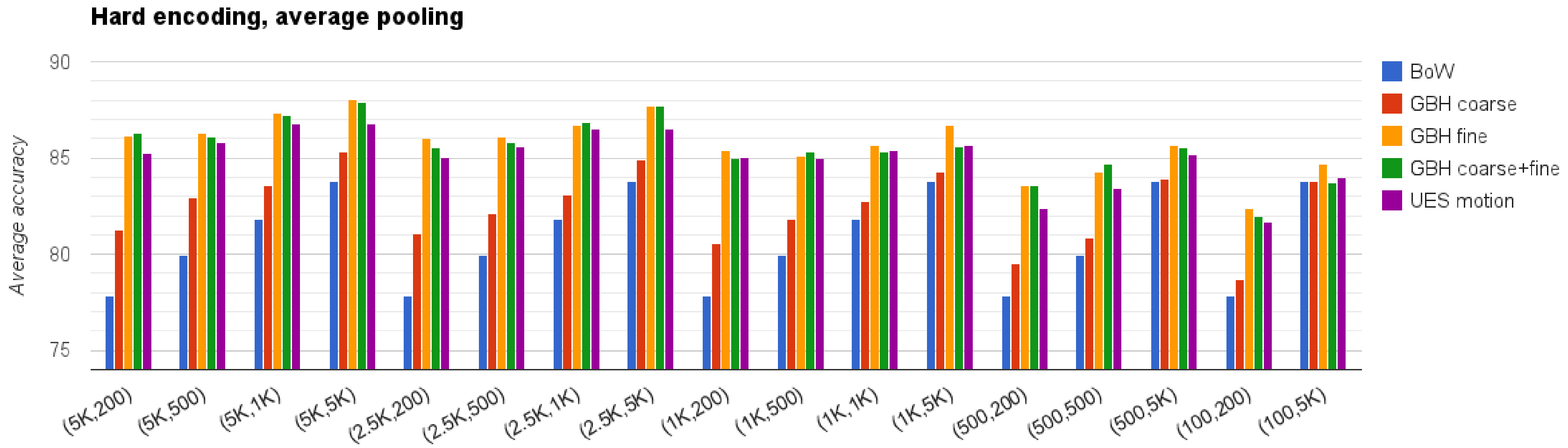


Figure 3.13: The fine GBH segmentation has better classification performance compared to other segmentations on the YouTube dataset. The average accuracy increases as the BoW codebook becomes sparser, and as the BMW codebook size increases. Codebook sizes are denoted in the format (K_{BMW}, K_{BoW}). In these experiments, we use the standard hard encoding and average pooling for both BoW and BMW.

features are not suitable for pooling over supervoxels since they are computed over cuboids of sizes much larger than the extracted supervoxels.

Codebook size. In Figure 3.12a we show classification accuracy on the Youtube dataset using Dense Trajectories. We compare to standard Bag of Words (BoW), which performs at 83.8% (comparable to the previously reported 84.1% by [97]), Spatio-Temporal Pyramids (STP) [97], with performance of 85.4, three supervoxel methods obtained with the streaming GBH algorithm (GBH course, GBH fine, and GBH coarse+fine), and supervoxels obtained from the UES algorithm with the "motion-ness" objective. In this graph, we vary the BoW codebook size, while holding the BMW codebook size fixed at 5000 centers. We find that coarse supervoxels (GBH coarse) are less suitable for pooling dense trajectories. They perform only slightly better than global pooling, and at the same level as STP (85.4). On the other hand, the other three supervoxel settings all outperform global pooling and STP. The finest level of the GBH supervoxel hierarchy is able to encode actions best, followed by the UES method and the combined hierarchy levels.

In Figures 3.12b, 3.12c, and 3.12d we fix the size of the Motion Words codebook and compare performance for varying sizes of the underlying BoW codebook. When the BoW codebook is very sparse (5000 centers), we obtain highest performance. Similarly, as the BMW codebook size increases, performance also increases. In Figure 4.39 we compare all codebook size settings together, which shows the relationship of increased performance to larger codebook sizes. Pooling over coarse supervoxels (GBH coarse) does not perform as well as fine supervoxels, regardless of the codebook size. However, it is comparable to standard BoW and it is still useful since the representation enables interpretability of the results (see Section 3.8).

We also evaluate the BMW representation on the challenging HMDB dataset. In Figure 3.14 we show classification performance compared to prior results reported by Wang *et al.* [97] using Dense Trajectories, and the improved Dense Trajectories features, Wang and Schmid [98]. We only evaluate fine supervoxels using Dense Trajectories and we use the same codebooks learned on the YouTube dataset. The results show that the Motion Words transfer well across datasets, and outperform the other representations. We attribute the good performance of the fine supervoxels to the ability of the segmentation method to preserve motion boundaries. BMW with the fine GBH segmentation obtains 58.8% classification accuracy, which is better than the 57.2% previously reported by Wang and Schmid [98].

Pooling and encoding methods. In Figure 3.15 we show results from applying different pooling and encoding methods for computing the BMW representation on the same classification task on the YouTube dataset. The standard hard encoding and average pooling performs as well

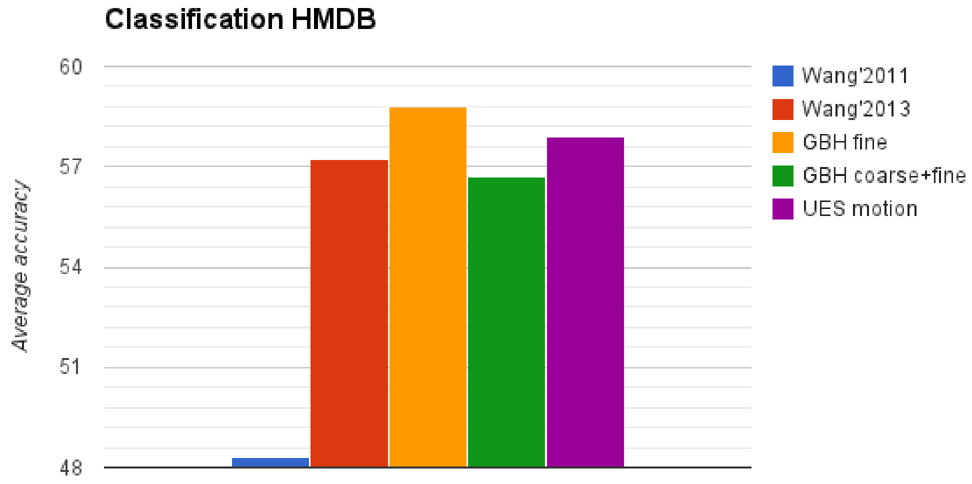


Figure 3.14: We report classification accuracy on the HMDB dataset using a BMW of 5000 Motion Words with an underlying BoW codebook of 5000 centers. Importantly, we use the codebook centers learned on the YouTube dataset and find that the learned Motion Words transfer across datasets and obtain better performance compared to other pooling methods.

as the other methods, or better when the codebook size is large. Hard encoding with max pooling performs comparable to the standard strategy, except for very sparse codebooks (5000). The sparse pooling method, LLC [6], performs better when combined with max pooling. Overall, the standard strategy obtains better performance for larger codebooks. We hypothesize that larger codebooks are sparser and are thus able to capture the descriptor encodings well when using hard encoding and average pooling.

3.7.3 Video retrieval

We analyze the usefulness of the proposed representation in the task of directly comparing videos using nearest neighbor. This is useful as we can evaluate the representation independently of finding good parameters for a classifier, which can be tricky, time consuming, and could lead to overfitting the data. In this nearest neighbor task, we also examine the effect the distance metric has on comparing videos represented in the Bag of Motion Words framework. We consider χ^2 distance with L1 histogram normalization. We simply concatenate the channels for each video, and treat the test set as query videos. A successful match is one where the retrieved video has the same action label as the query video.

In Figure 3.16 we report average recall from a nearest neighbor retrieval task on the YouTube dataset using χ^2 distance, where we use the provided action labels to determined correct retrieval.

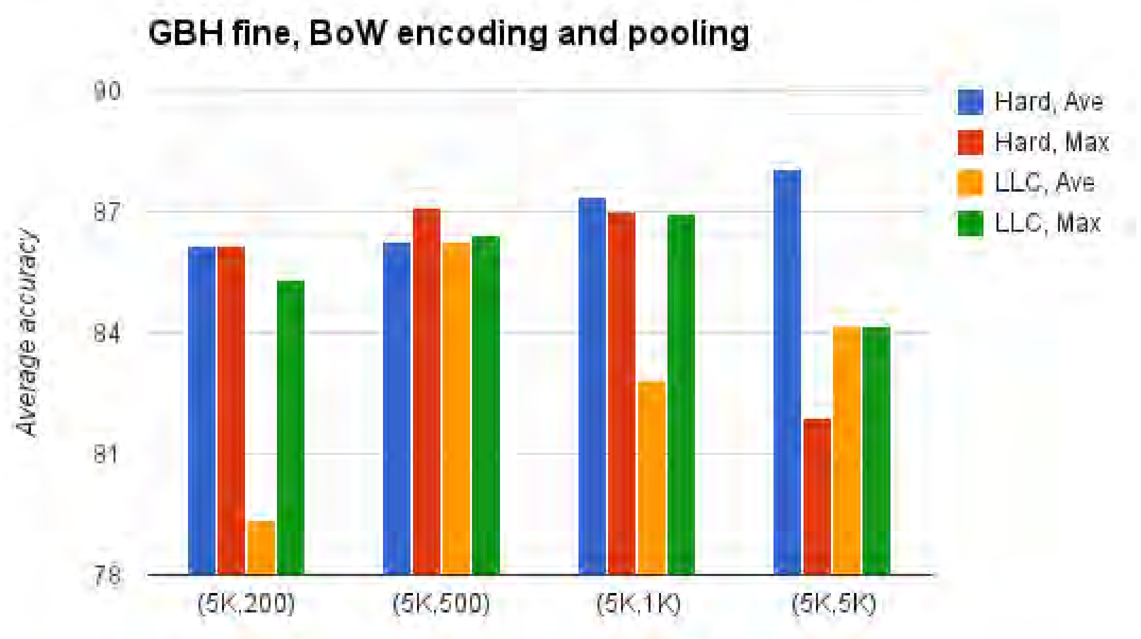


Figure 3.15: For the fine GBH segmentation, we evaluate several encoding and pooling strategies. The standard hard encoding and average pooling performs as well as the other methods, or better. Hard encoding with max pooling performs comparable to the standard strategy, except for very sparse codebooks (5000). The sparse pooling method, LLC [6], performs better when combined with max pooling. Overall, the standard strategy obtains better performance for larger codebooks. We hypothesize that larger codebooks are sparser and are thus able to capture the descriptor encodings well when using hard encoding and average pooling.

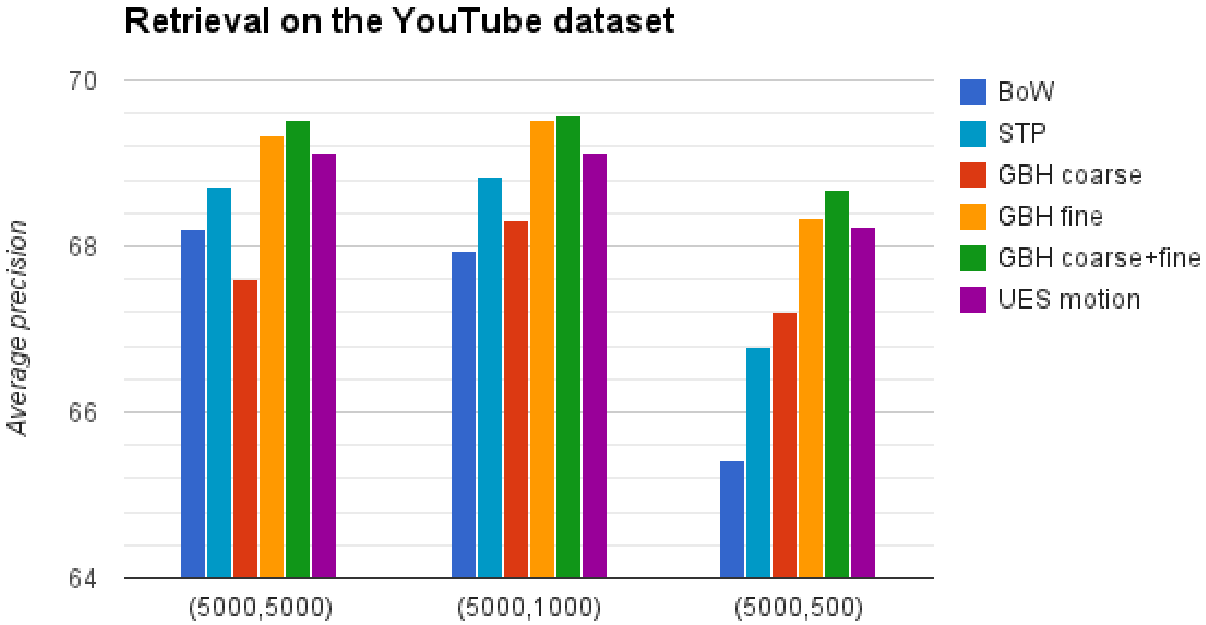


Figure 3.16: We report nearest neighbor retrieval results on the YouTube dataset. When pooling over non-coarse supervoxels, the Motion Words representation outperforms BoW and STP.

Similarly to the classification performance results, we find that coarse supervoxels do not provide good spatio-temporal support for pooling low-level features. However, we find that pooling over the finer supervoxels outperforms BoW and STP representations.

3.7.4 Sparsity Analysis

Further, we evaluate the proposed representation in terms of its "sparsity." We define "sparsity" as the number of zero bins in the video histogram, normalized by the length M of the histogram and averaged over all videos. Standard BoW with global pooling on the YouTube dataset generates a representation that has 43% sparsity, that is, 43% of the bins are unused, on average. Pooling over STPs generates a representation with 60% sparsity. On the other hand, the BMW representation with GCS is much sparser, with 90% of the bins being zeros, on average. In general, the biking and walking dog classes have less sparse representations on average. We hypothesize that since videos from these classes have a large amount of camera motion, more codebook centers are required to capture the supervoxel-based descriptors. On the other hand, action classes like golf require very few codebook centers since most of the videos are shot from static cameras and the golf swing motion is short and spatially constrained.

3.8 Qualitative evaluation

3.8.1 Interpretability

The BMW representation enables interpretation of the features common between videos. That is, given two videos, we can answer the question “which are the similar regions between the videos?” When using Dense Trajectories, which encode HOG, HOF, MBHX, MBHY and trajectory information, we can specifically ask for the regions similar in appearance, flow, horizontal motion, vertical motion, trajectory information, or any combination of these channels.

For example, in the tennis action we want to visualize regions that have similar motion, that is, supervoxel regions with similar horizontal (MBHX) and vertical motion (MBHY). In Figure 3.17 we show the original video frames for two videos and the Motion Words they share. We color each supervoxel based on the Motion Word id that they encode to. We see that the supervoxels corresponding to both players (colored in bright green) encode to the same Motion Word. Additionally, the supervoxels corresponding to the trees and ground also encode to the same Motion Words, since they are stationary. On the other hand, while we might expect the ground regions to also match, the motion induced by the second player’s shadow makes these regions distinct in terms of their motion descriptors (in fact, in terms of their HOG descriptors, too, since the edge information is different).

Similarly, in Figure 3.18 we visualize the HOG, MBHX and MBHY channels for two videos of walking a dog. The two videos are of the same person and dog, in the same environment, however there is a large amount of camera motion, zoom change and a large amount of blur and pixelation. We observe that in general, the regions corresponding to the grass and background encode to the same Motion Words. Likewise, the regions corresponding to the dog match well. In these videos, due to the camera motion, the segmentation is unable to segment the person from the ground very well, and since many of the pixels corresponding to the person are labeled with pixels of the ground, we do not get good matches.

Finally, in Figure 3.19 we show region matches from two challenging biking videos, where the bikers move quickly and the camera follows them, resulting in large and dominating camera motion. In this scenario the segmentation algorithm has a hard time finding consistent motion boundaries and the resulting segments are jittery and many are disjoint. Regardless of this noisy segmentation, when we use the low-level descriptors in these regions, we still obtain good matches between the videos. The regions that are similar include the road, the bikes, and one of the biker.

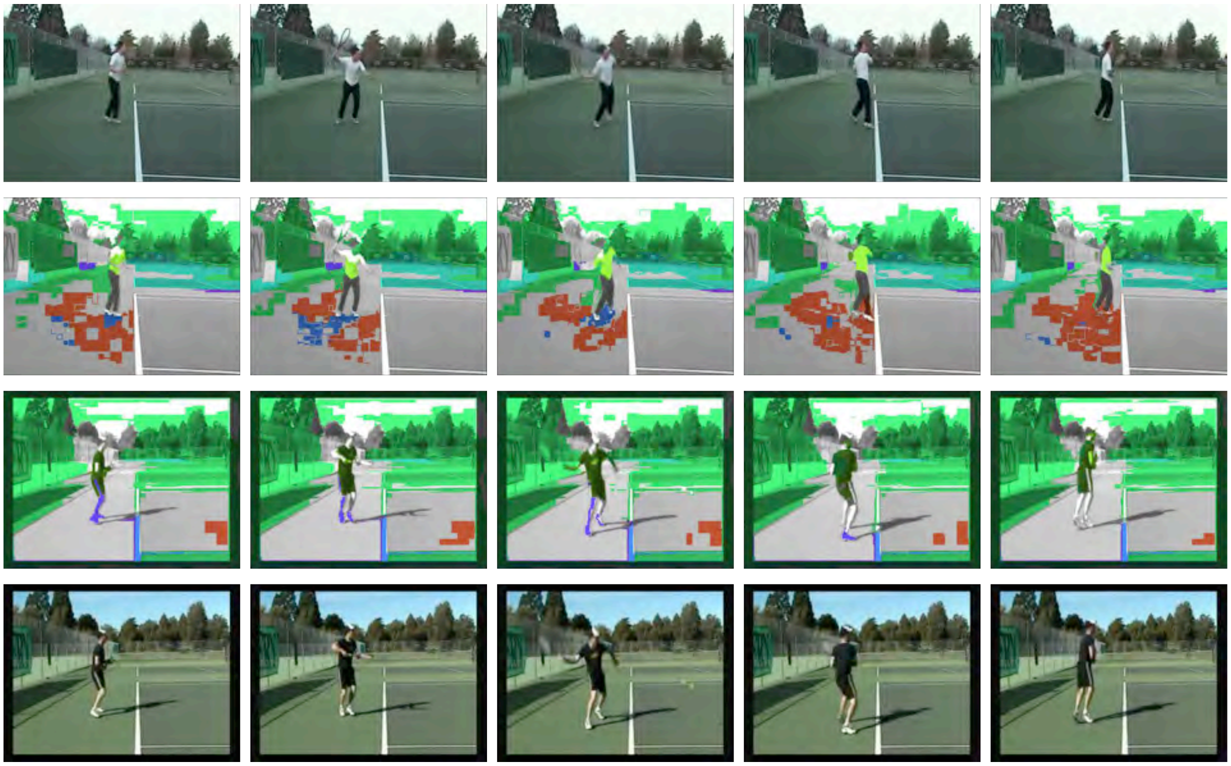


Figure 3.17: Two tennis videos and their matching Motion Words. The two videos are taken in the same environment, but at different locations (courts). The supervoxels corresponding to the players encode to the same Motion Word because they have similar texture and horizontal motion. The supervoxels corresponding to the trees also encode to one Motion Word. Note that the supervoxels corresponding to the ground have different horizontal descriptors due to the shadow of the second player.

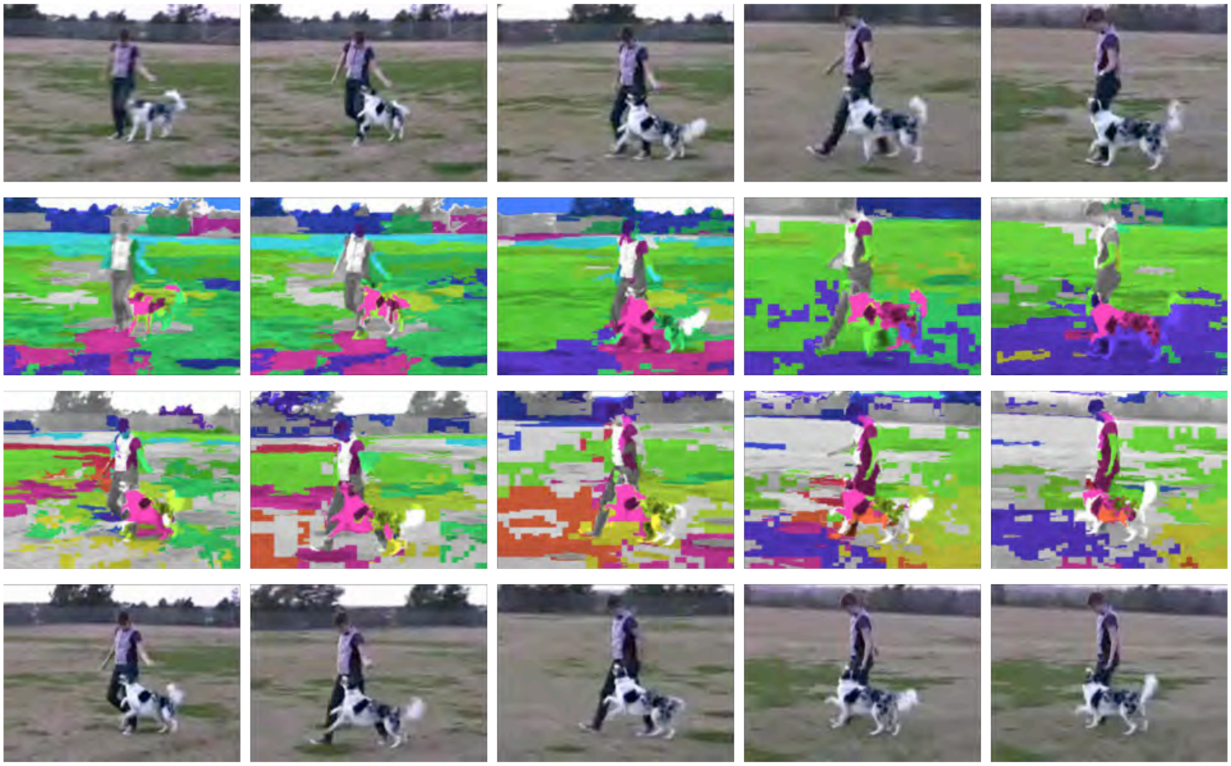


Figure 3.18: Two walking dog videos with large camera motion and jitter, causing a large amount of blur. Nevertheless, we still obtain common Motion Words for regions that have similar appearance and motion.

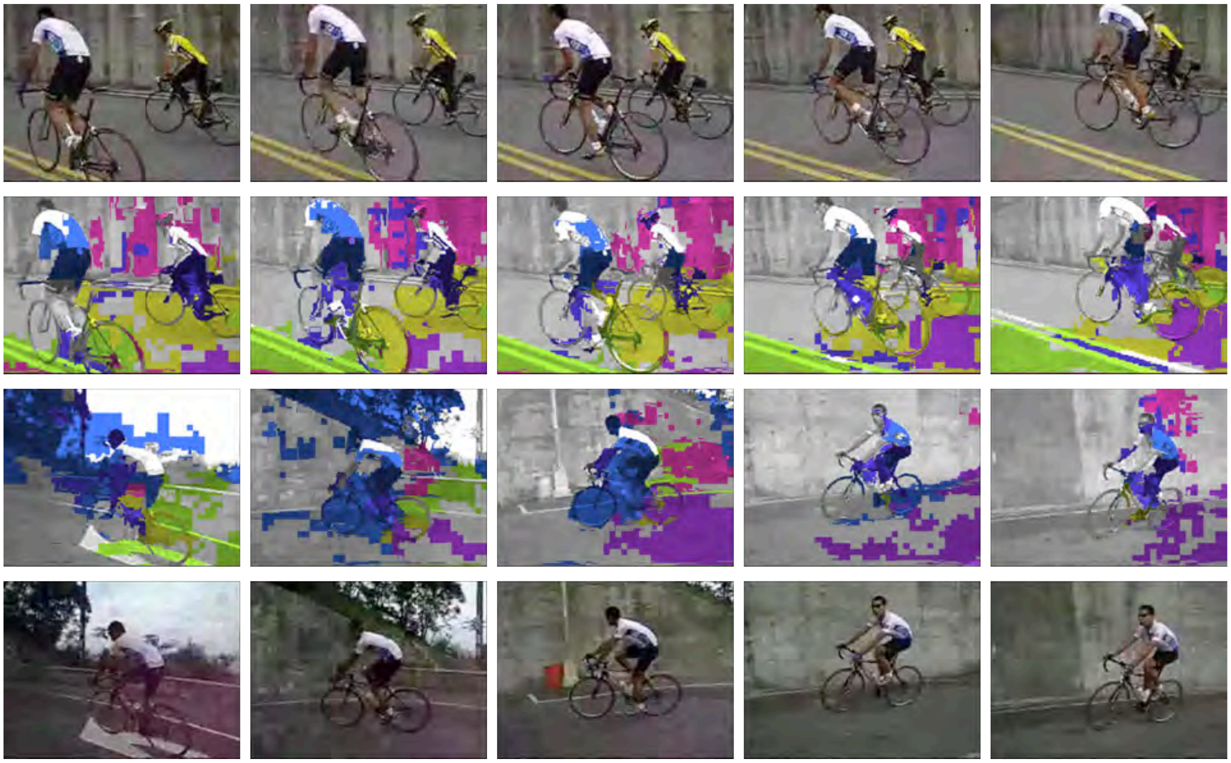


Figure 3.19: Two very challenging biking videos with large camera motion, blur, and quickly moving objects of interest in different environments. Common regions are those originating from Motion Words corresponding to the bikers and the lane markings on the road.

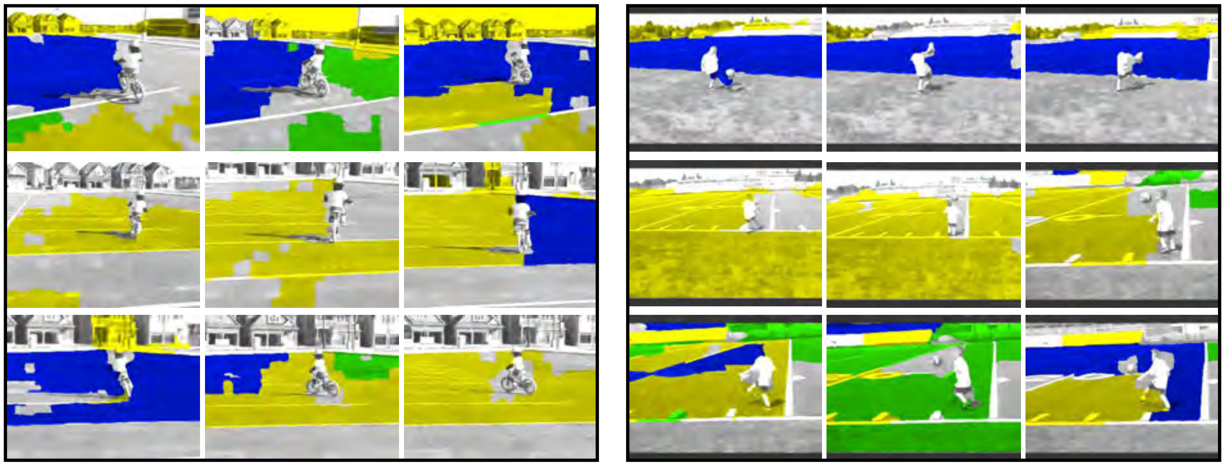


Figure 3.20: The query bike video (left) is mistakenly classified as juggle (right) in a nearest neighbor retrieval task. We can visualize Motion Words that are common between the two videos. Yellow denotes the HOF channel, green the trajectory, and blue denotes multiple channels (including HOG and MBH).

In addition, Motion Words make it easier to interpret incorrect results. This is especially useful in nearest neighbor search, since the only information we obtain from the retrieval is a single number corresponding to the distance between the videos. This however does not tell us *why* the videos are similar. For example, in a retrieval task with a query bike video and a retrieved juggle video (Figure 3.20), we can easily visualize the regions that make the two videos similar and gain understanding of the result. When we visualize the common regions (Figure 3.20), we observe that the regions of the soccer field in the juggle video have similar appearance to the parking lot in the biking video. Furthermore, these regions encode to the same Motion Words based on their flow - in both cases the camera starts out far away, then zooms in on the children and follows them. Since these regions dominate in both videos (the bike and the soccer ball comprise a very small number of pixels), we can now understand why the videos are similar to each other.

3.8.2 Unique Motion Words

In addition, we can evaluate how well the representation captures features specific for each action class. For example, Motion Words that appear only in videos from one action class are unique to that class. The total number of unique Words in the YouTube dataset using the BoW representation is only 83 out of $20,000$ Words, for STP it is 220, and for BMW it is 1280. We visualize a few of these Motion Words for the action juggle in Figure 3.21 and for the action diving in

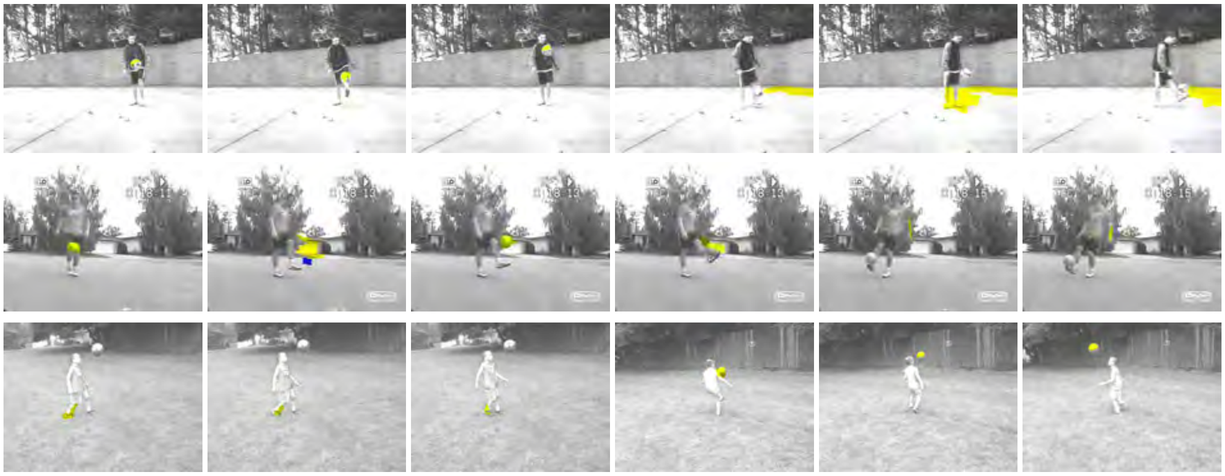


Figure 3.21: For the action juggle, we pick the Motion Words (codebook centers) that are used (expressed) only in this class. In the YouTube dataset, we pick only centers that are used by videos from more than one group.

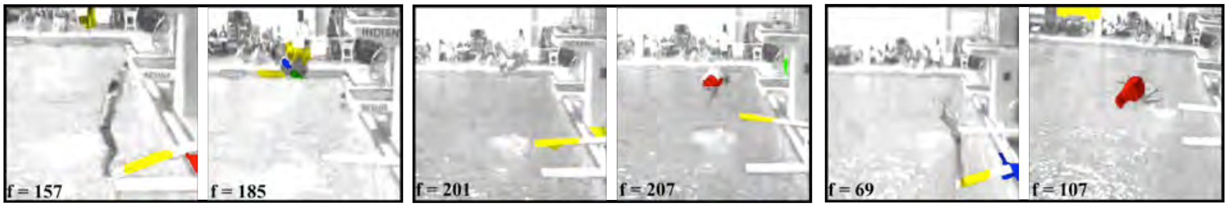


Figure 3.22: Motion Words unique to diving are present when the diving board oscillates up and down, and when the divers spin in the air. Interestingly, the upward and then downward motion of the divers is not unique to this class - the action jumping also features people moving up and down. Yellow are Motion Words matching in motion, and red are matching in appearance.

Figure 3.22. In juggle, the supervoxel regions roughly correspond to the soccer ball and the legs of the players, and in diving they correspond to the divers spinning in the air, the stands in the background, and to the characteristic oscillating motion of the diving board after the divers jump.

In Figure 3.23 we show some of the unique Motion Words for the riding action. In most videos, the camera is close to the horseman and there is a lot of blur as the camera attempts to keep him in the center of the video. Nevertheless, we find many supervoxels, mainly on the back of the horse and on the saddle, spanning temporal frames in the range of 7 to $100+$, which are unique to this action class. In this case we considered Motion Words which match in both HOG and MBHX since most of the actions happen along the horizontal axis and we wish the supervoxels to match in appearance, too.

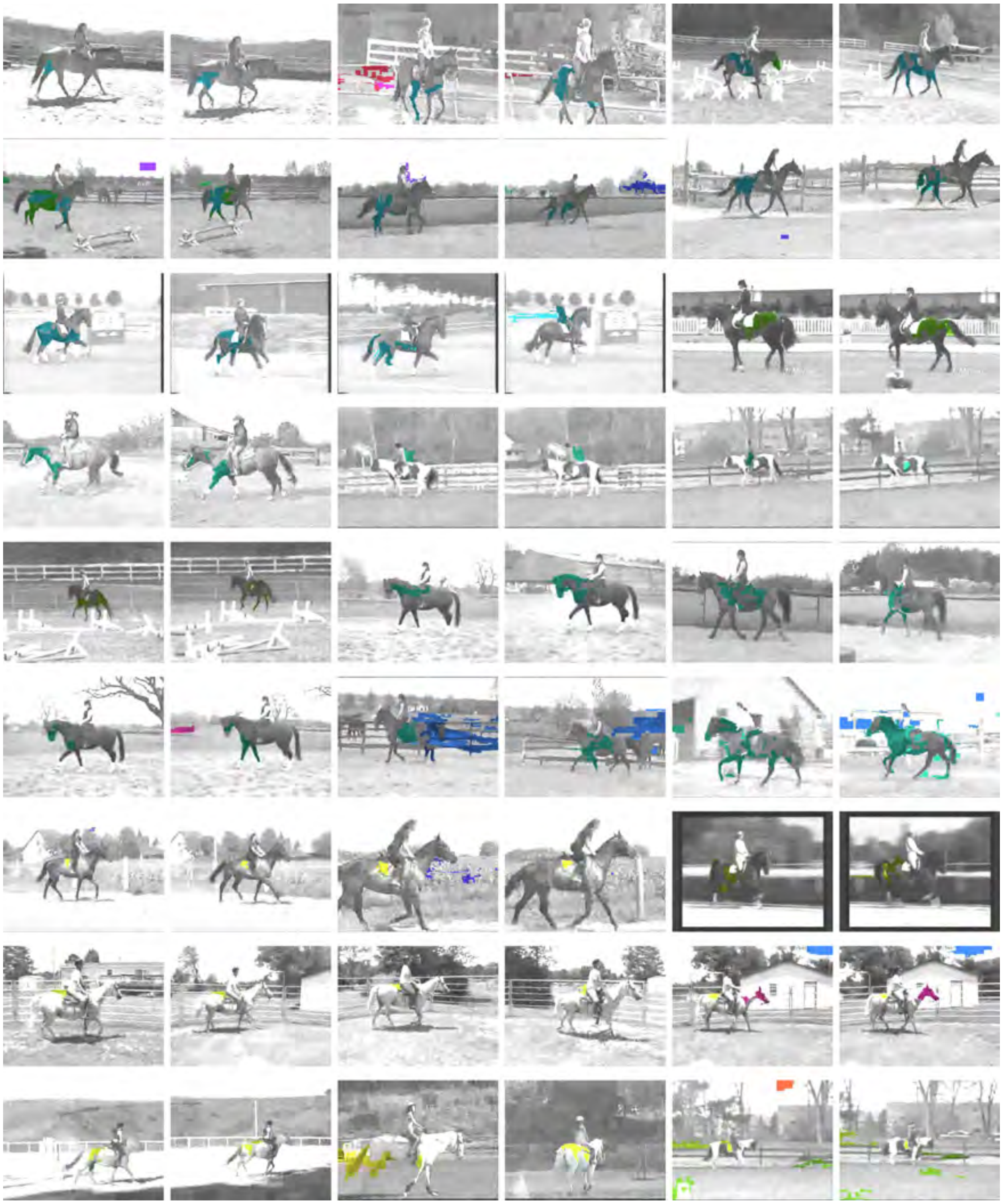


Figure 3.23: Even though the segmentation is often blocky due to the large amount of blur induced by the camera tracking the horsemen, we still find many unique Motion Words for the riding class, matching in both the HOG and the MBHX channels. In particular, most of the supervoxels are on the back of the horse or on the saddle. Each row displays frames from three different videos, with two frames per video. It is important to note that the supervoxels have a temporal span of 7 to 100+ frames.

3.9 Future work

3.9.1 Localization

Motion Words enable localization of regions in space and time. We have shown several ways of choosing which regions to localize - those common between two or more videos based on single low-level feature channels or their combinations, or those unique for a set of videos (for example, videos belonging to the same action class), or by manually picking one region and visualizing all others that encode to the same Motion Word. An additional and important application of the Bag of Motion Words representation is for action localization. That is, localizing the regions that correspond to the person performing the action. There are several approaches that can be taken to extend BMW for localization. Prior works generally use bounding box annotations on the training videos or person/objectness detectors to define multiple regions of interest, followed by pruning techniques [79, 80, 84]. In the case of Motion Words, such annotations can be intersected with the supervoxels in the videos to select the supervoxels to use for training. At testing time, all supervoxels will be considered and matched to the learned models. This method will not require special background pruning or selection algorithms. On the other hand, models can be trained on all supervoxels, but the Motion Words containing the annotated regions can be given a higher weight. This allows matching of the supervoxels in the testing video to all Motion Words for a more robust comparison.

3.9.2 Fisher vector encoding

Fisher vector encoding [68, 76, 101] and super vector encoding [14] have produced excellent results in image and video classification. The proposed Bag of Motion Words framework makes use of standard encoding strategies in two places - when encoding the low-level features to the BoW codebook, and then when encoding the motion features to the BMW codebook. In the experimental section we evaluated hard (standard) and soft encoding, showing that the former obtains better performance. In future work, we propose to evaluate more advanced encoding strategies like Fisher vector encoding.

3.10 Conclusion

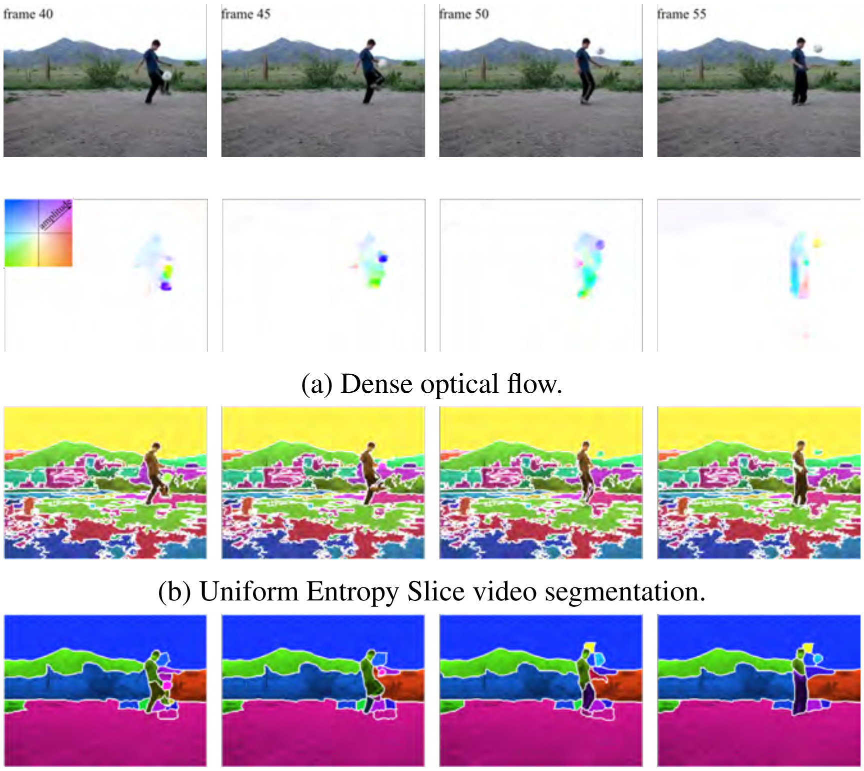
We propose a new video representation, Bag of Motion Words, which enables understanding of *why* videos are classified as similar, and achieves state-of-the-art performance in the tasks of activity classification and retrieval. The representations builds upon the popular and successful Bag of Words framework. Rather than pool features over cuboids of manually predetermined spatial and temporal sizes, we enable more flexible spatio-temporal support by pooling features over supervoxels. However, each video can have a different number of supervoxels, and thus, a different number of supervoxel-based features. To allow fast and robust comparison of videos, we propose a second quantization step which makes it possible to construct a robust and compact video descriptor. We show that this representation is well-suited for activity classification and retrieval, achieving state-of-the-art performance. We rigorously evaluate the design choices involved in the proposed representation, and show that highest improvement in performance is achieved when we encode Dense Trajectories to large codebooks and pool over non-coarse supervoxels.

Furthermore, Motion Words enable interpretability of the results, giving us the power to gain understanding of which regions make videos similar. This key feature of the representation is important and useful in several cases. First, it enables us to interpret classification and retrieval failures, where we are usually limited to a single numerical value (e.g., the distance between two videos in a retrieval task). By visualizing the regions that are common between videos, we can gain insight whether the videos are indeed similar, or whether the classifier is not suitable for the task. Second, we can easily visualize specific types of regions across all videos, e.g., a manually selected region corresponding to an object of interest. Third, the representation's sparsity properties capture regions that are unique per action class. Since we can interpret these regions, we gain an understanding of the distinctive features learned for each class.

Chapter 4

Globally Consistent Supervoxels

4.1 Introduction



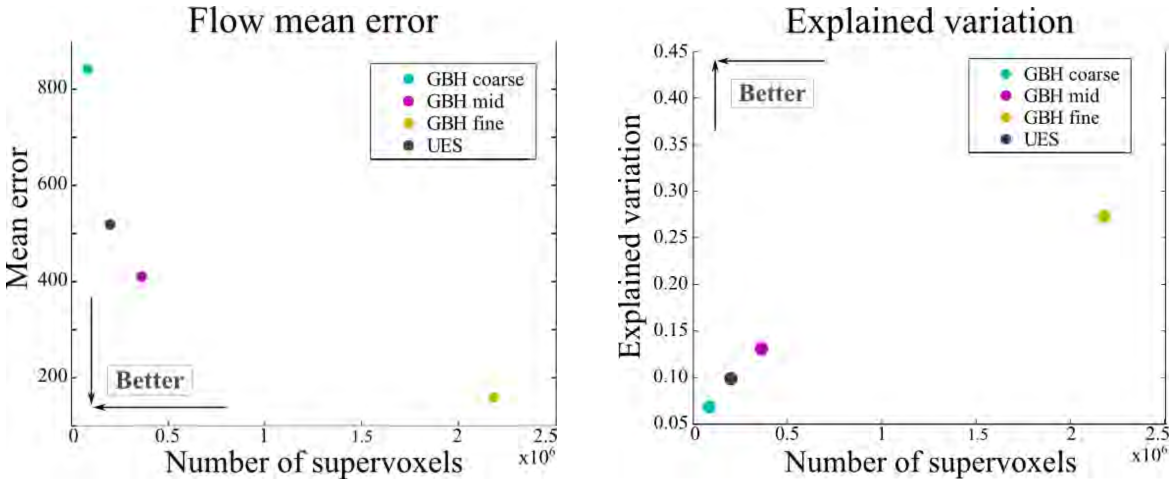
(a) Dense optical flow.

(b) Uniform Entropy Slice video segmentation.

(c) Supervoxel segmentation which emphasizes motion boundaries.

Figure 4.1: Frames from a juggling action, with their corresponding dense optical flow, show the soccer ball has motion different from its surroundings. However, state-of-the-art video segmentation methods, e.g. UES [104], group the ball with the background (third row). In the context of Motion Words, we seek segments that emphasize and respect motion boundaries as shown in the bottom row.

In Chapter 3, we showed that pooling features over supervoxels improves classification per-



(a)

(b)

Figure 4.2: Mean optical flow magnitude error 4.2a and explained variation 4.2b for four segmentations on all 1160 videos from the YouTube dataset. The difference between the oversegmentation (GBH3) and larger regions suggests that supervoxels generated by higher levels of hierarchy or those that optimize for "motionness" in UES, do not approximate and explain flow well. We hypothesize this implies that larger supervoxels group pixels with different motion and thus violate motion boundaries.

formance compared to pooling over rigid cuboids. Our evaluation of state-of-the-art video segmentation methods shows that fine supervoxels are better suited for Motion Words compared to coarse supervoxels. To better understand this difference in performance, we examine the supervoxel properties these methods seek to encode. One of the goals of video segmentation is to recover object boundaries. This often leads to grouping pixels with very different motion, especially if we require a segmentation into large segments. We hypothesize that regions which span motion boundaries are less suited for Motion Words compared to smaller regions which respect motion boundaries. This hypothesis is based on the better performance obtained when using small supervoxels, which intrinsically capture fine motion boundaries.

We can examine quantitatively how well a segmentation approximates the actual dense optical flow. We compute the difference between the actual flow and the mean flow in each supervoxel. Every pixel is assigned the mean flow magnitude in the supervoxel it belongs to. We report the mean squared error, that is, how well the average flow approximates the actual dense flow per pixel. In Figure 4.2 we show evaluation of four segmentations for all 1160 videos in the YouTube dataset. GBH1 is the highest level in the hierarchy, corresponding to a coarse segmentation, GBH2 is the mid-level of the hierarchy, GBH3 is an oversegmentation (lowest level) and UES is a segmentation into supervoxels, optimizing for "motionness." The x-axis is the total number of supervoxels. The oversegmentation, GBH3, which generates the highest number of supervoxels, has smallest error. This is expected, since, as the supervoxels become smaller and approach single pixels, the error decreases. The difference in mean error between the segmentations suggests that the higher-level supervoxels do not approximate the flow well. We hypothesize this implies that the larger supervoxels do not capture motion boundaries well.

Superpixel methods aim to oversegment object boundaries, decomposing large regions into smaller building blocks coherent in appearance. One of the goals of these methods is to respect object boundaries, that is, to seek a segmentation where pixels belonging to different objects have different segment ids. We propose a similar approach to supervoxel segmentation, namely, to oversegment spatio-temporal regions into smaller building blocks coherent in motion. We seek to preserve motion boundaries, that is, to assign different supervoxel ids to pixels which have different motion in the same frame. For example, in the GBH segmentation shown in Figure 4.4c, the pixels corresponding to the diver's torso are grouped with the pixels corresponding to the background into the supervoxel labeled A. However, these two sets of pixels have very different motion in that frame, t, as seen by the dense optical flow, shown in Figure 4.4b. Observe that supervoxel A groups pixels across a motion boundary that exists in frame t. Instead, we seek finer supervoxels which respect such motion boundaries (e.g., Figure 4.4d).

We can also examine motion boundary preservation qualitatively. In Figure 4.1 we present an example of regions which span motion boundaries, where we show several frames from a soccer juggling action, along with the corresponding dense optical flow. We see that the soccer ball has distinctly different motion from its surrounding pixels (second row). The third row shows the output from the state-of-the-art Uniform Entropy Slice (UES) video segmentation method [104]. The UES method automatically selects supervoxels from a hierarchical segmentation to balance the relative information based on a specified post hoc feature criterion. For example, the method supports "motionness," which selects larger supervoxels with less relative motion from coarser levels, and smaller supervoxels with more relative motion from finer-levels. However, in many cases, especially in videos with large motion, accurately recovering motion boundaries is a challenging task. For instance, even though the ball is its own video segment in frame 40 in Figure 4.1, it is grouped with the grass in frame 45, then with the mountains in frames 50 and 55, even when those segments become disjoint. While segmentation methods attempt to preserve motion boundaries, they have several limitations which need to be addressed for applications requiring good boundaries, e.g. Motion Words.

First, pixel-based methods, e.g. GBH [105], UES [104], [37], are significantly affected by noise from the underlying motion and appearance cues. This leads to drifting pixels, bleeding

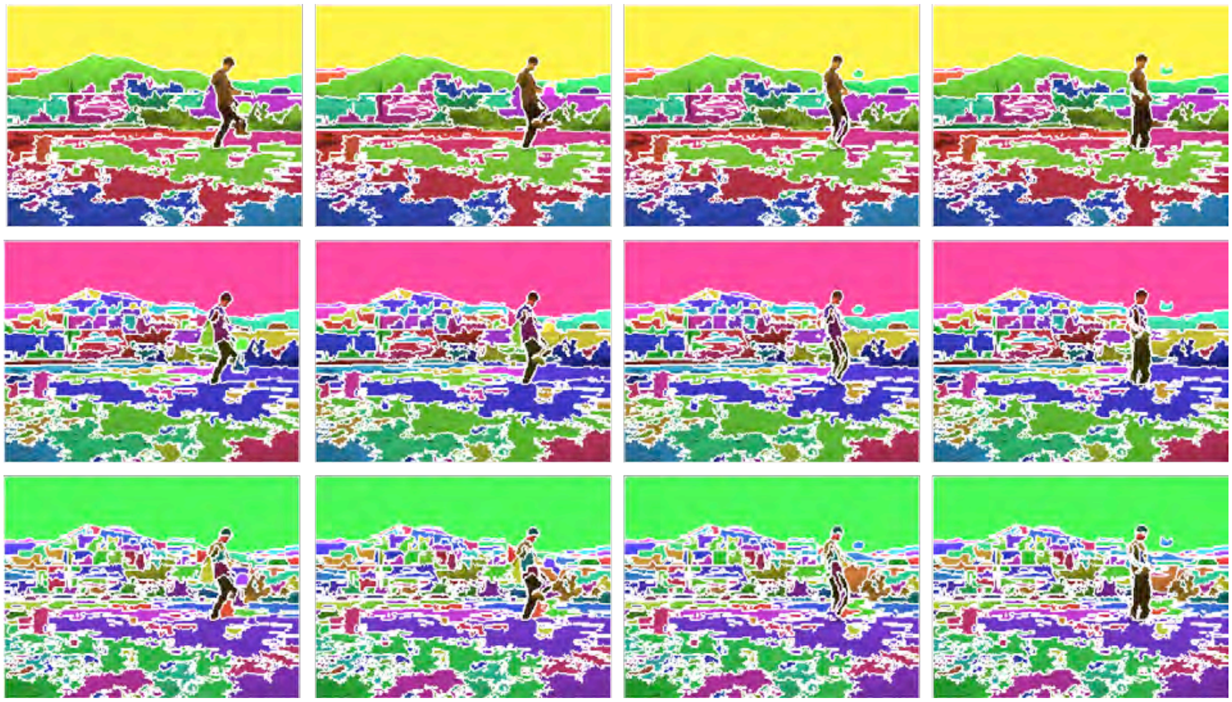
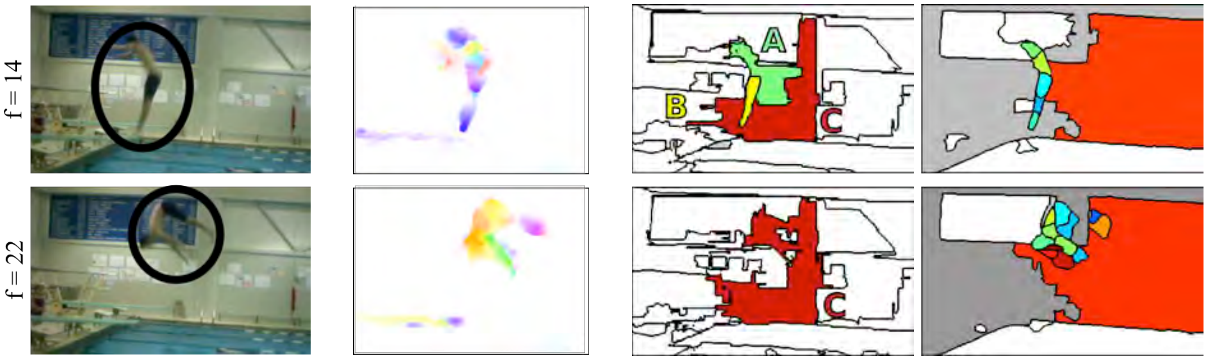


Figure 4.3: Three levels of segmentation hierarchy generated by the streaming GBH [105] algorithm. The top row shows the highest, most coarse level, while the bottom row shows the finest level of segmentation.

edges and to grouping pixels in one frame with pixels with distinctly different motion and appearance in subsequent frames. Furthermore, it leads to regions that are unstable in consecutive frames and thus lack spatiotemporal uniformity. In the example in Figure 4.1, the regions corresponding to the grass and bushes are finely segmented, leading to unstable and disjoint segments in future frames.

Second, failure to preserve motion boundaries at the finest level renders methods like UES [104], which rely on supervoxel hierarchies, also unable to capture these boundaries. This is the case even if we want to put more emphasis on motion, for example by optimizing using the "motionness" objective in the case of UES. In Figure 4.3 we visually inspect the Graph-based Hierarchical (GBH) streaming segmentation used by several methods that operate on supervoxel trees [41, 104]. At the finest level of the hierarchy, the motion boundary around the supervoxel corresponding to the moving soccer ball is preserved in frames 40 and 45. However, in the subsequent frames, the pixels corresponding to the ball are grouped with the stationary pixels of the background, spanning across motion boundaries. Subsequently, no matter which hierarchy level is chosen by the UES optimization, it will be unable to recover the motion boundaries.

Third, even though widely used, segmentation methods that aim to optimize for a specific



(a) Diving action. (b) Dense flow [9] (c) sGBH [105]. (d) Desired supervoxels.

Figure 4.4: Due to temporal quantization in the streaming GBH segmentation [105], the regions of the diver and the background are merged, even though they have different motion in the current and the previous frames.

energy function face critical computational limitations. The large number of pixels in a video renders optimization over all frames infeasible, making temporal quantization a necessity. To obtain an approximate solution, algorithms optimize over small overlapping sliding temporal windows (usually 10 frames). However, this procedure hinders the ability to capture global spatio-temporal relationships between supervoxels. Consider the following scenario: supervoxel i translates to the left during time interval $t_1 \in [1, 15]$, while its neighboring supervoxel, j remains stationary. Both i and j are then stationary for $t_2 > 15$ (and still neighbors). Temporal quantization will cause i and j to be grouped together for frames $t > 15$, losing the information about the initial different motion of the superpixels. For example, in Figure 4.4 we show how the pixels corresponding to the diver are merged with those of the background when the diver reaches the highest point of the jump and is nearly stationary for several frames. This leads to the diver being merged with the background in subsequent frames, when the diver has a very different motion relative to the background. If we could propagate the information that the regions of the diver and the background had different motion throughout all video frames, this undesirable artifact can be eliminated.

In the context of Motion Words, we hypothesize that it is important for regions to preserve motion boundaries. In this chapter we review different types of video segmentation methods in terms of the properties they are designed to encode. We refer to them as "current supervoxel properties." We then compare these properties with the ones we seek to encode when computing Motion Words, i.e., "desired supervoxel properties."

We propose Globally Consistent Supervoxels (GCS), where, rather than consider single pixels, we use superpixels as the smallest building blocks. We group these superpixels into supervoxels based on their appearance, motion, and their spatio-temporal relationships across all video frames, which ensures *global* consistency. We show that these supervoxels better encode the desired properties, are better suited for Motion Words, and produce visually suitable segments. Finally, we evaluate the proposed segmentation on motion boundary preserving properties on both a standard segmentation benchmark and the YouTube action dataset. Finally, we present qualitative comparison with state-of-the-art segmentation methods.

4.2 Related work

Most methods for video segmentation are approaches for still images that have been extended to video. Here we review the approaches most closely related to this part of the thesis work. They fall into three main categories: methods that pose the problem as an energy minimization function over motion and appearance cues per pixel, methods which group superpixels from an already existing hierarchical supervoxel tree segmentation, and methods which cluster trajectories.

Energy-based methods. For example, the method of Felzenszwalb and Huttenlocher [30] builds a graph based on color and is one of the most popular for image segmentation. It has been successfully extended for video by Xu and Corso [105]. They implement the Felzenszwalb and Huttenlocher method directly on the 3D video voxel graph, by building a graph over the spatiotemporal volume, in which voxels are nodes connected with 26 neighbors in 3D space-time [105]. Grundmann [37] proposes a hierarchical graph-based segmentation which extends this method. Their algorithm builds on an oversegmentation of the original image segmentation method. They iteratively construct a region graph over the obtained segmentation and for a bottom-up hierarchical tree strictre of the region graphs. They merge regions based on the χ^2 distance of *Lab* histograms. In addition, they allow a selection of the desired segmentation level, which is more intuitive than selecting a threshold to control region size, as it is done in the original image segmentation. Xu *et al.* [105] further create a streaming version that is computationally much more efficient and can be applied to videos with larger number of frames.

Meanshift methods, first introduced by Fukunaga and Hostetler [34], are mode-seeking methods. For example, Comaniciu and Meer [17] and Wang *et al.* [99] adapt the meanshift kernel to the local structure of the feature points, which improves results at the expense of computing time. Streaming approaches [21, 69] improve efficiency based on the ubiquitous use of the Gaussian kernel. Paris and Durand [70] interpret mean shift as a topological decomposition of the feature space into density modes and create a hierarchical segmentation by using topological persistence.

Their algorithm is more efficient than previous works especially on videos and large images.

Normalized Cuts [83] is a widely used graph partitioning criterion for image segmentation, which requires solving a generalized eigenvalue problem. To extend it for video, Fowlkes *et al.* [31] use the Nyström approximation to solve the above eigenproblem, which works well for short, low-resolution videos. Segmentation by weighted aggregation (SWA) [81] is an alternative approach to optimizing the normalized cut criterion that computes a hierarchy of sequentially coarser segmentations.

In recent work on video segmentation, Brendel and Todorovic [8] attempt to segment the video into spatiotemporal tubes, which represent spatiotemporal extents of moving objects. They build a simple, blocky segmenter, which uses compression error to split and then merge image regions (with a minimum threshold of 10 frames) based on HSV color values and Lucas-Kanade optical flow [58]. Other methods attempt to segment foreground objects while avoiding over segmentation [53]. Such pixel-based algorithms allow every adjoining pixel to have a different label, which introduces spatial and temporal artifacts, like bleeding pixels and the need for temporal quantization.

Supervoxel hierarchy methods. On the other hand, several methods have been developed that operate on supervoxel tree hierarchies. For example, the recently proposed Uniform Entropy Slice (UES) video segmentation method [104] automatically selects supervoxels from a hierarchical segmentation to balance the relative information based on a specified post hoc feature criterion. For example, the method supports "motionness," which selects larger supervoxels with less relative motion from coarser levels, and smaller supervoxels with more relative motion from finer-levels. Another work which is based on the hierarchical abstraction of the supervoxel graph is that of Jain *et al.* [41]. They propose an exact, general and efficient coarse-to-fine energy minimization strategy for video segmentation. They build upon the observation that the space of coherent labelings is significantly smaller than the space of all possible labelings. In related work, the Pylon model [54] and associative hierarchical CRFs [48] define a hierarchical cost function over supervoxels at all levels and thus solve a multilayer optimization problem. Van den Bergh *et al.* [24] propose a method for online extraction of video superpixels, defining an objective function which prefers superpixels with homogeneous color. The optimization is based on iteratively refining partitions by exchanging small blocks of pixels (e.g., 2×2) between superpixels. They use objectness, originally proposed for images [2], to generate tubes of bounding boxes throughout extended time intervals.

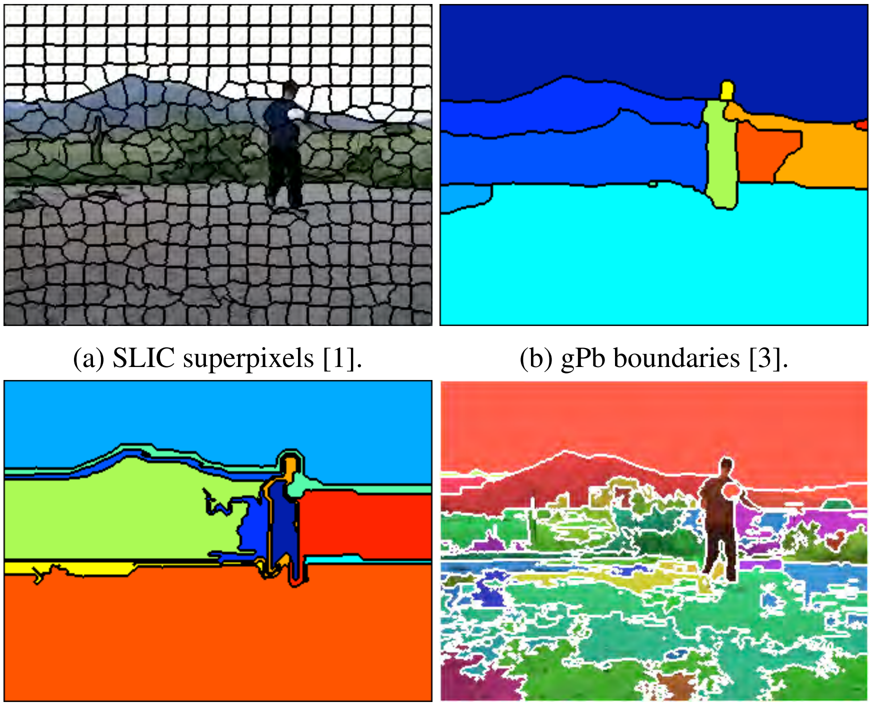
Trajectory-based methods. To encode long-range motion cues other works use clusters of long trajectories [10, 55]. However, these methods use long-range trajectories which are sparse and ignore background motion (which is often very informative). Further, the methods rely heavily on the underlying track clustering function. Lezama *et al.* [55] augment trajectories with local image information and seek a segmentation which respects object boundaries and associates these objects across frames. Raptis [71] also use clusters of long-term point trajectories, but they require bounding boxes annotation and assume fixed number of parts (clusters).

Videos recorded in unconstrained environments contain a large amount of camera motion, lightning changes, and jitter. Extracting robust pixel-based appearance and motion cues in such poses an extensive challenge. The noise in the underlying features, the temporal quantization required for energy optimization methods and the inherent reliance on the underlying supervoxels in a hierarchy graph present artifacts that are undesirable in the context of Motion Words.

The third contribution of this thesis is a supervoxel segmentation, Globally Consistent Supervoxels (GCS), which puts more emphasis on motion preserving properties. The segmentation is general, that is, it can incorporate a variety of user-defined cues to construct the supervoxels. It does not impose a limit to the number of supervoxels generated (and thus it does not require a priori knowledge of the number of segments). It operates bottom-up and top-down and thus it can recover from drifts and noise. We show that GCS encodes motion boundary preservation properties better than two state-of-the-art video segmentation methods, provides better spatio-temporal support for Motion Words, and produces visually meaningful segments.

4.3 Common supervoxel properties

Supervoxel methods have a great potential as a pre-processing step for video analysis, similarly to superpixel segmentation for images. Indeed, many of the properties considered important for supervoxel segmentation are inspired by properties desirable for superpixels [65], for example, coherence, locality and compactness. The ultimate goal of image and video segmentations is to respect object boundaries, which implies preserving appearance boundaries. Therefore, methods place more emphasis on keeping regions of different appearance separate, and less emphasis on separating regions that move differently. The recent evaluation of supervoxel methods [105] proposes that good supervoxels should respect appearance boundaries, encode spatiotemporal uniformity, should align with spatiotemporal boundaries and preserve object boundaries, should be parsimonious, and oversegmentation should not reduce the achievable performance of the application. In this section we review and consider these properties in the context of Motion Words.


(a) SLIC superpixels [1].

(b) gPb boundaries [3].

(c) FH color segmentation [30].

(d) UES video segmentation [104].

Figure 4.5: Current image and video segmentation methods can adequately recover appearance boundaries at different levels of granularity.

Tremendous progress in image and video segmentation methods have resulted in superpixels and supervoxels that successfully respect **appearance boundaries** for a variety of tasks, ranging from very fine edges, to only strong edges. For example, in Figure 4.5 we visualize several static superpixel segmentations from state-of-the-art methods applied to a video frame with relative minor motion and blur. Superpixel lattices oversegment an image into regions that conform to a regular grid [65]. In Figure 4.5a we show the output of the Simple Linear Iterative Clustering (SLIC) method, which clusters pixels in the combined five-dimensional color and image plane space to efficiently generate compact, nearly uniform superpixels. Such methods produce regions that respect very fine appearance boundaries. On the other hand, boundary detection methods seek to find count ours in the image plane that represent a change in pixel ownership from one object to another. The globalized probability of boundary (gPb) [3] method uses both local and global image cues and it is very successful at recovering object boundaries in images (Figure 4.5b). The popular color-based image segmentation of Felzenszwalb and Huttenlocher [30] can segment the image into mid-sized regions coherent in color (Figure 4.5c). Similarly, video segmentations can also adequately recover object boundaries (Figure 4.5d).

Spatiotemporal uniformity for supervoxels [105], is a property that encourages compact and uniformly shaped supervoxels in spacetime. It is analogous to the conservatism property for superpixels [65], which seeks segments in a superpixel lattice to be of regular size and to never greedily select huge image regions. In videos, this property embodies many of the basic Gestalt principles, e.g. proximity, continuation, closure, and symmetry, and helps simplify computation in later stages [105].

The superpixel property for **consistent spatial relations** [65], which enforces consistent and unambiguous spatial relationship between superpixels, has not yet been considered in supervoxel segmentation. We propose an equivalent property for supervoxels, **consistent global motion relations,** to require that if two spatially neighboring supervoxels have different motion in any frame f, they should always be labeled with different supervoxel ids (Figure 4.6).

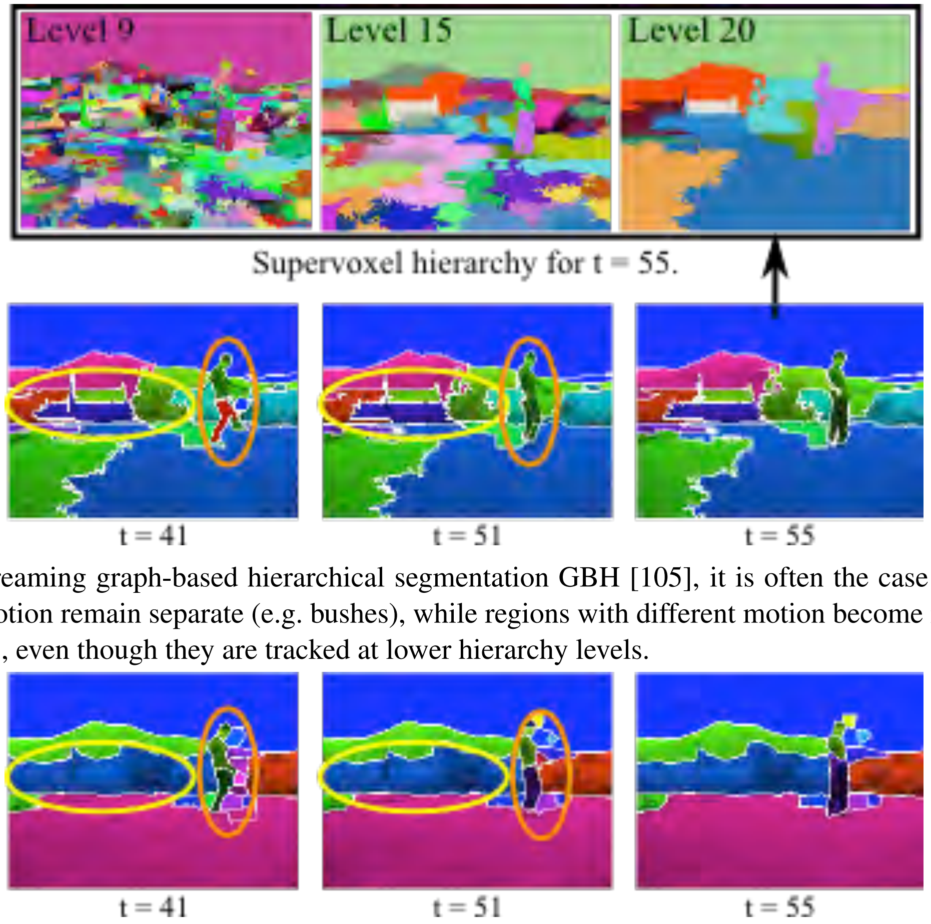
The **spatiotemporal boundaries and preservation** criterion states that supervoxel boundaries should align with object/region boundaries when they are present and the supervoxel boundaries should be stable when they are not present [105]. Furthermore, each supervoxel should overlap with only one object [105], and encourage a high degree of explained variation, a metric originally used to evaluate superpixels [65]. In the context of Motion Words, it is important to also extend this property to the motion of the supervoxels. That is, we seek to encode **motion boundaries and preservation**, where supervoxels should obey motion boundaries when they are present, and should be stable and consistent when they are not present. This criterion also aligns with the consistent temporal relations property.

While we seek supervoxels that respect object boundaries, it is also important that the number of supervoxels is not prohibitive, i.e., we seek a **parsimonious** segmentation. That is, we wish to encode these desirable properties with as few supervoxels as possible.

In the next section we discuss the above supervoxel properties in the context of Motion Words. We deem some of the properties less important, but we seek to place more emphasis on others. To achieve this, we define additional criteria that we wish to encode in a supervoxel segmentation and we ground our design choices in the context of these new properties.

4.4 Desired supervoxel properties

While successful for many tasks, video segmentation algorithms make a trade off between respecting object boundaries and respecting motion boundaries, giving more emphasis on the former. However, when motion boundaries are of higher importance for the task, it is essential to be able to put more emphasis on motion preservation. For example, to capture the characteristic



(a) In the streaming graph-based hierarchical segmentation GBH [105], it is often the case that regions of similar motion remain separate (e.g. bushes), while regions with different motion become merged (e.g. ball and sky), even though they are tracked at lower hierarchy levels.

(b) Instead, to have a parsimonious segmentation that respects fine motion boundaries, we want to merge neighboring regions with same global motion, and keep regions with different motion always separate.

Figure 4.6: We wish to give more emphasis on preserving fine motion boundaries without over-segmentation in appearance.

motion of the legs and arms of the soccer player in Figure 4.6 we need to over-segment regions with similar appearance (the body). The legs and the torso of the soccer player have different motion in frame $f = 41$, thus we want them to remain in different segments as long as we know their corresponding regions across time. On the other hand, there are many neighboring regions with different appearance (e.g. bushes) which never move apart since they have similar motion throughout the entire video, that is, they have similar *global* motion. Such regions do not contribute new motion information and grouping them will be beneficial in several ways. First, it will lead to a more parsimonious representation, and thus to more efficient computation. Second, the larger regions will be more robust to drifting and bleeding pixels, more uniform, and consequently help analysis and association in subsequent frames.

We seek to oversegment the video in terms of the motion cues while putting less emphasis on respecting fine appearance boundaries. Existing segmentation algorithms optimize at the pixel level for both appearance and motion. Most methods allow for adjusting parameters to define the level of segmentation. For instance, we can obtain a range of segmentations from many small regions to a few large ones (Figure 4.3). Nevertheless, we are currently unable to tune parameters to give more emphasis on preserving motion boundaries without this resulting in an over-segmentation in appearance.

In the example in Figure 4.6, the region corresponding to the soccer ball is successfully tracked in the finest level of the GBH segmentation hierarchy over the first two frames. However, this level of the hierarchy produces an extreme over-segmentation in appearance, which is undesirable. Instead, we want regions with similar motion (e.g. grass) to be grouped together. If we choose a higher segmentation hierarchy level, the optimization forces regions of different motion to be grouped together (e.g. the ball and the background). The recently introduced Uniform Entropy Slice [104] method selects supervoxels across hierarchies, optimizing for a user defined property, e.g. "motionness." However, such approaches are limited by the motion boundary preservation properties of the method generating the supervoxel hierarchy. Thus, when the underlying method fails to preserve motion boundaries, so do methods that operate on the hierarchy.

We illustrate this common problem with another example. In Figure 4.4 we show two frames from a diving action, half a second apart, and their corresponding dense optical flow. As seen by the motion cues at time $t = 14$, the segments corresponding to the diver (denoted by A and B), have a very dissimilar motion compared to the remaining regions. In particular, the motion is distinctively different from the region corresponding to the stationary wall in the background, (labeled as C). However, the streaming GBH algorithm groups the three regions at time $t = 22$, failing to maintain these important motion boundaries. In contrast, we want to retain this information and oversegment regions in terms of their motion cues (Figure 4.4d).

For computational feasibility purposes, graph-based methods require temporal quantization, which introduces undesirable artifacts. These methods process a small, local neighborhood of frames at a time (typically 10 frames), and can therefore capture only the local spatiotemporal relations between pixels (Figure 4.8). This temporal quantization forfeits many useful *global* properties like encoding that an object was in motion in the past, and should not be merged with the background, even if the object is stationary in the current neighborhood of frames. To illustrate the properties we would like to capture, we show several examples, where we pick one segment id and visualize all pixels which have been labeled with this id across frames. We include the remaining supervoxel boundaries in black for reference. In Figures 4.7, 4.9 and 4.10

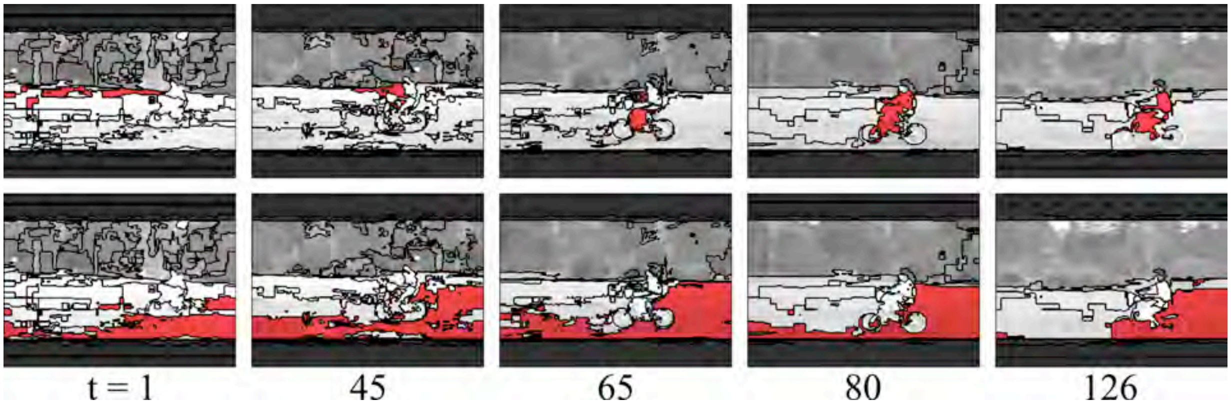


Figure 4.7: We visualize one supervoxel from the streaming GBH segmentation [105], colored in red, across time. We include the remaining supervoxel boundaries in black for reference. We notice "bleeding" of pixels from regions of one type of motion and appearance (the background moving to the right), into regions of very different characteristics (the biker, moving to the left).

we show supervoxels which start over one region with characteristic motion, but then get "picked up" (grouped) by regions with very different motion; regions which erroneously incorporate other regions of different motion and are unable to recover from this merge in subsequent frames; and regions that become disjoint, with different appearance and motion, but are still under the same supervoxel label. Instead, we want to ensure that supervoxels with different motion at any time will always be kept separate, thus encoding spatiotemporal relationships over all frames of the video (i.e., propagating this information globally). For example, in Figure 4.6, the legs and the torso of the soccer player have different motion in at least one frame, namely $f = 41$. We want to encode that these two regions have had different motion in subsequent frames, even if they have identical motion in the following frames. We want to assign different supervoxel ids in all frames for which we know their correspondences.

Finally, similarly to Lezama *et al.* [55], who over segment the video into regions that only respect, but do not necessarily comprise object boundaries, we do not seek to segment complete objects. However, our goal differs in that we do not want to impose that every pixel within a supervoxel correspond to the same real-world object. Rather, we want every pixel to correspond to parts of spatially adjacent objects that always have the same *instantaneous motion*. This idea is inspired by Moore and Jones [65] who define a superpixel as a spatially-coherent, homogeneous, structure which preserves information over scales or sampling resolutions. In the construction of supervoxels, we propose to retain these spatial properties and extend them to include global motion cues.

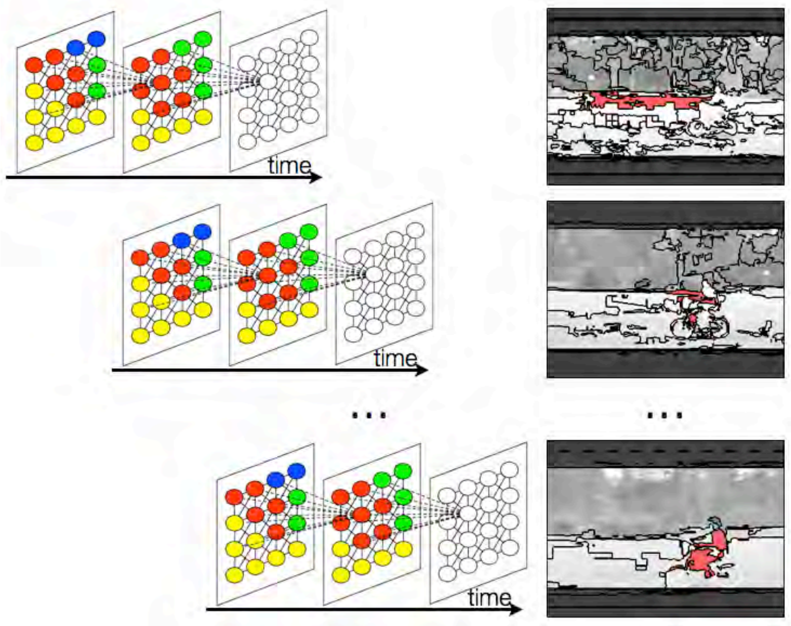


Figure 4.8: Graph-based methods require temporal quantization, which results in undesirable artifacts. For example, we cannot recover from mistakenly grouping pixels with different motion and we cannot use label information from earlier temporal windows.

4.5 Design choices

We seek a supervoxel segmentation that encodes the following properties:

P1. Robustness to noisy motion and appearance cues.

P2. Fine motion boundary preservation locally (per frame).

P3. Fine motion boundary preservation globally (taking into account all video frames).

P4. Consistent motion relationships between supervoxels.

P5. Parsimonious appearance boundaries.

To encode these properties, we need to take into account the *global* motion of supervoxels, that is, the relative motion of all pairs of adjacent supervoxels over all frames containing these supervoxels. Formulating an energy optimization over individual pixels will produce a very large graph that requires temporal quantization for computational reasons. With this technique we can examine only a small number of frames at a time, typically $7 - 10$. With such a small number of frames the algorithm can examine only a local neighborhood of pixels at a time. Thus, temporal quantization forfeits global motion information, i.e., information of how supervoxels relate to each other over all video frames. In contrast, we choose a rule-based method for reasoning about

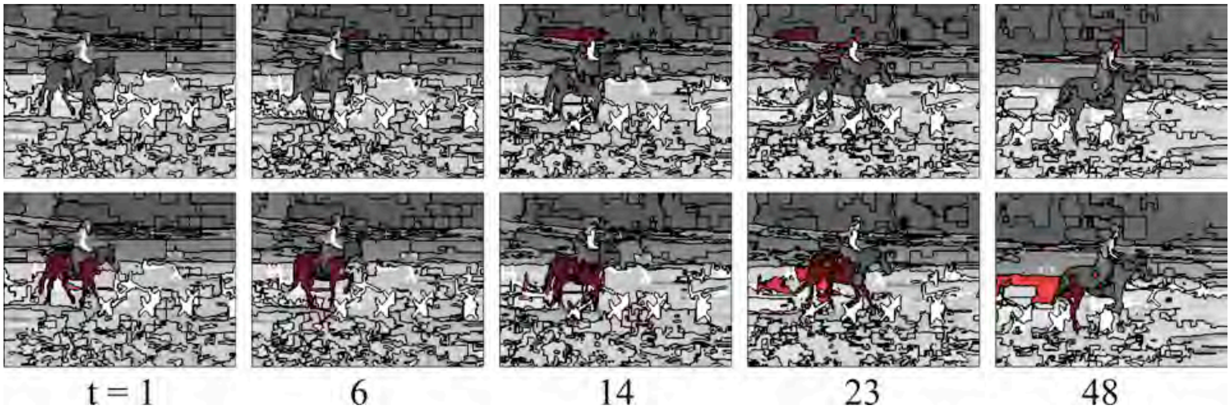


Figure 4.9: A common failure of the streaming GBH segmentation [105] are “picking up” are “leaving pixels behind” artifacts - the background superpixels become grouped with the head of the moving rider in the top row, while the pixels on the horse become grouped with the background in the bottom row.

groups of superpixels to construct supervoxels. That is, we do not formulate the segmentation problem as an energy minimization. Instead, we define a heuristic method for grouping, splitting and associating superpixels in space and time. This allows us to consider superpixels in terms of their relative *global* motion relationships and to construct Globally Consistent Supervoxels based on the superpixel associations.

The design choices for the proposed supervoxel segmentation are grounded based on the aforementioned properties. First, we present an overview of the design decisions we make to encode these properties, and in Section 4.6 we give a detailed description of the proposed Globally Consistent Supervoxels method.

P1. Robustness to noisy motion and appearance cues. First, to be robust to noisy motion and appearance cues, we chose to construct supervoxels using superpixel as the fundamental unit. That is, we use appearance and motion statistics computed over superpixels. This differs from graph-based methods which consider single pixels.

Second, we wish to be robust to the blur and illumination challenges arising from the varied, non-affine camera motion present in videos from unconstrained environments. Due to these various noise factors, no single granularity of appearance segmentations will suffice to capture object boundaries. While most segmentation methods use only one appearance cue, e.g. color, or color histograms (e.g., [8, 37, 105]), we propose to combine multiple appearance cues. We do so by intersecting multiple appearance-based segmentations and correcting for small regions.

P2. Fine motion boundary preservation locally (per frame). It is essential that the input superpixels respect fine motion boundaries in the local sense, that is, the boundaries defined by

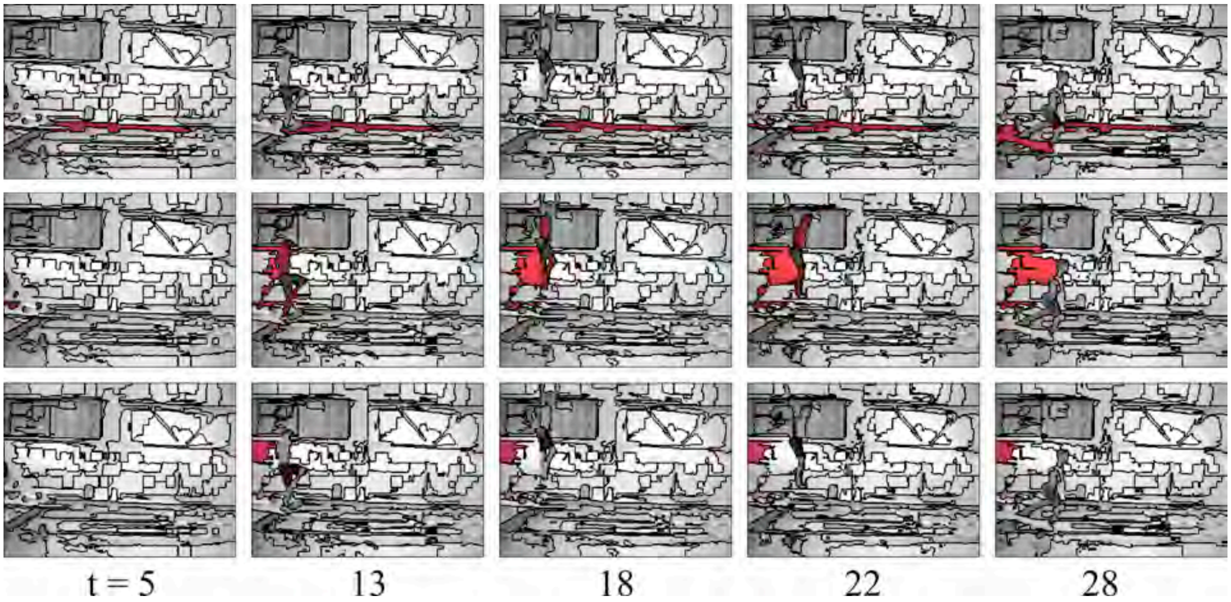


Figure 4.10: The "bleeding" and "picked up" pixel artifacts of the streaming GBH segmentation [105] are also common in videos recorded from a static camera. Pixels corresponding to the static background, the jumping diver, the moving diving board become grouped, even though they have very different motion and appearance.

the motion in the current frame. The most popular and successful motion cue is dense optical flow. The flow in one frame is computed based on two frames, the current and the succeeding frame. Here, we use the term "local motion" to denote the flow in a single frame, computed in this standard method, that is, using only the current and the successor frame.

Image segmentation methods work well for static images. However, appearance cues are extremely affected by motion, which leads to blur. Consequently, in such cases appearance boundaries are hard to recover. It is thus crucial to augment static image appearance segmentation with motion cues. Given dense optical flow, we compute a motion boundary segmentation for each frame. We augment the superpixels, splitting them if needed, to ensure they respect these motion boundaries. At this step, an oversegmentation in motion is desirable, as these segments will form the finest granularity of motion boundaries that we will consider in the subsequent analysis. In Section 4.6 we discuss the benefit of oversegmenting in motion and show that even an extreme oversegmentation does not hinder subsequent analysis.

P3. Fine motion boundary preservation globally (taking into account all video frames). Energy minimization methods can handle only a small number of frames at a time, and thus render encoding global motion relationships between supervoxels unfeasible. We propose a simple, rule-based association approach to find superpixels correspondences from one frame to the next.

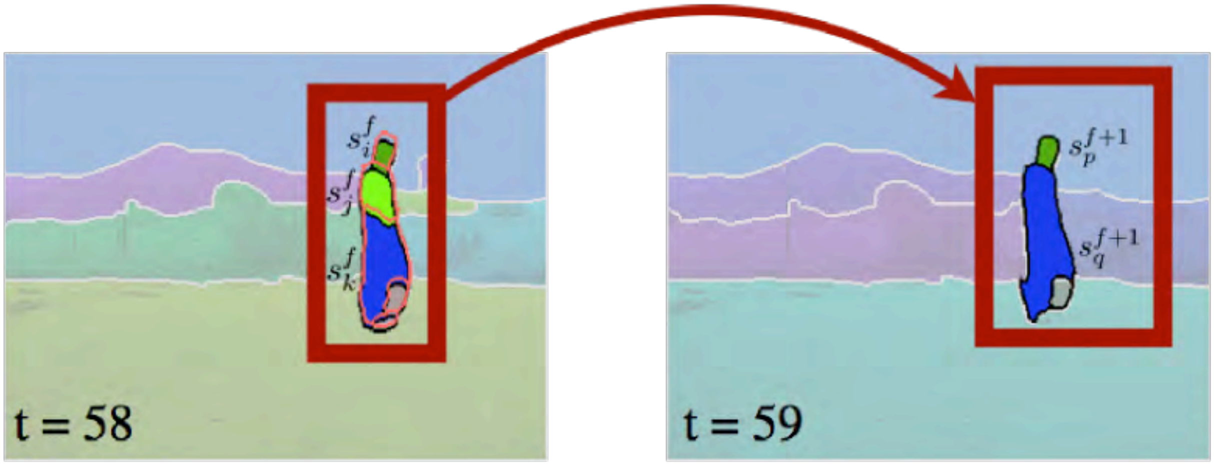


Figure 4.11: Given motion and appearance cues we define a set of rules for associating superpixels (described in Algorithm 2). Superpixel s_i^f is associated one-to-one with superpixel s_p^{f+1}. Both superpixels s_j^f and s_k^f are associated (in a many-to-one way) to superpixel s_q^{f+1}. To ensure consistent relationships between the pixels in the superpixels in f and those in $f + 1$, we will split the target superpixel, s_q^{f+1}.

The decision to associate superpixels is based on a cost function over the motion and appearance cues of the superpixels being considered. Starting with the first and second frame, we associate superpixels in each pair of frames. At the end of this process we will have correspondences across all video frames. Along with property **P4**, this ensures we encode global superpixel relationships. Furthermore, since the input superpixels respect fine motion boundaries, this property remains an invariant throughout this association process.

P4. Consistent motion relationships between supervoxels. Superpixels from a static image segmentation in one frame do not necessarily have a one-to-one mapping with the superpixels extracted in the consecutive video frame. Thus, we often have to handle cases where multiple superpixels with different motion descriptors are associated to one superpixel in the next frame, for example, as shown in Figure 4.11. If we simply allow the merge, we will violate the motion relationships between the pixels in frame t and those in frame $t + 1$. We wish to ensure that in $t + 1$ the pixels respect the motion boundaries present in t. That is, we want to propagate the information that the superpixels in t had different motion. To do so, we split the target superpixel in $t + 1$ using the superpixels in t and their motion information. Thus, we re-segment the target superpixel into regions which are consistent with the motion boundaries from frame t. This strategy also encodes property **P3**: motion boundaries that exist in one frame will be propagated in all subsequent frames, ensuring global motion boundary preservation.

P5. Parsimonious appearance boundaries. The previous properties, e.g., motion boundary preservation, require that we oversegment non-stationary regions. To obtain a parsimonious segmentation, we perform a smoothing step where we merge supervoxels with weak appearance boundaries which have similar motion. We merge only those supervoxels which are always spatio-temporal neighbors, with instantaneous motion (direction and magnitude) within a threshold of each other. Thus, for every pair of neighboring supervoxels and for every frame in which these supervoxels are present, we check if they are neighbors and if their relative motion is similar. If so, these supervoxels are merged.

Resolve many-to-many associations. We finalize the segmentation by resolving many-to-many associations. We consider the amount of overlap as we seek to create only one-to-one associations. If there is ambiguity and a many-to-many associations cannot be resolved, we terminate the supervoxels at frame t and assign new supervoxel ids to those in frame $t + 1$.

The Globally Consistent Supervoxels algorithm implements each of the above steps.

4.6 GCS Construction

We construct Globally Consistent Supervoxels with the following steps:

1. **Superpixels.** We generate a tree hierarchy of superpixels which respect strong appearance, fine motion boundaries, and encode properties **P1** (robustness to noisy motion and appearance cues) and **P2** (fine motion boundary preservation locally).
2. **Supervoxels.** For every set of corresponding superpixels in time, we assign an unique supervoxel id. During association of superpixels, if motion boundaries are violated, we split superpixels. This step encodes properties **P3** (fine motion boundary preservation globally) and **P4** (consistent motion relationships between supervoxels).
3. **Offline pass.** If segmenting in an offline fashion, repeat previous step starting at the last frame, and proceeding backwards, to ensure motion boundaries from $t + 1$ are preserved in t. This step encodes properties **P3** and **P4** in t by "peeking" into the future frames.
4. **Parsimony.** We perform a forward pass in which we merge supervoxels with similar motion, encoding property **P5** (parsimonious appearance boundaries).
5. **Globally Consistent Supervoxels.** In a forward pass, we finalize the segmentation by reducing many-to-many associations to one-to-one associations.

In this section we describe each step in details.

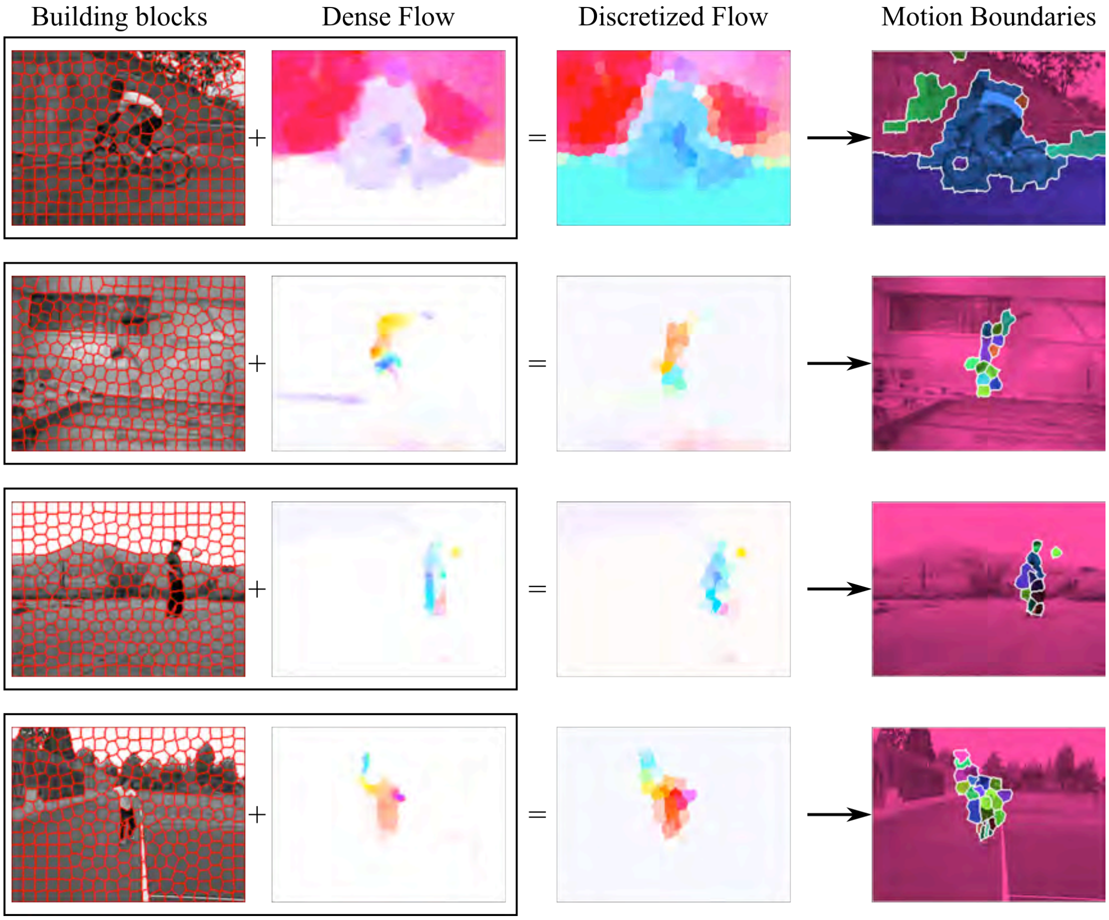


Figure 4.12: The basic building blocks, SLIC superpixels [1], are used to discretize the dense optical flow in each frame. The pixels in each SLIC superpixel are replaced with the mean flow vector for that region. Mean shift clustering on these flow vectors defines superpixels with similar (instantaneous) motion. These clusters define fine, local motion boundaries, i.e., motion boundaries which are agnostic to the motion of pixels in other video frames.

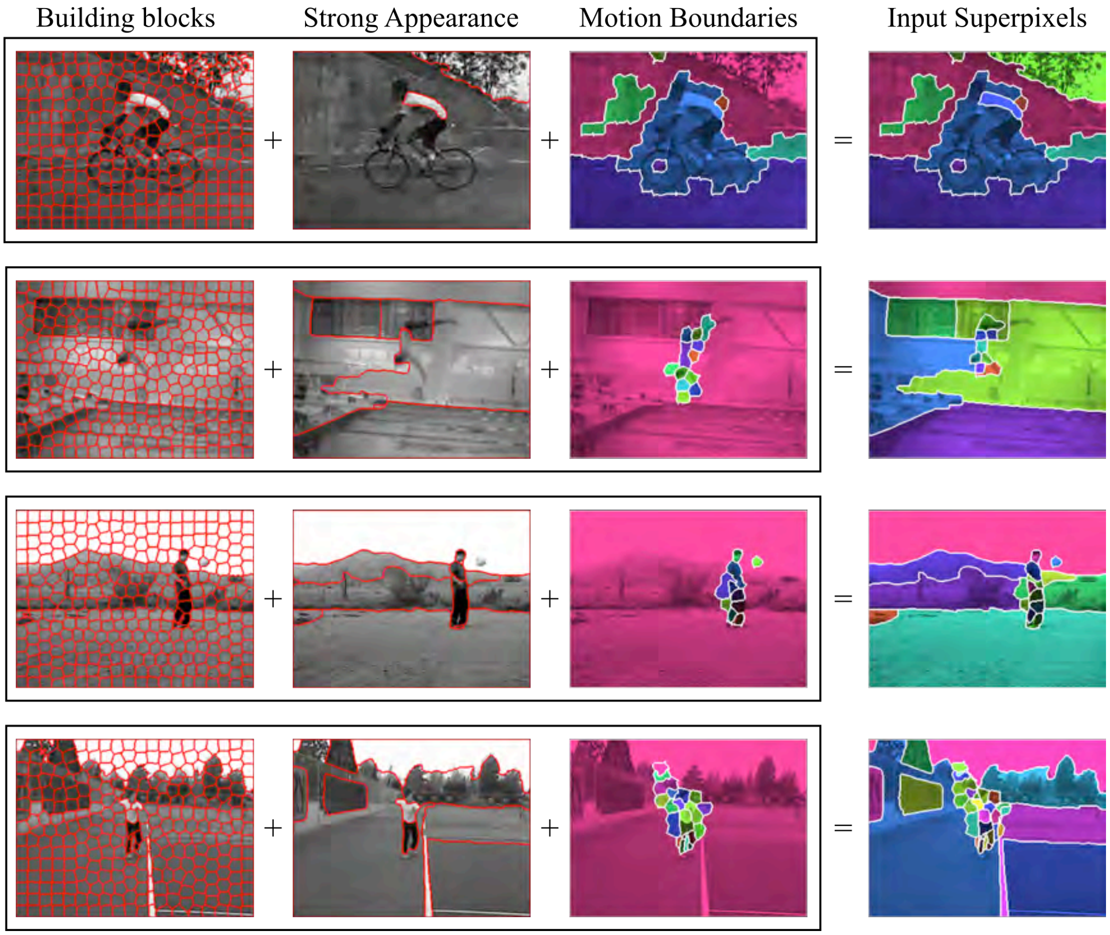


Figure 4.13: The basic building blocks, SLIC superpixels [1], the strong appearance boundaries produced by gPb [3], and the motion boundaries are combined to form input superpixels. Regions smaller than the basic building blocks are merged into larger superpixels. The resulting superpixels respect strong appearance boundaries and fine motion boundaries and are the input to the segmentation algorithm.

1. Superpixels. Instead of segmenting at the pixel level, we consider superpixels as the smallest units. Small, nearly uniform superpixels are well suited for encoding fine motion boundaries. On the other hand, larger superpixels are better for capturing strong appearance boundaries. For example, in Figure 4.5 we showed fine, coarse and mid-level image segmentations in a video frame with relatively low motion blur. The finest segmentation, the SLIC superpixels [1], do respect fine boundaries, but they present an extreme oversegmentation. Using such small regions for further analysis, e.g. association with superpixels in the next frame, will be very noisy and unstable. In contrast, the mid-sized color segmentation [30] captures larger regions. However, it relies heavily on color cues, leading to irregular, unstable, bleeding segments, e.g., regions around the person and the mountain. The gPb contour detector [3] has excellent performance in static images. However, when there is motion blur, segments often span object boundaries. Two such examples are shown in Figure 4.5 and Figure 4.13, where the quickly moving soccer ball is grouped with the background, the bike, road and sidewalk are grouped due to fast camera motion, respectively. In such cases contour detector produce undesirable undersegmentations.

We combine the benefits of both very fine, oversegmented regions, and coarse edges which respect strong object boundaries. We consider two levels of granularity. As the smallest building blocks we chose the fine, grid-like superpixels generated by SLIC (Simple Linear Iterative Clustering) [1]. The SLIC method clusters pixels in the combined five-dimensional color and image plane space and it efficiently generates compact, nearly uniform superpixels (Figure 4.13, first column). For the coarse level, we chose the robust contour segmentation generated by the gPb method [3].

The shapes of coarse superpixels varies greatly from one video frame to the next. Further, a coarse superpixel will not necessarily capture the motion boundaries from previous frames. To handle these cases, we want to be able to decompose a superpixel into its constituent building blocks. Thus, we intersect the SLIC superpixels with the gPb superpixels to produce a tree hierarchy of superpixels for each frame.

In the superpixel association step, we want to generate good hypotheses of superpixel correspondence in consecutive frames. We use appearance and spatial cues, where the latter include motion cues. Although color is a commonly used cue, its robustness in videos is hampered by the large amount of variations which cause the measured color values to vary significantly. A change in illuminant color, viewpoint, and camera motion, all influence the color values in a frame. We want an appearance cue which is robust to photometric changes and varying image quality. In the HSV color space, it is known that the hue becomes unstable near the grey axis. To this end,

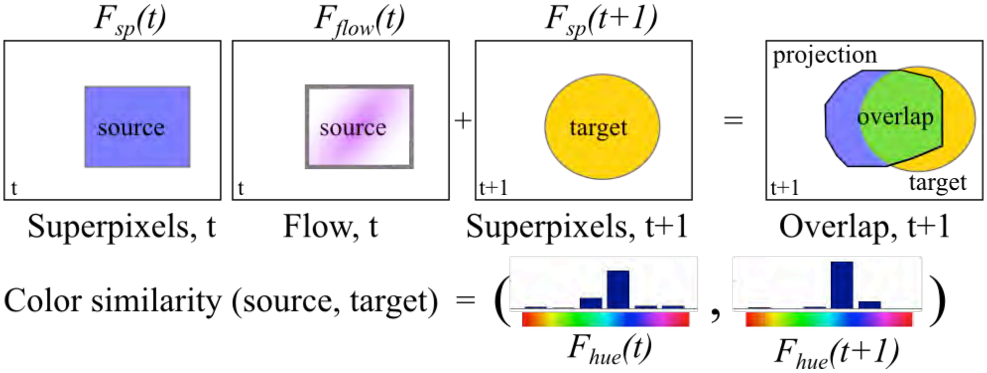


Figure 4.14: Inputs for basic superpixel association from frame t to frame $t + 1$: F_{sp}, the superpixel segmentation; $F_{flow}(t)$, the dense optical flow from t to $t + 1$; F_{hue}, the hue histograms, extracted densely at every 5th pixel.

Van de Weijer *et al.* [102] apply an error propagation analysis to the hue transformation. The analysis shows that the certainty of the hue is inversely proportional to the saturation. Therefore, the hue histogram is made more robust by weighing each sample of the hue by its saturation. The color model is scale-invariant and shift-invariant with respect to light intensity [94]. Based on the extensive analysis of color descriptors presented by van de Sande *et al.* [94], we choose the hue histogram descriptor, which is a good trade-off between efficiency in descriptor computation and fulfilling the above requirements.

For the spatial cues we extract dense optical flow to project superpixels from one frame to the next. We use the popular and successful dense optical flow implementation of Brox *et al.* [9]. To be robust to noisy flow estimation, we discretize the flow over the chosen building blocks, the SLIC superpixels. Given dense optical flow, we compute the average flow vector in each superpixel and apply mean shift clustering to discover groups of superpixels with similar flow vectors (Figure 4.12). The connected components of this segmentation form the motion boundary segmentation, $F_{\text{mb}}(t)$, for each frame t. To encode fine motion boundaries, we combine the tree hierarchical segmentation (SLIC combined with gPb) with the motion boundaries in the current frame. Regions smaller than the basic building blocks are merged into larger ones. The result is a superpixel segmentation, $F_{\text{sp}}(t)$, for each frame t, where the superpixels respect appearance boundaries (defined by gPb) and fine motion boundaries (defined by the dense flow clusters). These superpixels are the input to the next step in the algorithm (Figure 4.13).

2. Supervoxels. The input superpixels, $F_{\text{sp}}(t)$, form a static segmentation. For the first frame in a video, we assign supervoxel ids $F_{\text{sv}}(1)$ to each superpixel in $F_{\text{sp}}(1)$ based on the motion boundary segmentation, $F_{\text{mb}}(1)$. To construct supervoxels from superpixels, for all pairs of consecutive frames $(t, t+1)$, we propagate the supervoxel labels $F_{\text{sv}}(t)$ to the superpixels $F_{\text{sp}}(t+1)$ (Figure 4.18). Multiple spatially adjacent superpixels (forming a connected component) can have the same supervoxel id. To propagate the supervoxel labels, we find superpixel correspondences for all pairs of frames by defining an association cost based on overlap and appearance cues. It is important to note that we do not require exact, pixel-based correspondences. Instead, we generate hypotheses of corresponding superpixels for consecutive frames, as shown in Figure 4.19. We do not assume a predefined motion model, but rather, we use motion cues from the dense optical flow. In these hypotheses, we allow many-to-many associations, allow for superpixels to not be associated in the next frame ("lost"), and allow superpixels in $t+1$ to not be associated from t ("new"). We do not impose constraints on the number of associations, the number of "lost," or the number of "new" superpixels. Algorithm 2 presents the steps and conditions involved in constructing supervoxels.

An association of superpixels forms a hypothesis that the pixels they contain should belong to the same supervoxel. We create such hypotheses for a set of source superpixels in t to a set of target superpixels in $t+1$ based on the overlap and the hue histogram score. The overlap encodes the spatiotemporal relationships between superpixels, and the hue encodes the similarity in appearance.

The input to the superpixel association step is:

1. The superpixel segmentations, $F_{\text{sp}}(t), F_{\text{sp}}(t+1)$ (Figure 4.14).
2. The supervoxel labels, $F_{\text{sv}}(t)$.
3. The dense optical flow from t to $t{+}1$ and the inverse dense flow from $t{+}1$ to t (Figure 4.14).
4. Hue histogram descriptors [94], extracted densely at every 5th pixel (Figure 4.14).

To evaluate association hypotheses, we define several regions, shown in Figure 4.15. The region π_s^{t+1} is the projection of the superpixel s_i in t onto frame $t+1$ using the dense optical in the region. The region π_o^t is the projection of the overlapped superpixel in $t+1$, o_i, onto t using the inverse optical flow in $t+1$. We denote the intersections formed by the two projections with the original superpixels by $\pi_o^t \bigcap s_i$ and $\pi_s^{t+1} \bigcap o_i$. These regions are used to compute overlap scores, as shown in Figure 4.16. We define the "into score" to encode how well the source superpixel is explained by the overlapped superpixel. It is the ratio of the intersection in $t+1$ to the projection of s_i. We define the "onto score" to encode how well the overlapped superpixel is explained by the source superpixel. It is the ratio of the intersection in $t+1$ to the overlapped superpixel o_i.

For all sufficiently overlapped superpixels, the hue histogram [94] distance defines the appearance cost. Here, to ensure robust matching of superpixels of different shapes, we use the hue descriptors only in the appropriate overlapping regions of the superpixels being considered. That is, when considering the potential association between s_i and o_i, superpixel s_i is described by the average hue histogram of the descriptors in the region $\pi_o^t \bigcap s_i$, that is, in the intersection of s_i and the projected o_i onto t. Superpixel o_i is described by the average hue histogram of the descriptors in the region $\pi_s^{t+1} \bigcap o_i$, that is, in the intersection of the projection of s_i onto $t+1$ with o_i. Using only the overlap regions ensures we can fairly compare small to large regions, which could otherwise have greatly differing hue descriptors.

Algorithm 2

```
function SUPERPIXEL ASSOCIATION($(F_{sp}, F_{SLIC}, F_{hue}, F_{mb})$)
    $F_{sv}(1) \leftarrow F_{mb}(1)$ // initialize supervoxel ids to motion boundaries
    for $t \leftarrow 1...T_v$ do // video frames
        for $s_i \in F_{sp}(t)$ do // all superpixels in t
            $\pi_s^{t+1} \leftarrow$ projection of $s_i$ onto frame $t+1$ using dense optical flow
            $O \leftarrow \pi_s^{t+1} \bigcap F_{sp}(t+1)$ // overlapped superpixels
            for $o_i \in O$ do
                $\pi_o^t \leftarrow$ projection of $o_i$ onto frame $t$ using inverse dense optical flow
                $\Theta \leftarrow (\pi_s^{t+1} \bigcap o_i)/\pi_i^{t+1}$ // into score
                $\Omega \leftarrow (\pi_s^{t+1} \bigcap o_i)/o_i$ // onto score
                $H \leftarrow \chi^2(F_{hue}(t)(s_i \cap p_o^t) - F_{hue}(t+1)(o_i \cap p_i^{t+1}))$ // hue distance
                if accept association$(H, \Theta, \Omega)$ then
                    $\Lambda(o_i) \leftarrow \Lambda(o_i) \cup s_i$
                else
                    $o_i$ unassociated
                end if
            end for
        end for
        for $m_i \in F_{mb}(t+1)$ do // motion boundary segments
            $p_i \in F_{mb}(t+1) = m_i$ // superpixels within motion boundary
            if $\|$unique$(F_{mb}(t)(\Lambda(p_i)))\| = 1$ then
                // assign supervoxel label from associated superpixels in t
                $F_{sv}(t+1)(p_i) \leftarrow$ unique$(F_{mb}(t)(\Lambda(p_i)))$
            else
                $F_{sv}(t+1)(p_i) \leftarrow$ new supervoxel label
            end if
        end for
        for all motion boundary violations do
            associate offending superpixels to $F_{SLIC}(t+1)$
            create new superpixels, $F_{sp}(t+1)'$
            update $\Lambda$ and supervoxel labels in $F_{sv}(t+1)$
        end for
        $F_{assoc}(t) \leftarrow \Lambda$
    end for
    return $F_{sv}, F_{assoc}$
end function
```

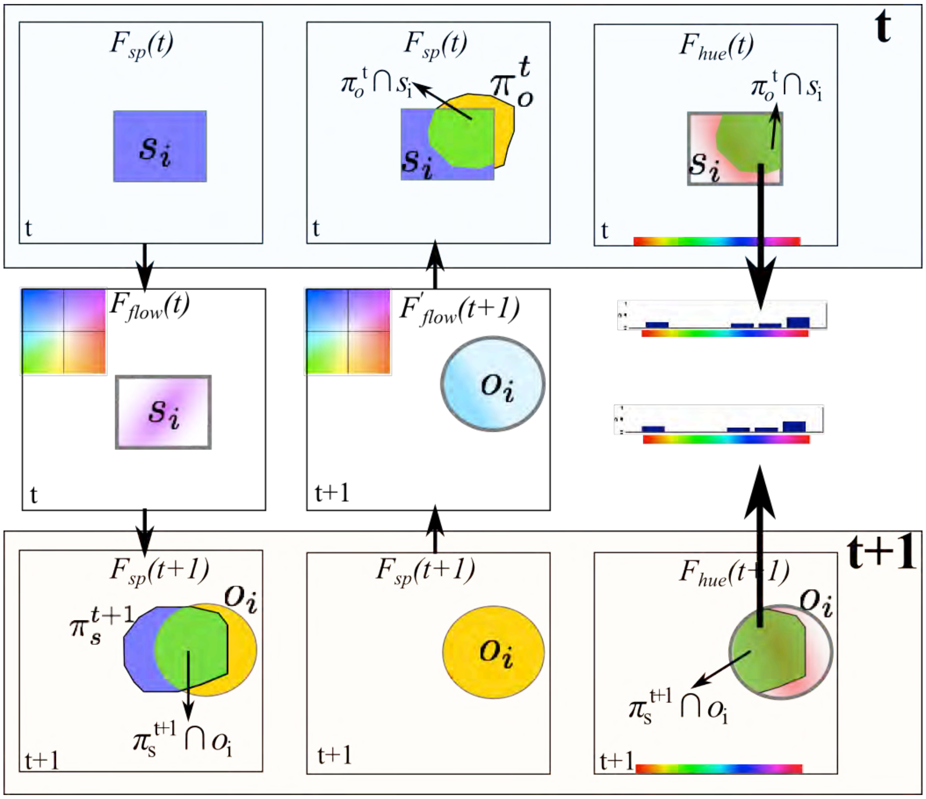


Figure 4.15: When evaluating association hypotheses, we consider the following regions: 1) the superpixel s_i from t, projected onto frame $t+1$ using the dense optical in the region, π_s^{t+1}; 2) the overlapped superpixel in $t+1$, o_i, projected backward, onto t using the inverse optical flow in the region, π_o^t; 3) the intersection of the two projections with the original superpixels, $\pi_o^t \cap s_i$ and $\pi_s^{t+1} \cap o_i$ are used to compute a hue histogram per superpixel, by taking the average of the hue descriptors that fall within each region.

We use these regions, overlap and hue scores to evaluate a small set of association rules, shown in Figure 4.17. If the superpixel s_i in t approximately retains its shape in $t+1$ and the dense flow estimation is accurate, the overlap and hue scores with the overlapped superpixel, o_i, will be high. This is the case for example with stationary superpixels. We refer to this type of association as "one-to-one." On the other hand, s_i could be merged with other regions in $t+1$, or split into multiple regions. A "many-to-one" association (merge) occurs when the projection of s_i is explained well by o_i (large "into" score), but multiple superpixels from t are required to explain o_i well (low "onto" score). A "one-to-many" association (split) occurs when o_i is explained well (large "onto" score), but multiple regions in $t+1$ are required to explain the projection of s_i (low "into" score). The hue distance between these regions should be small in order to accept the association hypothesis. These three rules capture the majority of superpixel relationships (including "many-to-many" cases) and form the primary rule set. We define two

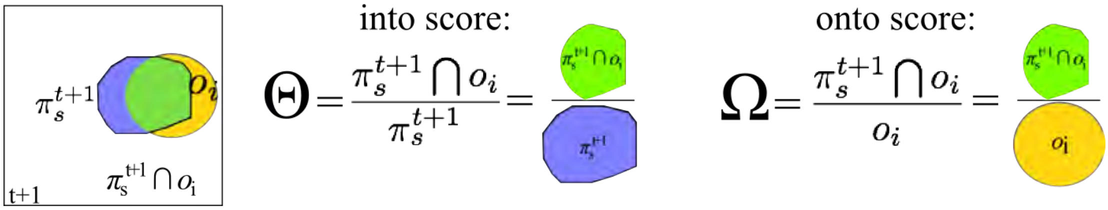


Figure 4.16: When considering overlap, we only take into account the overlapping regions in frame $t+1$. The "into score," which captures how well the projected s_i (blue) is explained by the overlap with o_i (orange), is the ratio of areas of the intersection region (green) to the projection of s_i (blue). The "onto score," which encodes how well the superpixel o_i (orange) is explained by the projected s_i (blue), is the ratio of areas of the intersection region (green) to the area of the superpixel o_i.

auxiliary rules for the remaining set of unassociated superpixels. To be robust to large object and camera motion, we allow low overlap scores, but require very similar hue descriptors. Unlike the other rules, in this case we use the hue descriptors in the original superpixel regions, since the overlap (if any), can be very small. Finally, to be robust to large illumination changes, we allow more dissimilar appearance (larger hue distance), but require very high overlap scores.

The output of this step is superpixel correspondences $F_{assoc}(t)$ from t to $t+1$ and a labeling of the superpixels in $t+1$ with supervoxel ids $F_{\text{sv}}(t+1)$. If all superpixels are successfully associated, the supervoxel ids in $t+1$ will be a subset of those in t. If there are "new" superpixels in $t+1$, i.e., superpixels that do not have an association from t, they get assigned a new supervoxel id based on the motion boundaries in $t+1$. This ensures that the motion boundaries are still respected even when new superpixels appear. If none of the superpixels are associated, the supervoxel ids in $t+1$ will be disjoint from the ones in t.

We illustrate these steps with two consecutive frames from the juggle video. In Figure 4.18 we visualize the input to the association step for frames t and $t+1$, and the output supervoxel ids after accepting all "one-to-one" association hypotheses which preserve motion boundaries. In this example, the person's torso and legs move differently in t, and thus are segmented into two superpixels with different supervoxel ids. In Figure 4.19 we examine the association process for these superpixels, $s_j^t, s_k^t \in F_{\text{sp}}(t)$, and the target superpixel, s_q^{t+1}. First, the dense optical flow projects the source superpixels from t onto $t+1$. This defines overlap scores with a small number of superpixels in $t+1$. For example, s_j^t overlaps 0.4 of the area of s_q^{t+1}, thus the overlap cost is $\text{In}_j = 0.4$. s_k^t overlaps 0.6 of the area of s_q^{t+1}, thus $\text{In}_k = 0.6$. Using the inverse flow, s_q^{t+1} overlaps s_j^t by 0.92, and s_k^t by 0.88, for an average overlap score of $\text{On}_q = 0.9$. Based on the overlap and hue scores, we accept the many-to-one association between s_j^t, s_k^t and s_q^{t+1}.

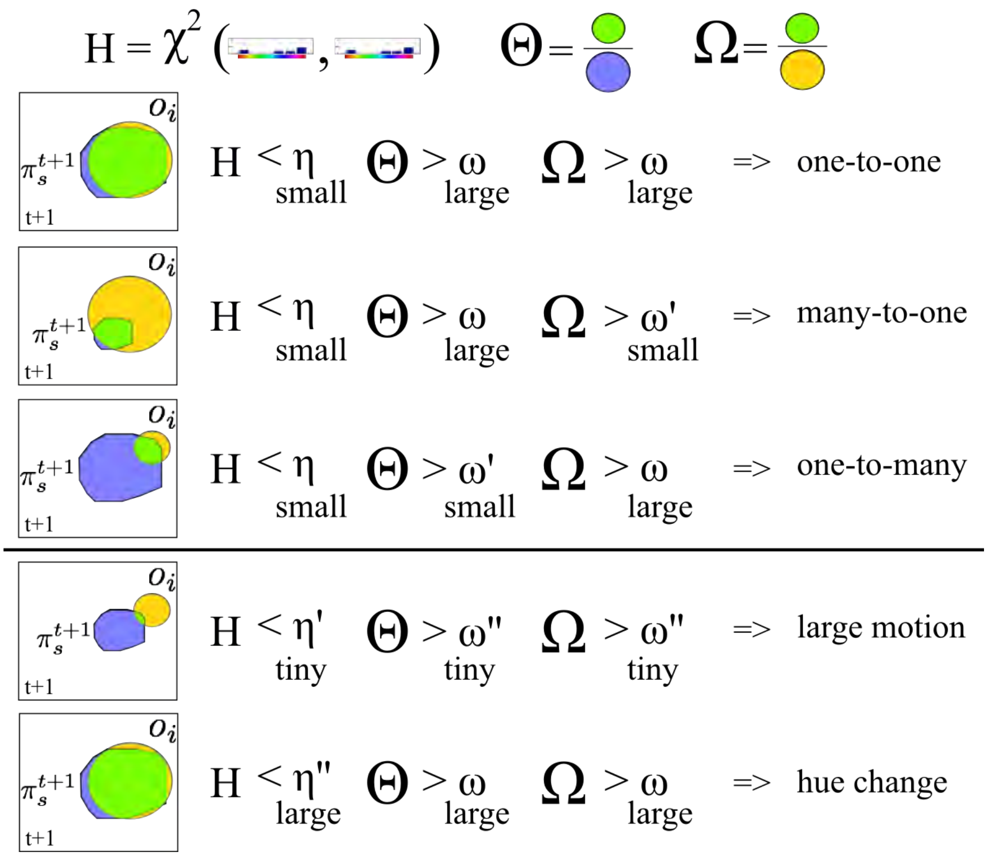
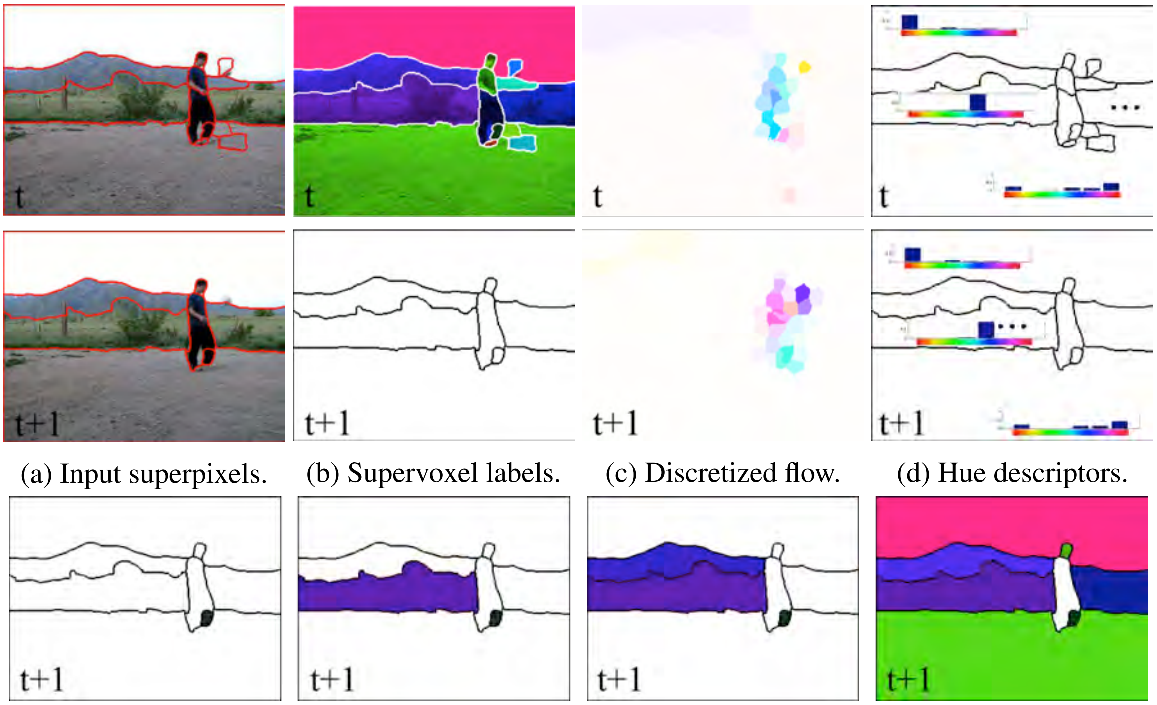


Figure 4.17: We define a small set of rules for associating superpixels from frame t to frame $t+1$ given the hue and overlap scores. The primary set of rules is applied to superpixels of similar color descriptors, while the secondary set captures large motion and illumination changes.

Many-to-many-associations might suggest merging or splitting of superpixels under the same supervoxel id, or under different ones. At this step we do not yet have information whether sets of superpixels should be merged - we do not yet know the global relationships between supervoxels (e.g., if two supervoxels always have similar motion). We postpone merging until a later step, when we will have this information. However, if two neighboring superpixels have different motion in $t+1$, this is sufficient information to assign different supervoxel labels, even if their associating superpixels from t voted for identical supervoxel ids. That is, when a supervoxel from t suggests a split in $t+1$ to superpixels which belong to different motion boundary segments.

Motion boundaries are violated when supervoxels in t with different supervoxel labels associate to one superpixel in $t+1$, forcing a merge of motion boundaries (Figure 4.19). We resolve such violations by re-segmenting superpixels in $t+1$. We can do this robustly via the underlying SLIC building blocks. Once we discover a violation based on a superpixel association (e.g., s_j^t, s_k^t and s_q^{t+1} in the above juggle example), we turn to the hierarchical tree segmentation of the input



(a) Input superpixels. (b) Supervoxel labels. (c) Discretized flow. (d) Hue descriptors.

(e) Intermediate results for associating superpixels into $t + 1$ and propagating supervoxel labels.

Figure 4.18: (a-d): the inputs to the superpixel association step: (a) the superpixels for both frames, (b) the supervoxel labels already computed for frame t, (c) the optical flow for t to project superpixels from t to $t + 1$ and the inverse optical flow to project superpixels from $t + 1$ to t, (d) the hue histogram descriptors for both frames. In (e) we show several steps of the label propagation procedure.

superpixels. Each source superpixel, e.g. s_k^t, is associated with the SLIC superpixels (rather than to superpixel s_q^{t+1}). The associated SLIC superpixels are labeled with the supervoxel id of s_k^t. We repeat this process for all source superpixels (Figure 4.21). In the end, the SLIC superpixels are grouped based on the assigned supervoxel ids, thus re-segmenting the target s_q^{t+1} into superpixels which respect the motion boundaries associated from t (Figure 4.20). This enforces the property that regions with different motion in t will be kept separate in all subsequent frames, as long as the superpixels are associated.

If we have the option to run the segmentation algorithm offline, we can perform an additional motion boundary preservation step. We can also enforce the property that superpixels which separate or move differently in any t are assigned different supervoxel ids in the previous frames. We do so by repeating this re-segmenting step backwards, from $t + 1$ to t for all frames.

Parsimony. We seek a segmentation which respects fine motion boundaries, which corresponds to an oversegmentation in areas of the video that have moving regions. However, in

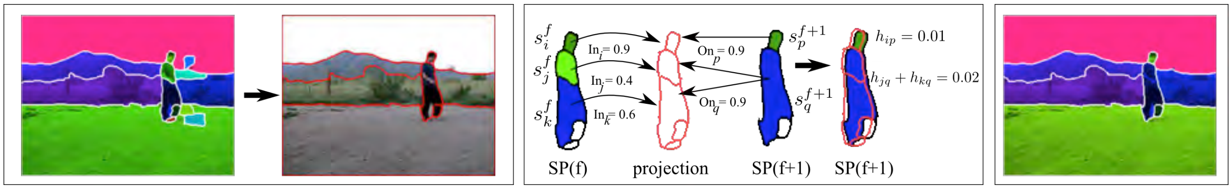


Figure 4.19: Supervoxel ids from t are propagated to $t + 1$ based on the underlying superpixel associations.

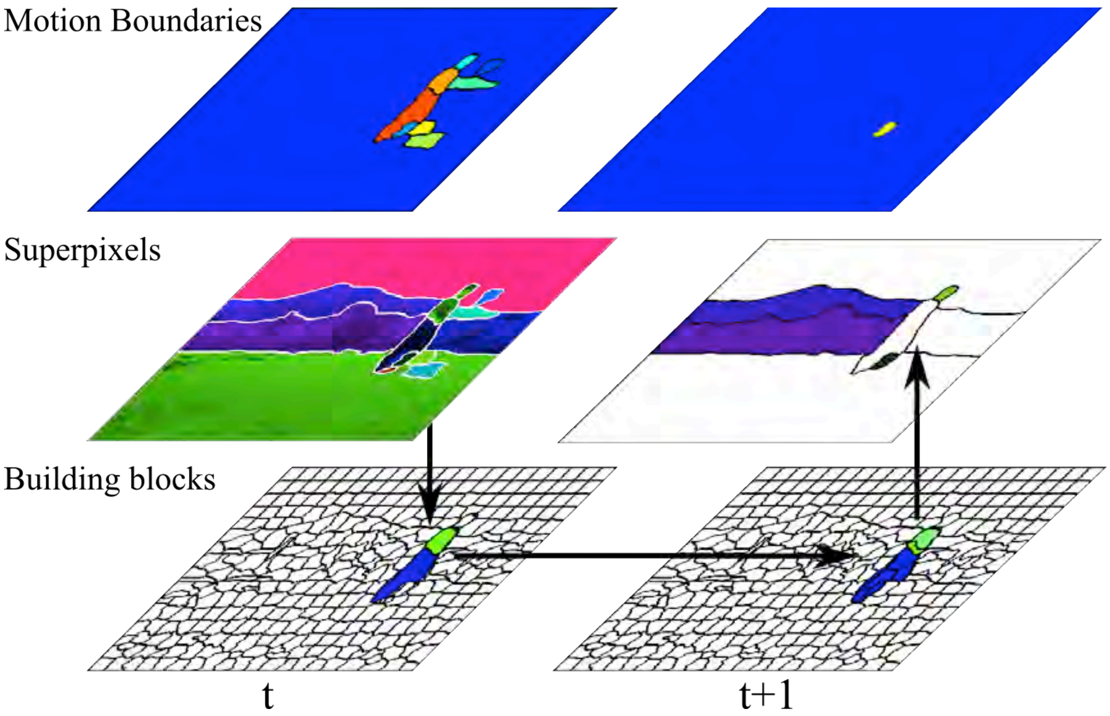


Figure 4.20: When the superpixel associations suggest that supervoxels should merge, the motion boundaries from t are not preserved. We use the SLIC superpixels to re-segment the target superpixel in $t + 1$ to regions which respect the motion boundaries, as associated from frame t.

terms of the appearance boundaries we seek a parsimonious representation. Thus, we merge supervoxels with similar motion. This process encodes property P5, parsimonious appearance boundaries.

In the previous steps we opted to oversegment superpixels and supervoxels when motion boundaries are violated or when superpixel associations are ambiguous. To obtain a parsimonious segmentation, we perform a smoothing step where we merge supervoxels with weak appearance boundaries which have similar motion. We merge only those supervoxels which are always spatio-temporal neighbors, with instantaneous motion (direction and magnitude) within a threshold of each other. Two supervoxels are considered to have similar motion based on the

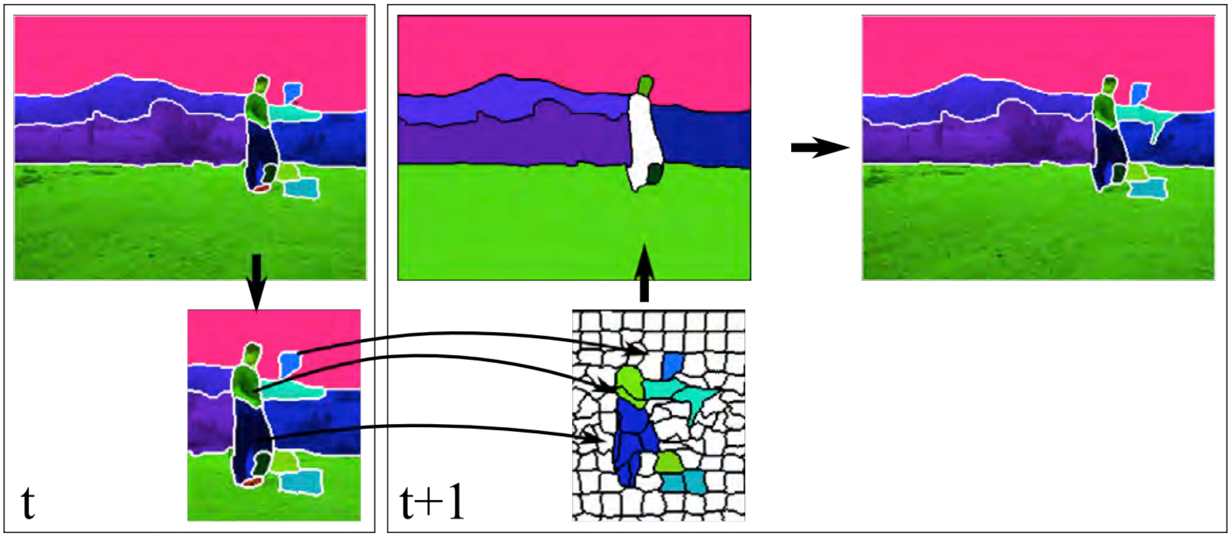


Figure 4.21: The process of associating superpixels from t to the smallest building blocks, SLIC superpixels in $t + 1$, when motion boundaries are violated. For example, the soccer ball in t has a different supervoxel label (denoted with blue) than the background (denoted with pink). However, both the superpixel corresponding to the ball and the one corresponding to the sky are associated to the pink superpixel in $t + 1$. Since they have different supervoxel ids, this merge violates motion boundaries. New superpixels are created in $t + 1$ based on the associations of the underlying SLIC superpixels and the supervoxel labels propagated from t.

angle of their mean optical flow vectors across all frames. Thus, for every pair of neighboring supervoxels and for every frame in which these supervoxels are present, we check if they are neighbors and if their relative motion is similar (Figure 4.23). If so, these supervoxels are merged.

Globally Consistent Supervoxels. We finalize the segmentation by resolving many-to-many associations. We use the overlap score to break ties in converting many-to-many associations to one-to-one associations. If there is ambiguity and a many-to-many association cannot be resolved, we assign new labels to the supervoxels in question in frame $t + 1$.

4.7 GCS quantitive analysis

We evaluate GCS on standard metrics used for video segmentations on a standard benchmark dataset. We use the LIBSVX 3.0 software library [104] and compare to the streaming GBH [105] method. The library provides ground truth segmentation and evaluation for the Chen [15] dataset, which is a subset of the popular `xiph.org` videos. The videos are densely labeled with semantic pixels into 24 classes, and have an average of 85 frames per video. In Figure 4.24 we show

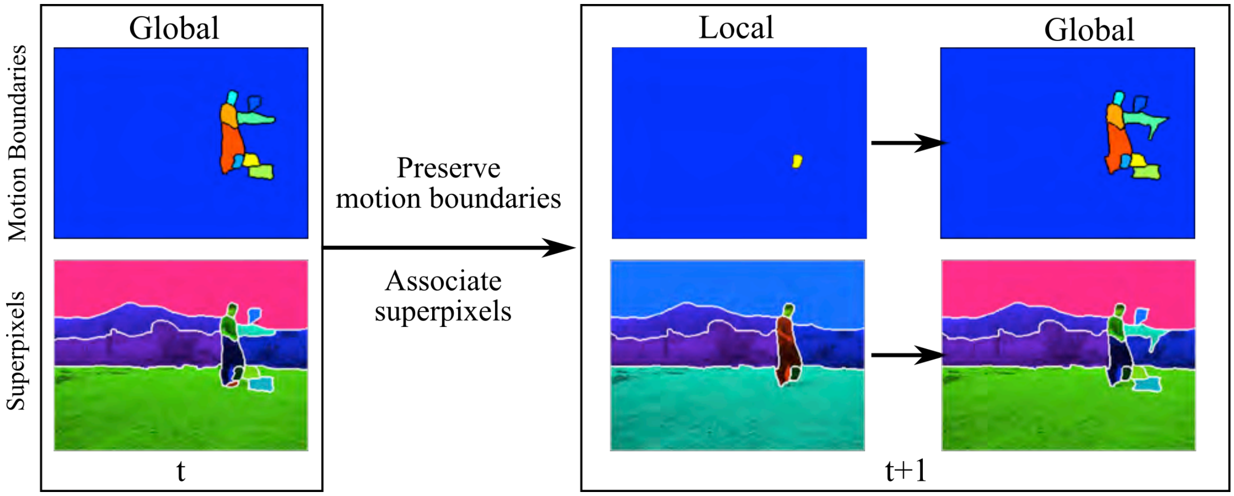


Figure 4.22: The final result of propagating supervoxel ids, re-segmenting superpixels, and propagating motion boundaries information to target frame. This process ensures that motion boundaries from the first frame of the video, through the current frame, $t + 1$, are preserved.

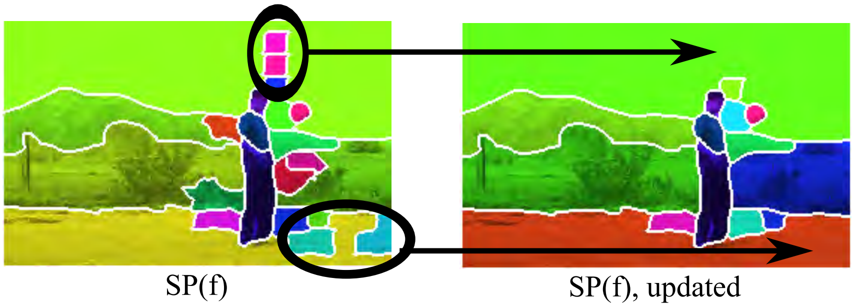


Figure 4.23: For every pair of superpixels that are always neighbors, check if their instantaneous motion is always the same, for all video frames which contain the two superpixels. If this condition is satisfied (e.g. the superpixels pointed to by the arrows), we consider these superpixels as similar in motion, and we merge them in all frames.

examples of several labeled frames from this dataset provided by the authors of [15].

4.7.1 Standard segmentation metrics

Similarly to [105], we present an evaluation of the proposed segmentation on four 3D volumetric performance metrics proposed by the authors to evaluate supervoxel desiderata. We use the LIBSVX library [105], which provides implementations for these metrics, includes ground truth segmentations, and results for the streaming GBH method. To evaluate video segmentations, the authors of LIBSVX generate several segmentations per video with a varying number of supervoxels, from 10 to 5000. They do so by varying parameters of each segmentation method. For

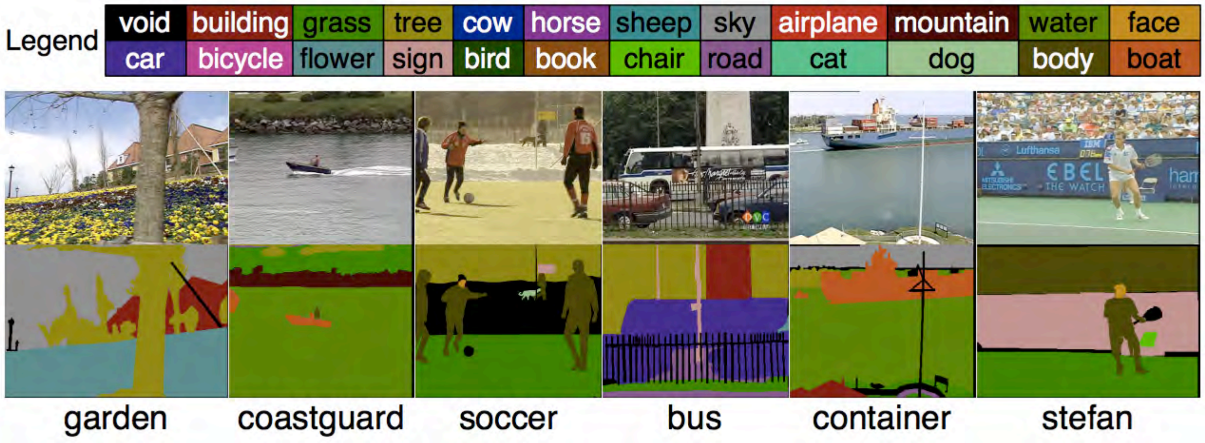


Figure 4.24: Examples of several manually labeled frames from the Chen segmentation dataset [15]. Image courtesy of [15].

GCS, we use one parameter set, the same one used in generating supervoxels for the YouTube dataset. We do not generate multiple segmentations with different number of supervoxels, and we do not tune parameters for GCS for the Chen dataset. Thus, we compare GCS with the average performance over all segmentation granularities for the streaming GBH algorithm.

3D Undersegmentation Error. This metric measures what fraction of voxels exceed the volume boundary of the ground-truth segment when mapping the supervoxels onto it. The authors take the average across all ground-truth segments in the video, giving equal weight to all ground-truth segments. The authors define this metric as:

$$\mathrm{UE}(g_i) = \frac{\sum_{\{s_j | s_j \cap g_i \neq 0\}} -\mathrm{Vol}(g_i)}{\mathrm{Vol}(g_i)},$$

which gives the 3D UE for a single ground-truth segment g_i in the video, where Vol is the segment volume. They take the average across all ground-truth segments in the video, giving equal weight to all ground-truth segments.

Figure 4.25 shows the GCS and GBH performances on this metric for each video in the dataset. Even though GCS was not designed with human labels in mind, the segmentation performs favorably on this metric compared to the state-of-the-art. Since in GCS we tend to group multiple objects, we expect to perform less well on this metric compared to a segmentation which seeks to segment out objects.

3D Boundary Recall. This metric measures the spatiotemporal boundary detection. Video segmentation methods seek a 3D boundary which will match the shape boundary of a 3D object, composed by surfaces. However, in videos, the 3D boundary face is not smooth, since voxels

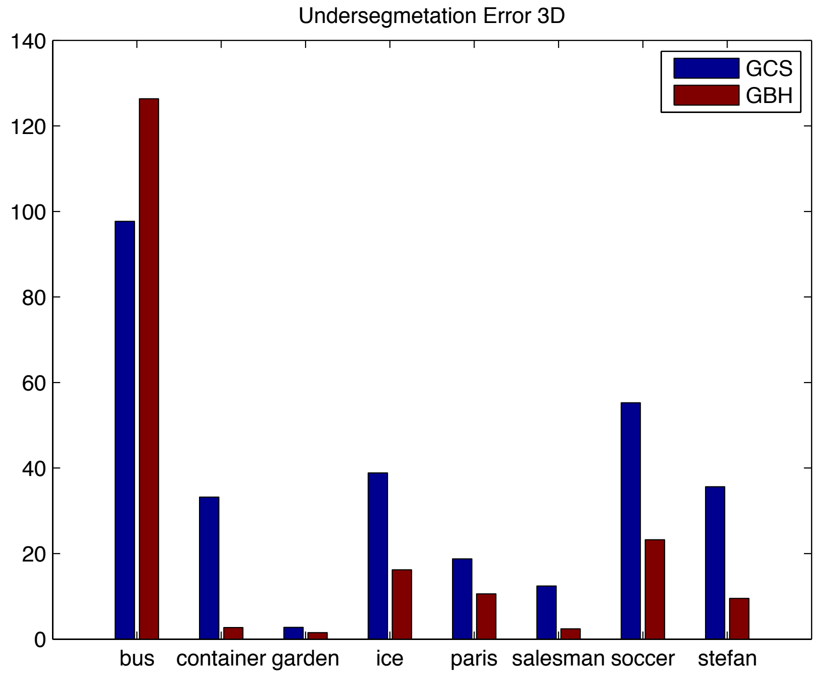


Figure 4.25: The 3D undersegmentation error metric measures what fraction of voxels exceed the volume boundary of the ground-truth segment when mapping the supervoxels onto it. GCS performs favorably compared to the GBH method on the popular Chen segmentation benchmark. This is the case even though GCS was not designed to segment objects and thus capture human ground truth segments.

are rectangular cubes. Thus, in the 3D boundary recall metric, the authors measure how well supervoxels capture both the 2D within-frame boundary and the between-frame boundary (see Figure 4.26). For each segment in the ground-truth and supervoxel segmentations, the authors extract the within-frame and between-frame boundaries and measure recall using the standard formula.

In Figure 4.27 we show the performance on this metric for GCS and GBH. The proposed GCS capture boundaries at a reasonable level compared to the streaming GBH. It is important to note the performance for the “container” video - this is a video with a ship (container) in a river, but all objects and camera are stationary, i.e., there is no motion. GCS groups all supervoxels into one - the entire video. Thus, it does not capture any of the human labeled object boundaries.

3D Segmentation Accuracy. This metric measures what fraction of a ground-truth segment is correctly classified by the supervoxels: each supervoxel should overlap with only one object/segment as a desired property proposed by the authors. We show results for this metric in

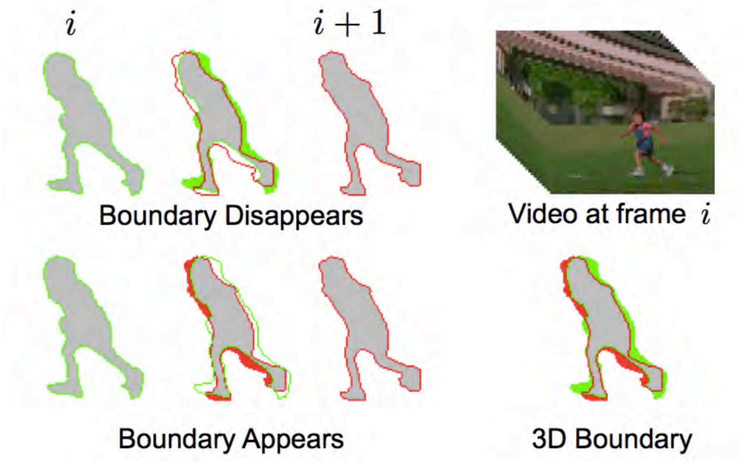


Figure 4.26: Xu and Corso [105] propose to capture 3D boundary recall using the 2D within-frame boundary and also the between-frame boundary. This figure shows the between-frame boundary concept using the ground-truth segment as an example. Image courtesy of [105].

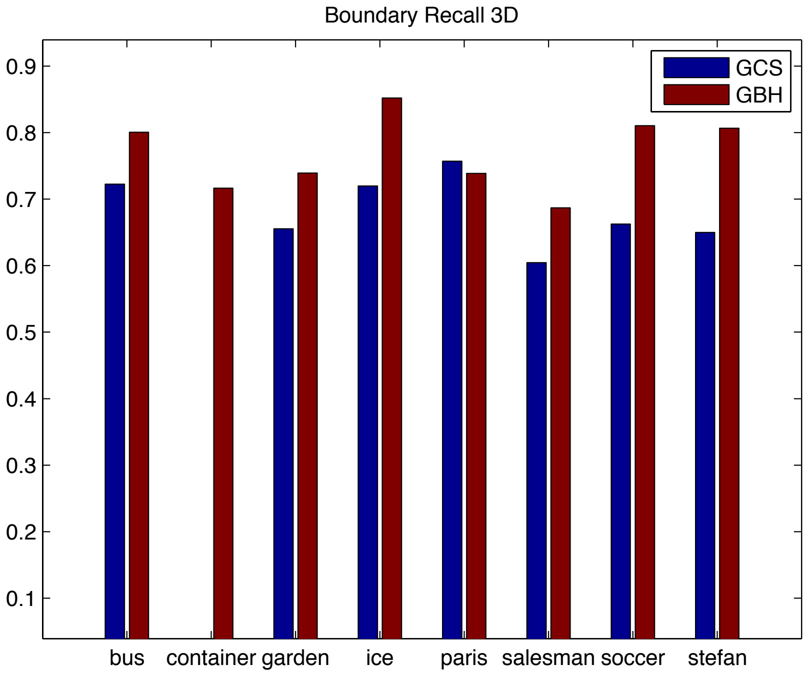


Figure 4.27: The 3D Boundary Recall metric measures the spatiotemporal boundary detection. The proposed GCS capture boundaries at a reasonable level compared to streaming GBH on the Chen dataset.

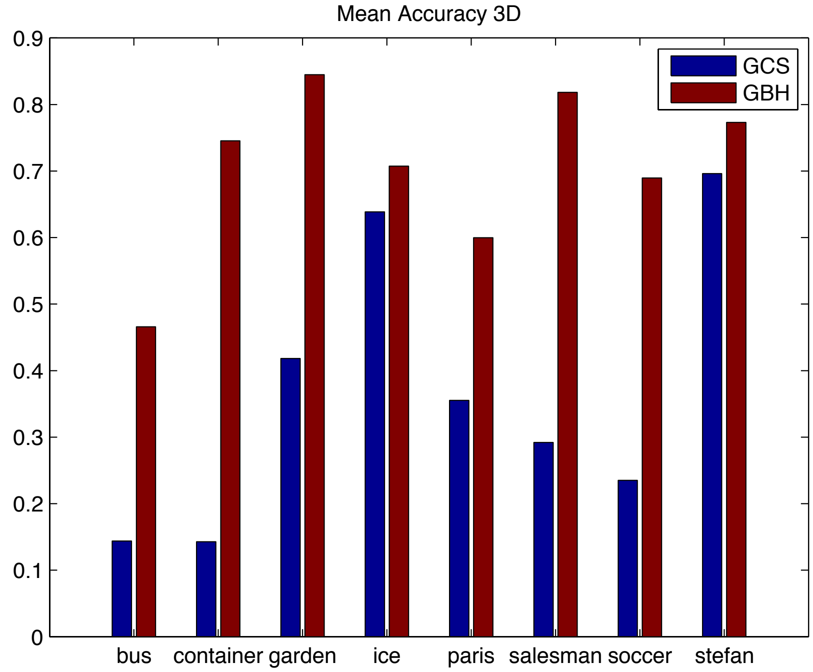


Figure 4.28: The mean 3D segmentation accuracy measures how well supervoxels respect the ground truth labeling, i.e., if the supervoxel overlaps with only on object/ground truth segment. In the Chen dataset, GCS groups multiple stationary regions which correspond to manually labeled objects. Since these stationary supervoxels span ground truth segment boundaries, GCS does not score high on this metric. The exceptions are videos where objects move - for example, the ice skaters ("ice") and the tennis player, tracked by a moving camera ("stefan"). In these videos, GCS performs favorably to GBH, since the regions that have different motion correspond to regions labeled as different objects.

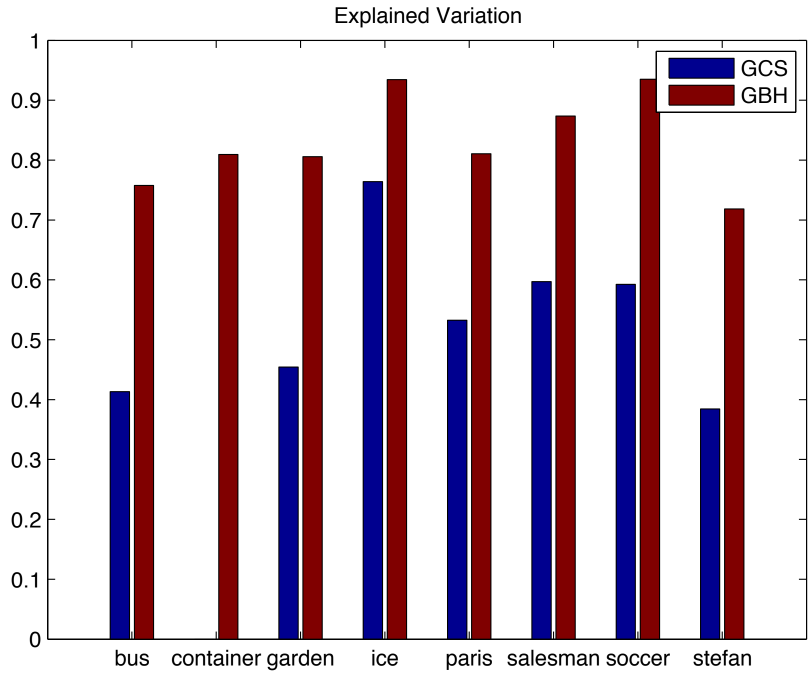


Figure 4.29: Explained variation captures how well the video is explained when the supervoxels are replaced with their mean color. Since GCS groups regions with varying colors if their motion is similar, we do not expect very high performance on this metric. However, for the ice skating video, where the moving regions (people) have somewhat homogenous color, GCS performs favorably.

Figure 4.28. Since in GCS we group regions that have relatively similar appearance, supervoxels will span object boundaries, and thus will perform low on this metric. However, when most objects in the video are moving, GCS should segment them out, and thus perform well. This is the case for the "ice" video, where ice skaters are the moving objects (see Figure 4.31), and the "stefan" video, where both player and camera have large motions. GCS performs well compared to GBH on this metric for these two videos, as expected.

Explained variation. This metric is proposed in [65] as a human independent metric, which is not susceptible to differences in human annotations, unlike the other metrics [105]. It considers the supervoxels as a compression method of a video in terms of the pixel color values. For superpixels, this metric describes the proportion of image variation that is explained when the detail within the superpixels is removed, i.e., when the superpixels are replaced with the mean color. In videos, LIBSVX extends this metric to supervoxels, using the mean color in each supervoxel. We show results for this metric in Figure 4.29. Since GCS groups regions with varying colors if

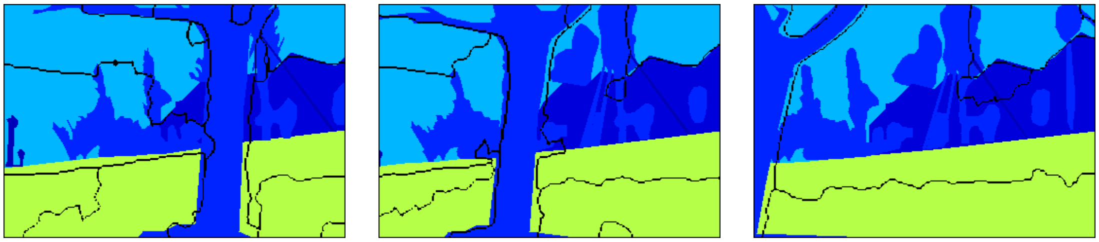


Figure 4.30: The colored regions are the human annotations for the “garden” video from the Chen dataset [15]. These regions are used as the ground truth segments in the evaluation of the above video segmentation metrics. The GCS boundaries, shown in black, produce a meaningful segmentation. They correspond to the tree, grass and sky regions. Since GCS seeks to group regions of similar appearance and motion, the regions corresponding to the bushes in the back are grouped with the houses and the sky, which have similar motion. This is desirable for applications like Motion Words, and it does not match human annotations who usually segment objects.

their motion is similar, we do not expect very high performance on this metric. However, for the ice skating video, where the moving regions (people) have somewhat homogenous color, GCS performs favorably. Unlike the high performance on the mean 3D segmentation accuracy, the tennis video (“stefan”) has many regions with different color values, which have similar motion, and thus the explained variation for this video is also lower.

Qualitative evaluation. Since the GCS segmentation was designed to segment out moving regions, it does not perform better than the streaming GBH method on the metric proposed by Xu and Corso [105], with the exception of videos where the objects annotated by humans are moving. We examine the GCS segmentation on the Chen qualitatively to confirm that, while performance on the above metrics is only comparable, the segmentation produces reasonable regions. In Figure 4.30 we show frames from the “garden” video. In color we show the ground truth segments used in the evaluation of the above metrics. In black we show the GCS boundaries, which correspond well to the tree, grass and sky regions, and produce a meaningful segmentation. In the ice skating video, Figure 4.31, since most ground truth regions correspond to regions with distinct motion, GCS produces as segmentation that is very close to the ground truth. In the soccer video, Figure 4.32, GCS also produces a segmentation comparable to the ground truth, especially for the regions corresponding to the moving soccer players. GCS oversegments the regions of the players, since their limbs move differently from their torsos.

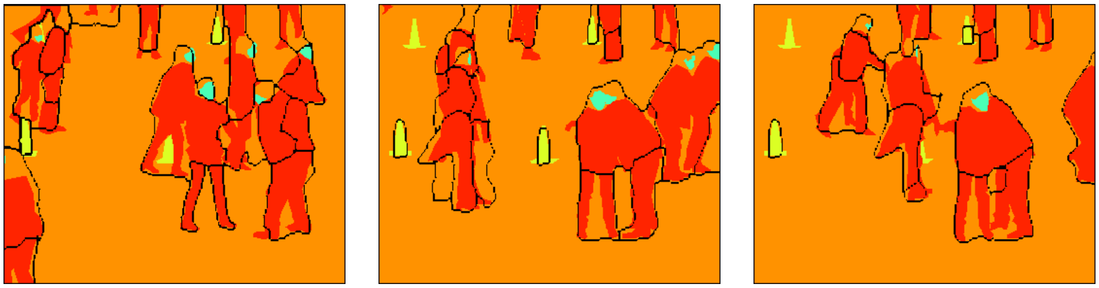


Figure 4.31: The colored regions are the human annotations, considered as ground truth, for the “ice” video from the Chen dataset [15]. The GCS boundaries, shown in black, segment the regions corresponding to the moving people very well.

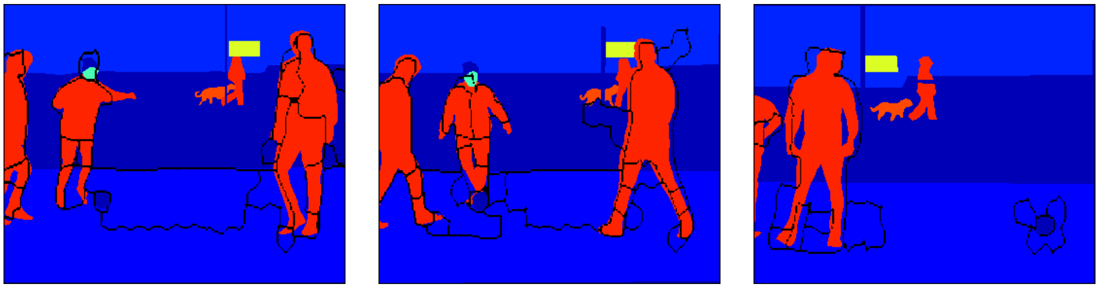
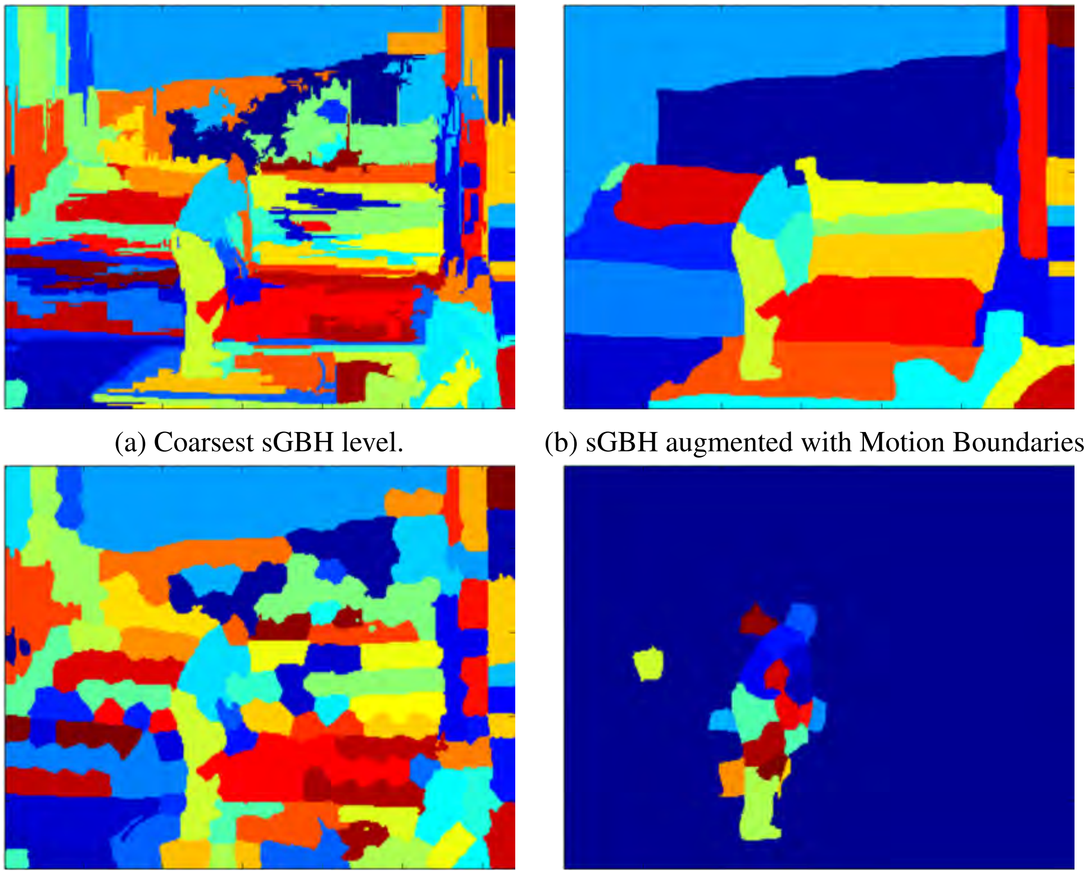


Figure 4.32: The colored regions are the human annotations, considered as ground truth, for the “soccer” video from the Chen dataset [15]. The GCS segments, outlined in black, favorably respect the boundaries corresponding to the soccer players. Those regions are oversegmented by GCS because the limbs and torsos of the players move differently throughout the video.

4.7.2 YouTube dataset evaluation

Since we apply the Globally Consistent Supervoxels segmentation for the task of action classification in large video datasets, we further evaluate GCS and several state-of-the-art segmentation methods on the videos in the YouTube dataset. The dataset contains 1160 videos from 11 action classes, many of which have large camera motion. We compare the proposed GCS to streaming GBH and UES.

In addition, since one of the shortcomings of GBH, and therefore, UES, was merging of motion boundaries and bleeding pixels (e.g., the supervoxels corresponding to the stationary background shown in Figure 4.33a), we propose two modified versions of the streaming GBH algorithm which we evaluate a modified streaming GBH version. To alleviate bleeding pixels, we augment the coarsest level of the GBH hierarchy (level 1) with SLIC superpixels. In every video frame, each SLIC superpixel gets assigned the mode of the supervoxel ids from GBH in the SLIC region, as shown in Figure 4.33c. We refer to this segmentation as GBH SLIC. To alleviate motion boundary merging, we augment the same coarse GBH level with the motion boundary



(a) Coarsest sGBH level.

(b) sGBH augmented with Motion Boundaries.

(c) sGBH augmented with SLIC segmentation.

(d) The GCS segmentation for this frame.

Figure 4.33: Original streaming GBH and augmented versions, compared to the GCS segmentation.

segmentation we computed for GCS (via mean shift clustering of the dense flow), as shown in Figure 4.33b. We refer to this segmentation as GBH MB. We evaluate the segmentations on two metrics that capture how well the supervoxels approximate motion, in terms of the optical flow.

Explained variation for optical flow. We apply this metric to the dense optical flow in videos to assess how well the motion data in the original pixels is represented by the supervoxels. We use same formula as the one for color:

$$R^2 = \frac{\sum_i (\omega_i - \omega)^2}{\sum_i (\upsilon_i - \omega)^2},$$

where we sum over all video pixels, i, υ_i is the actual flow vector at pixel i, ω is the global mean flow vector, and ω_i is the mean flow vector of the pixels assigned to the supervoxel that contains υ_i. R^2 takes the maximum possible value of 1 as the number of supervoxels increases and we recover the original pixels [65]. It takes the minimum possible value of 0 when there is

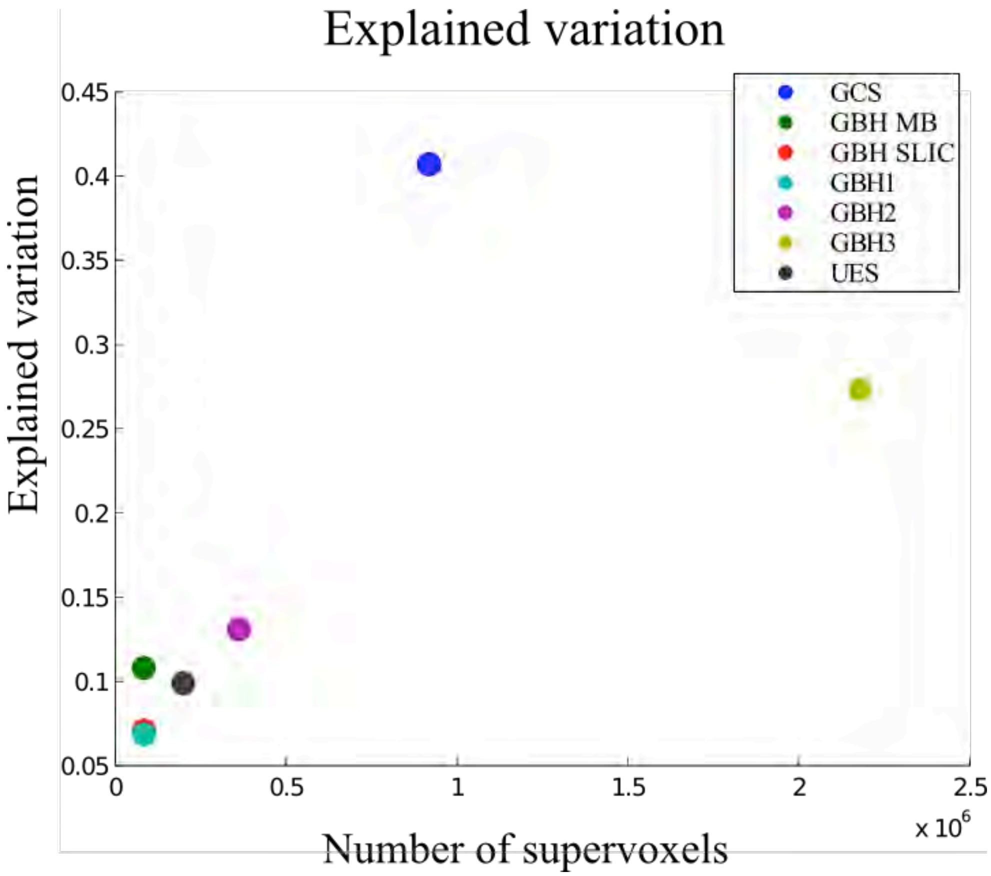


Figure 4.34: Explained variation for dense optical flow on the YouTube dataset. The GCS segmentation, with explained variation value of 0.40, outperforms GBH3 (0.27), while using half as many supervoxels. These results show that the mean flow in the GCS segmentation better represents the actual flow. The results are averaged over all supervoxels from all videos.

only one supervoxel (the video mean) [65]. While this metric penalizes supervoxels that contain consistent motion but have a large variance, it is still useful as a human independent metric.

In Figure 4.34 we show the performance on this metric as a function of the number of supervoxels. We compare GCS to UES, coarse, medium and fine streaming GBH segmentations (GBH1, GBH2, and GBH3, respectively), and the augmented GBH MB and GBH SLIC. The coarsest segmentation, GBH1, which also has the smallest number of supervoxels, has lowest performance - the large segments average flow vectors over regions with very different motion. Augmenting GBH1 with SLIC does not change performance significantly. However, augmenting with GBH1 with the motion boundary segmentation improves performance on this metric. It even exceeds the amount of explained variation compared to UES, which has more supervoxels. The medium level of the GBH hierarchy, GBH2 performs even better, since the supervoxel regions are smaller and the average flow can better represent the motion in the videos. The

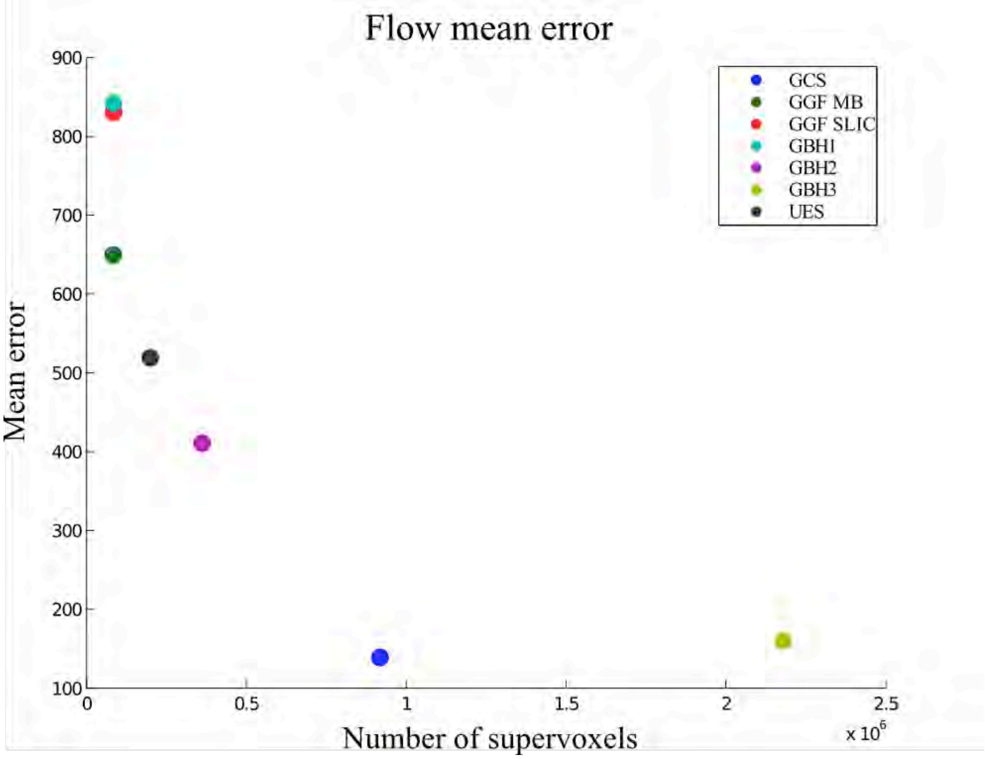


Figure 4.35: We evaluate, in an unsupervised way, how well supervoxels represent the actual motion in each frame, by computing the mean flow error between the actual flow vectors and the mean flow vector in each superpixel in all YouTube videos. GCS approximates flow best, using a relatively parsimonious segmentation.

finest segmentation, GBH3, which generates the highest number of supervoxels (2.2 billion), is better able to explain motion. The GCS segmentation, with explained variation value of 0.40, outperforms GBH3 (0.27), while using 0.9 million supervoxels, compared to 2.2 million.

Mean flow error. Labeling video frames from action datasets like YouTube or HMDB will be infeasible. We want to evaluate how well the supervoxels represent the actual motion in each frame in an unsupervised way. We compute the mean squared error of the dense optical flow in each supervoxel and the actual dense optical flow in the pixels which belong to each supervoxel. We average the errors independently for the horizontal and vertical flow components, over all supervoxels in all videos from the YouTube dataset.

In Figure 4.35 we show the mean error as a function of the number of supervoxels for each segmentation. We compare GCS to UES, a fine, mid-level and coarse GBH segmentation (GBH3, GBH2, and GBH1, respectively), and the augmented GBH1 with motion boundaries, GBH MB, and the augmented GBH1 with SLIC superpixels. We compute the mean error over all supervoxels from all videos. GBH1, which has the least number of supervoxels, but the most coarse

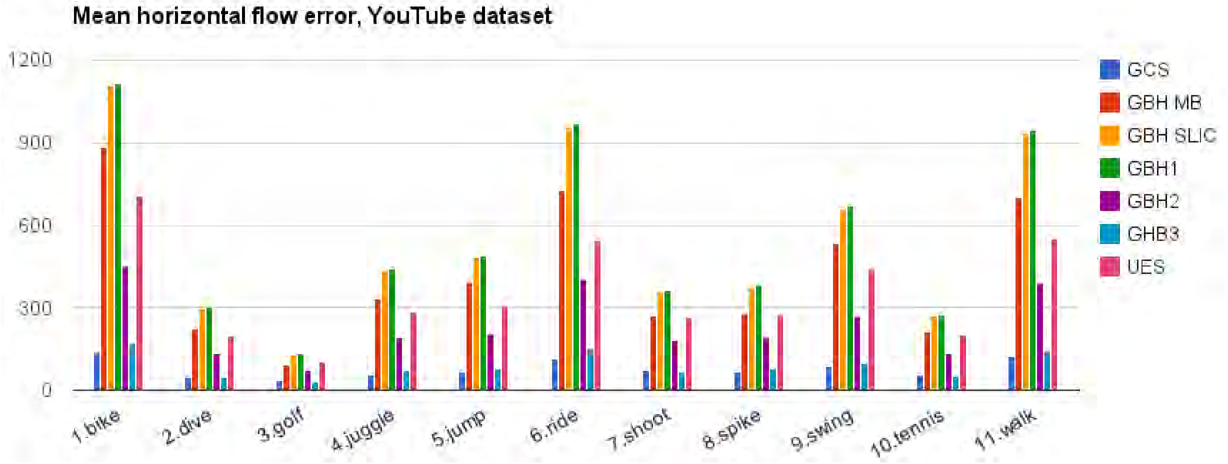


Figure 4.36: Mean error in approximating optical flow in the horizontal component per class on the YouTube dataset. Some action classes, like biking, riding, swing and walking dog feature many videos in which the predominant motion is horizontal, including moving camera. Approximating the flow well is more challenging for these action classes.

ones, has the highest error in approximating the flow. This level of the hierarchy merges regions with different motion to generate large supervoxels. Augmenting this segmentation with SLIC superpixels does not affect the performance for this metric. However, augmenting GBH1 with the motion boundary segmentation does improve performance, since the segments better respect motion boundaries. The number of supervoxels for these three segmentations is the same. The finest segmentation, GBH3 has a much lower error in approximating the flow. This is expected: as the size of the supervoxels approaches that of individual pixels, the error will approach 0. The mid-level GBH2 performs in between the finest and coarsest levels, both in number of supervoxels and in mean error, as expected. The UES segmentation (optimizing for "motionness"), which selects supervoxels among the three hierarchy levels, has more supervoxels than the coarsest level, but fewer supervoxels than the mid-level. This suggests that UES picks mostly coarse segments. Indeed, the error for UES is lower than the coarse segmentation, but higher than the mid-level segmentation. On the other hand, GCS, with only half the number of supervoxels, has lower error (138) than the finest GBH level (159). These results show that GCS better approximates flow using a relatively parsimonious segmentation.

We also examine the flow error independently for the horizontal component in Figure 4.36, and the vertical component in Figure 4.37. Some action classes, like biking, riding, swing and walking dog feature many videos in which the predominant motion is horizontal, including a moving camera, which tracks the objects of interest. Since fewer videos in this dataset feature vertical motion, the total error in the vertical component is lower than the total error in the

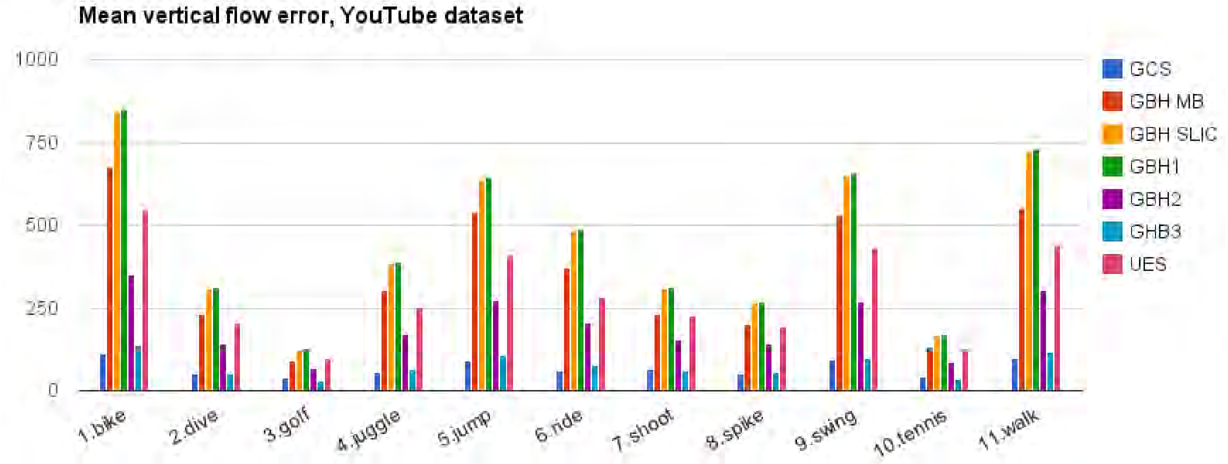


Figure 4.37: Mean error in approximating optical flow in the vertical component per class on the YouTube dataset. Compared to the horizontal flow error, the absolute error in the vertical component is lower. This is the case because the videos in this dataset feature less vertical motion. The exceptions are videos from the classes of biking (mainly due to the camera tracking fast bikers), jumping on a trampoline, swinging, and walking dog (tracking camera).

horizontal component. The exceptions are videos from the class biking, due to the typical camera motion of tracking the bikers; the jumping videos, featuring people on trampolines, the swing videos, which are usually recorded in profile view, and the walking dog videos, where the camera also tracks the people and animals.

In Figure 4.38 we compare across action classes how well each segmentation approximates the horizontal flow. This figure is identical to Figure 4.36, but to better understand performance per class, we group the results in the x-axis based on the segmentation. The class biking is the hardest to approximate across segmentations. However, in the case of GCS, this class is approximated as well as the other two hard classes, riding a horse, and walking a dog. This is not the case with other segmentations, which have a harder time when there is large camera motion. All segmentations have very low error on the golf class, which usually features a stationary camera with the only motion being that of the golfer swinging to hit the ball. Even so, all segmentations other than the fine GBH3 have error much larger than GCS. This result shows that even for scenarios where the motion is confined to a small region and has small amplitude, coarser state-of-the-art segmentations are unable to represent the motion well. This implies that the supervoxels they generate do not respect motion boundaries as well as an oversegmentation, or a segmentation like GCS, which was designed with this property in mind.

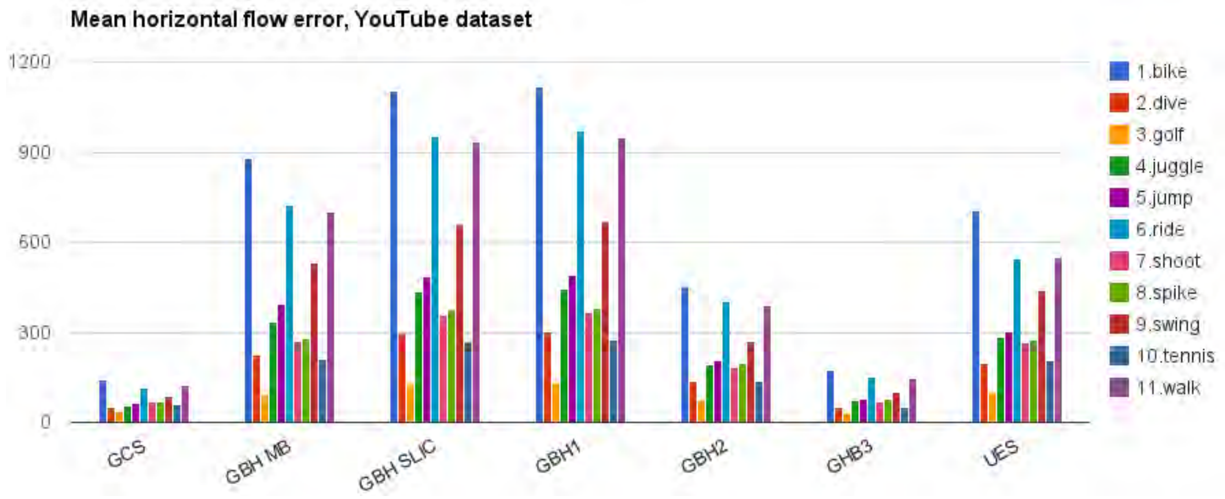


Figure 4.38: It is interesting to compare how well each segmentation approximates the horizontal flow across action classes. This figure is identical to Figure 4.36, but to better understand performance per class, we group the results in the x-axis based on the segmentation.

4.7.3 Action classification

We evaluate the GCS segmentation for the task of computing Motion Words. We compare action classification performance to Motion Words learned using the streaming GBH method and the UES method. We use GCS in the same experimental setup as the one used to validate Motion Words in Chapter 3, i.e., the same datasets, comparing to the same baselines. In Figure 4.39 we show results for several BoW codebook sizes, and several BMW codebook sizes on the YouTube dataset. Here we use the standard hard encoding and average pooling in both Bow and BMW. In all cases, pooling over a video segmentation outperforms standard BoW. The coarse segmentations have lower performance compared to the fine GBH3, the multi-level GBH4 (which includes the supervoxels from GBH3), and the proposed GCS.

4.7.4 Action retrieval

Furthermore, we are interested if GCS gives better performance for video retrieval and if the similar Words between videos are more semantically meaningful compared to other segmentations.

In Table 4.1 we report average recall from a nearest neighbor retrieval task on both datasets using L1 distance. Using supervoxels from the GBH3 segmentation gives lower performance on both datasets compared to GCS. The GBH regions combined pixels from regions with different motion, which makes this segmentation not well suited for this task. On the other hand, using GCS which better respects motion boundaries, obtains state-of-the-art performance on both

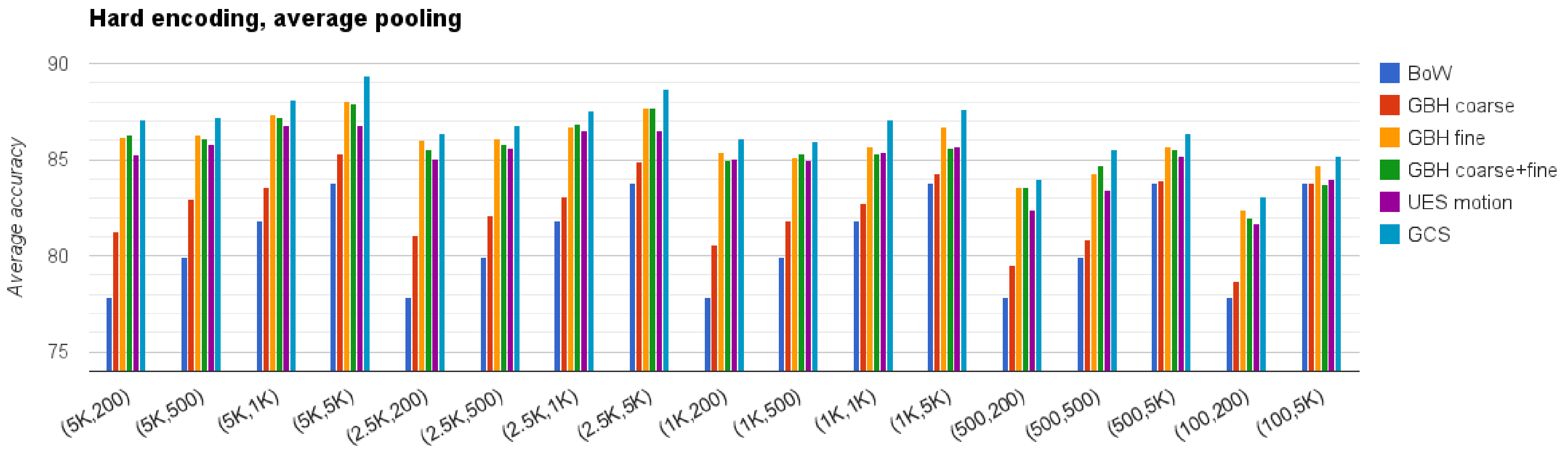


Figure 4.39: GCS has better classification performance than other segmentations on the YouTube dataset. We report mean average accuracy for several codebook sizes, denoted in the format (K_{BMW}, K_{BoW}), using hard encoding and average pooling for both BoW and BMW.

	BoW	BMW GBH3	BMW GCS
YouTube [57]	66.95	68.17	**70.13**
HMDB [47]	12.59	9.46	**13.13**

Table 4.1: In a nearest neighbor retrieval task, we pool Dense Trajectories globally, over the fine GBH3 and over the GCS segmentation. Chance on the YouTube dataset is 9.09 while on the HMDB it is 1.96 . The supervoxels from GCS are better suited for this task than the ones generated by GBH.

datasets.

4.8 GCS qualitative evaluation

We investigate the qualitative properties of the proposed GCS segmentation visually. In Figure 4.40 we show thee manually chosen supervoxels from a biking video. The first and last time frames displayed are also the first and last frames in which each supervoxel appears and disappears. In this video the camera moves smoothly as the biker moves to left, and thus the supervoxels are of reasonably long temporal length (e.g., $30 - 50$ frames). In the middle row we show a region which corresponds to the back wheel of the bike, which was merged with the grass in the UES segmentation. However, since the motion of those pixels is different from the grass, GCS separates the two regions. Furthermore, unlike UES and GBH, the regions away from the biker are large and coarse, but the ones corresponding to the biker are finer, and capture the varying motion in those pixels.

In Figure 4.41a we show manually selected supervoxels from another biking video, which features very large camera motion, which results in very large motion blur. Unlike, GBH and UES, the proposed GCS produces more coherent supervoxels which better capture motion information. Another video with similar large camera motion and blur is shown in Figure 4.42. GCS is again able to associate superpixels across time well, producing supervoxels of lower temporal length, but preserving motion boundaries well.

In Figure 4.42 we show the difficult diving video, where UES groups the background with the diver after frame 17. Here we see that GCS, while generating supervoxels of lower temporal length compared to UES, keeps the two regions separately.

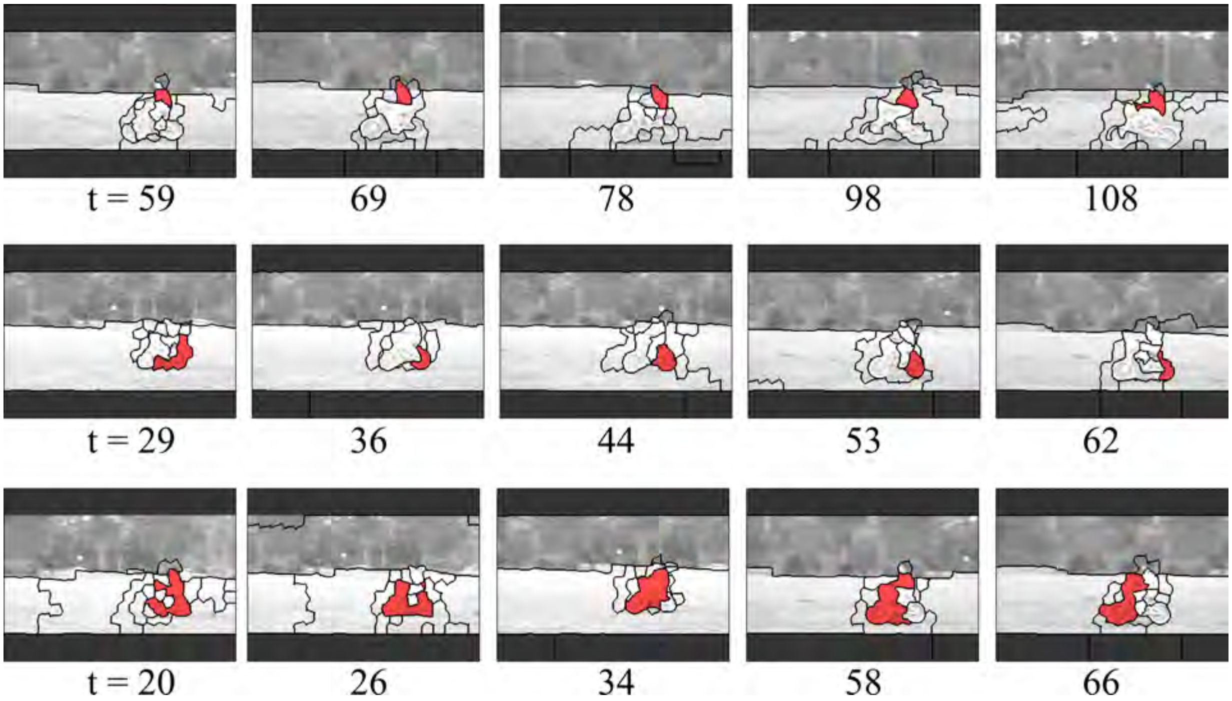


Figure 4.40: Manually chosen supervoxels in a biking video which has camera motion. GCS is able to associate superpixels across a reasonable number of frames, e.g. $30 - 50$ even for regions with large motion. In the middle row we show a region which corresponds to the back wheel of the bike, which was merged with the grass in the UES segmentation.

4.9 Conclusion

The proposed Globally Consistent Supervoxels (GCS) considers superpixels as the smallest building blocks to build a segmentation which respects motion boundaries. We group these superpixels into supervoxels based on their appearance, motion, and their spatio-temporal relationships across all video frames, which ensures *global* consistency. We show that these supervoxels better encode the desired properties, are better suited for Motion Words, and produce visually suitable segments. Evaluation of the proposed segmentation on motion boundary preserving properties on both a standard segmentation benchmark and the YouTube action dataset show excellent optical flow approximation, with relatively parsimonious supervoxels. Action classification results on the YouTube dataset show that this segmentation is more suitable for learning a Bag of Motion Words representation, obtaining state-of-the-art performance.

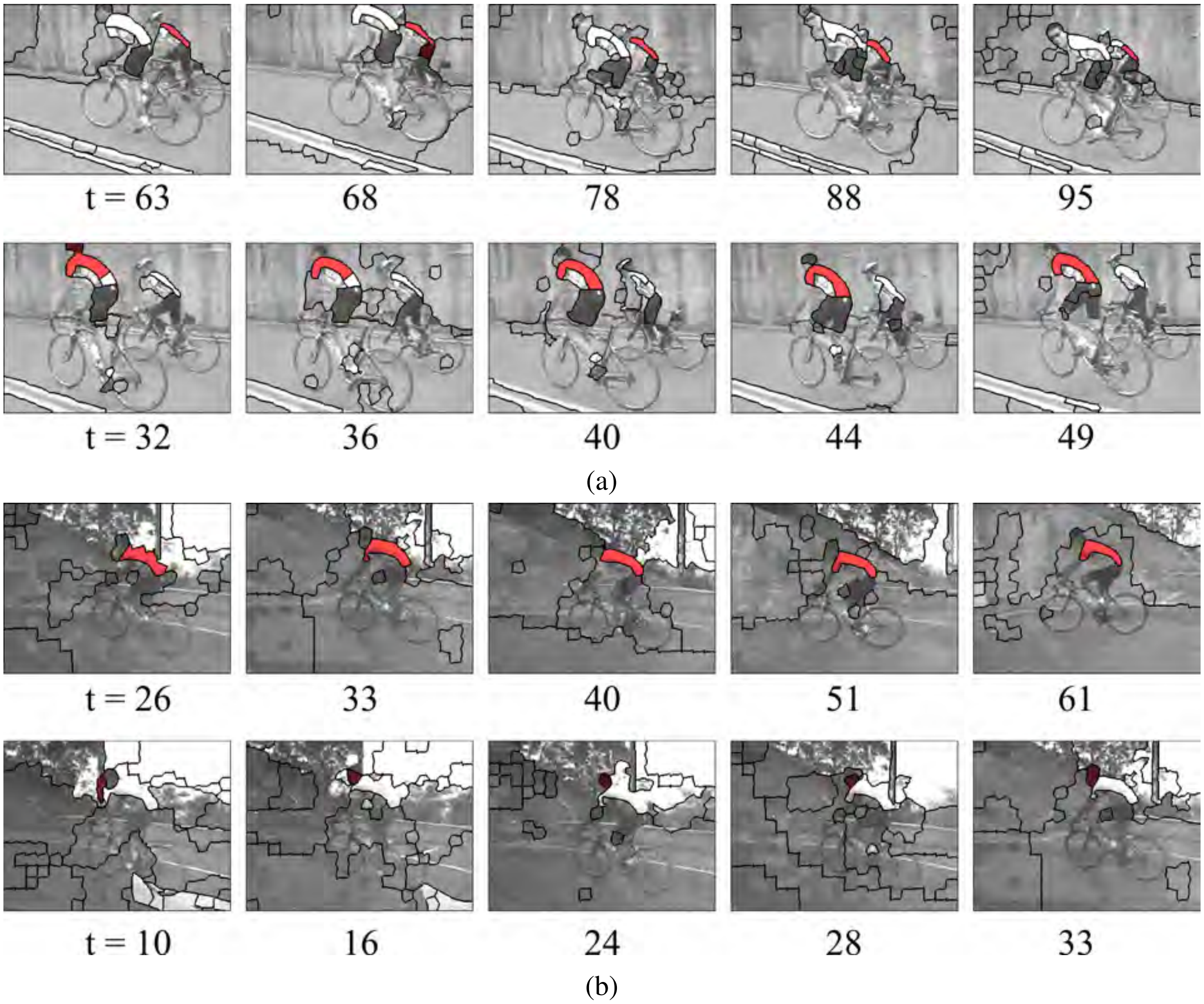


Figure 4.41: Manually chosen supervoxels from a biking video with very large camera motion and thus very large motion blur. Regardless of the noise, GCS is able to segment regions which span more than 10, preserving motion boundaries.

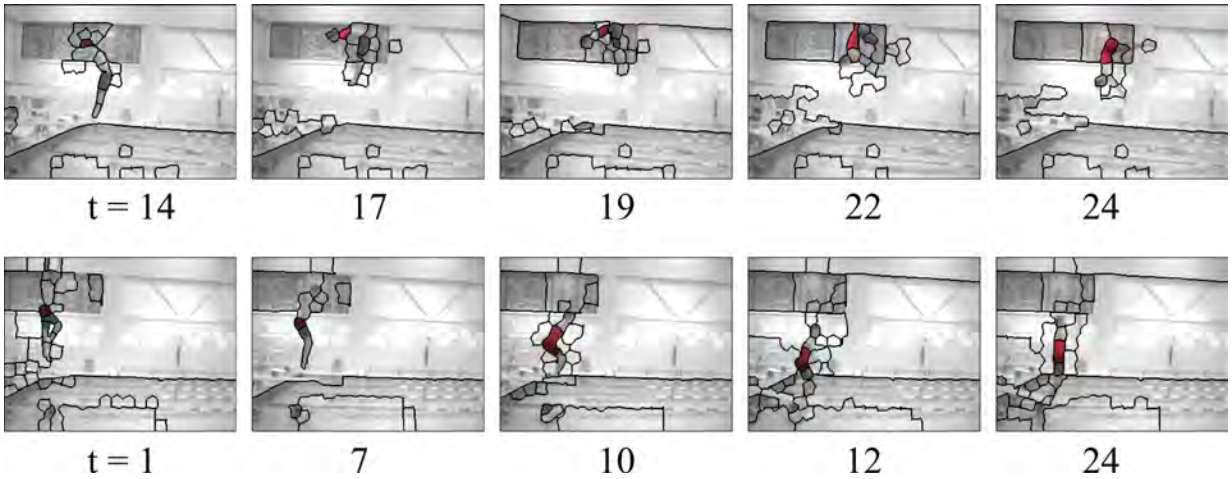


Figure 4.42: Manually chosen supervoxels for two diving videos. Previously, we showed that GBH and UES group the background with the diver after frame 17. On the other hand, GCS is able to preserve motion boundaries. However, to capture such fine motion boundaries, GCS generates temporally shorter supervoxels.

Chapter 5

Conclusion

This thesis provides new methods for quantization and pooling of features for videos for the task of action classification. First, we provide a new quantization method, Source Constrained Clustering, which targets clustering of features from data with large within-class variability. In particular, we study the problem of data quantization via K-means in the context of the popular Bag of Words (BoW) framework. The BoW framework relies heavily on the assumption that the clustering step produces a grouping of the samples which is meaningful for the classification task. However, when data comes from disparate sources the resulting clusters might not be suitable for distinguishing between the desired semantic classes. Example scenarios which use data from disparate sources include: activity recognition tasks where multiple subjects perform actions with varied styles in unconstrained environments; object recognition tasks in which the images come from environments with dissimilar characteristics; data from web sources with different presentation style and bias. Through evaluation on two complex video datasets, we show that the proposed Source Constrained Clustering method produces video representations that generalize better across activity execution styles.

Second, we propose robust spatio-temporal support for pooling features over supervoxel regions. In the task of activity recognition in videos, computing the video representation often involves pooling feature vectors over spatially local neighborhoods. The pooling is done over the entire video, over coarse spatio-temporal pyramids, or over pre-determined rigid cuboids. Similarly to pooling image features over superpixels in the image analysis community, it is natural to consider pooling spatio-temporal features over a video segmentation. We propose the Bag of Motion Words video representation, where we pool features over supervoxels. In our experiments we show how the representation enables understanding of *why* videos are classified as similar, while achieving state-of-the-art performance in the tasks of activity classification and retrieval.

Finally, we propose a new supervoxel segmentation method, Globally Consistent Supervoxels (GCS), which oversegments regions with motion, aiming to capture motion boundaries with high precision. We are inspired by superpixel methods, which aim to oversegment object boundaries, thus decomposing large regions into smaller building blocks coherent in appearance. One of the goals of these methods is to respect object boundaries, that is, to seek a segmentation where pixels belonging to different objects have different segment ids. We propose a similar approach to supervoxel segmentation, namely, to oversegment spatio-temporal regions into smaller building blocks coherent in motion. We seek to preserve motion boundaries, that is, to assign different supervoxel ids to pixels which have different motion in the same frame. We show that this segmentation produces meaningful regions and further improves classification performance in the context of Bag of Motion Words for action classification and retrieval.

Bibliography

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence*, 34(11):2274 – 2282, 2012. 67, 77, 78, 79

[2] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2189–2202, 2012. 65

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011. 67, 78, 79

[4] F. Bach and Z. Harchaoui. Diffrac: a discriminative and flexible framework for clustering. In *NIPS*, 2007. 3, 11, 13

[5] V. Bettadapura, G. Schindler, T. Ploetz, and I. Essa. Augmenting bag-of-words: Data-driven discovery of temporal and structural information for activity recognition. In *CVPR*, 2013. 3, 25, 27, 29, 35, 39

[6] Y.-L. Boureau, N. Le Roux, F. Bach, J. Ponce, and Y. LeCun. Ask the locals: Multi-way local pooling for image recognition. In *ICCV 2011*, pages 2651–2658, Nov 2011. 3, 26, 27, 28, 35, 41, 46, 47

[7] P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained K-Means Clustering. Technical report, Microsoft Research, 2000. 3, 11

[8] W. Brendel and S. Todorovic. Video object segmentation by tracking regions. In *International Conference on Computer Vision*, pages 833–840, 2009. 3, 35, 65, 73

[9] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision*, 2004. 63, 80

[10] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *European Conference on Computer Vision*, pages 282–295, 2010. 27, 36, 39, 65

[11] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In

CVPR, pages 60–65, 2005. 34

[12] Y. Cao, D. Barrett, A. Barbu, S. Narayanaswamy, H. Yu, A. Michaux, Y. Lin, S. Dickinson, J. M. Siskind, and S. Wang. Recognize human activities from partially observed videos. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 0:2658–2665, 2013. 27

[13] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic Segmentation with Second-Order Pooling. In *European Conference on Computer Vision*, October 2012. 27, 33

[14] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2011. 3, 41, 56

[15] A. Y. C. Chen and J. J. Corso. Propagating multi-class pixel labels throughout video frames. In *Image Processing Workshop (WNYIPW)*, pages 14–17, 2010. 88, 89, 90, 95, 96

[16] C.-Y. Chen and K. Grauman. Efficient activity detection with max-subgraph search. In *CVPR*, pages 1274–1281. IEEE, 2012. 34

[17] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, May 2002. 64

[18] J. J. Corso, E. Sharon, S. Dube, S. El-Saden, U. Sinha, and A. Yuille. Efficient multilevel brain tumor segmentation with integrated bayesian model classification. *Medical Imaging, IEEE Transactions on*, 27(5):629–640, 2008. 3

[19] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising with block-matching and 3d filtering. In *Electronic Imaging*, 2006. 34

[20] C. Dance, J. Willamowski, L. Fan, C. Bray, and G. Csurka. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, 2004. 2, 7

[21] Daniel DeMenthon and Remi Megret. Spatio-Temporal Segmentation of Video by Hierarchical Mean Shift Analysis. Technical Report LAMP-TR-090,CAR-TR-978,CS-TR-4388,UMIACS-TR-2002-68, University of Maryland, College Park, 2002. 64

[22] F. De la Torre, A. Bargeil, X. Martin, and J. Hodgins. Guide to the CMU Multimodal Activity (CMU-MMAC) Database. http://kitchen.cs.cmu.edu/. In *Technical report, Robotics Institute, CMU*, 2008. 2, 10, 20, 22, 23

[23] F. De la Torre and T. Kanade. Discriminative cluster analysis. In *International Conference on Machine Learning*, volume 148, pages 241 – 248, New York, NY, USA, June 2006.

ACM Press. 3, 11, 13, 19

[24] M. V. den Bergh, G. Roig, X. Boix, S. Manen, and L. J. V. Gool. Online video seeds for temporal window objectness. In *ICCV*, pages 377–384, 2013. 65

[25] K. G. Derpanis, M. Sizintsev, K. Cannons, and R. P. Wildes. Efficient action spotting based on a spacetime oriented structure representation. In *Computer Vision and Pattern Recognition*, pages 1990–1997, 2010. 34

[26] C. Ding and X. He. K-means clustering via principal component analysis. In *International Conference on Machine Learning*, volume 1, pages 225–232, 2004. 11

[27] C. Ding and T. Li. Adaptive dimension reduction using discriminant analysis and k-means clustering. In *International Conference on Machine Learning*, 2007. 3, 11, 13, 19

[28] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005. 3, 27, 33

[29] I. Everts, J. C. van Gemert, and T. Gevers. Evaluation of color STIPs for human action recognition. In *Computer Vision and Pattern Recognition*, June 2013. 28, 33, 35

[30] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, Sept. 2004. 64, 67, 79

[31] C. Fowlkes, S. Belongie, and J. Malik. Efficient spatiotemporal grouping using the nyström method. In *CVPR*, December 2001. 65

[32] Y. Fu, Y. Jia, and Y. Kong. Interactive phrases: Semantic descriptions for human interaction recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints):1, 2014. 27

[33] K. Fukunaga. *Introduction to Statistical Pattern Recognition, Second Edition*. Academic Press. Boston, MA, 1990. 12

[34] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Inf. Theor.*, 21(1):32–40, Sept. 2006. 64

[35] A. Gaidon, Z. Harchaoui, and C. Schmid. Activity representation with motion hierarchies. *International Journal of Computer Vision*, Nov. 2013. 27

[36] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, pages 1–8. IEEE, 2009. 26, 34

[37] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. *Computer Vision and Pattern Recognition*, 2010. 3, 35, 37, 61, 64, 73

[38] C. W. Hsu, C. C. Chang, and C. J. Lin. A practical guide to support vector classification. Technical report, National Taiwan University, 2005. 43

[39] ILOG. IBM ILOG CPLEX optimizer, 2010. `http://www.ilog.com/products/cplex/`. 19

[40] H. Izadinia and M. Shah. Recognizing complex events using large margin joint low-level event model. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part IV*, European Conference on Computer Vision, pages 430–444, Berlin, Heidelberg, 2012. Springer-Verlag. 33

[41] A. Jain, S. Chatterjee, and R. Vidal. Coarse-to-fine semantic video segmentation using supervoxel trees. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013. 62, 65

[42] A. Jain, A. Gupta, M. Rodriguez, and L. S. Davis. Representing Videos using Mid-level Discriminative Patches. In *Computer Vision and Pattern Recognition*, June 2013. 3, 25, 27, 29, 33, 39

[43] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311, jun 2010. 34

[44] Y.-G. Jiang, S. Bhattacharya, S.-F. Chang, and M. Shah. High-level event recognition in unconstrained videos. *International Journal of Multimedia Information Retrieval*, 2(2):73–101, 2013. 27

[45] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *IEEE International Conference on Computer Vision*, October 2007. 34

[46] A. Kläser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *British Machine Vision Conference*, pages 995–1004, sep 2008. 39

[47] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 2, 27, 42, 104

[48] L. Ladicky, C. Russell, P. Kohli, and P. H. S. Torr. Associative hierarchical crfs for object class image segmentation. In *ICCV*, pages 739–746, 2009. 65

[49] T. Lan, Y. Wang, and G. Mori. Discriminative figure-centric models for joint action localization and recognition. In *International Conference on Computer Vision (ICCV)*, 2011. 34

[50] I. Laptev and T. Lindeberg. Local descriptors for spatio-temporal recognition. In *International Workshop on Spatial Coherence for Visual Motion Analysis*, 2006. 20, 39

[51] M. H. Law, A. Topchy, and A. K. Jain. Clustering with Soft and Group Constraints. *Structural, Syntactic, and Statistical Pattern Recognition*, pages 662–670, 2004. 3, 11

[52] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Computer*

Vision and Pattern Recognition, 2011. 3, 28, 33, 37, 39

[53] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. *ICCV*, 2011. 65

[54] V. Lempitsky, A. Vedaldi, and A. Zisserman. A pylon model for semantic segmentation. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2011. 65

[55] J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *CVPR*, 2011. 35, 65, 66, 71

[56] J. Liu, B. Kuipers, and S. Savarese. Recognizing human actions by attributes. In *CVPR*, 2011. 3, 27, 33

[57] J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos ”in the wild”. In *Computer Vision and Pattern Recognition*, June 2009. 2, 25, 27, 28, 42, 104

[58] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, 1981. 65

[59] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *5-th Berkeley Symposium on Mathematical Statistics and Probability.Berkeley, University of California Press.*, pages 1:281–297, 1967. 3, 11

[60] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *ICCV*, pages 2272–2279, 2009. 34

[61] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2009. 9, 10, 20, 21, 22, 23

[62] S. Mathe and C. Sminchisescu. Dynamic Eye Movement Datasets and Learned Saliency Models for Visual Action Recognition. In *ECCV*, 2012. 3, 4, 33, 34

[63] R. Messing, C. Pal, and H. Kautz. Activity recognition using the velocity histories of tracked keypoints. In *Proceedings of the Twelfth IEEE International Conference on Computer Vision*, 2009. 10

[64] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *European Conference on Computer Vision*, volume 3021, pages 69–82. Springer Berlin, 2004. 2, 7

[65] A. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *Computer Vision and Pattern Recognition*, 2008. 3, 34, 35, 66, 67, 68, 71, 94, 97, 98

[66] M. H. Nguyen, L. Torresani, F. De la Torre, and C. Rother. Weakly supervised discriminative localization and classification: a joint learning process. Technical report, Carnegie Mellon University, 2009. 28, 33

[67] J. C. Niebles, C. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable

motion segments for activity classification. In *ECCV*, 2010. 3, 10

[68] D. Oneata, J. Verbeek, and C. Schmid. Action and Event Recognition with Fisher Vectors on a Compact Feature Set. In *ICCV*, pages 1817–1824, 2013. 3, 33, 41, 56

[69] S. Paris. Edge-preserving smoothing and mean-shift segmentation of video streams. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, ECCV '08, pages 460–473, Berlin, Heidelberg, 2008. Springer-Verlag. 64

[70] S. Paris and F. Durand. A Topological Approach to Hierarchical Segmentation using Mean Shift. In *IEEE CVPR*, pages 1–8, 2007. 64

[71] M. Raptis, I. Kokkinos, and S. Soatto. Discovering discriminative action parts from mid-level video representations. In *CVPR*, 2012. 34, 66

[72] M. Raptis and L. Sigal. Poselet key-framing: A model for human activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 34

[73] A. Ravichandran, C. Wang, M. Raptis, and S. Soatto. Superfloxels: A mid-level representation for video sequences. In A. Fusiello, V. Murino, and R. Cucchiara, editors, *Computer Vision ECCV 2012. Workshops and Demonstrations*, volume 7585 of *Lecture Notes in Computer Science*, pages 131–140. Springer Berlin Heidelberg, 2012. 36

[74] M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach: a spatio-temporal maximum average correlation height filter for action recognition. In *In Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2008. 34

[75] S. Sadanand and J. Corso. Action bank: A high-level representation of activity in video. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1234–1241, June 2012. 27, 39

[76] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek. Image Classification with the Fisher Vector: Theory and Practice. *International Journal of Computer Vision*, 105(3):222–245, Dec. 2013. 41, 56

[77] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA, 2001. 20

[78] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 32–36 Vol.3, 2004. 3

[79] N. Shapovalova, M. Raptis, L. Sigal, and G. Mori. Action is in the eye of the beholder: Eye-gaze driven model for spatio-temporal action localization. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Neural Information Processing*

Systems, pages 2409–2417, 2013. 33, 56

[80] N. Shapovalova, A. Vahdat, K. Cannons, T. Lan, and G. Mori. Similarity constrained latent support vector machine: an application to weakly supervised action classification. In *ECCV*, 2012. 4, 34, 37, 56

[81] E. Sharon, M. Galun, D. Sharon, R. Basri, and A. Brandt. Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442(7104):719–846, June 2006. 65

[82] N. Shental, T. Hertz, D. Weinshall, and M. Pavel. Adjustment learning and relevant component analysis. In *Proceedings of the 7th European Conference on Computer Vision-Part IV*, pages 776–792, London, UK, 2002. Springer. 3, 11

[83] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug. 2000. 65

[84] N. I.-C. Shugao Ma, Jianming Zhang and S. Sclaroff. Action recognition and localization by hierarchical space-time segments. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2013. 27, 33, 34, 56

[85] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering objects and their location in images. In *10th IEEE International Conference on Computer Vision*, volume 1, pages 370 – 377 Vol. 1, October 2005. 2, 7

[86] E. Spriggs, F. De la Torre, and M. Hebert. Temporal segmentation and activity classification from first-person sensing. In *CVPR Workshop on Egocentric Vision*, pages 17 –24, June 2009. 2, 8, 22, 24

[87] C. Sun and R. Nevatia. Active: Activity concept transitions in video event classification. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 913–920, Dec 2013. 27

[88] K. D. Tang, F.-F. Li, and D. Koller. Learning latent temporal structure for complex event detection. In *Computer Vision and Pattern Recognition*, pages 1250–1257. IEEE, 2012. 33

[89] M. Tenorth, J. Bandouch, and M. Beetz. The TUM Kitchen Data Set of Everyday Manipulation Activities for Motion Tracking and Action Recognition. In *IEEE Int. Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS)*, 2009. 10

[90] J. Tighe and S. Lazebnik. Superparsing: Scalable nonparametric image parsing with superpixels. In *European Conference on Computer Vision*, pages 352–365, 2010. 26, 34

[91] A. Torralba and A. Efros. Unbiased look at dataset bias. In *Conference on Computer Vision and Pattern Recognition*, 2011. 7

[92] A. Torralba and A. Oliva. Statistics of natural image categories. *Network: Computation in Neural Systems*, 14:391–412, 2003. 3, 20

[93] D. Tran, J. Yuan, and D. Forsyth. Video event detection: From subvolume localization to spatiotemporal path search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(2):404–416, 2014. 34

[94] K. E. A. Van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1582–1596, Sept 2010. 80, 81, 82

[95] A. Vazquez-Reina, S. Avidan, H. Pfister, and E. Miller. Multiple hypothesis video segmentation from superpixel flows. In *ECCV*, 2010. 35

[96] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained K-means Clustering with Background Knowledge. In *Proceedings of 18th International Conference on Machine Learning*, pages 577–584, 2001. 3, 11

[97] H. Wang, A. Kläser, C. Schmid, and L. Cheng-Lin. Action Recognition by Dense Trajectories. In *Computer Vision and Pattern Recognition*, 2011. 3, 4, 25, 26, 27, 28, 29, 30, 31, 33, 37, 39, 40, 41, 43, 45

[98] H. Wang and C. Schmid. Action Recognition with Improved Trajectories. In *ICCV 2013 - IEEE International Conference on Computer Vision*, Sydney, Australia, Dec. 2013. IEEE. 45

[99] J. Wang, B. Thiesson, Y. Xu, and M. F. Cohen. Image and video segmentation by anisotropic kernel mean shift. In T. Pajdla and J. Matas, editors, *ECCV (2)*, volume 3022 of *Lecture Notes in Computer Science*, pages 238–249. Springer, 2004. 64

[100] L. Wang, Y. Qiao, and X. Tang. Motionlets: Mid-level 3D parts for human motion recognition. In *CVPR*, 2013. 3, 4, 25, 27, 29, 33, 36, 39

[101] X. Wang, L. Wang, and Y. Qiao. A comparative study of encoding, pooling and normalization methods for action recognition. In K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, editors, *ACCV (3)*, volume 7726 of *Lecture Notes in Computer Science*, pages 572–585. Springer, 2012. 41, 56

[102] J. V. D. Weijer, T. Gevers, and A. Bagdanov. Boosting color saliency in image feature detection. *IEEE TRANS. PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 28:150–156, 2006. 80

[103] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, volume 15, pages 505–512, 2002. 3, 11

[104] C. Xu, S. Whitt, and J. Corso. Flattening supervoxel hierarchies by the uniform entropy slice. In *ICCV*, 2013. 26, 35, 40, 59, 61, 62, 65, 67, 70, 88

[105] C. Xu, C. Xiong, and J. Corso. Streaming hierarchical video segmentation. In *ECCV*, 2012. 26, 34, 35, 37, 40, 61, 62, 63, 64, 66, 67, 68, 69, 71, 73, 74, 88, 89, 92, 94, 95

[106] J. Ye, Z. Zhao, and M. Wu. Discriminative K-means for clustering. In *Advances in Neural Information Processing Systems*, 2007. 3, 11, 13

[107] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *CVPR*, pages 2442–2449, 2009. 34

[108] H. Zha, C. Ding, M. Gu, X. He, and H. Simon. Spectral relaxation for K-means clustering. In *Neural Information Processing Systems*, pages 1057–1064, 2001. 11

[109] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision*, 73:213–238, June 2007. 2, 7

[110] Y. Zhang, X. Liu, M. Chang, W. Ge, and T. Chen. Spatio-temporal phrases for activity recognition. In *ECCV*, 2012. 3, 27, 29, 35, 36, 39

[111] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, pages 141–154, 2010. 34

[112] Y. Zhu, N. M. Nayak, and A. K. Roy-Chowdhury. Context-aware modeling and recognition of activities in video. In *Computer Vision and Pattern Recognition*, June 2013. 27, 34