

**SEGMENTAÇÃO HIERÁRQUICA DE VÍDEO  
USANDO ESCALA DE OBSERVAÇÃO**



KLEBER JACQUES FERREIRA DE SOUZA

**SEGMENTAÇÃO HIERÁRQUICA DE VÍDEO  
USANDO ESCALA DE OBSERVAÇÃO**

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

ORIENTADOR: PROF. DR. ARNALDO DE ALBUQUERQUE ARAÚJO  
COORIENTADOR: PROF. DR. SILVIO JAMIL FERZOLI GUIMARÃES

Belo Horizonte  
Novembro de 2016



KLEBER JACQUES FERREIRA DE SOUZA

**HIERARCHICAL VIDEO SEGMENTATION  
USING OBSERVATION SCALE**

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

ADVISOR: PROF. DR. ARNALDO DE ALBUQUERQUE ARAÚJO  
CO-ADVISOR: PROF. DR. SILVIO JAMIL FERZOLI GUIMARÃES

Belo Horizonte

November 2016



© 2016, Kleber Jacques Ferreira de Souza.  
Todos os direitos reservados

**Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG**

Souza, Kleber Jacques Ferreira de

S729h Hierarchical video segmentation using observation  
scale / Kleber Jacques Ferreira de Souza – Belo  
Horizonte, 2016.  
xxii: 80 f.: il.; 29 cm.

Tese (doutorado) - Universidade Federal de Minas  
Gerais – Departamento de Ciência da Computação.

Orientador: Arnaldo de Albuquerque Araújo  
Coorientador: Sílvio Jamil Ferzoli Guimarães

1. Computação – Teses. 2. Processamento de  
Imagens. 3. Videodigital. I. Orientador. II. Coorientador.  
III. Título.

CDU 519.6\*84(043)




UNIVERSIDADE FEDERAL DE MINAS GERAIS  
INSTITUTO DE CIÊNCIAS EXATAS  
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

## FOLHA DE APROVAÇÃO

Hierarchical video segmentation using an observation scale

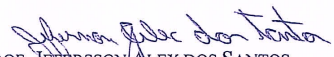
**KLEBER JACQUES FERREIRA DE SOUZA**

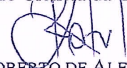
Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:


  
PROF. ARNALDO DE ALBUQUERQUE ARAÚJO - Orientador  
Departamento de Ciência da Computação - UFMG

  
PROF. SILVIO JAMIL FERZOLI GUIMARÃES - Coorientador  
Instituto de Informática - PUC/MG

  
PROF. HÉLIO PEDRINI  
Instituto de Computação - UNICAMP

  
PROF. JEFERSSON ALEX DOS SANTOS  
Departamento de Ciência da Computação - UFMG

  
PROF. ROBERTO DE ALENCAR LOTUFO  
Departamento Engenharia da Computação - UNICAMP

  
PROF. WILLIAM ROBSON SCHWARTZ  
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 24 de novembro de 2016.



# Acknowledgments

First and foremost, I would like to thank my supervisors, Prof. Arnaldo de Albuquerque Araújo and Prof. Silvio Jamil Ferzoli Guimarães for their continual support, guidance and inspiration. I am deeply grateful!

I would also like to thank the members of my examination board: our exchange of ideas enriched my understanding of my own work.

I am thankful to the NPDI lab and the VIPLAB lab for receiving me in their group.

Thanks to CAPES, CNPq and FAPEMIG for the financial support of this work. Without this funding, this work simply would not have happened.

My family has always been a strong source of support and inspiration. Thanks to my wife, Woolye Kuilayla, who underwent a series of personal and professional sacrifices so I could follow my dream, which he fully embraced as our dream. E por fim, eu serei eternamente grato aos meus pais, Salvador Ferreira e Neuza Ferreira; e meus irmãos, pelo amor, carinho, apoio e incentivo dado em todos os momentos de minha vida.



*“One picture is worth more than ten thousand words.”*  
(Unknown)



# Resumo

Este trabalho aborda o problema de segmentação hierárquica de vídeos. Segmentação hierárquica de vídeo é um conjunto de segmentações de vídeo em diferentes níveis de detalhe em que as segmentações em níveis de detalhe mais grosseiras podem ser produzidas a partir de fusões simples das regiões de segmentações em níveis de detalhes mais finos. O nível hierárquico corresponde a uma escala de observação, que pode ser processada de várias formas, dependendo da medida de dissimilaridade utilizada para calcular todas as escalas. Neste trabalho, a segmentação hierárquica de vídeo é transformada em um problema de particionamento de grafo em que cada parte corresponde a um *supervoxel* do vídeo. Assim, é apresentada uma nova abordagem para a segmentação hierárquica de vídeo, que calcula uma hierarquia de partições por uma reponderação do grafo original, usando uma medida de dissimilaridade simples em que uma segmentação *not too coarse* pode ser facilmente inferida. Usando a abordagem proposta foram desenvolvidos dois métodos de segmentação hierárquica de vídeo: um aplicado à segmentação de vídeos completo e outro aplicado a segmentação de vídeo por *streaming*. Também foi desenvolvida uma extensa análise comparativa, considerando as avaliações quantitativas que mostram a precisão, facilidade de uso e coerência temporal dos métodos propostos. Além disso, é proposto um método de *oversegmentation*, que melhora tanto a precisão quanto o custo computacional dos métodos propostos. De acordo com os resultados experimentais, a hierarquia inferida pelos métodos propostos produz bons resultados quando aplicada à segmentação de vídeo e comparados com os métodos encontrados na literatura.

**Palavras-chave:** Segmentação hierárquica de vídeo baseada em grafos, escala de observação, *supervoxel*.



# Abstract

This work addresses the problem of hierarchical video segmentation. Hierarchical video segmentation is a set of video segmentations at different detail levels in which the segmentations at coarser detail levels can be produced from simple merges of regions from segmentations at finer detail levels. The hierarchical level corresponds to a scale of observation, that can be processed in several ways, depending on the dissimilarity measure used to calculate all scales. In this work, the hierarchical video segmentation is transformed into a graph partitioning problem in which each part corresponds to one supervoxel of the video, thus we present a new approach for hierarchical video segmentation, which computes a hierarchy of partitions by a reweighting of the original graphs, using a simple dissimilarity measure in which a not too coarse segmentation can be easily inferred. Using the proposed approach, we developed two methods of hierarchical video segmentation, an applied full video segmentation; and another applied to streaming video segmentation. We also provide an extensive comparative analysis, considering quantitative assessments showing accuracy, ease of use, and temporal coherence of our methods. In addition, we propose an oversegmentation method, able to enhance the accuracy and the computational cost of our methods. According to the experimental results, the hierarchy inferred by our methods produced good results when applied to video segmentation and compared with the methods in the literature.

**Palavras-chave:** Hierarchical Graph-Based Video Segmentation, Observation Scale, Supervoxel.





# List of Figures

1.1	Example image: (a) original image, (b) labeled image. . . . .	1
1.2	Example of frames from video: The original frames are illustrated in the first row and the labeled frames in the second row. . . . .	2
1.3	Example of hierarchical segmentation. . . . .	4
1.4	Three different processing paradigms for video segmentation. [Xu et al., 2012]. . . . .	8
2.1	Segmentation result obtained by [Felzenszwalb and Huttenlocher, 2004].	14
2.2	Segmentation result obtained by GBH segmentation. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained by GBH with different hierarchical levels. . . . .	15
2.3	Example for computing the hierarchical scale for an edge-weighted graph. For this example, we suppose that all scales for regions $X$ and $Y$ are already computed, and we will calculate the hierarchical scale for the edge $\{B, G\}$ . . . . .	24
3.1	Outline of HOScale method: the video is transformed into a video graph (step 1); the hierarchy is computed from the video graph (step 2); the identification of video segments is made from hierarchy (step 3); and finally, (step 4) the graph is transformed in the segmented video. . . . .	27

3.2	Video segmentation with different video graph creations. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained using xyt graph, rgbxy graph and rgbxyt graph, respectively. . . . .	30
3.3	Video segmentation with different thresholds. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained using rgbxyt graph and <i>min_size</i> 0.3%, and 0.5%. . . . .	31
3.4	Video segmentation with different number of segments. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained using rgbxy graph, thresholds 0.5% and 3, 5 and 11 segments. . . . .	32
3.5	Outline of the StreamHOScale method: the video is transformed into a video graph (step 1); the hierarchy is computed from the video graph (step 2); the identification of video segments is made from hierarchy (step 3); the supervoxels output is calculated based on previously processed frames (step 4); and finally, the video graph is transformed in the video segmented (step 5). . . . .	33
3.6	Full and Streaming video segmentation. The original frames are illustrated in the first row. The full video segmentation, using FVHOScale, is illustrated in second row. The streaming video segmentation, using StreamHOScale with number of 2-sized frame block, is illustrated in third row. . . . .	35
3.7	Streaming video segmentation. Segmentation of the first block. . . . .	35
3.8	Streaming video segmentation. Segmentation of the second block. . . . .	36
3.9	Streaming segmentation. Segmentation of the first and second blocks. . . . .	36
3.10	Streaming video segmentation. Segmentation of the first and second blocks with the temporal propagation. . . . .	37
3.11	Streaming video segmentation. Segmentation of the first and second blocks with the temporal propagation. . . . .	37
3.12	Streaming video segmentation. Segmentation of the third and final block. . . . .	38
3.13	Streaming video segmentation. Segmentation of the all blocks with the temporal propagation. . . . .	38

3.14	An example of the temporal propagation using StreamHOScale method.	39
3.15	Outline of oversegmentation method: the video is transformed into a color graph (step 1); the hierarchy is computed from the color graph (step 2); the identification of colors segments is made from hierarchy (step 3); and finally, (step 4) the graph is transformed in the video, using the average of the colors of each connected component. . . . .	40
3.16	Video segmentation with oversegmentation. The original frames are illustrated in the first row. The results obtained using FVHOScale without oversegmentation are in the second row. The following rows, from top to bottom, illustrate the results obtained using oversegmentation with threshold of the minimum size of the supervoxels, 2, 10, 25 e 50, respectively. . . . .	42
4.1	Example images from SegTrack dataset. The original frames are illustrated in the first row and groundtruth frames in the second row. . . .	44
4.2	Example images from Chen dataset. The original frames are illustrated in the first row and groundtruth frames in the second row. . . . .	45
4.3	Example images from GaTech dataset. . . . .	45
4.4	A visual explanation of the distinct nature of 3D boundaries in video (please view in color). We overlap each frame to compose a volumetric video, the green colored area which is a part of girl in frame $i$ should not be counted as a part of girl in frame $i + 1$ , similarly the red area which is a part of girl in frame $i + 1$ should not be counted as a part of girl in frame $i$ . The lower right graph shows the 3D boundary along the time axis (imagine you are looking through the paper). From [Xu and Corso, 2012]. . . . .	47
4.5	A comparison with the parameters used in the creation of the video graph, separated by type of graph. The comparison is based on the Explained Variation metric. It shows the average value for the three datasets (SegTrack, Chen and GaTech). . . . .	49

4.6	A comparison with the parameters used in the creation of the video graph, separated by type of graph. The comparison is based on the Mean Duration metric. It shows the average value for the three datasets (SegTrack, Chen and GaTech).	49
4.7	A comparison with the variation of the threshold for minimum size supervoxel parameter, separated per dataset. The comparison is based on the Explained Variation metric.	51
4.8	A comparison with the variation of the threshold for minimum size supervoxel parameter, separated by dataset. The comparison is based on the Mean Duration metric.	52
4.9	A comparison with the variation of the size of frame block parameter, separated per dataset. The comparison is based on the Explained Variation metric.	53
4.10	A comparison with the variation of the size of frame block parameter, separated per dataset. The comparison is based on the Mean Duration metric.	53
4.11	A comparison with the variation of the threshold for minimum size supervoxel in oversegmentation, separated per dataset. The comparison is based on the Explained Variation metric.	55
4.12	A comparison with the variation of the threshold for minimum size supervoxel in oversegmentation, separated per dataset. The comparison is based on the Mean Duration metric.	55
4.13	A comparison among our methods, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale, and the methods GB, GBH, StreamGBH, SWA, Meanshift and Nyström when applied to SegTrack dataset.	57
4.14	A comparison among our methods, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale, and the methods GB, GBH, StreamGBH, SWA, Meanshift and Nyström when applied to Chen dataset.	58

4.15	A comparison among our methods, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale, and the methods GB, GBH, StreamGBH, SWA, Meanshift and Nyström when applied to GaTech dataset. . . . .	59
4.16	A comparison among our methods, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale, and the methods GB, GBH, StreamGBH, SWA, Meanshift and Nyström. It shows the average value for the three datasets (SegTrack, Chen and GaTech). . . . .	60
4.17	Examples of video segmentations for a video extracted from the Chen dataset. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained by GB, GBH, StreamGBH, SWA, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale respectively. The parameters were tuned to obtain about 50 supervoxels. . . . .	64
4.18	Examples of video segmentations for a video extracted from the Chen dataset. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained by GB, GBH, StreamGBH, SWA, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale respectively. The parameters were tuned to obtain about 100 supervoxels. . . . .	65
4.19	Examples of video segmentations for a video extracted from the GaTech dataset. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained by GB, GBH, StreamGBH, SWA, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale respectively. The parameters were tuned to obtain about 50 supervoxels. . . . .	66
4.20	Examples of video segmentations for a video extracted ofrom the GaTech dataset. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained by GB, GBH, StreamGBH, SWA, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale respectively. The parameters were tuned to obtain about 100 supervoxels. . . . .	67

4.21	A comparison among our methods, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale, and the methods GBH, StreamGBH and SWA when applied to SegTrack, Chen and GaTech datasets. The comparison is based on the following metrics: (a) time cost; (b) space cost. . . . .	68
A.1	A comparison with the parameters used in the creation of the video graph, separated by dataset using the xyt graph. The comparison is based on the Explained Variation metric. . . . .	78
A.2	A comparison with the parameters used in the creation of the video graph, separated by dataset using the xyt graph. The comparison is based on the Mean Duration metric. . . . .	78
A.3	A comparison with the parameters used in the creation of the video graph, separated by dataset using the rgbxy graph. The comparison is based on the Explained Variation metric. . . . .	79
A.4	A comparison with the parameters used in the creation of the video graph, separated by dataset using the rgbxy graph. The comparison is based on the Mean Duration metric. . . . .	79
A.5	A comparison with the parameters used in the creation of the video graph, separated by dataset using the rgbxyt graph. The comparison is based on the Explained Variation metric. . . . .	80
A.6	A comparison with the parameters used in the creation of the video graph, separated by dataset using the rgbxyt graph. The comparison is based on the Mean Duration metric. . . . .	80

# List of Tables

3.1	Variation of the number of colors according to the threshold. . . . .	41
-----	-----------------------------------------------------------------------	----





# Contents

<b>Acknowledgments</b>	<b>ix</b>
<b>Resumo</b>	<b>xiii</b>
<b>Abstract</b>	<b>xv</b>
<b>List of Figures</b>	<b>xvii</b>
<b>List of Tables</b>	<b>xxiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	5
1.2 Challenges . . . . .	7
1.3 Hypotheses . . . . .	9
1.4 Contributions . . . . .	9
1.5 Outline . . . . .	10
<b>2 Background Knowledge</b>	<b>13</b>
2.1 Related Work . . . . .	13
2.2 Graphs, Hierarchies and Properties . . . . .	17
2.3 Observation Scale . . . . .	20
<b>3 Hierarchical Video Segmentation</b>	<b>27</b>
3.1 Full Video Segmentation . . . . .	28
3.1.1 Video graph creation . . . . .	29
3.1.2 Observation scale computation . . . . .	31

3.1.3	Hierarchical level selection . . . . .	32
3.1.4	Segmented video creation . . . . .	33
3.2	Streaming Video Segmentation . . . . .	33
3.3	Video Segmentation with Oversegmentation . . . . .	39
3.4	Final Remarks . . . . .	41
<b>4</b>	<b>Experimental results</b>	<b>43</b>
4.1	Experimental setup . . . . .	43
4.1.1	Datasets . . . . .	43
4.1.2	Methods . . . . .	46
4.1.3	Metrics . . . . .	46
4.1.4	Implementation issues . . . . .	47
4.2	Parameter Learning . . . . .	48
4.2.1	Parameters for video graph creation . . . . .	48
4.2.2	Parameter for minimum size supervoxel . . . . .	51
4.2.3	Parameter for StreamHOScale . . . . .	52
4.2.4	Parameters for oversegmentation . . . . .	54
4.3	Comparison to state of the art . . . . .	56
4.3.1	Quantitative analysis . . . . .	56
4.3.2	Qualitative analysis . . . . .	61
4.3.3	Computational cost . . . . .	61
4.4	Final Remarks . . . . .	62
<b>5</b>	<b>Conclusions and future works</b>	<b>69</b>
	<b>Bibliography</b>	<b>71</b>
<b>A</b>	<b>Parameters for video graph creation per dataset</b>	<b>77</b>

# Chapter 1

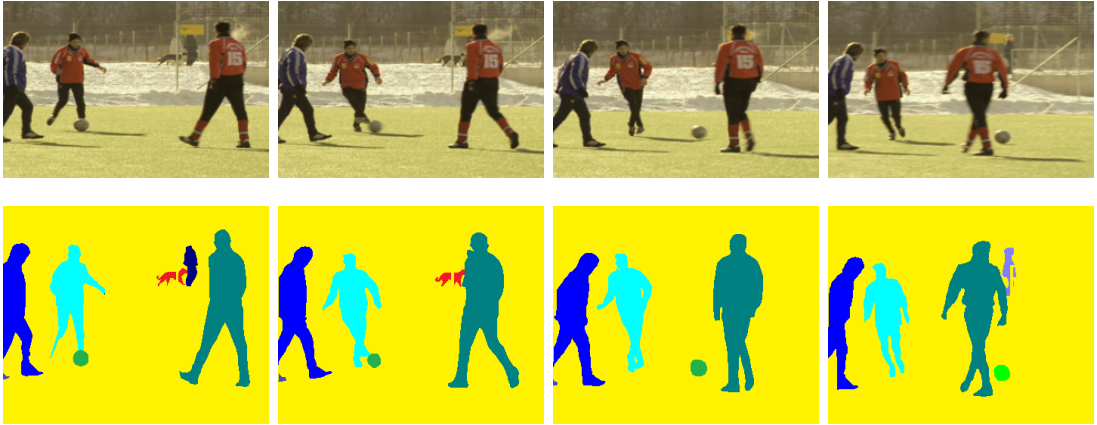
## Introduction

Content-Based Visual Information Retrieval (CBVIR) is still a major challenge for Computer Science. The main objective is the development of autonomous systems that reproduce the capabilities of the human visual system to interpret visual data that is able to adequately respond to visual stimuli. However, interpreting visual data is not an easy task for computers because an image may have thousands of pixels. So, an automatic image interpretation may become a difficult task and, often, very expensive. For example, what do you see in Figure 1.1(a)? Do you see people, a dog, a ball? For the human eye, it could be easy to identify these objects, but the computer must compute 38,400 pixels with 17,362 distinct colors. Thus, the computer needs to organize the pixels so that they can be represented as an object of interest, as illustrated in Figure 1.1(b), where the red color represents the dog, the green color represents the ball and so on.



**Figure 1.1.** Example image: (a) original image, (b) labeled image.

The process of grouping perceptually similar pixels into regions is known as image segmentation [Gonzalez and Woods, 2011] which, as shown in Figure 1.1, is not an easy task for the computer. The segmentation process organizes the input data in structures with relevant semantic content. These structures correspond to objects, or parts of objects, that will aid in the image interpretation and analysis process [Pedrini and Schwartz, 2007]. When working with videos, the challenge is even greater, since the visual information of each image, also called frame, must be temporally propagated according to the next image. As can be seen in Figure 1.2, the visual information (for example, the ball labeled with green color) changes into spatiotemporal form in the video frame sequences.



**Figure 1.2.** Example of frames from video: The original frames are illustrated in the first row and the labeled frames in the second row.

The interpretation of the data in a video is a complex activity. So a step of video segmentation may be necessary to partition the data set into structures with relevant semantic content to aid in the analysis process. We can find in the literature several algorithms of video segmentation, which mostly are extensions of image segmentation techniques. Some of these algorithms simply apply image segmentation techniques to the video frames without considering temporal coherence [Chen et al., 2007; Winnemoller et al., 2006]. Others can preserve the temporal information as supervoxels, which is a set of spatially continuous voxels<sup>1</sup> that have similar appearance (intensity, color, texture, etc.) [Veksler et al., 2010; Xu et al.,

<sup>1</sup>A voxel has three coordinates  $(x; y; t)$ , in which time is represented as the third dimension.

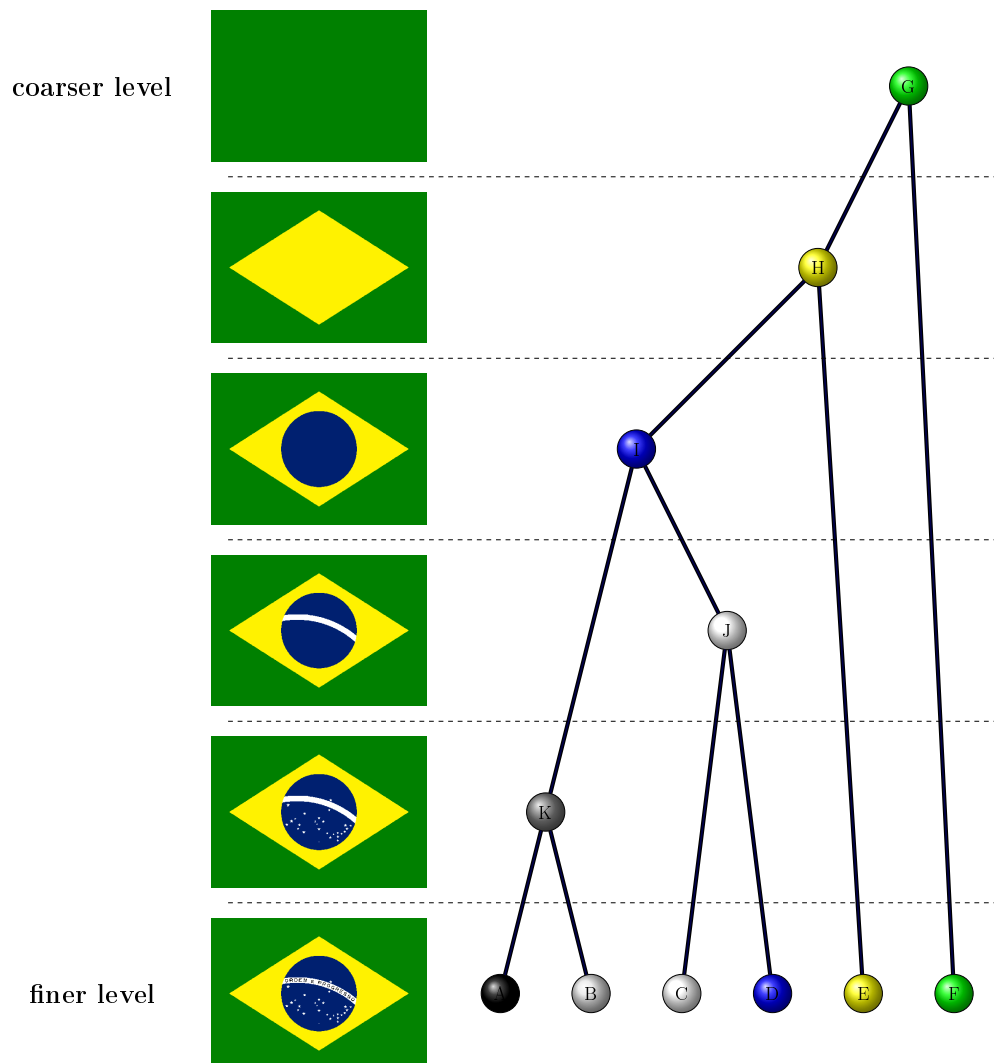
2012]. As can be seen in Figure 1.2, the ball labeled with green color and the dog labeled with red color are both supervoxels note that supervoxels may have different sizes and shapes that vary according to the visual content processed. Moreover, note that people (in Figure 1.2) are represented by different supervoxels. Once a supervoxel is only one spatially continuous voxel, each person would be a different supervoxel.

The video segmentation in supervoxels is not an easy task, since it is necessary to preserve the spatiotemporal coherence of segmented information in every frame of the video. In this work, we address the hierarchical video segmentation subject, that extends the concepts of hierarchical image segmentation, in order to consider the spatiotemporal information present in a video. A hierarchical image segmentation is a set of image segmentations at different detail levels in which the segmentations at coarser detail levels can be produced from simple merges of regions from segmentations at finer detail levels, as illustrated in Figure 1.3. Therefore, the segmentations at finer levels are nested with respect to those at coarser levels. Hierarchical methods have the interesting property of preserving spatial and neighboring information among segmented regions [Guimarães et al., 2012]. The hierarchical level corresponds to a scale of observation, that can be processed in several ways, depending on the dissimilarity measure used to calculate all scales, and the relationship among them, as proposed by Felzenszwalb and Huttenlocher [2004].

The majority of the hierarchical segmentation methods produces hierarchical levels from the merger of the previous level, and so on, but with a high computational cost, as a coarser level always depends on computation of all previous levels. Guimarães et al. [2012] proposed an efficient hierarchical segmentation method based on the observation scales, that produces the complete set of segmentations at every scale. Thereby, it is possible to provide all scales of observations, instead of only one segmentation level.

In this work, we present a new approach to hierarchical video segmentation, which extends the image segmentation method proposed by Guimarães et al. [2012] to video segmentation. We use observation scale applied to the hierarchical video segmentation. Our approach removes the need for parameter tuning in the calculations of hierarchical levels. In other words, the proposed hierarchical

video segmentation approach is not dependent on the hierarchical level, and consequently, it is possible to compute any level without computing the previous ones, thus the time for computing a segmentation is almost the same for any specified level. Our approach for hierarchical video segmentation using an observation scale computes a hierarchy of partitions by a reweighting of the original graph in which a segmentation can be easily inferred, and the temporal coherence is related to the graph transformation used. Exploring this graph transformation and the use of scale observation in hierarchical video segmentation is the subject of this thesis.



**Figure 1.3.** Example of hierarchical segmentation.

## 1.1 Motivation

Along with recent technological advances of computers in general, the video data has become more and more accessible and plays an increasingly important role in our everyday life. Today, even commonly used consumer hardware, such as notebooks, mobile phones, and digital photo cameras, allow to create videos. At the same time, faster internet access and growing storage capacities enable to directly publish and share videos with others.

However, despite the increasing importance of video data, the possibilities to analyze it in an automated fashion are rather limited. Computer vision systems are far behind the capabilities of human vision. For instance, video search in large scale databases archives is currently only feasible with costly manual annotation. Web search engines commonly rely mainly on textual data, such as descriptions or tags, in order to retrieve relevant videos.

Another example are human action classification applications. The task of human action classification is the process of naming human actions based on the video content and can be defined as follows: given a pre-determined set of actions, we need to classify an unlabelled action from an video in one of these types.

Action recognition and classification in videos is a topic of interest in many recent researches. Extending the two main stages discussed by Poppe [2010], several researches focus their models in (i) description, (ii) representation, (iii) classifiers and (iv) data filtering.

We remark the effort applied to the task due to some different points, as the relevance of the topic, such its application to security systems or data retrieval framework, or as the growing amount of data available requiring feasible computation with limited resources. This scenario creates a growing trend of using temporal video segmentation as preprocessing for action recognition, aiming to create a better representation of the action movement or data reduction for video processing. The idea is that the segmentation methods could partitionate videos into coherent constituent parts, to easily carry out the classification based on the obtained segments. Most segmentation works rely on creating a better representation of actions.

Niebles et al. Niebles et al. [2010] proposed a strategy for modeling temporal

structure of decomposable motion segments for activity classification. They used a discriminative model that encodes a temporal decomposition of video sequences, and appearance models for each motion segment. In Zhou and Wang [2012], Qiang and Gang proposed a new representation of local spatio-temporal cuboids based on atomic actions that represent the basic units of human actions.

In Sekma et al. [2013], the authors presented a motion descriptor for human action recognition that is based on both the accordion representation of the video and its temporal segmentation into elementary motion segments.

In Ma et al. [2013], the authors presented a representation called hierarchical space-time segments. This representation creates a tree with video segments containing body parts, detected using a boundary map. The tree is then pruned using visual cues exploring the relation between segments. The remaining segments are tracked forward and backward in time and represented by a Bag-of-Word (BoW) model. The goal of this approach is to create an unsupervised method to extract relevant space-time segments to represent a video.

In Guo et al. [2016], the authors focused in a mid-level representation of a graph-based approach, that combines two descriptions to represent the video: one spatio-temporal segmentation using motion word and dense trajectories, and an iterative procedure to obtain discriminative patches by a learning-based clustering. They use the identified patch to locate the discriminative area and describe the supervoxels that fall inside this area using a Histogram of Oriented Gradient.

Spatio-temporal motion and appearance context information around pixels can deliver more complex motion and appearance structures. In Peng et al. [2013], the authors proposed a motion boundary based dense sampling strategy, called Dense Trajectories Motion Boundary DT-MB, to reduce the number of trajectory preserving the power of dense trajectory using a group of spatio-temporal context descriptor. Likewise, in Yi and Lin [2015], the authors proposed a mid-level approach to represent and model the spatio-temporal relationship of video elements for the purpose of human activity classification in unconstrained environments. In Jain et al. [2014], the authors proposed the use of tubelets, *i.e.*, mid-level representation from successive mergings of the spatio-temporal segmentations, to perform action localization.

In the same way, we understand that video segmentation can be an important



task in the analysis and interpretation of video data, because it facilitates the decomposition of video in structures with relevant semantic content that to aid in the automatic processing by computers.

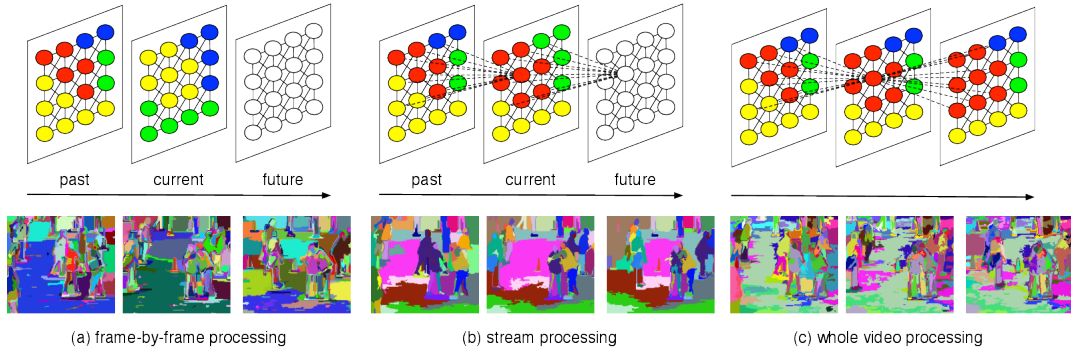
## 1.2 Challenges

According to Grundmann et al. [2010a], are three major challenges for developing methods of video segmentation comprise: temporal coherence, automatic processing and scalability.

Temporal coherence is related to the propagation of visual information present in each frame of the video consistently. When performing image segmentation, the expected result is the grouping of its visual information individually. When working with the video segmentation, the result expected is that the individual information of each frame is propagated consistently to subsequent frames, thus, increasing the efficiency of segmentation. Getting a good video segmentation and keeping the most of the spatiotemporal coherence is a major challenge because of the volume of data that is processed. There are at least three different processing paradigms for video segmentation [Xu et al., 2012] that can influence the treatment of temporal coherence, as illustrated in Figure 1.4:

- **Frame-by-frame:** processing such that each frame is independently segmented, but no temporal information is used. Even though it is fast, the results and temporal coherence are poor. Although frame-by-frame processing is efficient and can achieve high-performance in spatial respects, its temporal stability is limited.
- **Stream processing:** segmenting the video by parts where the segments of the current frame is based only on a few previously processed frames. It is an online processing, and the results are good and efficient in terms of time and space complexity, where each video frame is processed only once and does not change the segmentation of previous frames.
- **Whole video processing (full video):** uses all the video frames in the segmentation process, creating a 3D volume processing that represents a model for

the whole video. It is a bidirectional multi-pass process. The results are the best, but the complexity is too high to process long and streaming videos. This approach has high computational cost and often impractical in long videos.



**Figure 1.4.** Three different processing paradigms for video segmentation. [Xu et al., 2012].

Automatic processing is related to the correct identification of video objects (semantically related regions) and its variation over the video, without knowing a priori, which regions to track, what frames contain those regions, or the time-direction for tracking (forward or backward). The objective is to get a good segmentation of video objects without the need for a manual process, i.e., without the need for a user to indicate any feature which facilitates the segmentation process.

Since scalability is related to computer resources required to perform the video segmentation, given the large amount of pixels or features in a video, video segmentation approaches tend to be slow and to need large memory. Consequently, previous advances concentrate on short video sequences (usually less than a second) or reduce complexity, which can adversely affect long-term temporal coherence. The majority of the full video segmentation methods end up consuming a lot of memory and so is applied to short videos, thus limiting their use. In addition, the streaming methods have better scalability because they require a smaller amount of memory.

Despite the large amount of existing video segmentation methods, they still have limitations related to these challenges. This thesis proposes a hierarchical

video segmentation approach that uses the observation scale with the goal of extending the good results found in the image segmentation application. Thereby, we obtain a video segmentation method that is scalable, that has a good coherence spatiotemporal and present a good segmentation of the objects presents in the video.

### 1.3 Hypotheses

Our main hypothesis is *that applying the Observation Scale method to the video segmentation will produce an effective hierarchical video segmentation.*

Also, the following secondary hypotheses are considered and validated in this thesis:

- The ways to transform the video data into a graph can change the effectiveness of the observation scale method.
- By using the observation scale method, it is possible to obtain a video segmentation method that has a good temporal coherence, is scalable and presents a correct segmentation.

Formally, the thesis problem statement can be formulated as follows.

*Given a video, how to represent its visual content to performing an effective segmentation by using an observation scale?*

### 1.4 Contributions

The main result of this thesis is a graph-based hierarchical video segmentation approach using an observation scale that produces good quantitative and qualitative results when applied to video segmentation. We propose two methods to apply our approach; using the full video and another using the streaming method. The great advantage of streaming method is the ability to run a video stream without the need of having all the video in memory, achieving to segment of consecutive frames blocks considering the temporal information present throughout the video.

In addition, we propose an oversegmentation method, which also uses our observation scale approach to improve the accuracy and the computational cost of our methods for video segmentation.

During the development of this research, parts of the work were published in order to disseminate the results achieved, namely:

## Journal

- De Souza, K. J. F., Araújo, A. de A., Patrocínio Jr., Z. K. G., and Guimarães, S. J. F. (2014). **Graph-based hierarchical video segmentation based on a simple dissimilarity measure**. *Pattern Recognition Letters*, 47(0):85 - 92.

## Conferences

- De Souza, K. J. F., Araújo, A. de A., Patrocínio Jr., Z. K. G., Cousty, J., Najman, L., Kenmochi, Y., Guimarães, S. J. F. (2016). **Decreasing the number of features for improving human action classification**. In *SIBGRAPI 2016 (XXIX Conference on Graphics, Patterns and Images)*, São José dos Campos, Brazil.
- De Souza, K. J. F., Araújo, A. de A., Guimarães, S. J. F., Patrocínio Jr., Z. K. G., and Cord, M. (2015). **Streaming graph-based hierarchical video segmentation by a simple label propagation**. In *SIBGRAPI 2015 (XXVIII Conference on Graphics, Patterns and Images)*, Salvador, Brazil.
- De Souza, K. J. F., Araújo, A. de A., Patrocínio Jr., Z. K. G., Cousty, J., Najman, L., Kenmochi, Y., and Guimarães, S. J. F. (2013). **Hierarchical video segmentation using an observation scale**. In *SIBGRAPI 2013 (XXVI Conference on Graphics, Patterns and Images)*, Arequipa, Peru.

## 1.5 Outline

This work is organized as follows.

**Chapter 2 - Background Knowledge.** In Chapter 2, we present the related works in hierarchical video segmentation with a discussion of the characteristics related to the objectives of this work. Moreover, we present some fundamental concepts which serve as basis for the understanding and development of this work.

**Chapter 3 - Hierarchical Video Segmentation.** In Chapter 3, we provide a detailed description of our approach to hierarchical video segmentation.

**Chapter 4 - Experimental Results.** In Chapter 4, we present the experimental results with a detailed quantitative and qualitative analysis.

**Chapter 5 - Conclusions and future works.** Finally, we present our concluding remarks and discuss future work directions.

**Appendix A - Parameters for video graph creation per dataset.** In Appendix A, we present some additional results of our experiments.



# Chapter 2

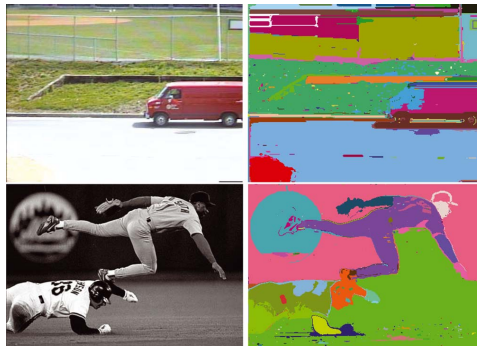
## Background Knowledge

In this chapter, we present some fundamental theoretical knowledge to the understanding of this work. In Section 2.1, we present related works on hierarchical video segmentation with a discussion of the characteristics aiming to the objectives of this work. In Section 2.2 presents some fundamental concepts about graphs, hierarchies and the dissimilarity measure adopted in this work. Finally, in Section 2.3 we present the concept of the observation scale used in this work.

### 2.1 Related Work

In this work, video segmentation refers to the segmentation of videos into different spatiotemporal regions, called supervoxels. It has been an active research topic in the last few years, and some dataset are available to compare different video segmentation algorithms [Xu and Corso, 2012; Galasso et al., 2013; Xu and Corso, 2016]. More specifically, we address the subject of hierarchical graph based video segmentation. Graph-based segmentation methods are commonly employed for video segmentation, the nodes represent the superpixels or supervoxels and edges represent the similarities [Felzenszwalb and Huttenlocher, 2004; Grundmann et al., 2010a]. Each supervoxel defines a spatiotemporal region in video that has similar information, like color, texture, motion, etc. The supervoxel hierarchies contain rich multiscale decompositions of video content, where various structures can be found at various levels.

A hierarchy can be represented by a tree, specially, as minimum spanning tree, and its use for image segmentation was popularized by Felzenszwalb and Huttenlocher [2004], who proposed a graph-based algorithm for image segmentation (so-called GB), that define a predicate for measuring the evidence for a boundary between two regions using a graph-based representation of the image. Their algorithm runs in time nearly linear in the number of image pixels, which makes it suitable for extension to spatiotemporal segmentation. A sample output from this segmentation algorithm is shown in Figure 2.1.

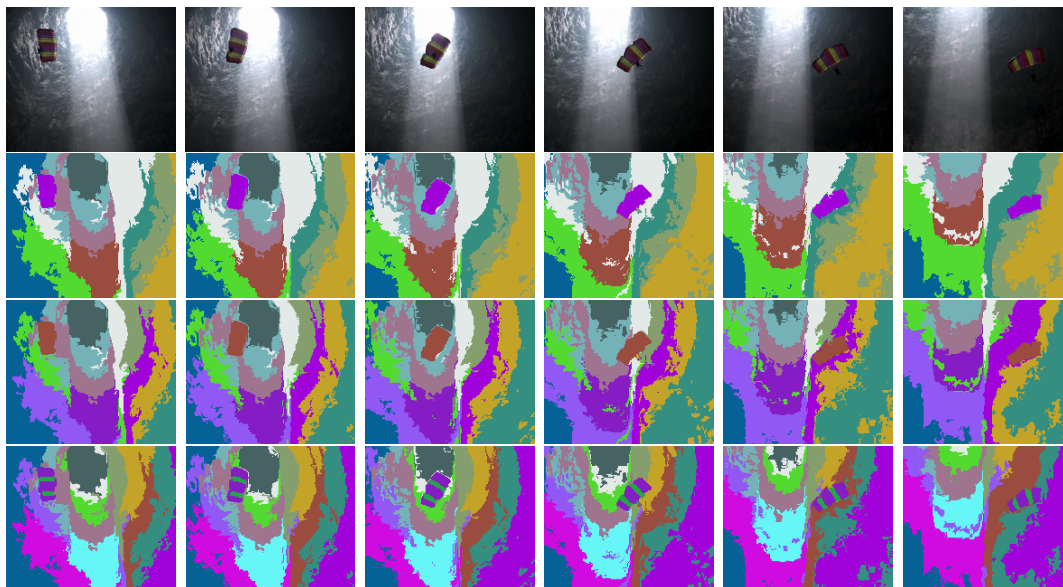


**Figure 2.1.** Segmentation result obtained by [Felzenszwalb and Huttenlocher, 2004].

In [Grundmann et al., 2010a], the authors proposed a hierarchical graph-based video segmentation algorithm (so-called GBH), that applies the same technique presented by Felzenszwalb and Huttenlocher [2004] to merge regions. Regions are described by local Lab histograms. At each step of the hierarchy, the edge weights are set to be the  $\chi^2$  distance between the Lab histograms of the connected two regions. Their algorithm builds on an oversegmentation of the above spatiotemporal graph-based segmentation. It then iteratively constructs a region graph over the obtained segmentation, and forms a bottom-up hierarchical tree structure of the region (segmentation) graphs. Even though this method presents high quality segmentations with a good temporal coherence and with stable region boundaries, for computing a video segmentation according to a specified level, it is necessary to compute all lower (finer) segmentations. A sample output from GBH segmentation is shown in Figure 2.2.

The methods based on Nyström method [Fowlkes et al., 2001, 2004] and





**Figure 2.2.** Segmentation result obtained by GBH segmentation. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained by GBH with different hierarchical levels.

segmentation by weighted aggregation (so-called SWA) [Sharon et al., 2000, 2006; Corso et al., 2008] optimize the same normalized cut criterion [Shi and Malik, 2000]. In [Fowlkes et al., 2001], the Nyström approximation was proposed to solve the eigenproblem. Their work demonstrated segmentation on relatively low-resolution, short videos and randomly sample points from the first, middle, and last frames. However, this method is not scalable as the number of supervoxels and the length of video increase. Sampling too many points makes the Nyström method require too much memory, while sampling too few gives unstable and low performance. The approach based on SWA proposed by [Sharon et al., 2000, 2006; Corso et al., 2008] is other alternative approach to optimize the normalized cut criterion. The SWA computes iteratively the hierarchy considering for high hierarchical levels the previous ones. Moreover, it uses an algebraic multigrid solver to efficiently compute the hierarchy. The algorithm is nearly linear in the number of input voxels, and produces a hierarchy of segmentations, which motivates its extension to a supervoxel method.

The mean shift segmentation (so-called MeanShift) proposed in [Comaniciu

and Meer, 2002] for image segmentation was applied to temporal sequences in [Paris and Durand, 2007]. Moreover, this work also introduced the Morse theory to interpret mean shift as a topological decomposition of the feature space into density modes. A hierarchical video segmentation is created by using topological persistence.

Although these methods have presented good results for segmenting videos, as shown in Xu and Corso [2012], there are still some factors that prevent their use in real applications, *i.e.*, memory consumption and processing time make unfeasible to apply those methods to medium and large videos. In [Xu et al., 2012], the authors proposed the first streaming hierarchical video segmentation method (so-called StreamGBH). The experimental results indicate that StreamGBH outperforms other streaming video segmentation methods and performs nearly as well as the full-video hierarchical graph-based method GBH. But, since it adopts GBH [Grundmann et al., 2010a] as a component, it also has the same limitations, *i.e.*, it is necessary to compute all lower (finer) segmentations. Furthermore, in [Tripathi et al., 2014], the authors proposed an improvement to StreamGBH method by incorporating motion information using early and mid-level visual processing. Similarly, a Sub-Optimal Low-Rank Decomposition method was introduced in [Li et al., 2015, 2016] for the graph-based streaming video segmentation.

We can find in the literature other methods that exploit the streaming video segmentation by using temporal superpixels. The temporal superpixel method computes the superpixel segmentation on the first frame and then extends the existing superpixels to subsequent frames (one by one) in a video. Therefore, this set of methods [Chang et al., 2013; Tsai et al., 2010; Bergh et al., 2013; Reso et al., 2013], by their nature, computes supervoxels in a streaming fashion, which is similar to streamGBH with a streaming window of one frame. The temporal superpixel method [Chang et al., 2013] uses a Gaussian Process for the streaming segmentation.

In this work, we propose two methods of video segmentation using an observation scale, one applied to the full video and another applied to streaming video. In order to provide a comparative analysis, we used the benchmark and library LIBSVX proposed in [Xu and Corso, 2012], which contains the implementation of the following video segmentation methods: GB [Felzenszwalb and Huttenlocher,

2004], GBH [Grundmann et al., 2010a], StreamGBH [Xu et al., 2012], Nyström [Fowlkes et al., 2001] and SWA [Corso et al., 2008].

## 2.2 Graphs, Hierarchies and Properties

First, we define a supervoxel according to Xu and Corso [2012]. Given a 3D lattice  $\Lambda^3$  (the voxels in the video), a supervoxel  $sv$  is a subset of the lattice  $sv \subset \Lambda^3$  such that the union of all supervoxels comprises the lattice and they are pairwise disjoint:  $\bigcup_i sv = \Lambda^3$  and  $sv_i \cap sv_j = \emptyset, \forall i, j$  pairs.

Let  $V$  be a set. We denote by  $\mathcal{P}(V)$  the set of all subsets of  $V$ . Let  $x$  be an element of  $V$ . Let  $\mathcal{P}_x(V)$  be the set of all subsets of  $V$  which contains the element  $x$ . A subset  $\mathbf{P} \subseteq \mathcal{P}(V)$  is called a *partition (of  $V$ )* if the intersection of any two distinct elements of  $\mathbf{P}$  is empty and if the union of all elements in  $\mathbf{P}$  is equal to  $V$ . If  $\mathbf{P}$  is a partition, each element of  $\mathbf{P}$  is called a *region (or class)* of  $\mathbf{P}$ . The set of all partitions of  $V$  is denoted by  $\Pi_V$  and the set of all regions of all partitions of  $V$  is denoted by  $\Gamma_V$ . Let  $\mathbf{P}$  and  $\mathbf{P}'$  be two partitions of  $V$ . We say that  $\mathbf{P}'$  is a *refinement* of  $\mathbf{P}$  if any region of  $\mathbf{P}'$  is included in a region of  $\mathbf{P}$ . A set  $\mathcal{H} = \{\mathbf{P}_\lambda \in \Pi_V \mid \lambda \in \mathbb{N}\}$  of (indexed) partitions is called a (*indexed*) *hierarchy* if for any two positive integers  $\lambda_1$  and  $\lambda_2$  such that  $\lambda_1 \geq \lambda_2$ , the partition  $\mathbf{P}_{\lambda_2}$  is a refinement of  $\mathbf{P}_{\lambda_1}$ .

We define a (undirected) *graph* as a pair  $G = (V, E)$  where  $V$  is a finite set and  $E$  is composed of unordered pairs of  $V$ , *i.e.*,  $E$  is a subset of  $\{\{x, y\} \subseteq V \mid x \neq y\}$ . Each element of  $V$  is called a *vertex of  $G$* , and each element of  $E$  is called an *edge of  $G$* . A graph is said to be *connected* if every pair of vertices in the graph is connected. An *edge-weighted graph* is a pair  $W = (G, w)$  where  $G$  is a graph and  $w$  is a map from  $E(G)$  into  $\mathbb{N}$ . A *tree* is a graph  $T = (V, E)$  which is connected and has only  $|V| - 1$  edges and an *edge-weighted tree* is a pair  $U = (T, w)$  in which  $T$  is a tree and  $w$  is a map from  $E(T)$  into  $\mathbb{N}$ . A *spanning tree* of a connected, undirected graph  $G = (V, E)$  – represented by  $T(G)$  – is a tree  $T = (V', E')$  in which  $V' = V$  and  $E' \subseteq E$ . For an edge-weighted graph  $W = (G, w)$  one associates an *edge-weighted spanning tree*  $U(G) = (T(G), w)$  in which  $T(G)$  is a spanning tree of  $G$  and  $w$  is a map from  $E(T)$  into  $\mathbb{N}$ . This map could be used to assign a weight to a spanning tree by computing the sum of the

weights of all edges in that spanning tree, *i.e.*,  $w(U) = \sum_{\{x,y\} \in E(T(G))} w(\{x,y\})$ . A *minimum spanning tree* (MST) of  $G$  – represented by  $\text{MST}(G)$  – is then a spanning tree of  $G$  with weight less than or equal to the weight of every other spanning tree of  $G$ .

Let  $W = (G, w)$  be an edge-weighted graph. Then, to any threshold  $\lambda \in \mathbb{N}$ , one may associate the partition  $\mathbf{P}_\lambda^W \in \mathcal{P}(V)$  induced by the graph made by  $V(G)$  and the edges in  $E(G)$  whose weight is less than  $\lambda$ . This partition is called the *partition induced by  $W$  at level  $\lambda$* . It is well known [Cousty and Najman, 2011; Morris et al., 1986] that for any two values  $\lambda_1$  and  $\lambda_2$  such that  $\lambda_1 \geq \lambda_2$ , the partition  $\mathbf{P}_{\lambda_2}^W$  is a refinement of  $\mathbf{P}_{\lambda_1}^W$ . Hence, the set  $\mathcal{H}^W = \{\mathbf{P}_\lambda^W \mid \lambda \in \mathbb{N}\}$  is a hierarchy. This hierarchy is called the *hierarchy induced by  $W$* .

Let us remember some definitions of region-merging criterion. The criterion for region-merging in Felzenszwalb and Huttenlocher [2004] measures the evidence for a boundary between two regions by comparing two quantities: one based on intensity differences across the boundary, and the other based on intensity differences between neighboring pixels within each region. More precisely, in order to know whether two regions must be merged, two measures are considered: the *internal difference*  $\text{Int}(X)$  and the *difference*  $\text{Diff}(X, Y)$  between two neighboring regions  $X$  and  $Y$ . The *internal difference*  $\text{Int}(X)$  of a region  $X$  is the highest edge weight among all the edges linking two vertices of  $X$  in the MST, while the *difference*  $\text{Diff}(X, Y)$  between two neighboring regions  $X$  and  $Y$  is the smallest edge weight among all the edges that link  $X$  to  $Y$ . Thus, for merging two adjacent regions  $X$  and  $Y$ , it is necessary to verify the following region merging predicate:

$$\text{MergePred}(X, Y) = \begin{cases} \text{true} & \text{if } \text{Diff}(X, Y) \leq \text{MInt}(X, Y) \\ \text{false} & \text{otherwise} \end{cases} \quad (2.1)$$

where the minimal internal difference  $\text{MInt}(X, Y)$  is defined as:

$$\text{MInt}(X, Y) = \min\{\text{Int}(X) + \tau(X), \text{Int}(Y) + \tau(Y)\} \quad (2.2)$$

and the threshold function  $\tau$  controls the degree to which the difference between two components must be greater than their internal differences in order for there

to be evidence of a boundary between them. For small components,  $Int(X)$  is not a good estimate of the local characteristics of the data. In the extreme case, when  $|X| = 1$ , we have  $Int(X) = 0$ . Therefore, in Felzenszwalb and Huttenlocher [2004], a threshold function based on the size of the component is used, *i. e.*:

$$\tau(X) = \frac{k}{|X|} \quad (2.3)$$

with a constant parameter  $k$ .

Then, two regions  $X$  and  $Y$  are merged when:

$$Diff(X, Y) \leq \min \left\{ Int(X) + \frac{k}{|X|}, Int(Y) + \frac{k}{|Y|} \right\} \quad (2.4)$$

where  $k$  is a parameter used to prevent the merging of large regions (*i. e.*, larger  $k$  forces smaller regions to be merged).

The merging criterion defined by Eq. (2.4) depends on the scale  $k$  at which the regions  $X$  and  $Y$  are observed. More precisely, let us consider the (*observation*) scale  $S_Y(X)$  of  $X$  relative to  $Y$  as a measure based on the difference between  $X$  and  $Y$ , on the internal difference of  $X$  and on the size  $|X|$  of  $X$ :

$$S_Y(X) = (Diff(X, Y) - Int(X)) \times |X|. \quad (2.5)$$

Then, the scale  $S(X, Y)$  is simply defined as:

$$S(X, Y) = \max(S_Y(X), S_X(Y)). \quad (2.6)$$

Thanks to this notion of a scale, Eq. (2.4) can be written as:

$$k \geq S(X, Y). \quad (2.7)$$

Thus, two adjacent regions  $X$  and  $Y$  must be merged at scale  $k$  if their dissimilarity measure is smaller than or equal to  $k$ .

Let  $\mathbf{P}$  be a partition and let  $X$  be a region of  $\mathbf{P}$ . Let  $Y$  be a region of  $\mathbf{P}$ . We say that  $X$  and  $Y$  are *two adjacent regions* if there exists an edge  $\{x, y\}$  of  $T$  such that  $x$  belongs to  $X$  and  $y$  belongs to  $Y$ ; in this case, we also say that the edge  $\{x, y\}$

links  $X$  and  $Y$ . Furthermore, if  $X$  and  $Y$  are adjacent, the (*observation*) *scale*, denoted by  $S(X, Y)$ , is given by  $S(X, Y) = \max(S_Y(X), S_X(Y))$ . In the following, we define some properties.

Intuitively, a partition is too fine when there exists two adjacent regions that should be merged according to scale instead of being separated.

Let  $\mathbf{P}$  and  $\mathbf{Q}$  be two partitions. If  $\mathbf{Q}$  is a refinement of  $\mathbf{P}$  and  $\mathbf{Q} \neq \mathbf{P}$ , we say that  $\mathbf{Q}$  is a proper refinement of  $\mathbf{P}$ .

Intuitively, a partition is too coarse if the splitting of one of its regions leads to a partition which is not too fine.

Given an edge weighted graph  $W$  and an integer  $\lambda$ , the algorithm proposed in Felzenszwalb and Huttenlocher [2004] produces a partition that is neither too fine nor too coarse at scale  $\lambda$ . However, as discussed in Section 2.1, when the scale  $\lambda$  varies, the set of obtained partitions, indexed by scales, is not a hierarchy. Moreover, there is no hierarchy of partitions such that, for each index  $\lambda \in \mathbb{N}$ , the partition indexed by  $\lambda$  is neither too fine nor too coarse at scale  $\lambda$ . Therefore, in order to obtain a hierarchy based on scales, one of the two properties on partitions must be relaxed. Thus, we define a hierarchy too coarse as follows.

It can be easily seen that there exist hierarchies that are not coarse, in which for any index  $\lambda \in \mathbb{N}$ , the partition indexed by  $\lambda$  is not too fine at scale  $\lambda$ . The trivial partition is composed by singletons, in which there does not exist a proper refinement that is not too fine.

## 2.3 Observation Scale

In this work, we extend the observation scale method, proposed by Guimarães et al. [2012], to apply it to video segmentation. We use the video graph to produce another edge-weighted graph, the scale map  $\mathcal{M}$ , in which the edge weights correspond to the observation scales. Actually, once that initial video graph  $W = (G, w)$  is generated, it is further simplified into a MST  $U_v(G) = (T(G), w)$  using the edge weights. Then, an ordering of the edges of  $E(T)$  is used to control the creation of a scale map  $\mathcal{M}$ , which is in fact an edge-weighted graph, *i.e.*,  $\mathcal{M} = (G_{\mathcal{M}}, w_{\mathcal{M}})$ , in which  $V(G_{\mathcal{M}}) = V(T)$  and  $E(G_{\mathcal{M}}) \subseteq E(T)$ .

Our methodology for hierarchical segmentation does not explicitly produce a hierarchy of partitions, but instead it produces a scale map from which the desired hierarchy can be inferred. It starts from a minimum edge-weighted spanning tree  $U(G) = (T(G), w)$  of the video graph  $W = (G, w)$  (which was built from the video). In order to compute the scale associated with each edge of  $T$ , our methodology iteratively considers the edges of  $E(T)$  in a non-decreasing order of their original weights  $w$  (illustrated in Algorithm 1), *i.e.*, the method is based on the analysis of a fusion tree oriented by merging of regions according to non-decreasing edge weight. Starting with an empty map  $\mathcal{M}_0 = (G_{\mathcal{M}}^0, w_{\mathcal{M}})$ , in which  $V(G_{\mathcal{M}}^0) = V(T)$  and  $E(G_{\mathcal{M}}^0) = \emptyset$ , for every edge  $e$  of  $E(T)$ , the new scale map  $\mathcal{M}_{i+1} = (G_{\mathcal{M}}^{i+1}, w_{\mathcal{M}})$  is generated by adding the edge  $e$  to  $E(G_{\mathcal{M}}^i)$ , *i.e.*,  $E(G_{\mathcal{M}}^{i+1}) = E(G_{\mathcal{M}}^i) \cup \{e\}$ , and by setting its weight to the hierarchical scale computed from the Algorithm 2. Note that, the computation of the new weights depends on the predicate that is used for comparing two adjacent regions. In this work, we use the predicate proposed by Felzenszwalb and Huttenlocher [2004]. Moreover, the kernel of our methodology, presented in Algorithm 2, is based on identification of the lower scale value that can be used to merge a region to another one while guaranteeing that there are no other two regions with another scale smaller than that one.

**Algorithm 1** Compute scale map  $\mathcal{M} = (G_{\mathcal{M}}, w_{\mathcal{M}})$ . Let  $U(G) = (T(G), w)$  be the MST of an image graph  $W = (G, w)$ .

- (0) Generate an empty map  $\mathcal{M}_0 = (G_{\mathcal{M}}^0, w_{\mathcal{M}})$ , in which  $E(G_{\mathcal{M}}^0) = \emptyset$ .
- (1) Sort  $E(T)$  into  $\pi = (e_1, \dots, e_m)$  by non-decreasing edge weights.
- (2) Let  $\mathbf{P}^{V(G)} = \{\{v_i\} \mid v_i \in V(G)\}$  be an initial partition of  $V(G)$ .
- (3) Let  $x$  and  $y$  be two vertices of  $V(G)$  that are connected by  $i$ -th non evaluated edge  $e$  in the ordering.
- (4) Find the region  $X$  of  $\mathbf{P}^{V(G)}$  that contains  $x$ .
- (5) Find the region  $Y$  of  $\mathbf{P}^{V(G)}$  that contains  $y$ .
- (6) Considering that  $\mathcal{M}_i = (G_{\mathcal{M}}^i, w_{\mathcal{M}})$ , compute the hierarchical observation scale using  $\mathcal{M}_i$

$$hscale = \max\{S_Y(x), S_X(y)\}$$

- (7) Generate a new scale map  $\mathcal{M}_{i+1} = (G_{\mathcal{M}}^{i+1}, w_{\mathcal{M}})$  by adding the edge  $e$  to  $E(G_{\mathcal{M}}^i)$ , *i.e.*,  $E(G_{\mathcal{M}}^{i+1}) = E(G_{\mathcal{M}}^i) \cup \{e\}$ , and by setting its weight to the hierarchical scale computed, *i.e.*,  $w_{\mathcal{M}}(e) = hscale$ .
- (8) Let  $\mathbf{P}^{V(G)} = \mathbf{P}^{V(G)} \setminus \{X, Y\} \cup \{\{z_i \mid z_i \in X \cup Y\}\}$ .
- (9) Repeat steps (3)-(8) until there is no edge left.
- (10) Set final scale map to that last map calculated, *i.e.*,  $\mathcal{M} = \mathcal{M}_{|V|-1} = (G_{\mathcal{M}}^{|V|-1}, w_{\mathcal{M}})$ .
- (11) Return  $\mathcal{M}$ .

**Algorithm 2** Compute hierarchical scale of a region that contains  $x$  relative to  $Y - S_Y(x)$ . Here, we consider the MST  $U(G) = (T(G), w)$  such that  $\mathcal{M}_i = (G_{\mathcal{M}}^i, w_{\mathcal{M}})$ , as defined in Algorithm 1,  $Int^{\mathcal{M}_i}(X)$  is the *internal difference* of  $X$  in  $\mathcal{M}_i$ , *i.e.*,  $Int^{\mathcal{M}_i}(X) = \max\{w_{\mathcal{M}}(\{x, y\}) \mid \{x, y\} \in E(G_{\mathcal{M}}^i), x \in X, y \in X\}$ , and  $S_Y^U(X)$  is calculated according to Eq. (2.5) using MST  $U(G)$ .

- (0) Let  $\mathcal{H}^{\mathcal{M}_i}$  be the hierarchy induced by  $\mathcal{M}_i$  and let  $\Gamma_{V(G_{\mathcal{M}}^i)}$  be the set of all regions of all partitions of  $\mathcal{H}^{\mathcal{M}_i}$ .
- (1) Let  $\mathcal{P}_x(V(G)) \subseteq \Gamma_{V(G_{\mathcal{M}}^i)}$  be the set of regions which contains the element  $x$ .
- (2) Find the largest set,  $X^* \in \mathcal{P}_x(V(G))$ , in terms of size, in which  $S_Y^U(X^*) \geq Int^{\mathcal{M}_i}(X^*)$ , *i.e.*,  $S_Y^U(X^*) \geq Int^{\mathcal{M}_i}(X^*)$  and  $|X^*| \geq |X'|, \forall X^*, X' \in \mathcal{P}_x(V(G))$ .
- (3) Compute  $\mathbf{C}_{X^*} = \{C_i \in \mathcal{P}_x(V(G)) \mid X^* \subset C_i\}$ .
- (4) If  $\mathbf{C}_{X^*} = \emptyset$  then  $S_Y(x) = S_Y^U(X^*)$  else  $S_Y(x) = \min\{\min\{Int^{\mathcal{M}_i}(C_i) \mid C_i \in \mathbf{C}_{X^*}\}, S_Y^U(X^*)\}$ .
- (5) Return  $S_Y(x)$ .

**Lemma 1.** *Hierarchical scale* At step (6) of Algorithm 1, the hierarchical observation scale  $hscale(e)$  is the lower scale of edge  $e$  in which some region of  $X$  will be merged to some region of  $Y$  such that for any value greater than or equal to  $hscale(e)$ , the dissimilarity measure between them is smaller than it.

**Proof.** Assume without loss of generality that  $hscale(e) = S_Y(x)$ , which is related to the largest set that contains the vertex  $x$  with respect to the regions of  $Y$  (step 2 of Algorithm 2). Suppose now that there exists  $hscale(e)' < hscale(e)$  in which



the region of  $X$  will be merged to the region of  $Y$ . In this case, the dissimilarity measure between  $X$  and  $Y$  should not be greater than  $h\text{scale}(e)'$ . This is a contradiction since  $h\text{scale}(e)' < h\text{scale}(e) = S_Y(\mathcal{M}_i, x)$ . ■

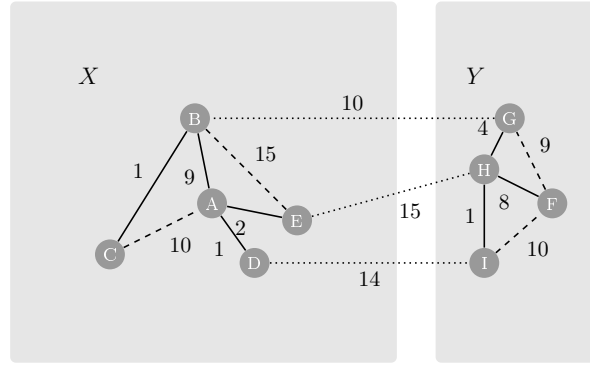
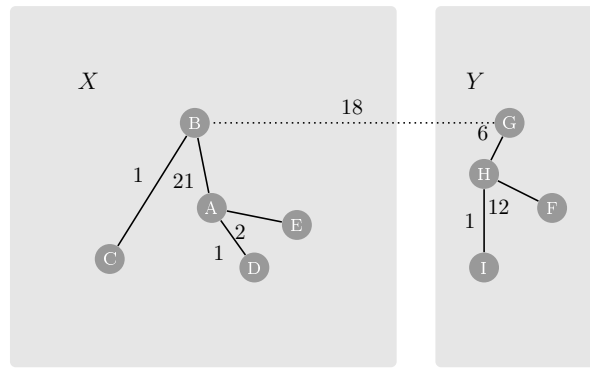
**Theorem 1.** *The hierarchy of partitions is not too coarse The hierarchy of partitions computed by Algorithm 1 is not too coarse.*

*Proof.* Let  $\mathcal{H} = \{\mathbf{P}_\lambda \mid \lambda \in \mathbb{N}\}$  be the hierarchy of partitions computed by Algorithm 1. Suppose that  $\mathcal{H}$  is too coarse, thus there exists  $\lambda \in \mathbb{N}$  such that  $\mathbf{P}_\lambda$  is too coarse at scale  $\lambda$ . Therefore,  $\mathbf{P}_\lambda$  should have a proper refinement  $\mathbf{P}_R$  that is not too fine at scale  $\lambda$ , *i.e.*, for every pair of adjacent regions  $X$  and  $Y$  in  $\mathbf{P}_R$ ,  $S(X, Y) > \lambda$ . But Lemma 1 guarantees that for every pair of adjacent regions  $X$  and  $Y$  the dissimilarity value between them is smaller than the hierarchical scale calculated at step (6) of Algorithm 1, *i.e.*,  $S(X, Y) < \lambda$ . This is a contradiction and, therefore, the hierarchy of partitions computed by Algorithm 1 is not too coarse. □

Thanks to Theorem 1, the segmentations inferred from the hierarchy of partitions are not too coarse.

We will explain the steps of our algorithms using an example. Let us illustrate the computation of a hierarchical observation scale on the graph of Fig. 2.3(a). To this end, we consider the iteration of the algorithm at which the edge  $e$  linking  $B$  to  $G$  is analyzed. At this step, the edges of the MST of weight below  $w(e) = 10$  have been already processed. Therefore, the hierarchical observation scale of these edges (depicted by continuous lines in the figure) is already known, as shown in Fig. 2.3(b). The regions  $X$  and  $Y$  obtained at steps (4) and (5) are set to  $\{A, B, C, D, E\}$  and  $\{F, G, H, I\}$ , respectively. Then, in order to find the value  $w_{\mathcal{M}}(e)$  at steps (6) and (7) of Algorithm 1, all partitions (for each region) must be considered.

Firstly, let us analyse the region  $X$  of Fig. 2.3(b). The set of all subsets of  $X$  which contains the vertex  $x$  and the induced graph has only one connected component which is  $\mathcal{P}_x(V(G)) = \{\{B\}, \{B, C\}, \{A, B, C, D, E\}\}$ . In step 2 (Algorithm 2), we look for the largest set in which the hierarchical observation scale is greater than or equal to the internal difference of the new re-weighted tree. Thus,

(a) Video graph  $W$ (b) Scale map  $\mathcal{M}$ 

**Figure 2.3.** Example for computing the hierarchical scale for an edge-weighted graph. For this example, we suppose that all scales for regions  $X$  and  $Y$  are already computed, and we will calculate the hierarchical scale for the edge  $\{B, G\}$ .

suppose that  $X^* = \{A, B, C, D, E\}$ , as  $S_Y^U(X^*) = (10 - 9) \times 5 = 5$  is smaller than  $Int^{\mathcal{M}_i}(X^*) = 21$  (which is the highest edge weight of  $X^*$ ), then it is necessary to verify for another set. Now, for  $X^* = \{B, C\}$ , the  $S_Y^U(X^*) = (10 - 1) \times 2 = 18$  is greater than or equal to  $Int^{\mathcal{M}_i}(X^*) = 1$ , then  $S_Y(x) = 18$ , which is the minimum between 18 and 21 (steps 2, 3 and 4 of Algorithm 2). The same process is made for  $S_X(y)$ . The set of all subsets of  $Y$  which contains the vertex  $y$  and the induced graph has only one connected component, which is  $\mathcal{P}_y(V(G)) = \{\{G\}, \{G, H, I\}, \{F, G, H, I\}\}$ , here  $S_X(y) = 12$ , which is the minimum between 12 and 18. Finally, the hierarchical observation scale of  $X$  and  $Y$  is 18 ( $= \max\{S_Y(x), S_X(y)\} = \max\{18, 12\}$ ).

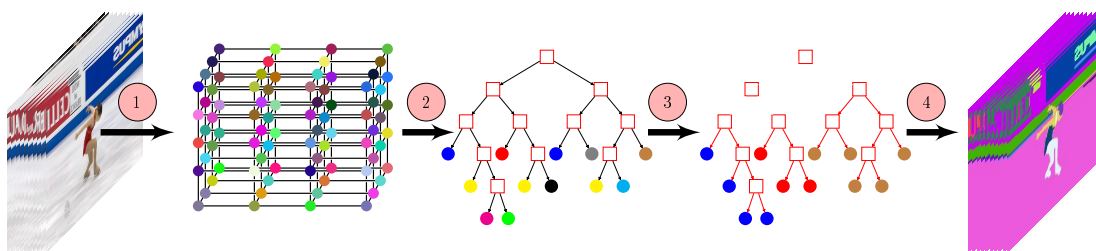
Finally, suppose that we will segment the graph illustrated in Fig. 2.3(b) in two regions and, for sake of simplicity, that all scales have distinct values. There are two possibilities for this operation: either we try to find directly a threshold for the scales that correctly identifies the desired number of regions or, after sorting the edges in descending order with respect to scale values, we remove those with larger values until the desired number of regions is obtained.



## Chapter 3

# Hierarchical Video Segmentation

Our strategy for hierarchical video segmentation using an observation scale, called HOScale, is illustrated in Fig. 3.1. Our approach is based on computation of (hierarchical) scales that indicate when any two adjacent regions must be correctly merged according to a specified predicate. Furthermore, instead of computing these scales directly from the video, the scales are calculated on a graph generated from the video to be segmented. Thereby, we propose a new approach to hierarchical video segmentation, which extends the method proposed by Guimarães et al. [2012] to video segmentation using spatiotemporal information.



**Figure 3.1.** Outline of HOScale method: the video is transformed into a video graph (step 1); the hierarchy is computed from the video graph (step 2); the identification of video segments is made from hierarchy (step 3); and finally, (step 4) the graph is transformed in the segmented video.

As illustrated in Fig. 3.1, the video is transformed into a video graph (an edge-weighted graph, step 1 in Fig. 3.1), that will be used to produce another edge-weighted graph, called *scale map*  $\mathcal{M}$  (step 2 in Fig. 3.1) in which the edge weights correspond to the scales for which two adjacent regions connected by this edge are correctly merged, *i.e.*, there are no two subregions of these regions that might be merged before these regions for that scale value. Moreover, instead of computing the hierarchy of partitions, any desired hierarchy can be inferred from the scale map produced before (step 3 in Fig. 3.1), *e.g.*, by removing from the map those edges from the map whose weight is greater than a specific value of scale. In other words, for partitioning the graph it is sufficient to apply a thresholding on the edge weights to remove them for creating connected components on the graph. Remember that connected components of the graph are related to the supervoxels on the video, thus, each supervoxel is transformed into a unique segment of video, characterized by a specific color in segmented video (step 4 in Fig. 3.1).

We propose two video segmentation methods; using the full video and another using the streaming method. In Section 3.1, we present our first method for hierarchical video segmentation using an observation scale applied to full video. In Section 3.2, we present an evolution of the method HOScale to video streaming. In addition, we proposed an oversegmentation method, which is used as a preprocessing step on the video, in order to perform a segmentation of the color space and makes our methods faster, as we present in Section 3.3. Finally, in Section 3.4, we present the main contributions of our methods.

## 3.1 Full Video Segmentation

Our full video segmentation approach, called FVHOScale (Full Video Hierarchical segmentation using an Observation Scale), is the application of HOScale approach to the whole video processing. This means that all stages of HOScale will run in the full video. For this it is necessary to define and understand how each of the present steps will be implemented in the HOScale, as detailed in the following sections.

### 3.1.1 Video graph creation

The graph creation (step 1 in Fig. 3.1) is a very important step in this kind of application since it models the type of information to be used into vertices, and the relationships between the elements of the video, into edges. In fact, a major difficulty is to design an adequate edge-weighted graph to well represent the video content. The most common way to perform the graph creation is to split the video into frames and set the frame pixels as vertices. The edges are 26-adjacency pixel relationship (spatiotemporal) weighted by a simple color gradient computed by the Euclidean distance in the RGB color space [Felzenszwalb and Huttenlocher, 2004; Grundmann et al., 2010a]. In this work, we propose to use other ways for transforming a video into a graph. Our goal is to improve the representativeness of the video information, for this we relate the color space with the spatiotemporal information of the video pixels. Here, we consider three ways:

1. **xyt graph**: the underlying graph is the one induced by the 26-adjacency pixel relationship, where the edges are weighted by a simple color gradient computed by the Euclidean distance in the RGB color space, using the Equation 3.1. Here, only the spatiotemporal information of the pixel in the space is evaluated.

$$\sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2} \quad (3.1)$$

2. **rgbxy graph**: the underlying graph is the xyt graph together with the  $K$  nearest neighbors in rgbxy space, where the edges are weighted by a gradient computed by the Euclidean distance in the rgbxy space, where each pixel is represented by its color, in RGB color space, and its spatial coordinates  $(x,y)$ , in the current frame, using the Equation 3.2.

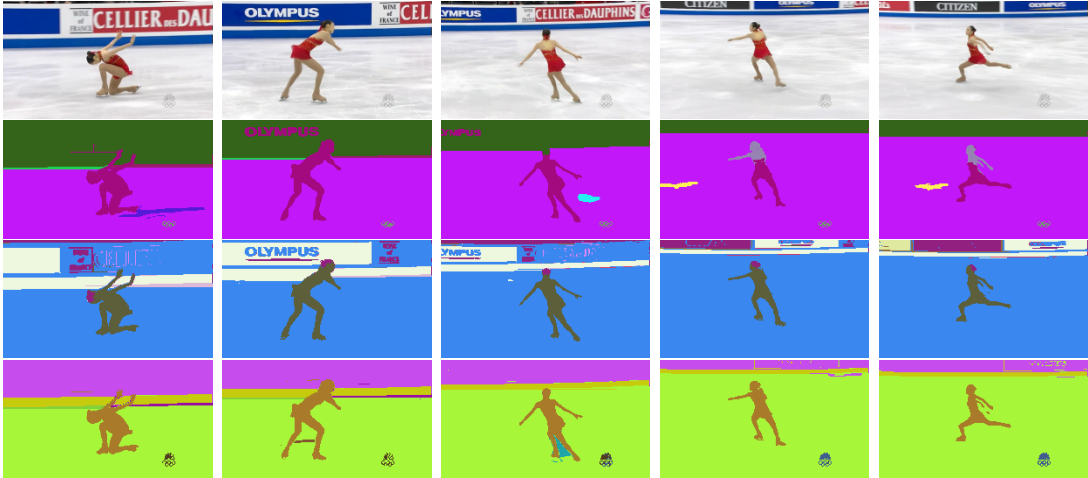
$$\sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2 + (X_1 - X_2)^2 + (Y_1 - Y_2)^2} \quad (3.2)$$

3. **rgbxyt graph**: the underlying graph is the xyt graph together with the  $K$  nearest neighbors in rgbxyt space, where the edges are weighted by a gradient

computed by the Euclidean distance in the  $rgbxyt$  space, where each pixel is represented by its color, in RGB color space, its spatial coordinates  $(x,y)$ , and its temporal coordinates  $(t)$ , in the current frame, using the Equation 3.3.

$$\sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2 + (X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (T_1 - T_2)^2} \quad (3.3)$$

In Figure 3.2, we illustrate different results achieved with the three ways to generate the video graph. We can see that the way to create the graph and relate the pixels can change the result of the segmentation significantly. Moreover, there are other ways of relating the video pixels as well as different ways to represent them, for example, using other color spaces, like HSV, YCbCr, Lab, etc. It is also possible to apply a 2D image filter (e.g. Gaussian filter) to each frame to remove noise. Thereby, in our approach we did not set the way to transform the video to a graph. Thus, we can explore various ways, which will be discussed in Chapter 4.



**Figure 3.2.** Video segmentation with different video graph creations. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained using  $xyt$  graph,  $rgbxy$  graph and  $rgbxyt$  graph, respectively.

Furthermore, it is important to note that it is at this stage, where we will

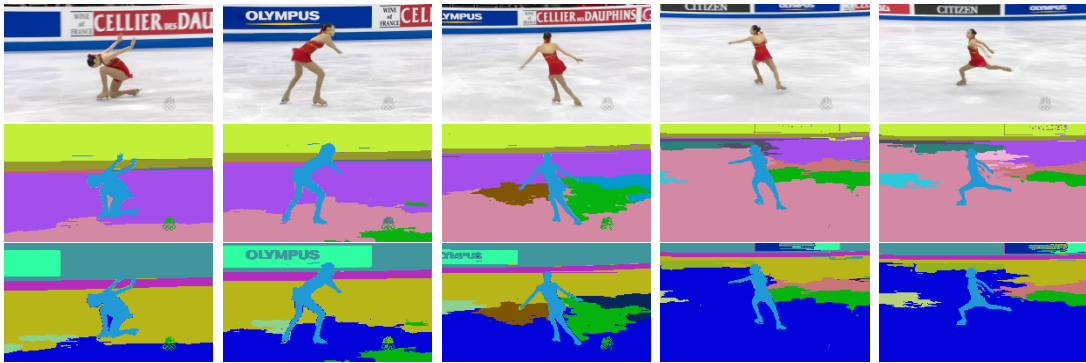


generate a huge amount of information, because we use all video frames. For example, if a video has resolution of 240x160 pixels and 100 frames, this means that the graph created has 3,840,000 vertices and 49,217,396 edges (disregarding the edge pixels that have less edges). Thus, it is impractical to apply this method to large videos.

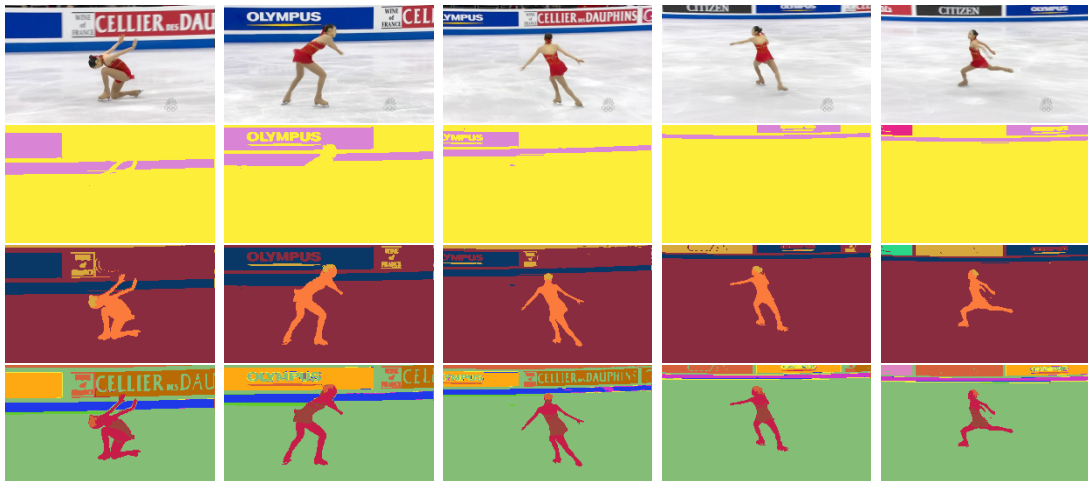
### 3.1.2 Observation scale computation

Using the graph created in the previous step we will produce a new graph, performing the observation scale computation, as shown in Section 2.3. The new graph is the scale map  $\mathcal{M}$  and will correspond to the minimum spanning tree in which the edge weights correspond to the hierarchical scales.

After the generation of the scale map  $\mathcal{M}$ , a low visual representation in finer levels can occur, for example, a video segment with few pixels that are not visually observed. Therefore it is necessary to perform a post-processing step that limits the minimum size of a segment generated. For this, we use the same approach used in [Grundmann et al., 2010a], that performs the union of all segments smaller than a threshold (*min\_size*) until there are no segments with size smaller than the threshold informed. This threshold may be a constant value (e.g., amount of component pixel) or variable according to the video (e.g., minimum percentage of pixels according to all information that is being processed).



**Figure 3.3.** Video segmentation with different thresholds. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained using rgbxyt graph and *min\_size* 0.3%, and 0.5%.



**Figure 3.4.** Video segmentation with different number of segments. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained using rgbxy graph, thresholds 0.5% and 3, 5 and 11 segments.

In Figure 3.3, we illustrate the use of a threshold that can influence the visual result, where the threshold represents the percentage of all pixels present in the video. As we can see in Figure 3.3, the value of threshold is related to the size of the objects present in the video. When we use a small threshold, it generates small objects and when we use a large threshold it eliminates small objects and enhances large objects.

### 3.1.3 Hierarchical level selection

The scale map  $\mathcal{M}$  is represented by a minimum spanning tree. Thus, to set the observation scale that you want to display of the segmented video, just hold the graph partitioning according to the level of details to be reached. If you want a level with more details, it is required to perform more partitions on the graph when you want a coarser segmentation, just do less cuts on the graph. To generate  $S$  supervoxels, it is necessary to cut  $S - 1$  edges of the minimum spanning tree.

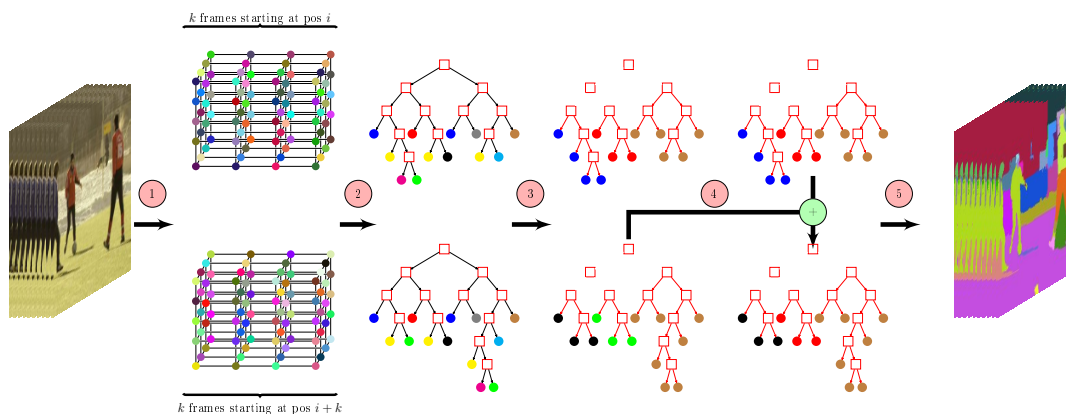
In Figure 3.4, we show how the choice of the number of supervoxels can change the result of segmentation. We can have more or less details, depending on the chosen hierarchical level.

### 3.1.4 Segmented video creation

Finally, the partitioned graph is converted into a video in which each partition represents a supervoxel. To facilitate the identification and visualization, each supervoxel is colored with a random and distinct color.

## 3.2 Streaming Video Segmentation

Although some video segmentation methods have presented good results, as shown in [Xu and Corso, 2012, 2016], the memory consumption and processing time are prohibitive issues for their use in real applications, mainly due to huge quantity of video information to be processed. Thus, instead of considering all video information, we divide the video into  $k$ -sized frame blocks in order to cope with those problems without losing the qualitative performance for the segmentations. So, we propose a graph-based streaming video segmentation, so called StreamHOScale, in which we apply the HOScale method to segment each  $k$ -sized frame block, followed by a new and simple strategy for merging the segmentations of two consecutive blocks. Our streaming method can be outlined in five steps, as illustrated in Figure 3.5.

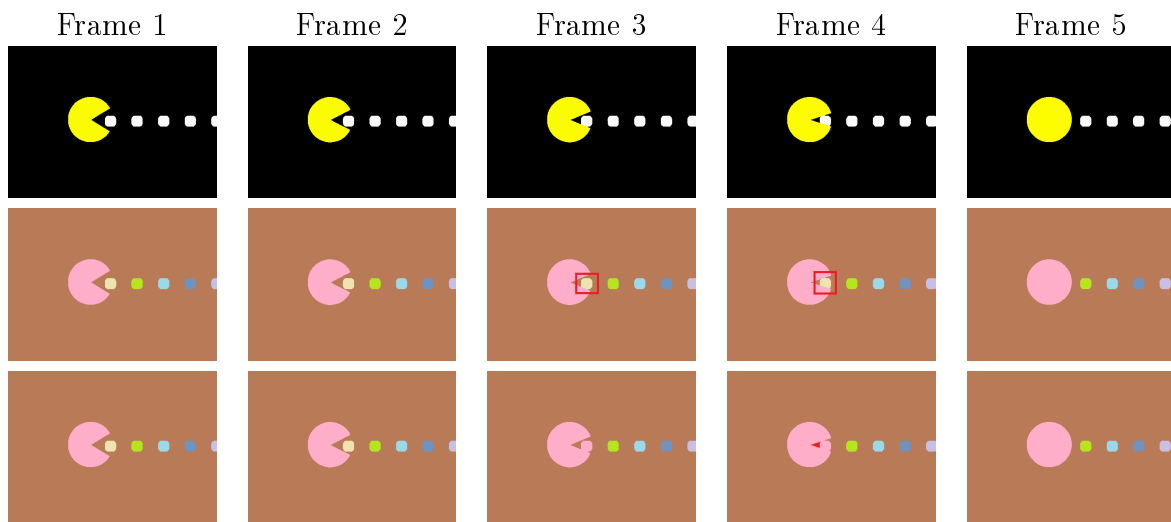


**Figure 3.5.** Outline of the StreamHOScale method: the video is transformed into a video graph (step 1); the hierarchy is computed from the video graph (step 2); the identification of video segments is made from hierarchy (step 3); the supervoxels output is calculated based on previously processed frames (step 4); and finally, the video graph is transformed in the video segmented (step 5).

The steps performed in the StreamHOScale method are very similar to the steps performed in the FVHOScale method. The difference is that the segmentation is performed in blocks, instead of being the whole video and a new step has been added to make the propagation of the spatiotemporal information of a block to another. As can be seen in Fig. 3.5, our method can be outlined in some steps: creation of a video graph for each  $k$ -sized frame block (step 1 in Fig. 3.5); computation of hierarchical scales for each video graph (step 2 in Fig. 3.5); identification of video segments for each block (step 3 in Fig. 3.5); computation of temporal coherence between video segments belonging to consecutive blocks (step 4 in Fig. 3.5); and creation of a segmented video (step 5 in Fig. 3.5). The great differential of StreamHOScale method is the step 4, which is responsible for making the spatiotemporal propagation between segmented blocks, trying to preserve, as much as possible, the accuracy of video segmentation

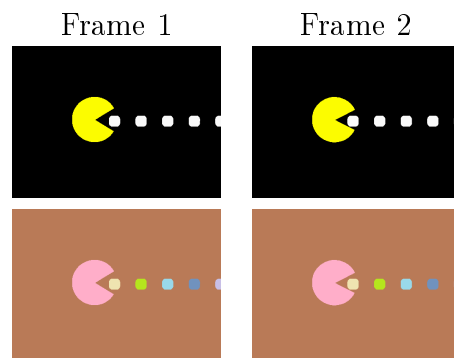
The main challenge of our StreamHOScale method is to compute video segmentation as well as the FVHOScale method preserving the spatiotemporal coherence. Thus, after computing segments for two consecutive  $k$ -sized frame blocks, a temporal coherence must be computed for producing consistent video segments. Regarding the temporal coherence, we use a new and simple strategy for merging two consecutive segmented blocks for identifying continuous supervoxels in the time. In order to do that, starting on the second block, the last frame of the previous block is incorporated at the beginning of following block. Therefore, a block consists of one “old” (processed) frame and  $k$  “new” (unprocessed), *i.e.*, two consecutive frame blocks are overlapped by one frame for providing the temporal coherence.

In Figure 3.6, we present a comparison of results obtained using FVHOScale and StreamHOSale. We can see that when we use the FVHOScale method it takes into account the global visual information of the video. However, the StreamHOSale method use the local information of each block which is preserved and propagated. We observed in the Figure 3.6, that in Frames 2 and 3, using FVHOScale there was a difference in the result (highlighted in red), where the object obtained a temporal propagation greater than that presented in the StreamHOScale. To better explain the temporal propagation used in StreamHOSale, we will illustrate, step by step, how the segmentation presented in Figure 3.6 was performed.



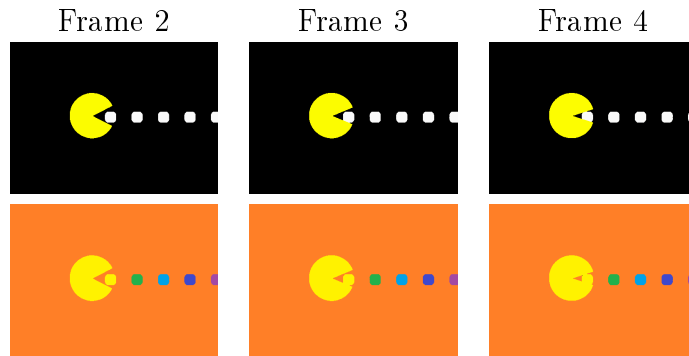
**Figure 3.6.** Full and Streaming video segmentation. The original frames are illustrated in the first row. The full video segmentation, using FVHOScale, is illustrated in second row. The streaming video segmentation, using StreamHOScale with number of 2-sized frame block, is illustrated in third row.

The segmentation shown in Figure 3.6 was performed using 2-sized frame blocks. Thus, the first processing step is to segment the Frames 1 and 2, using the FVHOScale, as shown in Figure 3.7.



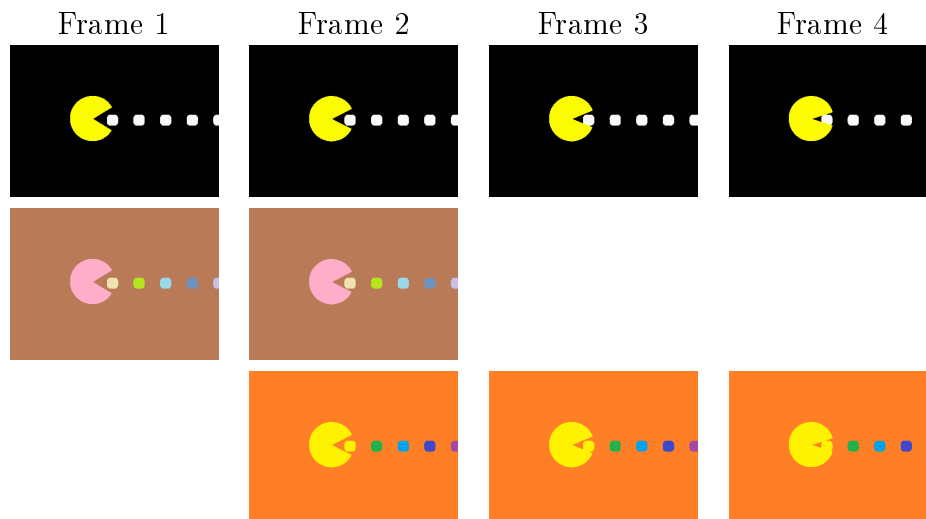
**Figure 3.7.** Streaming video segmentation. Segmentation of the first block.

After the segmentation of the first block, the segmentation process of the second block will be started, where the Frames 2, 3 and 4 are segmented. The Frame 2 is segmented again, because the last frame of the previous block is incorporated at the beginning of following block. Producing the result shown in Figure 3.8.



**Figure 3.8.** Streaming video segmentation. Segmentation of the second block.

Thus, the two consecutive blocks that have been segmented independently and have a common frame (the Frame 2), shown in Figure 3.9. Now, the method will use the common frame to propagate the temporal information from the first block to the second block.

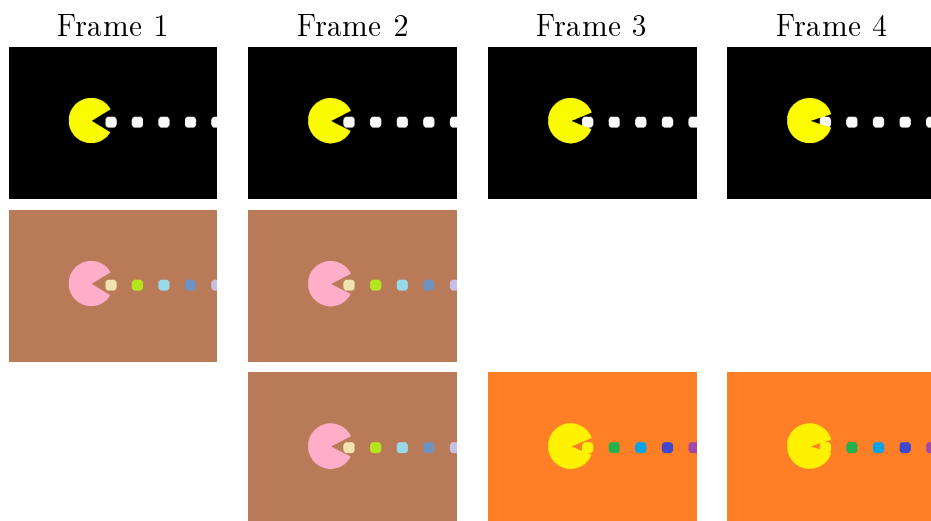


**Figure 3.9.** Streaming segmentation. Segmentation of the first and second blocks.

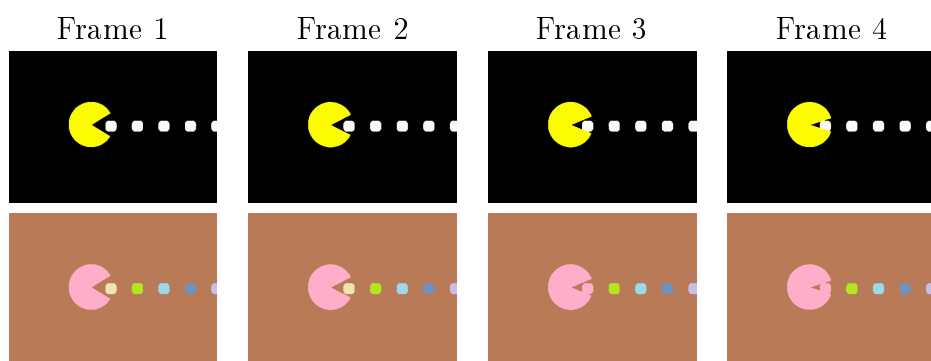
In temporal propagation, the segments of Frame 2 of the second block will be labeled according to the segments of Frame 2 of the first block. According to the following decisions:

- The segments will be labeled according to their size, from largest to smallest.

- The segment of the second block that has an intersection with one, and only one, segment of the first block, will receive the label of this segment.
- The segment of the second block that intersects with more than one segment of the first block will receive the label of the largest segment.
- Each segment can only receive a unique label, so that segments that are not labeled at the end of the process will receive a new label.

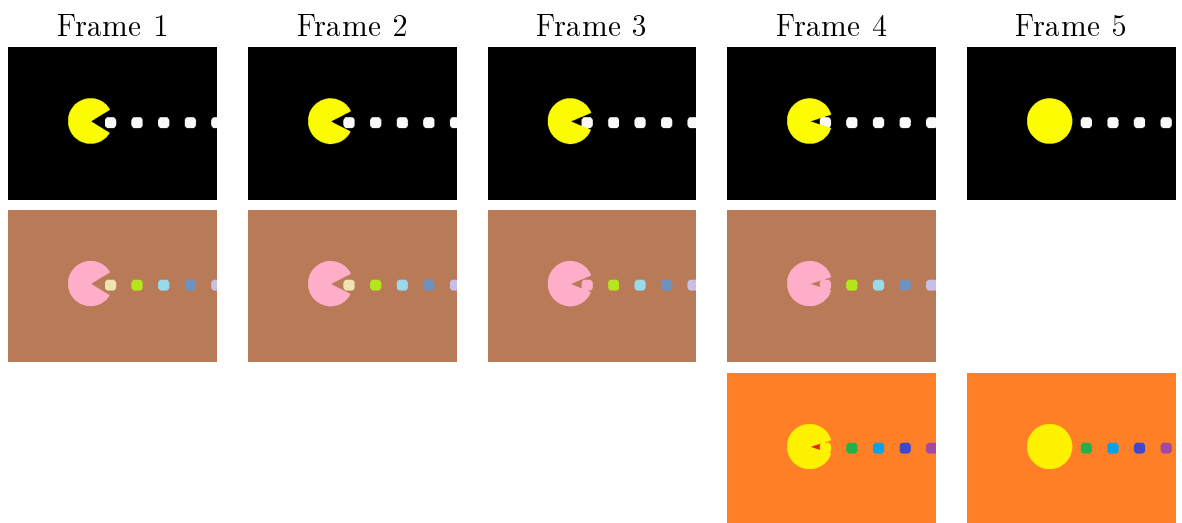


**Figure 3.10.** Streaming video segmentation. Segmentation of the first and second blocks with the temporal propagation.

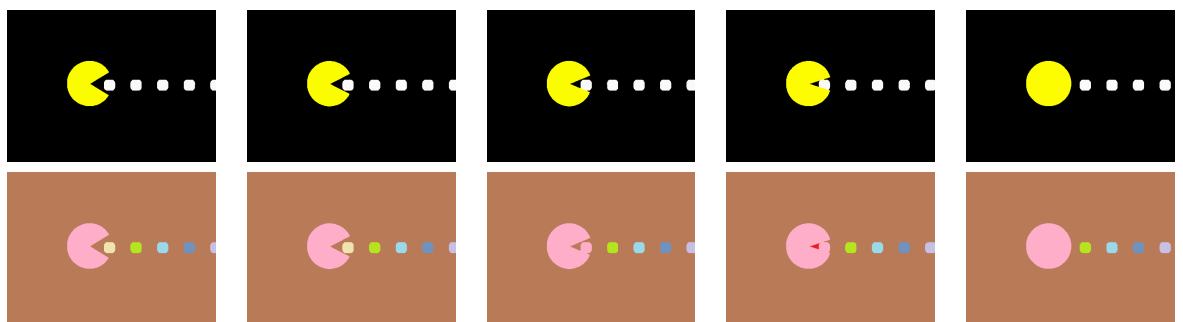


**Figure 3.11.** Streaming video segmentation. Segmentation of the first and second blocks with the temporal propagation.

Thus, the Frame 2 of the second block is labeled according to Frame 2 of the first block, shown in Figure 3.10. Once the first frame of the second block has already been labeled, simply propagate the label to the other frames, since they are strongly connected components, shown in Figure 3.11. Thereby, this approach is repeated for all other video frames blocks, as shown in the figures 3.12 and 3.13. Segments that are not labeled according to the previous block will receive a new label because it is a new segment in the scene, shown in Figure 3.13, segment in red.



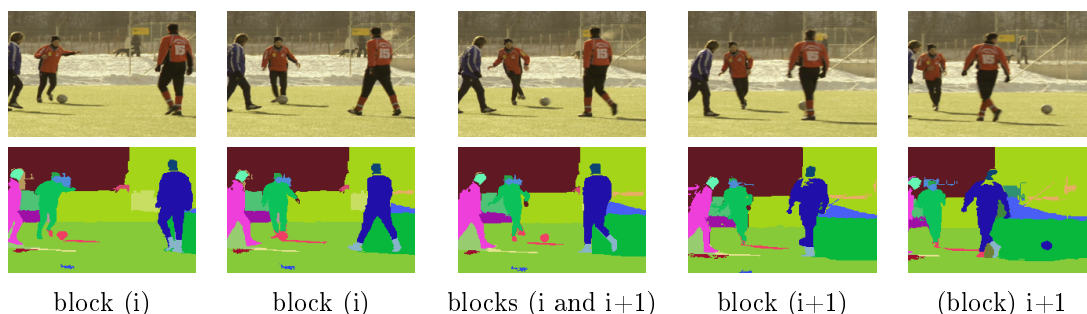
**Figure 3.12.** Streaming video segmentation. Segmentation of the third and final block.



**Figure 3.13.** Streaming video segmentation. Segmentation of the all blocks with the temporal propagation.



In Figure 3.14, it is possible to see an example of this process, where we can see that from one block to another there is a small blurring in supervoxels. Note the blue supervoxel representing the man's body, and note that in the blocks  $i$  and  $i+1$  there was a little difference in the continuation of visual information, but despite this, there was a good propagation of the temporal information between the blocks.



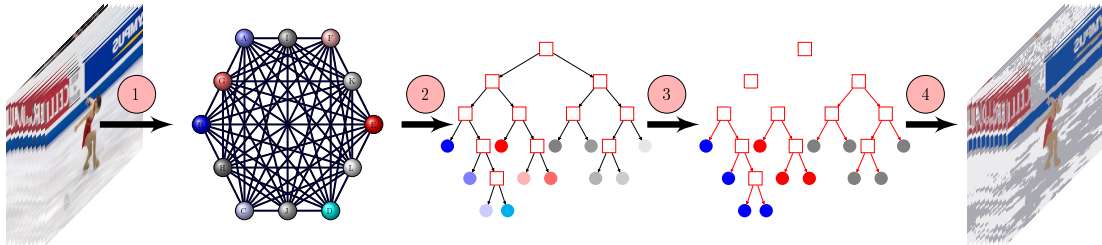
**Figure 3.14.** An example of the temporal propagation using StreamHOScale method.

The main idea for performing temporal coherence is to identify, in the overlapped frame, the supervoxels that contain voxels of both consecutive  $k$ -sized blocks. After this identification, we propagate the labels of previous blocks to new ones. Thus, we identify the supervoxels which are continuous in the time. The great advantage of StreamHOScale method is the ability to run a video stream without the need of having all the video in memory, achieving to the segment of consecutive frame blocks considering the temporal information present throughout the video.

### 3.3 Video Segmentation with Oversegmentation

Oversegmentation is the process by which the objects being segmented are themselves segmented or fractured into subcomponents. In this work, we propose to use the oversegmentation technique as a step of pre-processing of our segmentation method, in order to simplify and pre-group the pixels to facilitate and accelerate the segmentation process. Our approach to oversegmentation can be outlined in

four steps, as illustrated in Figure 3.15, where the goal is to perform a segmentation in the color space.



**Figure 3.15.** Outline of oversegmentation method: the video is transformed into a color graph (step 1); the hierarchy is computed from the color graph (step 2); the identification of colors segments is made from hierarchy (step 3); and finally, (step 4) the graph is transformed in the video, using the average of the colors of each connected component.

When performing video segmentation using FVHOScale or StreamHOScale method we observe that the first step, the conversion of pixels to a video graph, is a very expensive task and has an huge amount volume of information. Therefore our goal to perform an oversegmentation is to perform a pre-grouping of pixels in order to facilitate the segmentation process.

To perform oversegmentation, we use the same method of observation scale used in the segmentation process, i.e., the HOScale approach. The goal is to generate small supervoxels to be used in the segmentation process, instead of pixels. In the oversegmentation, instead of using the video pixels we propose to use the distinct colors present in video. Thus, we performed a simplification of colors before running the video segmentation, where the goal is to join the colors that have greater similarity in an attempt to increase the discriminative power of these colors in the images in the segmentation process. After running the oversegmentation, each supervoxel produced is represented by the average of the colors of the pixels. Thus, we can use the supervoxels generated at this stage as the components to be used in the segmentation process, in both methods FVHOScale

and StreamHoscale.

The oversegmentation does not define the amount of supervoxels to be generated, thus, the amount of supervoxel generated will be defined by the value of the threshold applied to the minimum size of the supervoxels. The higher the threshold is lower the amount of generated supervoxels in oversegmentation. In Figure 3.16, we illustrate some results using the over segmentation with different threshold of the minimum size of the supervoxels, where you can see an improvement in the quality of segmentation. Moreover, we can observe that as we increase the threshold we decrease the amount of color to be used in targeting, so the segmentation process is faster, as shown in Table 3.1. When we are performing segmentation without over segmentation we use 3,240,000 pixels with 24,666 distinct colors, but when we use the oversegmentation using threshold 10, will be used only 864 distinct colors in the segmentation process. Thus, the advantage of using oversegmentation is that as the segmentation process will use supervoxels instead of pixels, it implies in less amount of data, so the segmentation process is much faster, as will be present in Chapter 4.

**Table 3.1.** Variation of the number of colors according to the threshold.

<b>threshold</b>	<b><i>min_size</i></b>	<b>number of colors</b>
-		24666
2		6164
10		864
25		290
50		142

## 3.4 Final Remarks

In this chapter, we presented our approach to hierarchical video segmentation. Our proposal is to extend the hierarchical image segmentation method proposed by Guimarães et al. [2012] to hierarchical video segmentation considering the spatiotemporal information using an observation scale. The hierarchical video segmentation is transformed into a graph partitioning problem in which each part corresponds to one supervoxel of the video. Thus, a new approach to hierarchical



**Figure 3.16.** Video segmentation with oversegmentation. The original frames are illustrated in the first row. The results obtained using FVHOScale without oversegmentation are in the second row. The following rows, from top to bottom, illustrate the results obtained using oversegmentation with threshold of the minimum size of the supervoxels, 2, 10, 25 e 50, respectively.

video segmentation is proposed, which computes a hierarchy of partitions by a reweighting of the original graph in which a segmentation can be easily inferred, and the temporal coherence is related to the graph transformation used.

We presented two different methods of transforming a video into a graph and thus performing the segmentation: FVHOScale (which segments the whole video at once) and StreamHOScale (which performs the streaming video segmentation). Furthermore, we proposed the use of oversegmentation before the segmentation process in order to reduce the processing time. In Chapter 4, we present the results of our experimental methods parameters making an analysis together with comparisons of other methods proposed in the literature.

# Chapter 4

## Experimental results

In the following sections, we present experimental results of our hierarchical video segmentation methods. We detail our experimental setup along Section 4.1. A study of the parameters of our methods is then given in Section 4.2. Section 4.3 compares our results with to the state of the art. Finally, Section 4.4 presents the final remarks of our experiments.

### 4.1 Experimental setup

In order to provide a comparative analysis, we used the benchmark and library LIBSVX proposed in [Xu and Corso, 2012], since the implemented methods are the state of the art for hierarchical video segmentation. The benchmark is composed, among others, by: three datasets, implementations of the methods for hierarchical video segmentation and a suite of metrics to implement 3D quantitative evaluation criteria for good supervoxels.

#### 4.1.1 Datasets

The LIBSVX proposes the use of three datasets to evaluate supervoxel and video segmentation:

- **SegTrack**<sup>1</sup> [Tsai et al., 2010]: this dataset provides a set of human-labeled single-foreground objects with the videos stratified according to difficulty on color, motion and shape. SegTrack has six videos, an average of 41 frames-per-video (fpv), a minimum of 21 fpv and a maximum of 71 fpv, leading to a total of 244 annotated frames. As can be seen in Figure 4.1, the annotated frames (groundtruth) correspond to only one specific object of the video, thereby the evaluations using these annotated frames do not evaluate all the objects in the video, but only the labeled object.



**Figure 4.1.** Example images from SegTrack dataset. The original frames are illustrated in the first row and groundtruth frames in the second row.

- **Chen**<sup>2</sup> [Chen and Corso, 2010]: This dataset is a subset of the well-known xiph.org videos that have been supplemented with a 24-class semantic pixel labeling set (the same classes from the MSRC object-segmentation dataset [Shotton et al., 2009]). The eight videos in this set are densely labeled with semantic pixels and have an average 85 fpv, minimum 69 fpv and maximum 86 fpv, leading to a total of 639 annotated frames. This dataset allows us to evaluate the supervoxel methods against human perception. As can be seen in Figure 4.2, the annotated frames correspond to semantic pixels, thus objects spatially disconnected have the same label, thereby the evaluations using these annotated frames do not offer an accurate label placement of supervoxels in the videos.

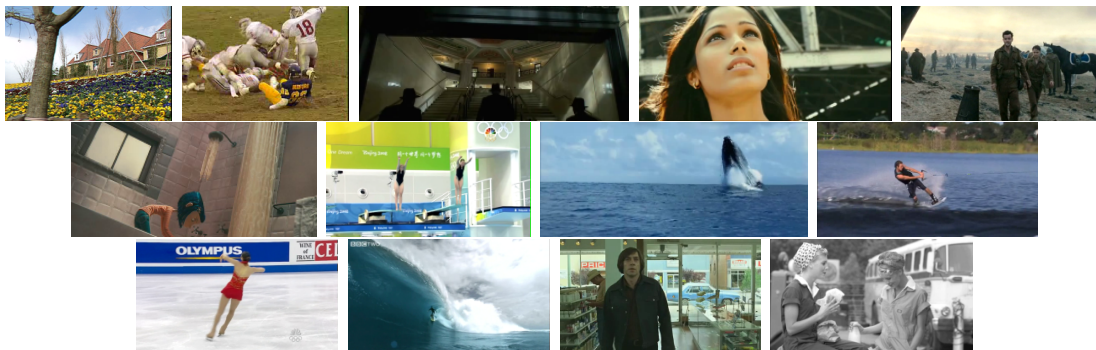
<sup>1</sup><http://cpl.cc.gatech.edu/projects/SegTrack/>

<sup>2</sup><http://www.cse.buffalo.edu/~jcorso/r/labelprop.html>



**Figure 4.2.** Example images from Chen dataset. The original frames are illustrated in the first row and groundtruth frames in the second row.

- **GaTech**<sup>3</sup> [Grundmann et al., 2010a]: It comprises 15 videos of varying characteristics, but predominantly with a small number of actors in the shot. In order to run all the methods included in the benchmark library, we restrict the videos to a maximum of 100 frames (they have an average of 86 fpv and a minimum of 31 fpv, leading to a total of 1,293 frames). Unlike the other two previous datasets, this dataset does not have a groundtruth segmentation, see examples of this dataset in Figure 4.3.



**Figure 4.3.** Example images from GaTech dataset.

In order to run all the experiments such as in [Xu and Corso, 2012], the input video resolution was scaled to 240x160 pixels.

<sup>3</sup><http://www.cc.gatech.edu/cpl/projects/videosegmentation>

### 4.1.2 Methods

The LIBSVX includes five methods for video segmentation (four offline and one streaming):

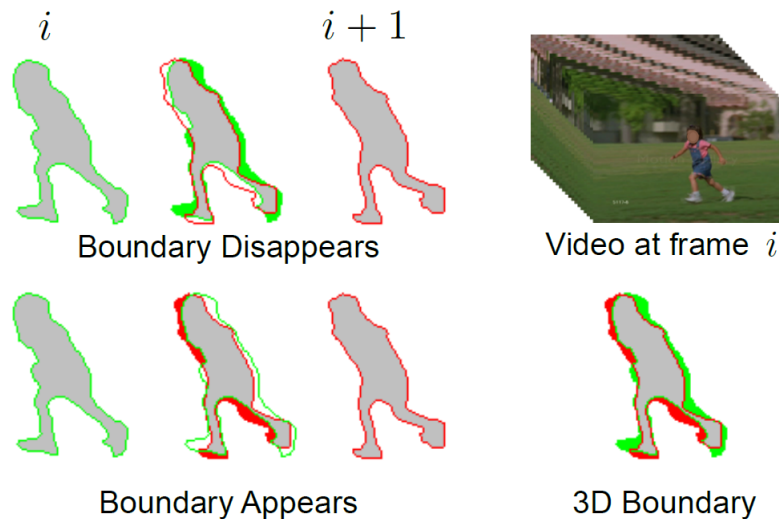
- **Graph-Based (GB)**: Implements the Felzenszwalb and Huttenlocher [2004] directly on the 3D video voxel graph;
- **Graph-Based Hierarchical (GBH)**: Implements the Grundmann et al. [2010b] that performs the graph-based method iteratively in a hierarchy;
- **Nyström Normalized Cuts (Nyström)**: Implements the Fowlkes et al. [2001] approach of using the Nyström approximation to solve the normalized cuts. Their implementation is in Matlab;
- **Segmentation by Weighted Aggregation (SWA)**: Implements the Sharon et al. [2000] hierarchical algebraic multigrid solution to the normalized cut criterion in a 3D manner as was done in [Corso et al., 2008];
- **Graph-based Streaming Hierarchical Video Segmentation (StreamGBH)**: Implements the graph-based hierarchical segmentation method [Xu et al., 2012].

### 4.1.3 Metrics

The LIBSVX provides a suite of the volumetric video-based 3D metrics. Xu and Corso [2012] extended to 3D space-time metrics some quantitative superpixel evaluation 2D metrics that were proposed by [Moore et al., 2008; Levinshtein et al., 2009; Veksler et al., 2010; Liu et al., 2011; Zeng et al., 2011].

- **3D Boundary Recall (3D BR)**: this metric measures the spatiotemporal boundary detection, see in Figure 4.4 an example of the metric calculation;
- **Explained Variation (EV)**: proposed in [Moore et al., 2008] as human-independent metric, thus, does not use the groundtruth;





**Figure 4.4.** A visual explanation of the distinct nature of 3D boundaries in video (please view in color). We overlap each frame to compose a volumetric video, the green colored area which is a part of girl in frame  $i$  should not be counted as a part of girl in frame  $i + 1$ , similarly the red area which is a part of girl in frame  $i + 1$  should not be counted as a part of girl in frame  $i$ . The lower right graph shows the 3D boundary along the time axis (imagine you are looking through the paper). From [Xu and Corso, 2012].

- **Mean Duration (MD):** proposed in [Xu et al., 2012] it measures the average temporal extent of all supervoxels in a video. According to Xu et al. [2012], the mean duration of segments is a more important metric, as it measures the temporal coherence of a segmentation method more directly.

#### 4.1.4 Implementation issues

To efficiently implement our methods, we used some data structures similar to the ones proposed in Guimarães et al. [2012]; in particular, the management of the collection of partitions is made using Tarjan’s union-find. Furthermore, we made some algorithmic optimizations to speed up the computations of the hierarchical observation scales. So, in order to create the video graphs, and when it is necessary, we employed a  $KD$ -tree for identifying the  $K$ -nearest neighbors, we used the ANN library [Arya et al., 1998] to build a  $KD$ -tree. Our method is implemented in C++.

We ran all experiments in a Intel(R) Xeon(R) CPU E5-2650 2.00GHz 64 GB RAM with Ubuntu 14.04.2 LTS. Moreover, we have computed the segmentation results for each video but with a distribution of supervoxel numbers varying from less than 200 to more than 500. To facilitate comparison of the methods for each dataset, we used linear interpolation to estimate each methods' metric outputs densely like Xu and Corso [2012].

## 4.2 Parameter Learning

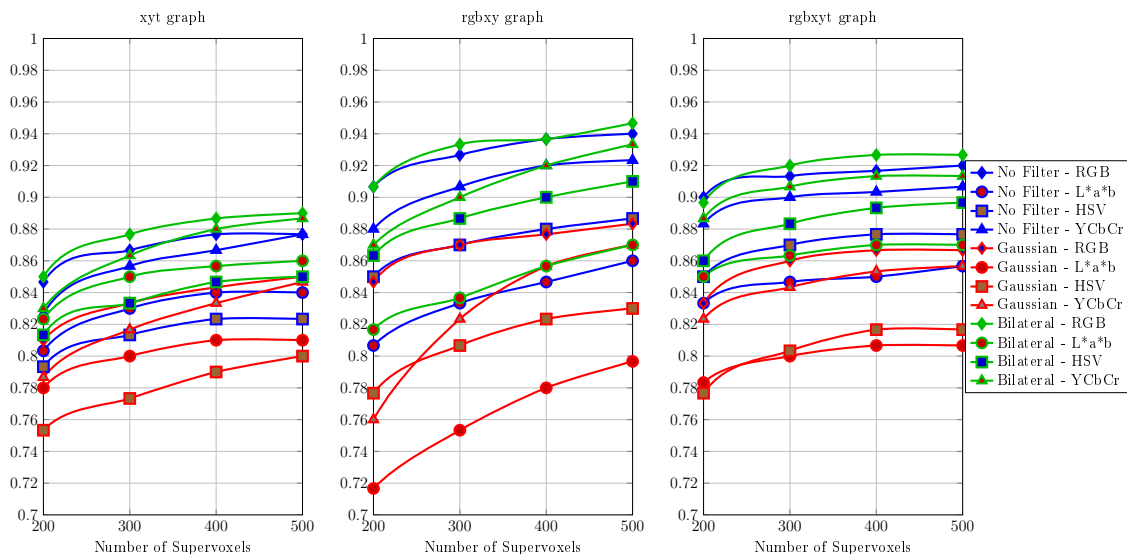
In order to determine an appropriate set of parameters for our hierarchical video segmentation methods introduced in Chapter 3, we optimized parametric settings on the training data of three datasets proposed in LIBSVX [Xu and Corso, 2012]: SegTrack, Chen and GaTech. Since not all datasets have groundtruth, we used only the Explained Variation and Mean Duration metrics to find the best parameters.

### 4.2.1 Parameters for video graph creation

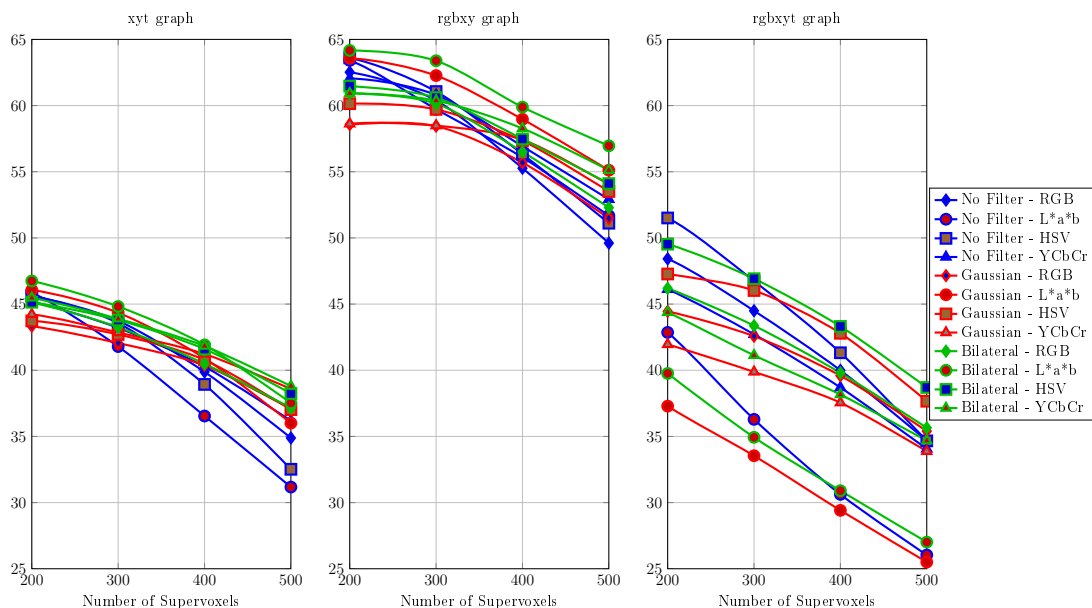
The first step of our HOScale approach is the creation of video graph. In this step, we evaluated the following parameters with their values:

- **Image Filtering:** No filter, Gaussian filter and Bilateral filter [Tomasi and Manduchi, 1998].
- **Color Space:** RGB, Lab, HSV and YCbCr.
- **Type of Graph:** xyt, rgbxy and rgbxyt.

We performed experiments varying all parameters with all possibilities using the FVHOScale method (the threshold for minimum size supervoxel was set at 0.5% related to percentage of video pixels). In Figures 4.5 and 4.6, we present the results of this experiment with the average values of three datasets, varying the Image Filtering, Color Space and Type of Graph. The individual results, of each dataset, are available in Appendix A. In Figure 4.5, we present the average results of Variation Explained metric and, in Figure 4.6 we present the average results of Mean Duration metric.



**Figure 4.5.** A comparison with the parameters used in the creation of the video graph, separated by type of graph. The comparison is based on the Explained Variation metric. It shows the average value for the three datasets (SegTrack, Chen and GaTech).



**Figure 4.6.** A comparison with the parameters used in the creation of the video graph, separated by type of graph. The comparison is based on the Mean Duration metric. It shows the average value for the three datasets (SegTrack, Chen and GaTech).

In Figure 4.5, we can observe the following features in the results:

- In general, the use of Gaussian filter has the poorest results.
- The Bilateral filter showed slightly better results than not using filters.
- The RGB color space obtained the best results, followed by YCbCr, HSV and Lab.
- The best result was obtained using the rgbxy graph.

In Figure 4.6, we can observe the following features in the results:

- The results were very different, depending on the type of graph used.
- The graph rgbxy showed the best results compared to other graphs.
- Using the xyt and rgbxy graphs, we observe that there are small differences in the results and that the Bilateral filter and the Lab space color obtained the best results. Moreover, the rgbxyt graph obtained results well differentiated.

According to the experimental results shown in Figures 4.5 and 4.6, we decided to set for the next experiments the following parameters:

- **Image Filtering:** No filter.
- **Color Space:** RGB.
- **Type of Graph:** rgbxy.

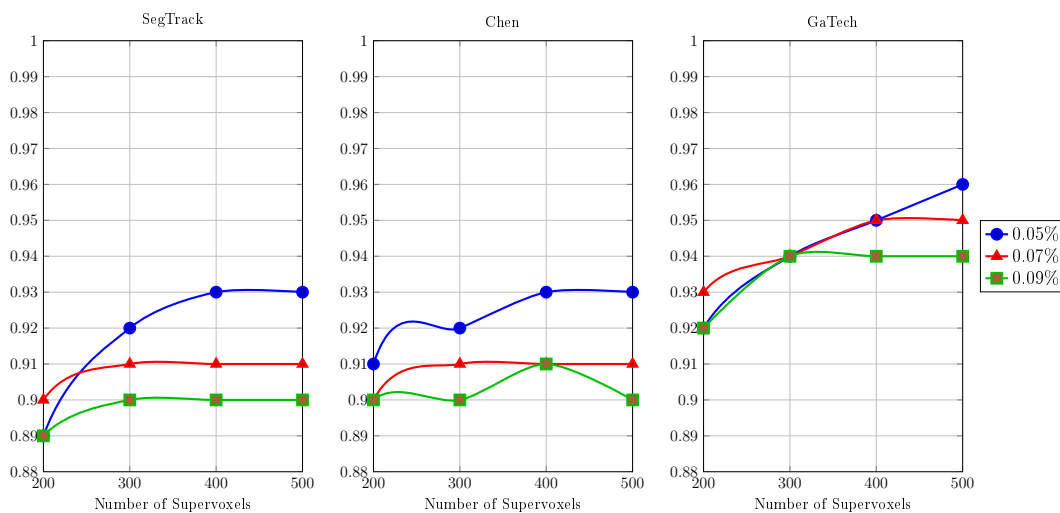
Although the Bilateral filter presented slightly better results than without the filter, the implementation of the Bilateral filter means one more step processing, thus increasing the processing time. So, we preferred to continue the experiments without using image filtering, because this approach also obtain good result. In addition, we set the RGB color space, because it obtained the best results in Explained Variation metric and rgbxy graph because it obtained the best results in the Mean Duration metric.

### 4.2.2 Parameter for minimum size supervoxel

We performed experiments to investigate the behavior of the results when we changed the threshold for minimum size supervoxels used in step of Observation Scale computation of our HOScale approach. For this, we changed the threshold parameter with the following values: 0.05%, 0.06%, 0.07%, 0.08%, 0.09% and 0.10% (related to the percentage of pixels present in the video). We used the FVHOScale method, without image filter, RGB Space color and the rgbxy graph. In Figures 4.7 and 4.8, we present the experimental results separated per dataset. In Figure 4.7 we present the results of Explained Variation metric and, in Figure 4.8, we present the results of Mean Duration metric.

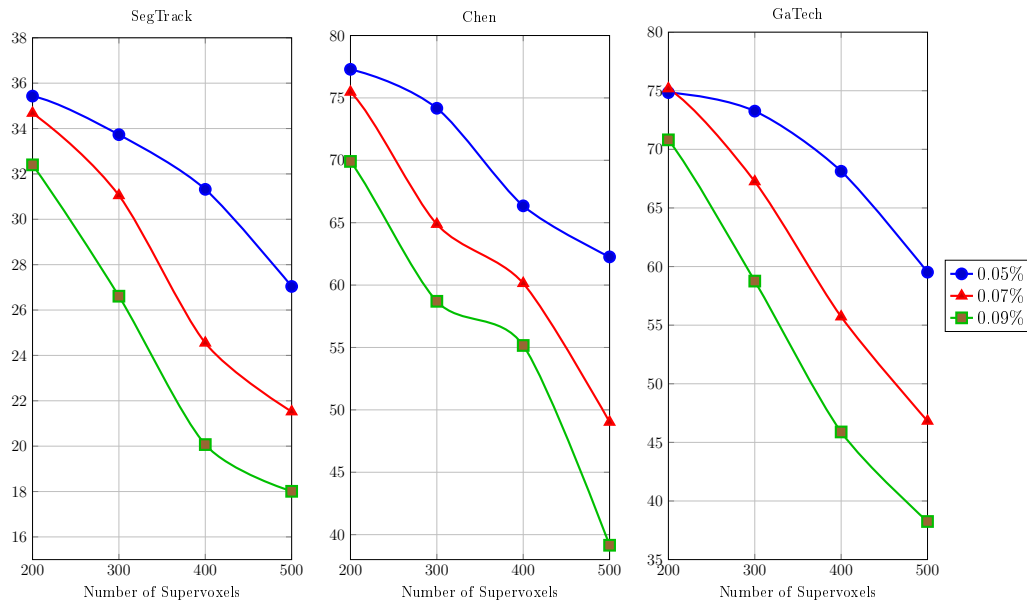
In Figure 4.7, we can observe the following features in the results:

- The results of SegTrack dataset show that as we increase the number of supervoxels, we get the best results with the lowest threshold for minimum size supervoxel. However, this ratio is lower in the other datasets.
- The threshold for minimum size supervoxel is related to the amount of information being processed. Note that the SegTrack dataset is the smallest and simplest dataset and GaTech is the largest and most complex dataset.



**Figure 4.7.** A comparison with the variation of the threshold for minimum size supervoxel parameter, separated per dataset. The comparison is based on the Explained Variation metric.

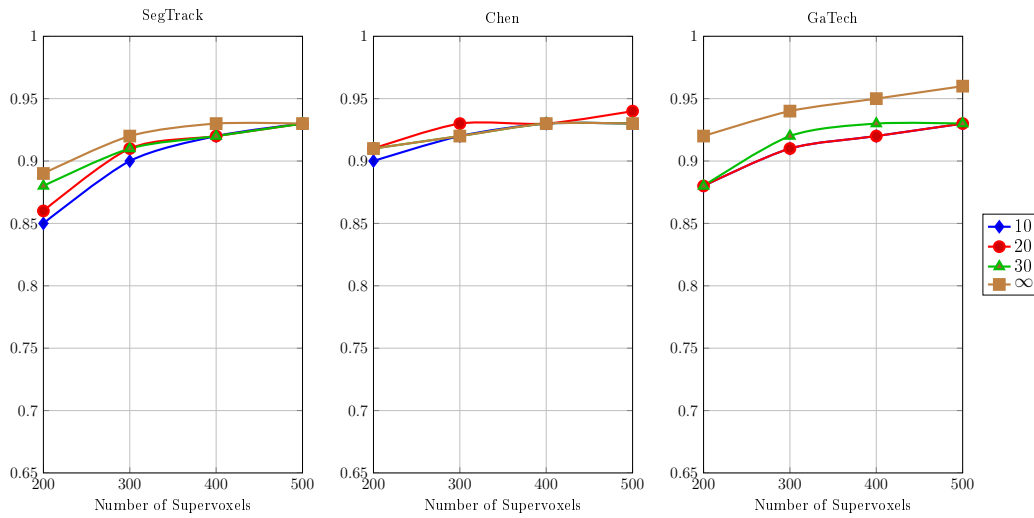
In Figure 4.8, we present the results of Mean Duration metric in the three databases, varying the threshold for minimum size supervoxel. We can observe that the smaller the value of threshold for minimum size supervoxel, the greater the value of Mean Duration, and thus the greater the temporal coherence. Therefore, we set the threshold in the next experiments to the value 0.05%, because it showed the best results in both Explained Variation and Mean Duration metrics.



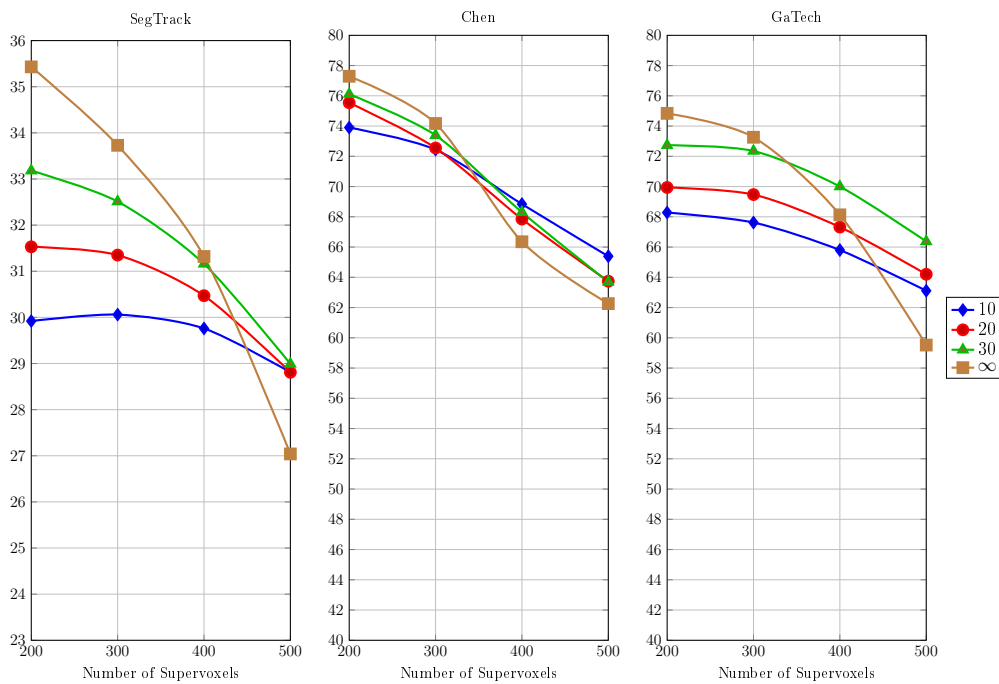
**Figure 4.8.** A comparison with the variation of the threshold for minimum size supervoxel parameter, separated by dataset. The comparison is based on the Mean Duration metric.

### 4.2.3 Parameter for StreamHOScale

Besides the parameters already studied, StreamHOScale method has the block size parameter to be segmented. Thereby, we performed experiments to investigate the variation of the size of frame blocks in StreamHOScale method. In Figures 4.9 and 4.10, we present the results of varying the block size with the values 10, 20 and 30 frames. We present the experimental results separated per dataset. In Figures 4.9 and 4.10, we present the experimental results separated per dataset. In Figure 4.9, we present the results of Explained Variation metric and, in Figure 4.10, we present the results of Mean Duration metric.



**Figure 4.9.** A comparison with the variation of the size of frame block parameter, separated per dataset. The comparison is based on the Explained Variation metric.



**Figure 4.10.** A comparison with the variation of the size of frame block parameter, separated per dataset. The comparison is based on the Mean Duration metric.

In the results shown in Figure 4.9, we can see that the values for the Explained Variation metric does not have a big difference to the variation of block size, but in the result shown in Figure 4.10, we can see that the higher the size block the greater will be the Mean Duration metric. Note that, the larger the block size, the higher the cost of time and space. For the next experiments, we fixed the block size with value 10, because it was the value used for comparison with the state of the art in the streaming method.

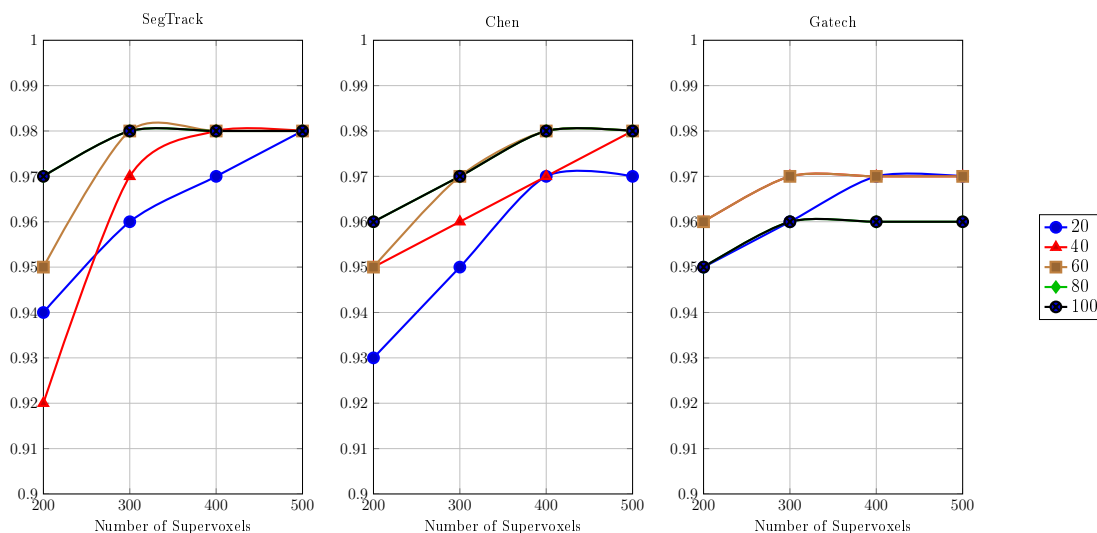
#### 4.2.4 Parameters for oversegmentation

Besides the FVHOScale and StreamHOScale methods, we also proposed to use the HOScale approach to perform an oversegmentation before performing the segmentation process. The parameters of oversegmentation are similar to HOScale approach. Therefore we used the same parameters found so far to run the oversegmentation. Furthermore, the use of a new type of graph was proposed in oversegmentation, the RGB graph which performs an oversegmentation on the distinct colors present in the video.

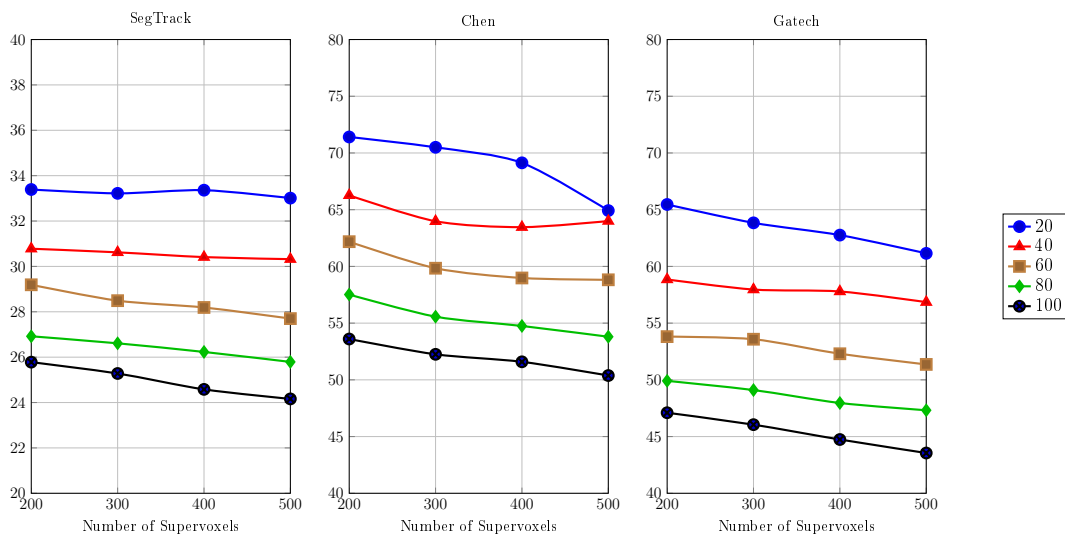
In Figures 4.11 and 4.12, we present the results of varying the threshold for minimum size supervoxel in oversegmentation. We present the experimental results separated per dataset, using the FVHOscale method with RGB graph whitout image filtering, RGB space color and size of frame blocks equal to 10. As it is a process of oversegmentation the threshold is defined in number of pixels, and should be used a small value, since the objective is not to segmentation, but eliminate noise joining lower-level information, i.e. the pixel colors in information more descriptive, such as in small supervoxels.

In Figure 4.12, we can observe that the greater the value of threshold for minimum size supervoxel in oversegmentation the greater the value of Mean Duration metric. However, the higher the threshold size, the lower is the consumption of time and space at the time of segmentation. Therefore the larger the threshold size, the lower the amount of generated supervoxels in oversegmentation and thus the lower the quantity of information to be segmented. However, from Figures 4.12, we can observe that the smaller the value of threshold for minimum size supervoxel in oversegmentation the greater the value of Mean Duration metric.





**Figure 4.11.** A comparison with the variation of the threshold for minimum size supervoxel in oversegmentation, separated per dataset. The comparison is based on the Explained Variation metric.



**Figure 4.12.** A comparison with the variation of the threshold for minimum size supervoxel in oversegmentation, separated per dataset. The comparison is based on the Mean Duration metric.

## 4.3 Comparison to state of the art

Using the library LIBSVX, we performed a comparative analysis of the results obtained by our methods using the HOscale approach to the methods of the state of art. We performed a comparative analysis in three aspects: quantitative analysis, qualitative analysis and computational cost.

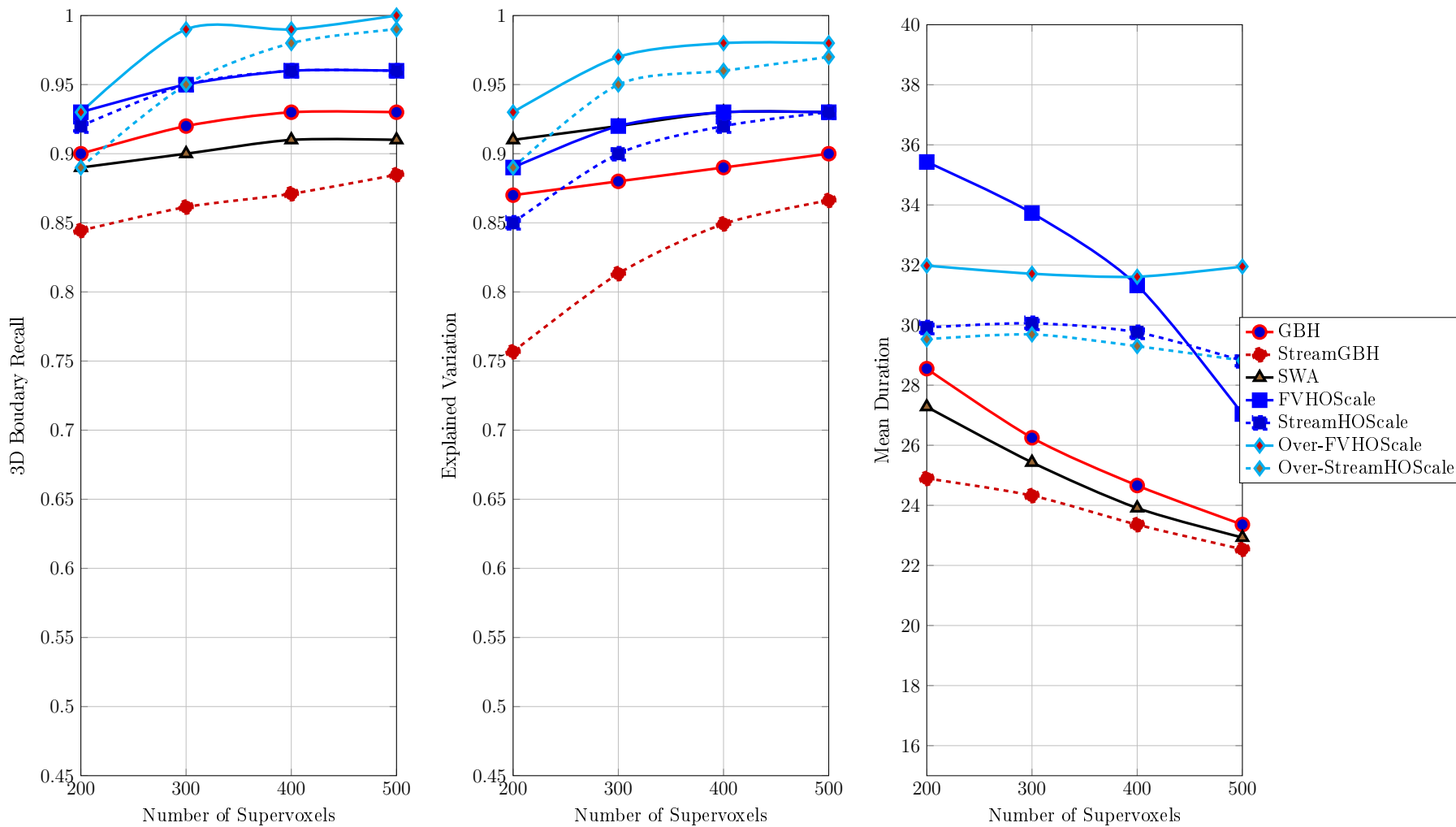
### 4.3.1 Quantitative analysis

In the quantitative analysis, we compared the experimental results of our methods with the methods available in LIBSVX using the following metrics: 3D Boundary Recall (groundtruth is needed); Explained Variation; and Mean Duration. Unlike other methods, we used the same parameters for all datasets, as shown in Section 4.2. For GBH, StreamGBH, SWA, Meanshift and Nyström methods, the parameters were tuned per dataset. For GB method, the parameters were tuned per video. In Figures 4.13, 4.14 and 4.15, we present the individual results per dataset. In Figure 4.16, we present the average results of all datasets.

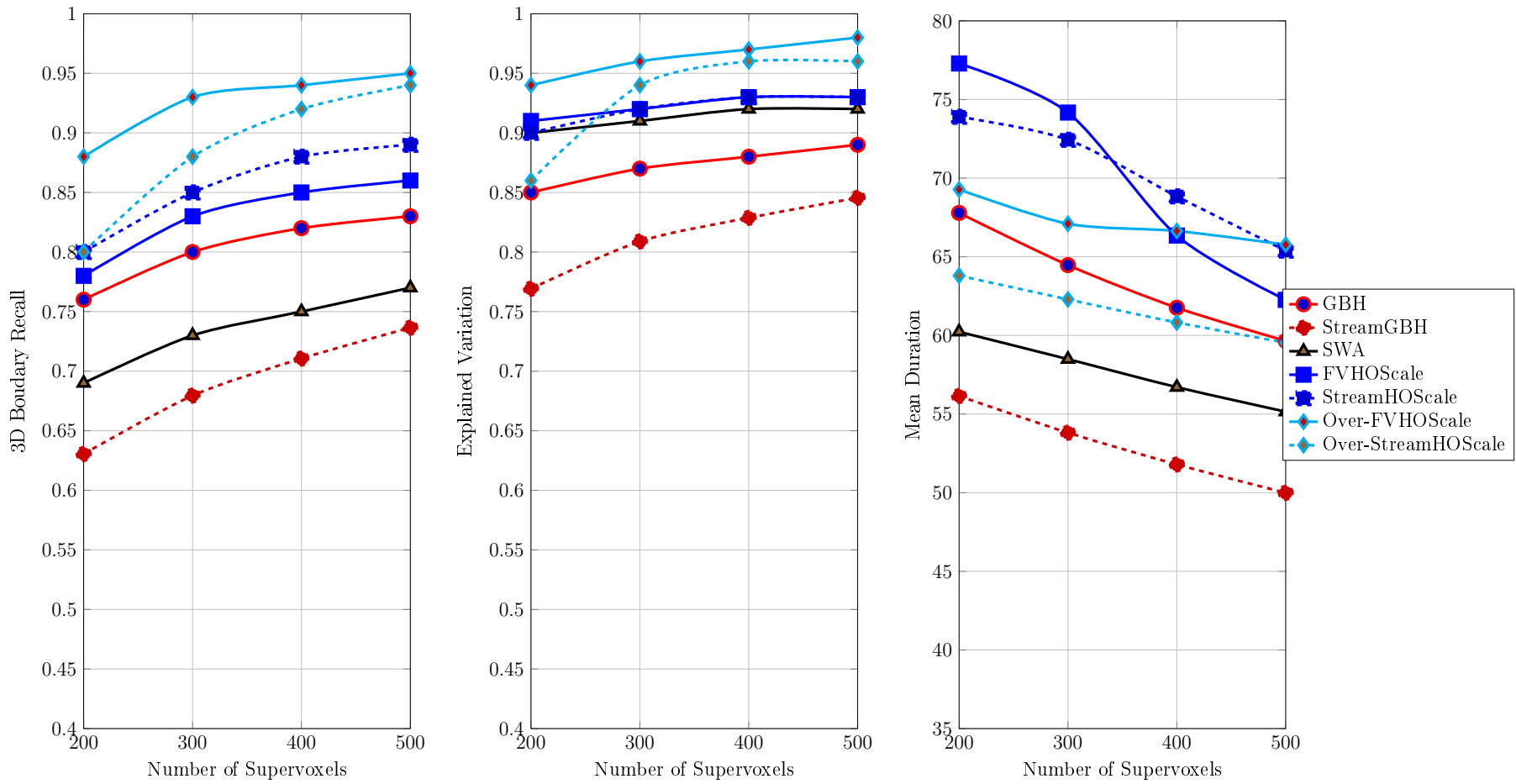
In Figure 4.16, we can observe the following features in the results:

- The results are similar to the 3D Boundary Recall and Explained Variation metric, with minor differences.
- The proposed approaches showed the best results.
- Our methods obtained better results when using the oversegmentation for the 3D Boundary Recall and Explained Variation metric. However, using the Mean Duration metric the best results were obtained when not using the oversegmentation.
- Even our stream approach showed better results when compared to state of the art.

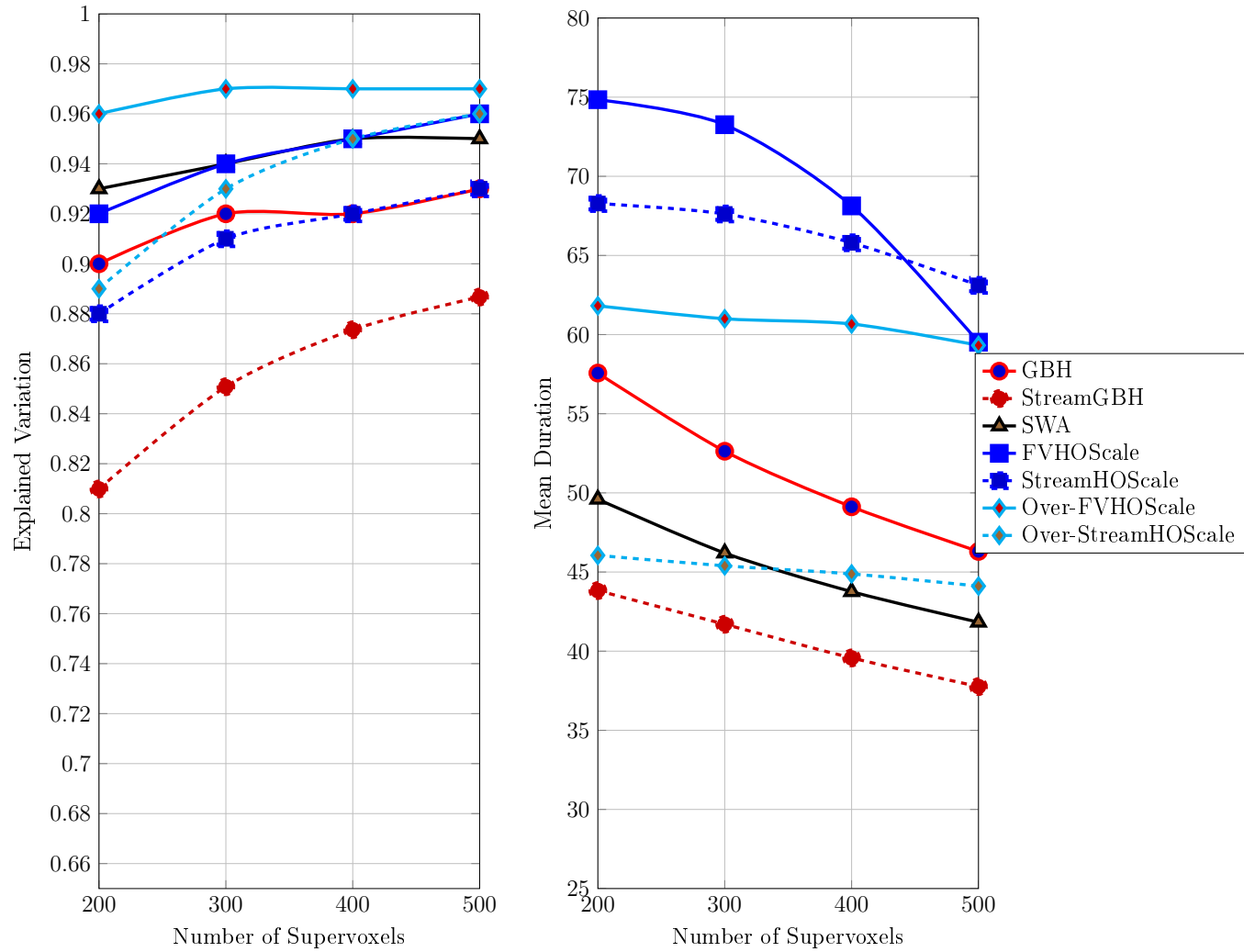
Regarding only mean duration, we can see that our methods showed far superior results to the state of the art. According to Xu et al. [2012], the mean duration of video regions is the most important metric, as it measures the temporal coherence of a segmentation method more directly.



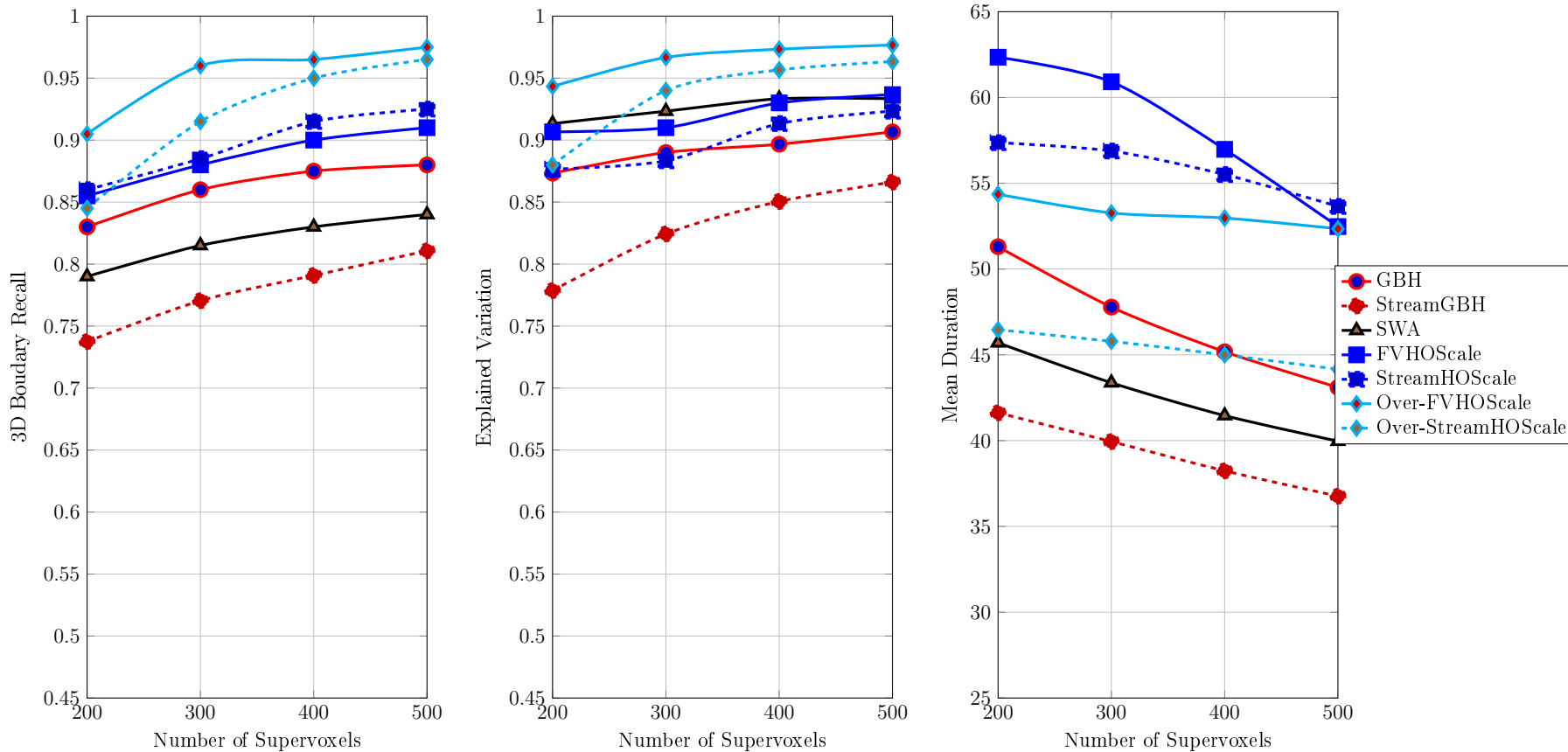
**Figure 4.13.** A comparison among our methods, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale, and the methods GB, GBH, StreamGBH, SWA, Meanshift and Nyström when applied to SegTrack dataset.



**Figure 4.14.** A comparison among our methods, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale, and the methods GB, GBH, StreamGBH, SWA, Meanshift and Nyström when applied to Chen dataset.



**Figure 4.15.** A comparison among our methods, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale, and the methods GB, GBH, StreamGBH, SWA, Meanshift and Nyström when applied to GaTech dataset.



**Figure 4.16.** A comparison among our methods, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale, and the methods GB, GBH, StreamGBH, SWA, Meanshift and Nyström. It shows the average value for the three datasets (SegTrack, Chen and GaTech).

### 4.3.2 Qualitative analysis

Despite the quantitative results, we also compared visually some of the tested methods in order to illustrate the behavior when we varied the number of video segments (50 and 100) to be computed.

In Figures 4.17, 4.18, 4.19 and 4.20, we illustrate results obtained when we applied the methods GB, GBH, StreamGBH, SWA, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale to the videos extracted from Chen and GaTech datasets. Note that, unlike of our methods, there is no guarantee that the four other methods will obtain the specified number of video segments, thus we compute a segmentation containing a number of regions, as close as possible, for the specified threshold.

Regarding the segmentation for each frame, we may observe that our methods presented good results when visually compared to the other methods. Moreover, we can observe that the use of oversegmentation improved the results of our method, especially in our streaming method. The same behavior may be observed for temporal coherence, where our methods also showed the best results.

### 4.3.3 Computational cost

Regarding computational cost, we compared our methods with the methods that presented the best quantitative results: SWA and GBH. Also with the streaming method: StreamGBH.

In Figure 4.21, we presented a comparison among our methods, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale, and the methods GBH, StreamGBH and SWA when applied to SegTrack, Chen and GaTech datasets. In Figure 4.21, we show the comparison of the processing time.

In Figure 4.21(a), we can observe the following features in the results, related to time cost comparison:

- In the SegTrack dataset, the results were very similar, except for our StreamHoscale method and using oversegmentation which showed much lower processing time than the others.

- In the Chen and GaTech datasets, our FVHOScale method had higher the processing time than other literature methods, but when using oversegmentation processing time was much lower.
- The StreamHOScale method got the lowest processing time. By using oversegmentation the processing time increased a little.

In Figure 4.21(a), we can observe the following features in the results, related to space cost comparison:

- The SWA method was the one that had the highest consumption of memory, followed by GBH method.
- Our methods showed low consumption of memory.
- The streaming methods showed to have a low consumption of memory.

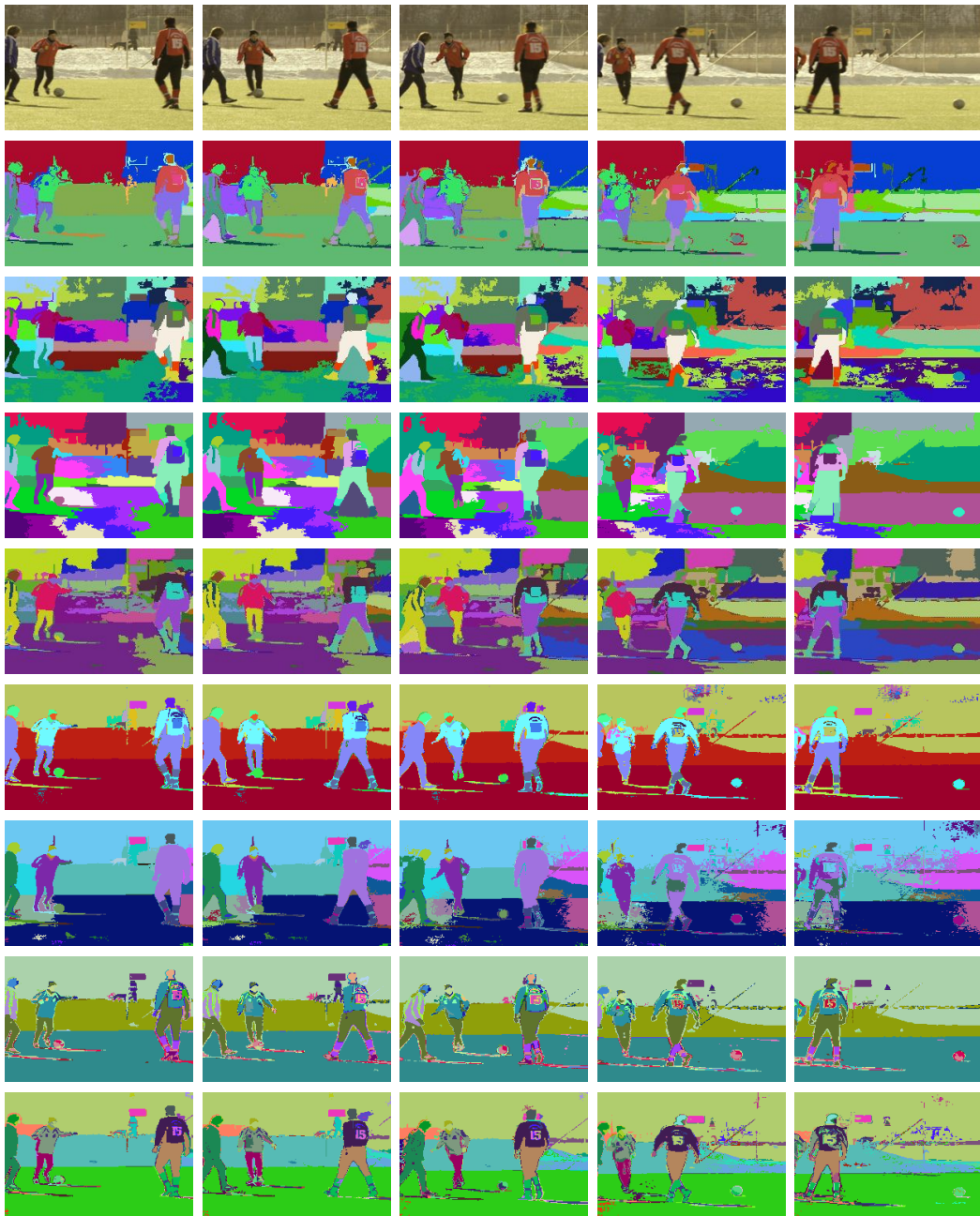
## 4.4 Final Remarks

In this chapter, we presented the experimental results of our hierarchical video segmentation approach using an observation scale. In order to provide a comparative analysis, we took into account the benchmark and library LIBSVX proposed in [Xu and Corso, 2012], since the implemented methods are the state of the art for video segmentation, including streaming video segmentation. The benchmark is composed, among others, by: (i) two datasets with groundtruth - Chen Xiph.org [Chen and Corso, 2010], SegTrack [Tsai et al., 2010]; (ii) one dataset without groundtruth - GaTech dataset [Grundmann et al., 2010a]; and (iii) implementations of the methods GB [Felzenszwalb and Huttenlocher, 2004], GBH [Grundmann et al., 2010a], StreamGBH [Xu et al., 2012], Nyström [Fowlkes et al., 2001] and SWA [Corso et al., 2008] applied to video segmentation. Using the library LIBSVX, developed by Xu and Corso [2012], it is possible to compute, among others, the following metrics: (i) 3D boundary recall; (ii) explained variation; and (iii) mean duration. For computing the first metric, a groundtruth is needed.

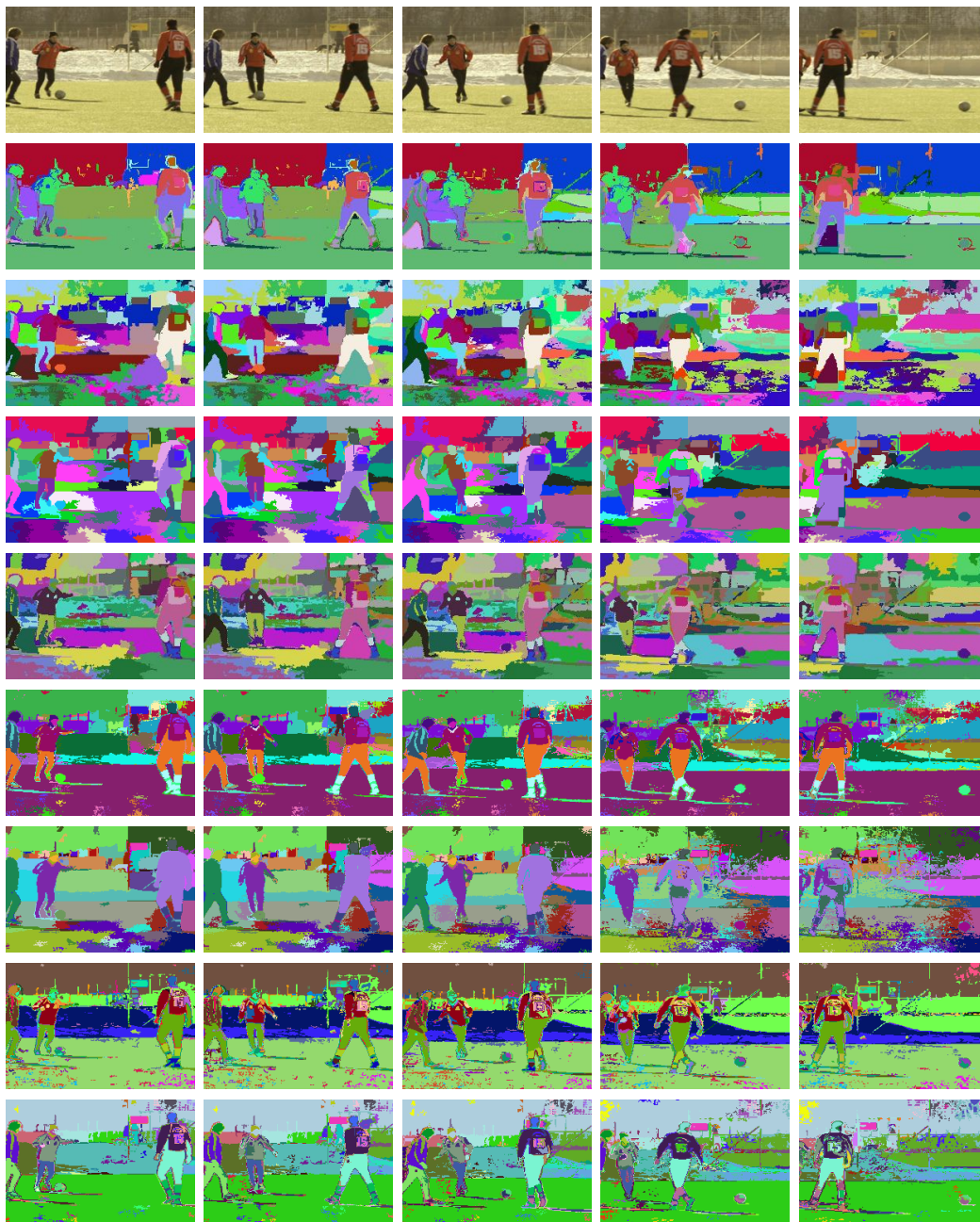


Experiments were performed using four versions of our approach: FVHOScale applied to the full video, StreamHOScale applied to video stream and the Over-FVHOScale and Over-StreamHOScale versions corresponding to execution of an oversegmentation process before performing the above methods.

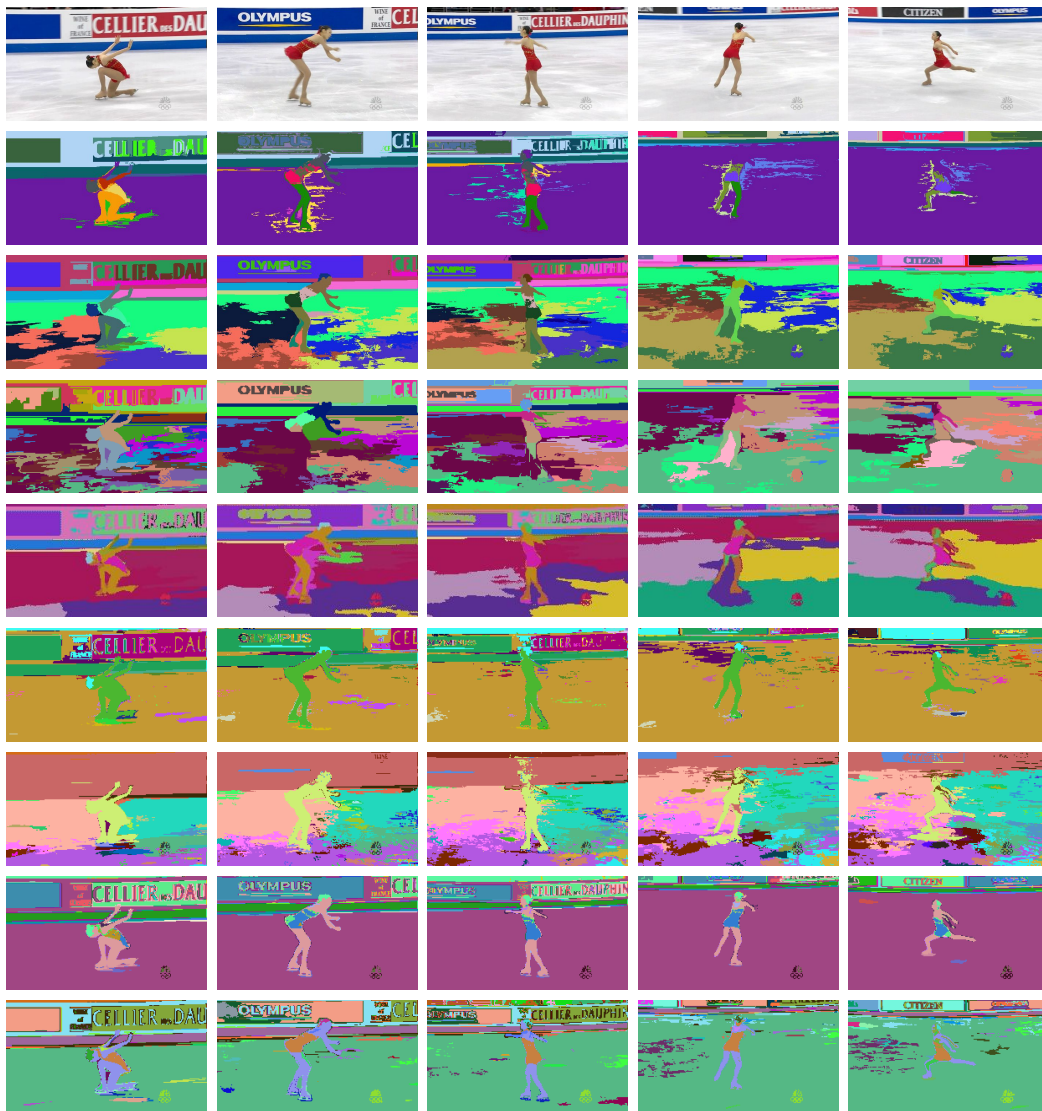
In Section 4.2, we presented experimental results to find the best parameters of our methods. After, in Section 4.3, we performed a comparative analysis of the results obtained by our methods using the HOScale approach to the methods of the state of the art. Our methods presented good quantitative and qualitative results when compared to the state of the art methods. Our methods shown to be better about spatiotemporal coherence. Concerning memory consumption and processing time cost, our methods showed less consume of memory than the other methods. Furthermore, the use of oversegmentation improved the quantitative and qualitative results, and also decreased the processing time.



**Figure 4.17.** Examples of video segmentations for a video extracted from the Chen dataset. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained by GB, GBH, StreamGBH, SWA, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale respectively. The parameters were tuned to obtain about 50 supervoxels.



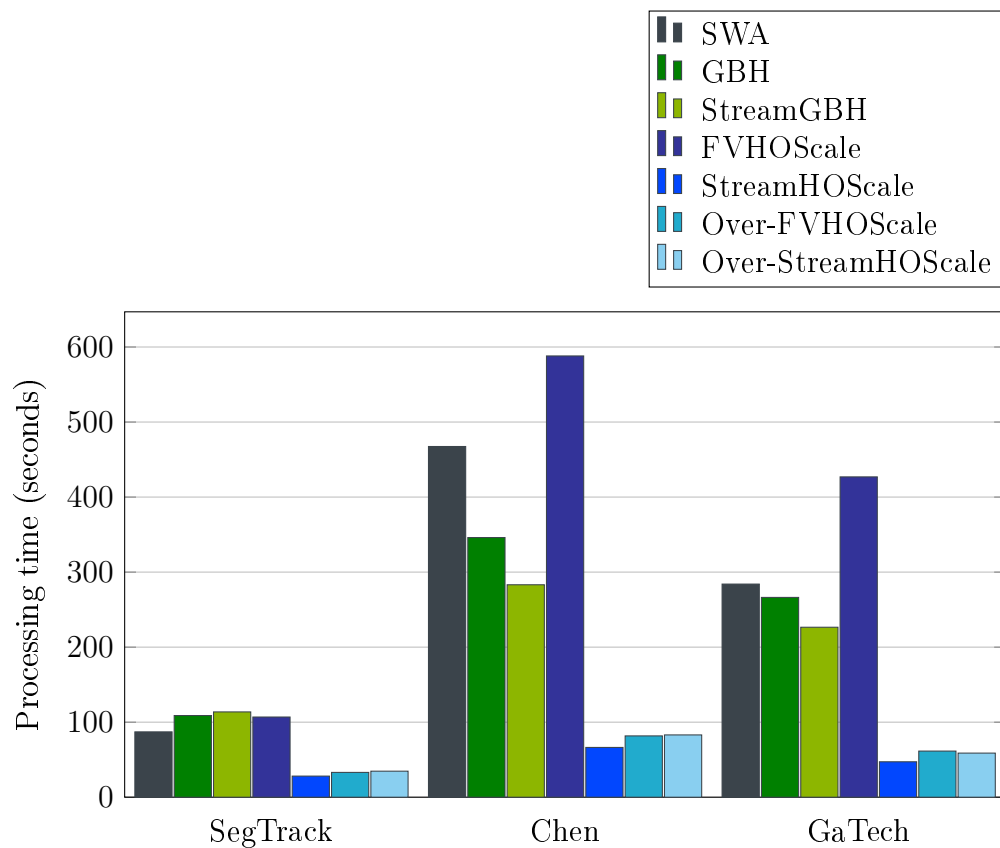
**Figure 4.18.** Examples of video segmentations for a video extracted from the Chen dataset. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained by GB, GBH, StreamGBH, SWA, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale respectively. The parameters were tuned to obtain about 100 supervoxels.



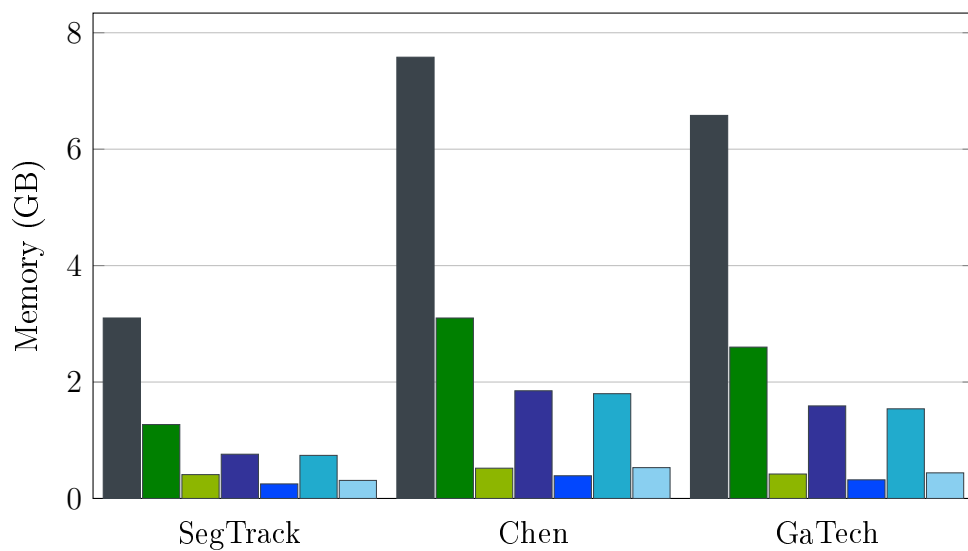
**Figure 4.19.** Examples of video segmentations for a video extracted from the GaTech dataset. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained by GB, GBH, StreamGBH, SWA, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale respectively. The parameters were tuned to obtain about 50 supervoxels.



**Figure 4.20.** Examples of video segmentations for a video extracted ofrom the GaTech dataset. The original frames are illustrated in the first row. The following rows, from top to bottom, illustrate the results obtained by GB, GBH, StreamGBH, SWA, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale respectively. The parameters were tuned to obtain about 100 supervoxels.



(a) Time cost.



(b) Space cost.

**Figure 4.21.** A comparison among our methods, FVHOScale, StreamHOScale, Over-FVHOScale and Over-StreamHOScale, and the methods GBH, StreamGBH and SWA when applied to SegTrack, Chen and GaTech datasets. The comparison is based on the following metrics: (a) time cost; (b) space cost.

# Chapter 5

## Conclusions and future works

In this work, we presented a new approach to hierarchical video segmentation based on computation of hierarchical observation scales. To compute our hierarchical scales, we proposed an approach to reweighting the minimum spanning tree computed from the video graph based on a criterion that measures the evidence for a boundary between two regions by comparing the intensity differences across the boundary and the intensity difference between neighboring voxels within each region. Finally, the partitioning of the graph, after the reweighting, is based on removing the edges whose weights (which represent the scales) are greater than or equal to a specified scale. Each graph region represents a video segment.

Our approach to hierarchical video segmentation can be divided into four main steps: (i) graph creation; (ii) computation of hierarchical scales; (iii) inference of video segmentation using thresholding; and finally, (iv) the graph is transformed in the segmented video. The graph creation is a very important step in this kind of application since it models the type of information to be used into vertices, and the relationships between the elements of the video, into edges. Therefore, we studied three possibilities for producing the video graph in order to verify the influence of color and pixel location: xyt graph, rgbxy graph and rgbxyt graph. In our experimental results we identified that the best results were obtained using rgbxy graph. In addition, we note that, in this step the best results were using the RGB color space, without using any image filter. Despite the Bilateral filter show slightly better results, it has a very high computational cost, thus we prefer

not to use any image filter.

We used the approach proposed to develop two video segmentation methods: FVHOScale, applied to full video segmentation and StreamHOScale, applied to streaming video segmentation. Unlike the other methods proposed in the literature, our methods provide all scales of observations instead of only one segmentation level. The great advantage of streaming method is the ability to run a video stream without the need of having all the video in memory, achieving to segment of consecutive frames blocks considering the temporal information present throughout the video. To perform our streaming method, we proposed a new and simple strategy for merging the results of two consecutive  $k$ -sized frame blocks, we produce good results preserving as much as possible the same quality measure of original one.

According to our experiments, the hierarchies inferred by our two methods, FVHOScale and StreamHOScale, produce good quantitative and qualitative results when applied to video segmentation. Furthermore, our oversegmentation method improve the accuracy and the computational cost of our methods. Moreover, unlike other tested methods, our methods are not influenced by the number of supervoxels to be computed, as shown in the experimental analysis, and present a low space and time cost.

For future works, we intend to investigate:

- New predicates to calculate the observation scale: we use the same predicate proposed by Felzenszwalb and Huttenlocher [2004], but we need to explore new ways of calculating of hierarchical observation scales using our approach.
- Automating the parameters: we need to search ways to find the best parameters for each video automatically.
- Apply our method in a real environment: using our segmentation method in other applications, to assist in the Content-Based Visual Information Retrieval process.



# Bibliography

- Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891--923.
- Bergh, M. V. D., Roig, G., Boix, X., Manen, S., and Gool, L. V. (2013). Online video seeds for temporal window objectness. In *2013 IEEE International Conference on Computer Vision*, pages 377--384.
- Chang, J., Wei, D., and Fisher III, J. W. (2013). A video representation using temporal superpixels. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '13*, pages 2051--2058, Washington, DC, USA. IEEE Computer Society.
- Chen, A. Y. C. and Corso, J. J. (2010). Propagating Multi-class Pixel Labels Throughout Video Frames. In *Proceedings of Western New York Image Processing Workshop*.
- Chen, J., Paris, S., and Durand, F. (2007). Real-time edge-aware image processing with the bilateral grid. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA. ACM.
- Comaniciu, D. and Meer, P. (2002). Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603--619.
- Corso, J. J., Sharon, E., Dube, S., El-Saden, S., Sinha, U., and Yuille, A. (2008). Efficient Multilevel Brain Tumor Segmentation With Integrated Bayesian Model Classification. *Medical Imaging, IEEE Transactions on*, 27(5):629--640.

- Cousty, J. and Najman, L. (2011). Incremental Algorithm for Hierarchical Minimum Spanning Forests and Saliency of Watershed Cuts. In *ISMM*, volume 6671 of *LNCS*, pages 272--283. Springer.
- Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient Graph-Based Image Segmentation. *IJCV*, 59(2):167--181.
- Fowlkes, C., Belongie, S., Chung, F., and Malik, J. (2004). Spectral grouping using the Nystrom method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):214--225.
- Fowlkes, C., Belongie, S., and Malik, J. (2001). Efficient spatiotemporal grouping using the Nystrom method. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I--231--I--238 vol.1.
- Galasso, F., Nagaraja, N. S., Cardenas, T. J., Brox, T., and Schiele, B. (2013). A unified video segmentation benchmark: Annotation, metrics and analysis. *Proceedings of the IEEE International Conference on Computer Vision*, pages 3527--3534.
- Gonzalez, R. and Woods, R. (2011). *Digital Image Processing*. Pearson Education.
- Grundmann, M., Kwatra, V., Han, M., and Essa, I. (2010a). Efficient Hierarchical Graph Based Video Segmentation. *Ieee Cvpr*.
- Grundmann, M., Kwatra, V., Han, M., and Essa, I. (2010b). Efficient hierarchical graph-based video segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2141--2148.
- Guimarães, S. J. F., Cousty, J., Kenmochi, Y., and Najman, L. (2012). A Hierarchical Image Segmentation Algorithm Based on an Observation Scale. In *SSPR/SPR*, pages 116--125.
- Guo, Y., Ma, W., Duan, L., En, Q., and Chen, J. (2016). Human action recognition based on discriminative supervoxels. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 3863--3869. IEEE.

- Jain, M., van Gemert, J., Jegou, H., Bouthemy, P., and Snoek, C. (2014). Action localization with tubelets from motion. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 740–747.
- Levinshtein, A., Stere, A., Kutulakos, K. N., Fleet, D. J., Dickinson, S. J., and Siddiqi, K. (2009). Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2290–2297.
- Li, C., Lin, L., Zuo, W., Wang, W., and Tang, J. (2016). An approach to streaming video segmentation with sub-optimal low-rank decomposition. *IEEE Transactions on Image Processing*, 25(5):1947–1960.
- Li, C., Lin, L., Zuo, W., Yan, S., and Tang, J. (2015). Sold: Sub-optimal low-rank decomposition for efficient video segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5519–5527.
- Liu, M. Y., Tuzel, O., Ramalingam, S., and Chellappa, R. (2011). Entropy rate superpixel segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2097–2104.
- Ma, S., Zhang, J., Ikizler-Cinbis, N., and Sclaroff, S. (2013). Action recognition and localization by hierarchical space-time segments. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2744--2751.
- Moore, A. P., Prince, S. J. D., Warrell, J., Mohammed, U., and Jones, G. (2008). Superpixel lattices. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8.
- Morris, O. J., Lee, M. J., and Constantinides, A. G. (1986). Graph theory for image analysis: an approach based on the shortest spanning tree. *Communications, Radar and Signal Processing, IEE Proceedings F*, 133(2):146--152.
- Niebles, J. C., Chen, C.-W., and Fei-Fei, L. (2010). Modeling temporal structure of decomposable motion segments for activity classification. In *Proceedings of the 11th European Conference on Computer Vision: Part II*, pages 392–405, Berlin, Heidelberg.

- Paris, S. and Durand, F. (2007). A Topological Approach to Hierarchical Segmentation using Mean Shift. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1--8.
- Pedrini, H. and Schwartz, W. R. (2007). *Análise de Imagens Digitais: Princípios, Algoritmos e Aplicações*. Editora Thomson Learning Edições Ltda.
- Peng, X., Qiao, Y., Peng, Q., and Qi, X. (2013). Exploring motion boundary based sampling and spatial-temporal context descriptors for action recognition. In *The 24th British Machine Vision Conference (BMVC)*, pages 1--11.
- Poppe, R. (2010). A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976 -- 990.
- Reso, M., Jachalsky, J., Rosenhahn, B., and Ostermann, J. (2013). Temporally consistent superpixels. In *2013 IEEE International Conference on Computer Vision*, pages 385--392.
- Sekma, M., Mejdoub, M., and Ben Amar, C. (2013). Human action recognition using temporal segmentation and accordion representation. In Wilson, R., Hancock, E., Bors, A., and Smith, W., editors, *Computer Analysis of Images and Patterns*, volume 8048 of *Lecture Notes in Computer Science*, pages 563--570.
- Sharon, E., Brandt, A., and Basri, R. (2000). Fast multiscale image segmentation. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 70--77 vol.1.
- Sharon, E., Galun, M., Sharon, D., Basri, R., and Brandt, A. (2006). Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442(7104):810--813.
- Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888--905.
- Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2009). Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *Int. J. Comput. Vision*, 81(1):2--23.

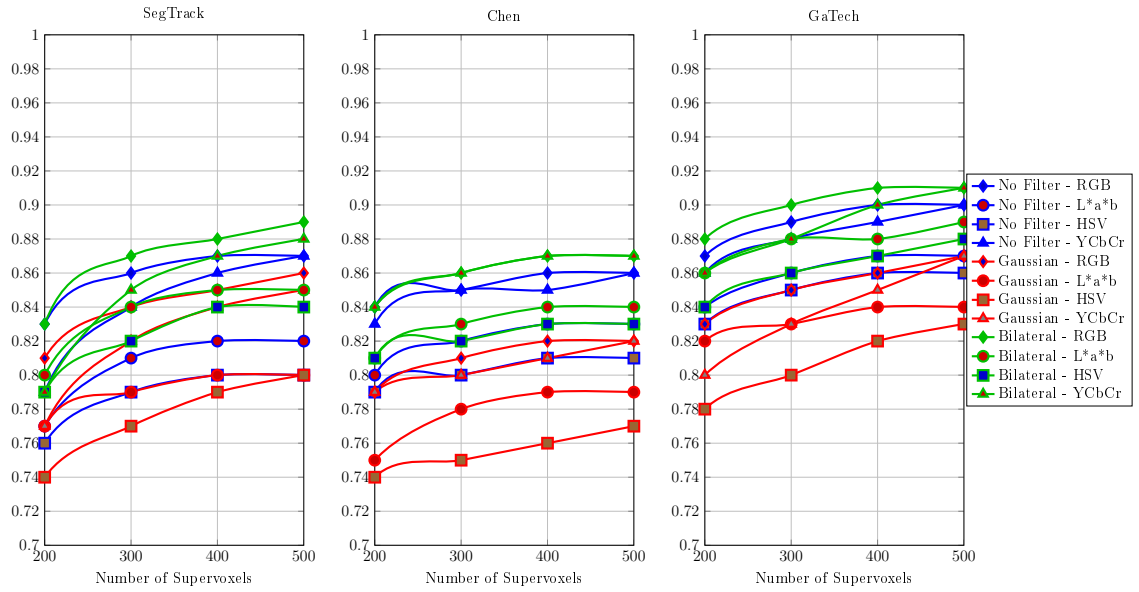
- Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Proceedings of the Sixth International Conference on Computer Vision, ICCV '98*, pages 839--846, Washington, DC, USA. IEEE Computer Society.
- Tripathi, S., Hwang, Y., Belongie, S., and Nguyen, T. (2014). Improving streaming video segmentation with early and mid-level visual processing. In *IEEE Winter Conference on Applications of Computer Vision*, pages 477--484.
- Tsai, D., Flagg, M., and M.Rehg, J. (2010). Motion Coherent Tracking with Multi-label MRF optimization. *BMVC*.
- Veksler, O., Boykov, Y., and Mehrani, P. (2010). Superpixels and supervoxels in an energy optimization framework. In *Proceedings of the 11th European Conference on Computer Vision: Part V, ECCV'10*, pages 211--224, Berlin, Heidelberg. Springer-Verlag.
- Winnemoller, H., Olsen, S. C., and Gooch, B. (2006). Real-time video abstraction. *ACM Trans. Graph*, 25:2006.
- Xu, C. and Corso, J. J. (2016). Libsvx: A supervoxel library and benchmark for early video processing. *International Journal of Computer Vision*, 119(3):272--290.
- Xu, C. and Corso, J. J. J. (2012). Evaluation of super-voxel methods for early video processing. ... *Vision and Pattern Recognition (CVPR), 2012* ... , pages 1202--1209.
- Xu, C., Xiong, C., and Corso, J. J. J. (2012). Streaming Hierarchical Video Segmentation. In *Proceedings of European Conference on Computer Vision*, pages 626--639.
- Yi, Y. and Lin, M. (2015). Human action recognition with graph-based multiple-instance learning. *Pattern Recognition*.
- Zeng, G., Wang, P., Wang, J., Gan, R., and Zha, H. (2011). Structure-sensitive superpixels via geodesic distance. In *2011 International Conference on Computer Vision*, pages 447--454.

Zhou, Q. and Wang, G. (2012). Atomic action features: A new feature for action recognition. In *Computer Vision - ECCV 2012 Workshops and Demonstrations*, volume 7583 of *Lecture Notes in Computer Science*, pages 291–300.

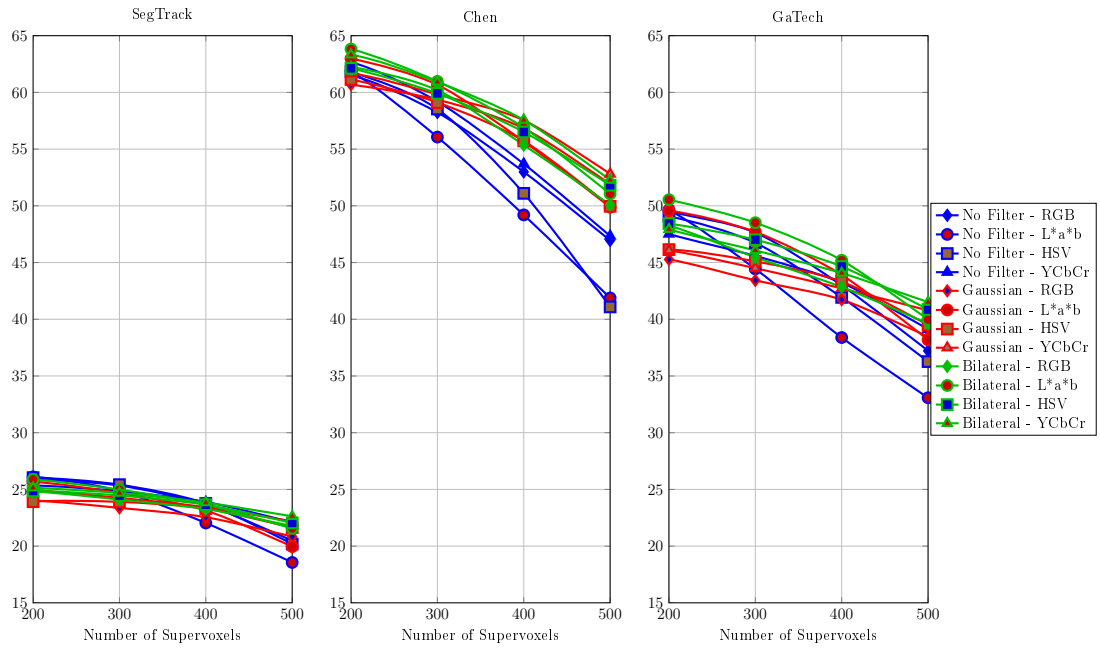
# Appendix A

## Parameters for video graph creation per dataset

Experimental results, of each dataset (SegTrack, Chen and GaTech), obtained in the search for the best parameters in the video graph creation step of FVHOScale method, as shown in Section 4.2.1.

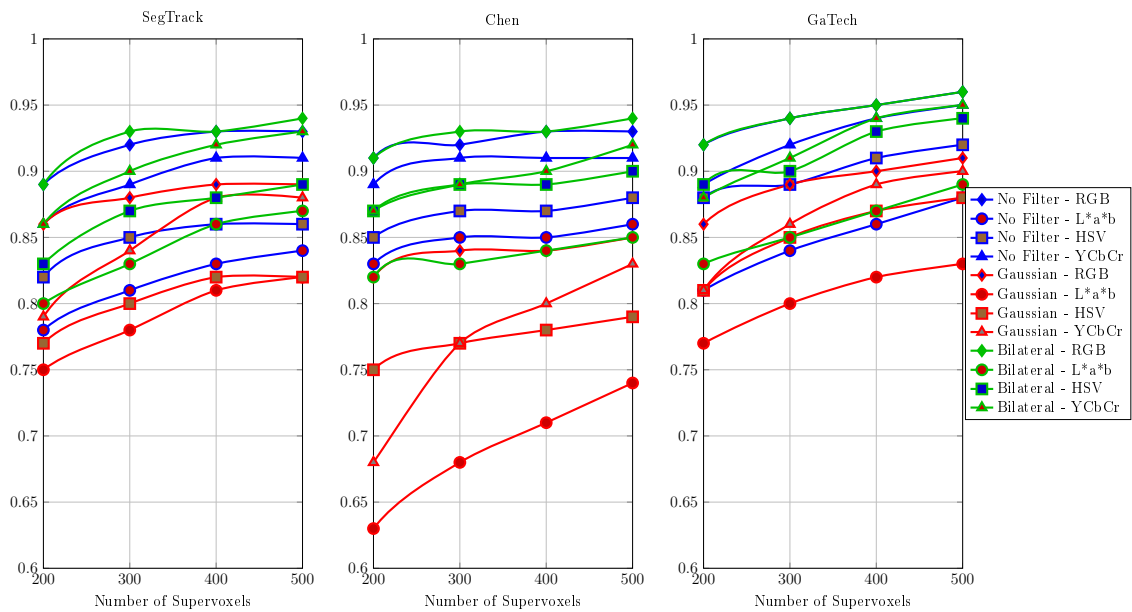


**Figure A.1.** A comparison with the parameters used in the creation of the video graph, separated by dataset using the xyt graph. The comparison is based on the Explained Variation metric.

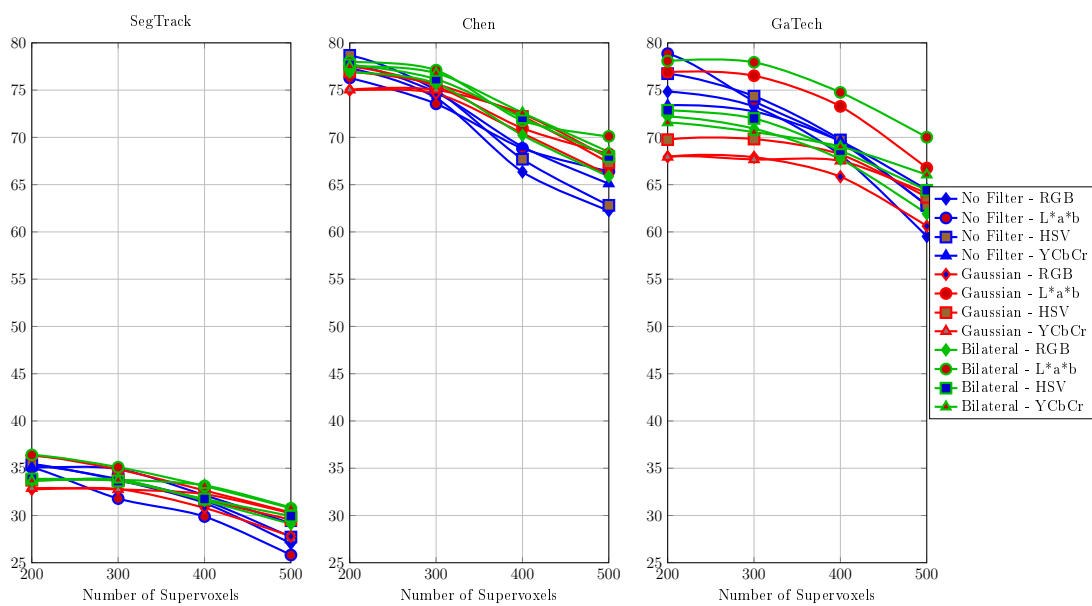


**Figure A.2.** A comparison with the parameters used in the creation of the video graph, separated by dataset using the xyt graph. The comparison is based on the Mean Duration metric.

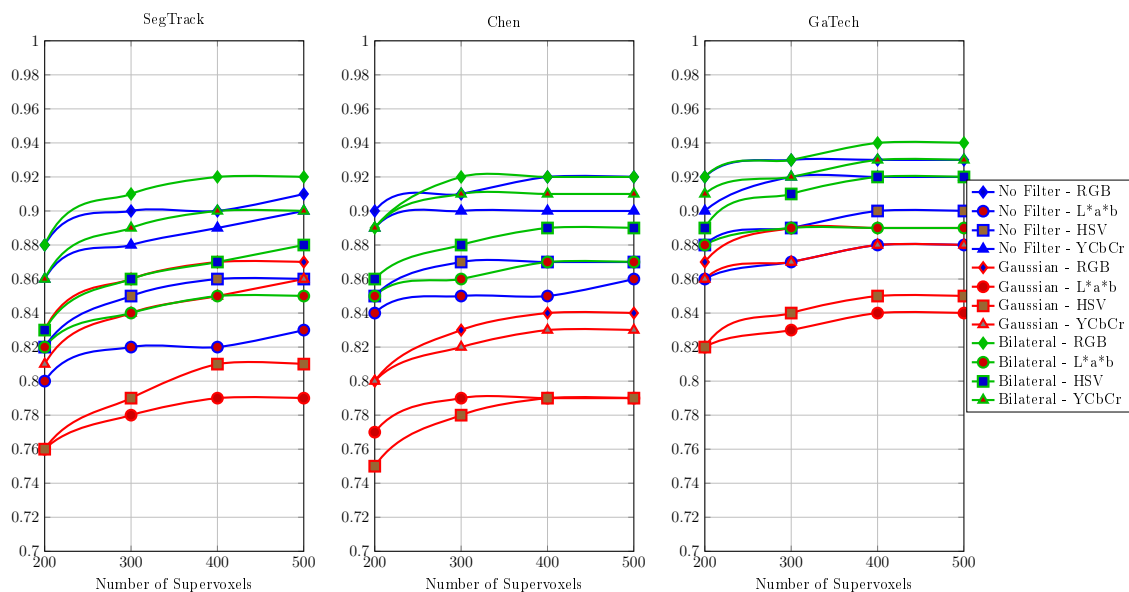




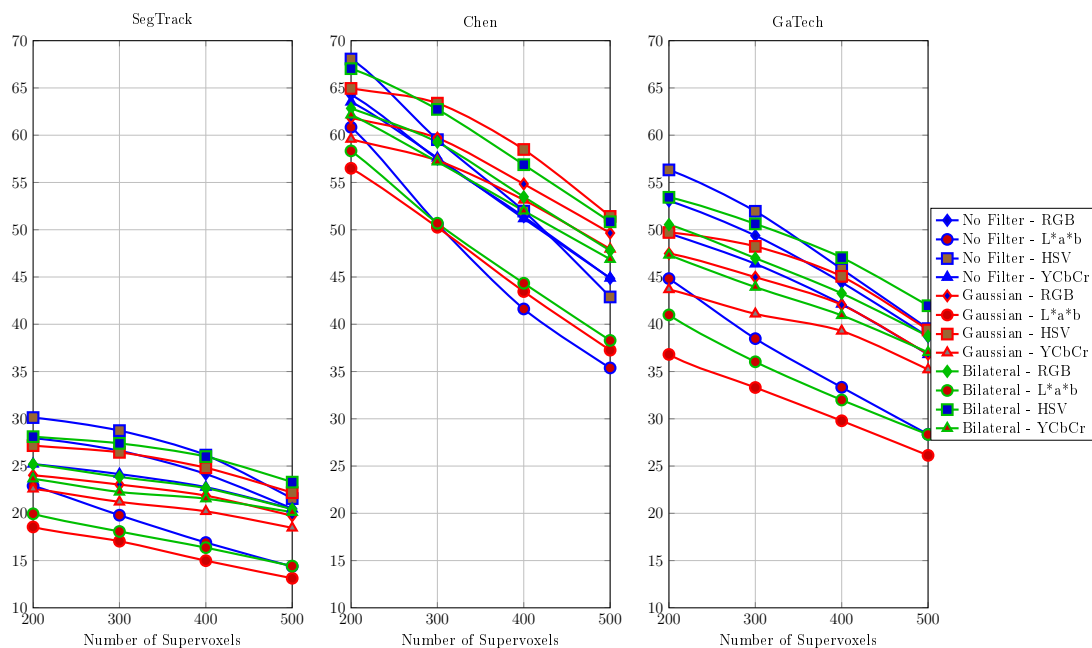
**Figure A.3.** A comparison with the parameters used in the creation of the video graph, separated by dataset using the rgbxy graph. The comparison is based on the Explained Variation metric.



**Figure A.4.** A comparison with the parameters used in the creation of the video graph, separated by dataset using the rgbxy graph. The comparison is based on the Mean Duration metric.



**Figure A.5.** A comparison with the parameters used in the creation of the video graph, separated by dataset using the rgbxyt graph. The comparison is based on the Explained Variation metric.



**Figure A.6.** A comparison with the parameters used in the creation of the video graph, separated by dataset using the rgbxyt graph. The comparison is based on the Mean Duration metric.