UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

EFFICIENT EXPLORATION OF REGION HIERARCHIES FOR SEMANTIC SEGMENTATION

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Míriam Bellver Bueno

In partial fulfilment

of the requirements for the degree in

SCIENCES AND TELECOMMUNICATION TECHNOLOGIES ENGINEERING

Advisors

Xavier Giró i Nieto

Carles Ventura

Barcelona, July 2015

Abstract

The motivation of this work is the efficient exploration of hierarchical partitions for semantic segmentation as a method for locating objects in images. While many efforts have been focused on efficient image search in large-scale databases, few works have addressed the problem of locating and recognizing objects efficiently within a given image.

My work considers as an input a hierarchical partition of an image that defines a set of regions as candidate locations to contain an object. This approach will be compared to other state of the art algorithms that extract object candidates for an image.

The final goal of this work is to semantically segment images efficiently by exploiting the multiscale information provided by a hierarchical partition, maximizing the accuracy of the segmentation when only a very few regions of the partition are analysed.

Resum

La motivació d'aquest treball és l'exploració eficient d'un arbre jeràrquic per tal de segmentar semànticament imatges com a mètode per reconèixer objectes. Molts treballs han tractat la cerca eficient d'objectes en imatges des del punt de vista global de la imatge en grans bases de dades, però no s'han dedicat tants esforços en resoldre el problema de localitzar i reconèixer objectes eficientment dins la pròpia imatge.

Aquesta tesis treballa amb particions jeràrquiques d'una imatge que defineixen un conjunt de regions candidates per contenir un objecte. Segmentar imatges utilitzant aquestes regions es compararà amb els resultats obtinguts amb candidats objecte extrets mitjançant altres algorismes de l'estat de l'art.

L'objectiu final és segmentar imatges semànticament de forma eficient aprofitant la informació entre nivells de l'arbre jeràrquic, maximitzant la qualitat de la segmentació quan només un conjunt molt reduït de zones de l'arbre són analitzades.

Resumen

La motivación de este trabajo es explorar eficientemente un árbol jerárquico para segmentar semánticamente imágenes como método para reconocer objetos. Muchos trabajos han tratado la búsqueda eficiente de objetos en imágenes desde el punto de vista global de la imagen en grandes bases de datos, pero no se han dedicado tantos esfuerzos en resolver el problema de localizar y reconocer objetos dentro de la propia imagen.

Esta tesis trabaja con particiones jerárquicas de una imagen que definen un conjunto de regiones candidatas para contener un objeto. Segmentar imágenes utilizando estas regiones se comparará con los resultados obtenidos a partir de candidatos de objeto extraídos mediante otros algoritmos del estado del arte.

El objetivo final es segmentar imágenes semánticamente de forma eficiente aprovechando la información entre niveles del árbol jerárquico, maximizando la calidad de la segmentación cuando sólo un conjunto muy reducido de zonas del árbol son analizadas.

Acknowledgements

First of all I would like to thank my advisor Xavi for all his dedication to this work, how he has made the impossible to gather all his students at least once a week so we could comment and share our findings, and feel that researching is better done in community. I also wanted to thank Carles for helping me in all the stages of the project, he has made that things looked much easier, and it is all thanks to his supportive attitude. I also wanted to thank Albert Gil and Josep Pujal for their technical support and for helping me in the most terrible moments of the research, when nothing works.

Mostly I wanted to thank my family, my parents and sister, who have always been there for me, being my best support. Finally I wanted to thank my partner for being always by my side and making me better day after day.

Revision history and approval record

Revision	Date	Purpose
0	6/07/2015	Document creation
1	6/07/2015	Document revision
2	9/07/2015	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Míriam Bellver Bueno	miriam.bellver@alu-etsetb.upc.edu
Xavier Giró i Nieto	xavier.giro@upc.edu
Carles Ventura	carles.ventura@upc.edu

Written by:		Reviewed and approved by:	
Date	9/07/2015	Date	9/07/2015
Name	Míriam Bellver Bueno	Name	Xavier Giró i Nieto
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures:

List of Tables:

1. Introduction

1.1. Statement of purpose

This thesis intends to efficiently **explore a hierarchical partition** for semantically segment an image in order to locate its objects and boundaries. A **semantic segmentation** of an image aims **at obtaining a reduced set of regions** defined by precise boundaries, labelling each region by its class category. The input of our algorithm is a set of object proposals that can be computed using different techniques.

In this research, the different **regions defined by a hierarchical partition [2, 3]** are considered candidate locations to contain an object. Other approaches in the state of the art are techniques based on the **generation of a ranked list of accurate segments [5, 11]** or **bounding boxes [10]** that are more likely to locally describe the objects of the image. Our work will compare the object proposals generated with the hierarchical partition with those from a popular baseline technique for object proposals: **CPMC [5].**

Efficient object recognition is essential for many computer vision applications such as object retrieval. The **goal** of this research is **gaining efficiency at the expense of losing accuracy by means of using hierarchical partitions**. Efficiency will be measured by the **number of regions** analysed and the **computation time** of each region by the algorithm at issue, whereas **accuracy** will determine the quality of the image semantic segmentation prediction compared to Ground-Truth.

Considering the scope of this research, hierarchical partitions have the advantage that provide **multiscale information**, which is fundamental in order to guide an efficient exploration through the tree partition. However, **techniques based on object candidates generate more accurate object proposals** than the regions obtained from a hierarchical partition.

Another goal of the project is to compare the performance achieved when describing the different types of object proposals with **handcrafted features [12]** or state of the art descriptors extracted from a **convolutional neural network [8]**.

This project aims at continuing the research that Xavier Giró, my advisor, started with his PhD Thesis, developing some of the ideas that arose during his research. Xavier Giró's thesis addressed **Part-based Object Retrieval with Binary Partition Trees [1]**. He guided a research about **Binary Partition Trees [2],** comparing the regions the algorithm generates to objects that have a semantic meaning. In order to retrieve objects from images he worked with **Bag of Words aggregation [16]**, developing an efficient strategy to explore the binary tree partition. This strategy consisted of **bottom-up extracting features of the hierarchical partition**, so that a region was described by all the objects contained in the region and not only by the region itself. Using such descriptors, some branches of the tree could be **discarded** while trying to detect an object through a top-down exploration of the tree.

Following the steps of Xavier Giró, this work is based on developing an efficient strategy to semantically segment images taking advantage of the **multiscale information** that hierarchical partitions provide. For this research **Ultrametric Contour Maps (UCM) [3]** hierarchical partitions and **Second Order Pooling (O2P) descriptors [4]** will be combined in the first place. For the purpose of comparing hierarchical partitions regions

to object-candidates-based algorithms, **Constrained Parametric Min-Cuts (CPMC) [5]** object candidates will be tested. The software developed to perform this research draws from a source code of **João Carreira [4, 5]** that **Carles Ventura,** PhD candidate and coadvisor of this thesis, had already used for his research. The code computes semantic segmentations of images using **CPMC object candidates [5]** and **O2P descriptors [4]**. Carles Ventura has provided me this code and also a code to generate **UCM partitions [3]**.

To perform the final experiments of this project, we replaced the **handcrafted O2P descriptors** with **SDS deep learning descriptors [6]** extracted from **convolutional neural networks [8]. Eduard Fontdevila** BSc student has provided a code to extract these descriptors used in the framework of his **BSc thesis [17].**

All this research has been developed using **Matlab** and the **Image Processing Group** platform in UPC.

1.2. Work packages

In the following tables there are the work packages of this project.

Project: Project Proposal	WP 1	
Major constituent: Documentation		
Short description: Planning and description of the project.	Planned start date: 19/02/15 Planned end date: 6/03/15	
	Start event: 19/02/15 End event: 6/03/15	
T1: Definition of the goal of the project. **T2**: Planning of the project **T3**: Documentation about the project proposal **T4**: Revision of the project proposal **T5**: Validation of the project proposal	Deliverables: Project Proposal	Dates: 6/03/15

Project: Research about the state of the art	WP 2	
Major constituent: Documentation		
Short description: Reading state of the art papers about semantic segmentation, hierarchical partitions, object candidates and object recognition in images.	Planned start date: 9/02/15 Planned end date: 25/02/15	
	Start event: 9/02/15 End event: 25/02/15	
T1: "*Part-based Object Retrieval with Binary Partition Trees*", Xavier Giró, Phd thesis **T2**: "*Object Recognition by Ranking Figure-Ground Hypotheses*" ,Joao Carreira. **T3**: "*Segmentation as Selective Search for Object Recognition*", Koen E. A. van de Sande and Theo Gevers	Deliverables: Blog posts on Bitsearch summing up papers.	Dates: Undefined.

<table>
<tr><td>Project: Critical Revision</td><td colspan="2">WP 3</td></tr>
<tr><td>Major constituent: Documentation</td><td colspan="2"></td></tr>
<tr><td rowspan="2">Short description: Compiling the work carried out until that moment.</td><td colspan="2">Planned start date: 1/04/15
Planned end date: 8/04/15</td></tr>
<tr><td colspan="2">Start event: 13/04/15
End event: 24/04/15</td></tr>
<tr><td>T1: Compile all the work done during these first weeks.
T2: Write the critical revision following the template.
T3: Revision of critical revision.
T4: Validation of critical revision
T5: Deliver it in Atenea.</td><td>Deliverables:
Critical Revision</td><td>Dates:
24/04/2015</td></tr>
</table>

<table>
<tr><td>Project: Adaptation of software</td><td colspan="2">WP 4</td></tr>
<tr><td>Major constituent: Software development</td><td colspan="2"></td></tr>
<tr><td rowspan="2">Short description: The goal of this package is adapting the already existing programs in GPI to make measurements of efficiency depending on the number of segments or object candidates analyzed.</td><td colspan="2">Planned start date: 2/03/15
Planned end date: 13/04/15</td></tr>
<tr><td colspan="2">Start event: 2/03/15
End event: 18/05/15</td></tr>
<tr><td>T1: Learn how to work with GPI servers.
T2: Understanding the code of Carles Ventura of CPMC object candidates.
T3: Adapting this software to calculate the accuracy of the segmentation depending on the number of objects candidates analyzed.
T4: Understanding Carles Ventura's matlab code to work with UCM.
T5: Adapting UCM software to generate its masks and descriptors, and generating a naive list of regions.
T6: Adapting Carles Ventura's software to make a selective search using UCM.
T7: Deliver the resulting codes to the Group of Image Processing with its documentation.</td><td>Deliverables:
Resulting code and documentation</td><td>Dates:
18/05/15</td></tr>
</table>

<table>
<tr><td>Project: Scientific publication redaction</td><td>WP 5</td></tr>
<tr><td>Major constituent: Documentation</td><td></td></tr>
<tr><td rowspan="2">Short description: Writing a scientific publication for CVPR WiCV.</td><td>Planned start date: 23/03/15
Planned end date: 23/04/15</td></tr>
<tr><td>Start event: 23/03/15
End event: 23/04/15</td></tr>
</table>

T1: Writing publication for CVPR WiCV.	Deliverables: Deliver publication.	Dates: 23/04/15

Project: Experimentation	WP 6	
Major constituent: Research and assessment of results		
Short description: Perform the comparison between the efficiency achieved using hierarchical partitions and the efficiency obtained with algorithms that use object candidates such as CPMC. Also try to test the best configurations using deep learning features.	Planned start date: 13/04/15 Planned end date: 15/06/15	
	Start event: 13/03/15 End event: 15/06/15	
T1: Calculate accuracy per category using CPMC object candidates. **T2:** Calculate accuracy per category using a naive list of UCM regions. **T3**: Calculate accuracy per category using UCM regions and a efficient exploration throughout the partition. **T4:** Analyze results comparing in terms of accuracy and efficiency. **T5**: Train the algorithm to use SDS features **T6**: Test best configurations with SDS features. **T7:** Think about further possible improvements. **T8:** Prepare the results for the memory of the project.	Deliverables: Part of this experimentation should be on the Critical Revision Document	Dates: 30/06/15

Project: Redaction of the memory and oral defence	WP ref: 7	
Major constituent: Documentation		
Short description: Writing the final memory for the TFG and preparing the oral defense.	Planned start date: 15/05/15 Planned end date: 15/07/15	
	Start event: 30/05/15 End event: 24/07/15	
T1: Writing the memory **T2:** Revision of memory **T3:** Deliver the memory **T4:** Prepare the oral defense **T5:** Rehearsal of the presentation **T6:** Doing the final presentation	Deliverables: Final Memory of the TFG	Dates: Delivery of final memory: 10/07/15 Oral defense: 20/07/15

1.3. Milestones

In the following table the milestones of this project are listed:

WP#	Task#	Short title	Milestone / deliverable	Date (week)
1	1	Validation of Project Proposal	Project Proposal	6/03/15
2	2	Read Xavier Giró's PhD	Understanding of the content	3th week
2	3	Read papers about segmentation	Research about state of the art	4th week
4	4	Adapting CPMC code	Obtain CPMC code that enables to determine segments used	13/03/15
3	5	Critical Revision	Critical Revision deliverable	24/04/15
4	4	Adapting UCM code	Adapt code to study efficiency with UCM	18/05/15
5	5	Write scientific publication	Deliver to CVPR for WiCV the publication.	23/04/15
6	6	Compare efficiency of different segmentation techniques	Obtain results of the experimentation	25/06/15
6	7	Train the algorithm with SDS descriptors and check results	Obtain results of the experimentation and compare them to other features	30/06/15
6	8	Further improvements	Reflect about the results obtained.	30/06/15
7	9	Write the memory of the project	Deliverable of the final memory	10/07/15
7	10	Rehearse the project presentation	Rehearsal of the oral presentation	17/07/15
7	11	Oral presentation of the project	Oral presentation of the project	20/07/15

1.4. Gantt Diagram

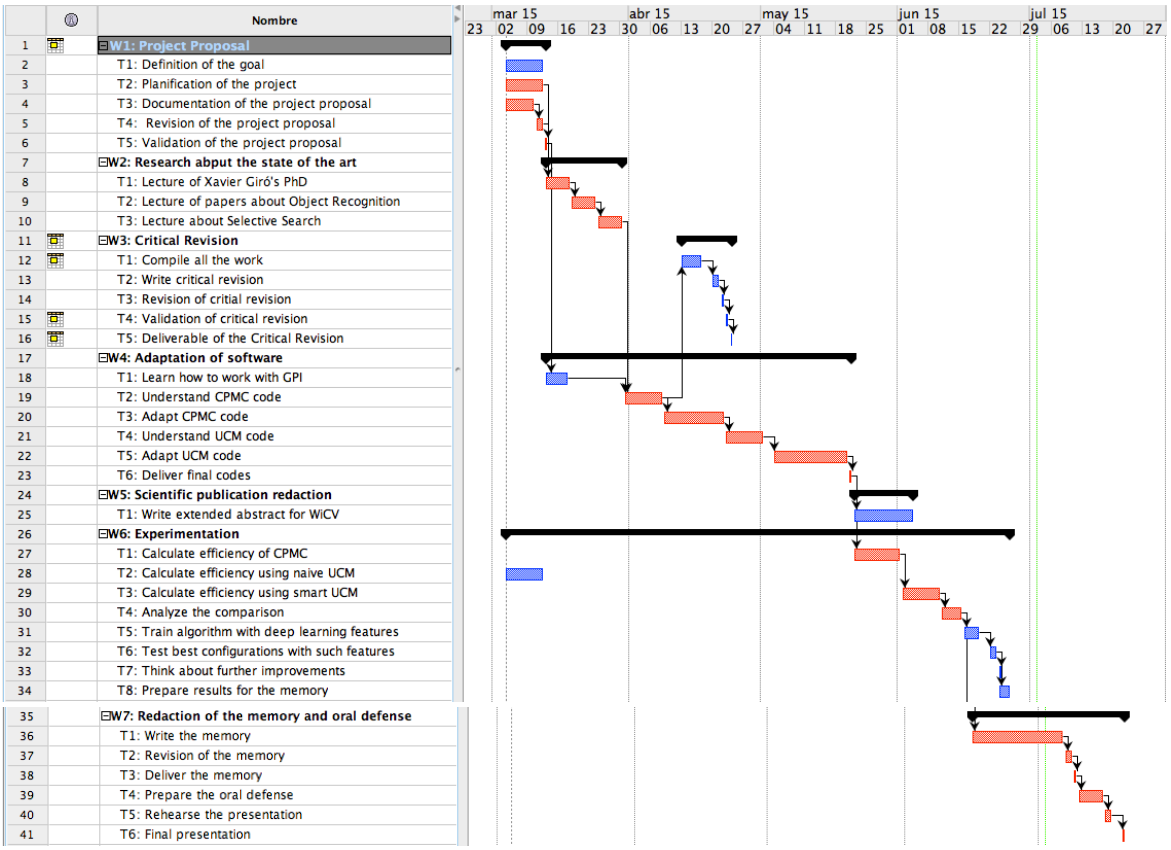


Figure 1: Gantt diagram

1.5. Incidences

Since the critical review submission, a few incidences have changed slightly the work packages, milestones and Gantt diagram.

For instance, the WP5 tasks have changed, due that the extended abstract **(Appendix III)** submitted to **Woman in Computer Vision (WiCV)** workshop of **International Conference on Computer Vision and Pattern Recognition (CVPR)** was not accepted, and therefore no poster had to be prepared. Moreover, it was originally planned a submission to the **LSUN** workshop of Computer Vision but we considered that the results we got on the date of submission were not promising enough yet.

Regarding the **WP6** of experimentation, a few tasks have been included, such as **training and testing the algorithm with deep learning SDS descriptors** in order to see whether they obtain better results.

The other work packages remain the same, considering a slight change of dates due the two incidences previously explained.

2. State of the art of the technology used or applied in this thesis:

2.1. Overview of the state of the art

The traditional local analysis of an image is based on scanning with a sliding window different locations and scales of an image. This exhaustive approach was combined with an efficient feature extraction in the classic work by **Viola and Jones [7]** and is also at the core of popular **convolutional neural networks [8]**.

An alternative to exhaustive search is using class-agnostic image processing to estimate the most feasible locations in an image to contain an object. A first family of solutions is based on the **saliency maps [9]**, which typically assign to each pixel a likelihood value that predicts the user attention. These solutions though do not directly provide a region of support for the object.

A different approach aims at reducing the amount of possible locations for an object by clustering the pixels with a **segmentation algorithm**. In these cases, a reduced set of regions is automatically defined with precise boundaries. However, flat segmentations tend to focus their analysis at a certain spatial scale, which is not rich enough when the size of the object is unknown. As an alternative, **hierarchical segmentations [2, 3]** provide a nested set of regions that capture a broad range of scales.

A last group of techniques are based on generating a ranked list of **object candidates** in the image, whether as **bounding boxes [10]** or **accurate segments [3, 5, 11]**. These techniques try to model the generic appearance of an object so that class-specific detectors are trained on them.

This thesis compares the hierarchical segmentations generated by **Ultrametric Contours Map (UCM) [3]** with a popular technique that uses object candidates, **Constrained Parametric Min-Cuts (CPMC) [5]**. The features that will describe both the regions of the **UCM [3]** hierarchical partition and the object candidates of the **CPMC [5]** are **Average Pooling of SIFTs (O2P) [4]** and **SDS [6]**. The following sections aim at further developing these concepts that will be relevant for this research.

2.2. Semantic Segmentation

Semantic segmentations are an approach to solve the **object recognition task**. In a semantic segmentation every pixel of the image is labelled by its **object class category**. The result is an image divided into **segments** that correspond to the objects of the scene. In the following example we can see the difference between object segmentation and semantic segmentation.

Figure 2: The second image is an object segmentation, labelling each object with a different colour. The third image, however, corresponds to a semantic segmentation, identifying that both objects in the image belong to the same class category. This image belongs to PASCAL VOC dataset [14]

In an object segmentation the algorithm detects and segments objects in an image without classifying their class category. In a semantic segmentation each pixel is **labelled by its class category,** without distinguishing the different instances of the class that may appear in the image. In the above example the object segmentation identifies two objects whereas the semantic segmentation provides information only of the semantic class associated to the pixels. By combining the object and semantic segmentation it is possible to both distinguish the amount of object instances and their associated semantic classes. The evaluation of our system focuses on the semantic segmentation of the image, although in our case we could also provide its object segmentation.

2.3. Ultrametric Contour Map (UCM)

An **Ultrametric Contour Map [3]** is a hierarchical partition which is generated by a graph-based region merging algorithm, that iteratively fuses pairs of similar regions, i.e. the two adjacent regions separated by the minimum weight boundary, starting from a fine partition at a super pixel level.

The base level of the hierarchical partition is formed by those regions that are more easily merged. On the other hand the upper levels of the partition are formed by those nodes that are more difficult to generate because they are more dissimilar.

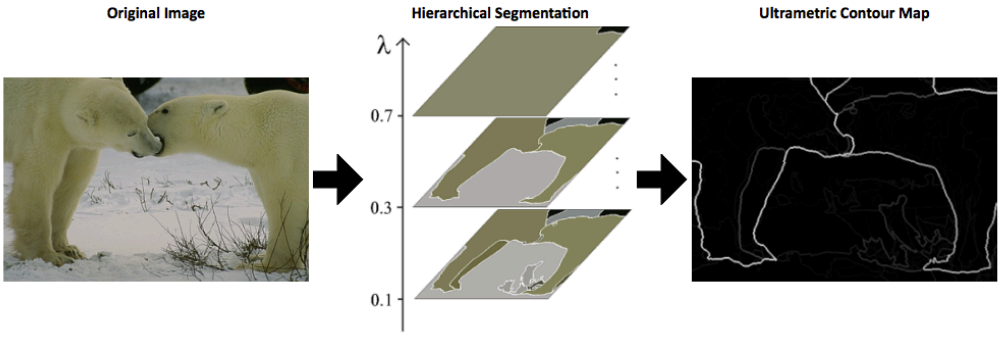


Figure 3: Hierarchical segmentation and Ultrametric Contour Map (UCM). This image is from [3].

2.4. Constrained Parametric Min-Cuts (CPMC)

CPMC [5, 15] generates a ranked list of object candidates without previous knowledge of the objects contained in the image. This is achieved by solving a sequence of **Constrained Parametric Min-Cuts**. In order to initialize the algorithm, some pixels should be considered as **background** and others as **foreground**. Once made this

selection, a sequence of **Min-Cuts [5,15]** will segment the image into a **set of accurate regions**, which should be filtered and regrouped so that regions not accurate enough or redundant are discarded.

Finally these object candidates are **scored** and **sorted** depending on the likelihood that each region is an object. A **learning scoring function** for each object **category** is used to obtain the score. Finally we obtain a list of ranked object candidates according to their likelihood of being an actual object.

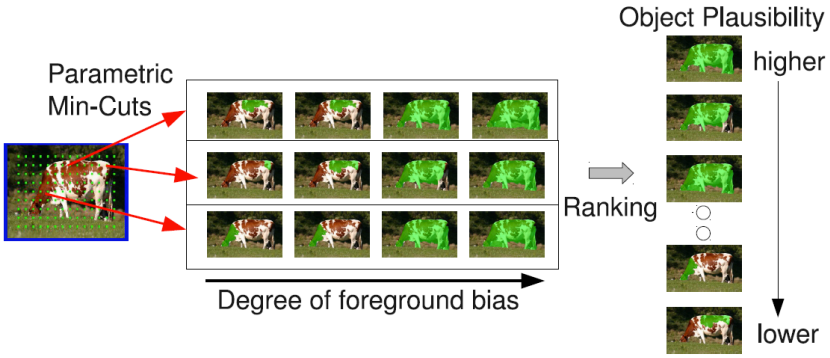


Figure 4: This picture shows a list of CPMC object candidates ranked by its likelihood of being an object. The image is from [5].

2.5. Second Order Pooling (O2P)

Scale-invariant feature transform (SIFT) [12] is an algorithm that detects and describes local features in images. It provides a feature description in terms of texture and contours. SIFT local features are extracted over square patches centered at image locations with a certain pixel width.

Pooling is the procedure that produces a global description of an image region. **Second Order Pooling [4, 13]** is a pooling technique that obtains second-order statistics by computing the outer products of SIFT local features. In order to do the final aggregation there are two different types of pooling:

- **Average Pooling**: Averaging the second order **SIFT's**.
- **Max Pooling**: Selecting the maximum element for each position of the matrix among all second order **SIFT's**.

The aggregated matrix is two-dimensional and **symmetric**, so half of the matrix contains all the information of the matrix. To reshape this two-dimensional descriptor into a single vector, half of the aggregated matrix is **zigzag scanned [13].**

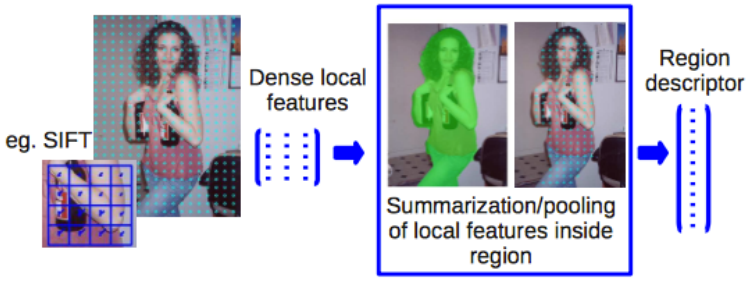


Figure 5: Pooling of SIFT descriptor in order to obtain an O2P region descriptor. This picture is from [13].

2.6. Simultaneous Detection and Segmentation

Simultaneous detection and segmentation (SDS) [6] solves the task of detecting objects of a certain class category of an image and labelling the pixels that belong to such class. The SDS algorithm consists of first generating region object proposals, for which **Multiscale Combinatorial Grouping (MCG) [11]** object proposals are used.

MCG [11] generates accurate object candidates from the combination of regions that **UCM hierarchical partition [3]** generates. Their approach consists of extracting features of these object proposals using a **convolutional neural network (CNN) [8]*** that extracts features from both the **bounding box** as well as the **region** of the object candidate. Two networks, one for the bounding boxes and another for the regions, are **fine-tuned*** and jointly trained to obtain **descriptors tailored for the semantic segmentation task on the Pascal VOC dataset [14]**. After that, a **SVM*** is trained to obtain for each object candidate a **score** that captures the confidence of belonging to each object class category.

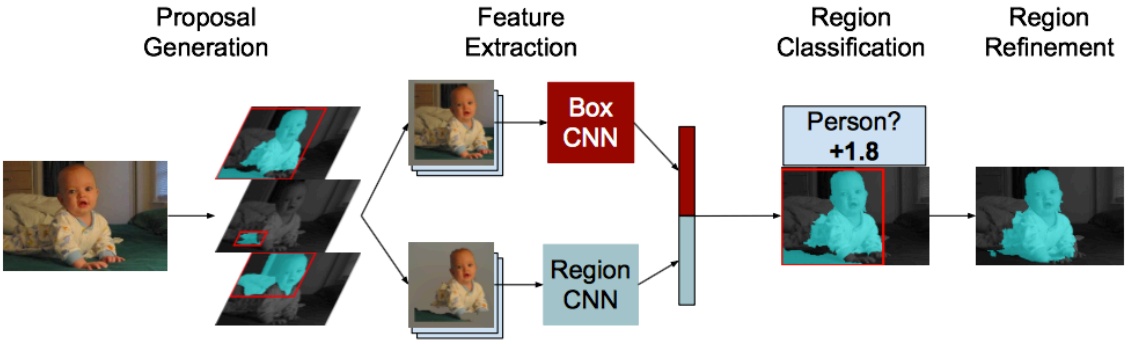


Figure 6: Pipeline followed for the SDS task. The first CNN is for the bounding boxes whereas the second is for the regions. The concatenation of the descriptors obtained from both networks generates the final descriptor.

* CNN, fine-tuning and SVM are described at the glossary.

3. Methodology / project development:

3.1. Dataset: Pascal VOC

This research is assessed on the segmentation challenge of the **Pascal Visual Object Classes Challenges** (**PASCAL) benchmark [14**]. **PASCAL VOC** is a network of excellence in the research field of computer vision that provides standardised image data sets for object class recognition and also runs challenges assessing performance on recognition tasks from a number of visual object classes in realistic scenes. This work is based on the segmentation challenge that PASCAL VOC provides, which is defined as following:

"*Generating pixel-wise segmentations giving the class of the object visible at each pixel, or "background" otherwise."*

The **segmentation challenge aims at detecting and segmenting 20 classes of objects**. The data set of PASCAL that was given to accomplish this challenge consists of

- 1112 training images
- 1111 validating images
- Testing images

The segmentation challenge provides a set of **training** images to train the algorithm and a set of **validating** images to test its performance before submitting the task. The final assessment of the algorithm on the challenge is computed using the **testing** images that only PASCAL possesses.

3.2. Metric: Accuracy Per Category (AAC)

The segmentation accuracy is measured as the **Intersection over Union (IoU) metric** of the predicted segmentation and the Ground-Truth, averaged across all object classes and the background. In this work we will refer to this measure Average Accuracy per Category (AAC) [15].

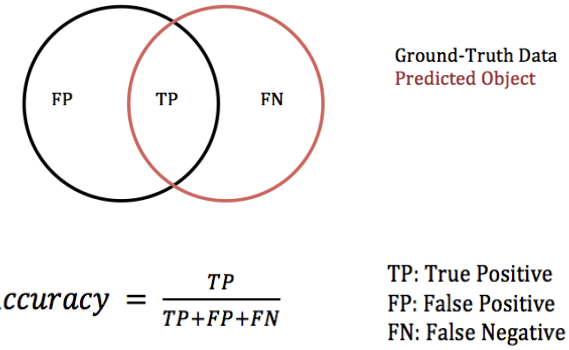


Figure 7: This figure depicts the overlap between the predicted object and the ground truth, and the nomenclature of the pixels to compute AAC.

3.3. Experimentation

Now that the dataset and the metric used to assess performance have been defined, this section will address the experiments performed during this thesis.

3.3.1. Pipeline

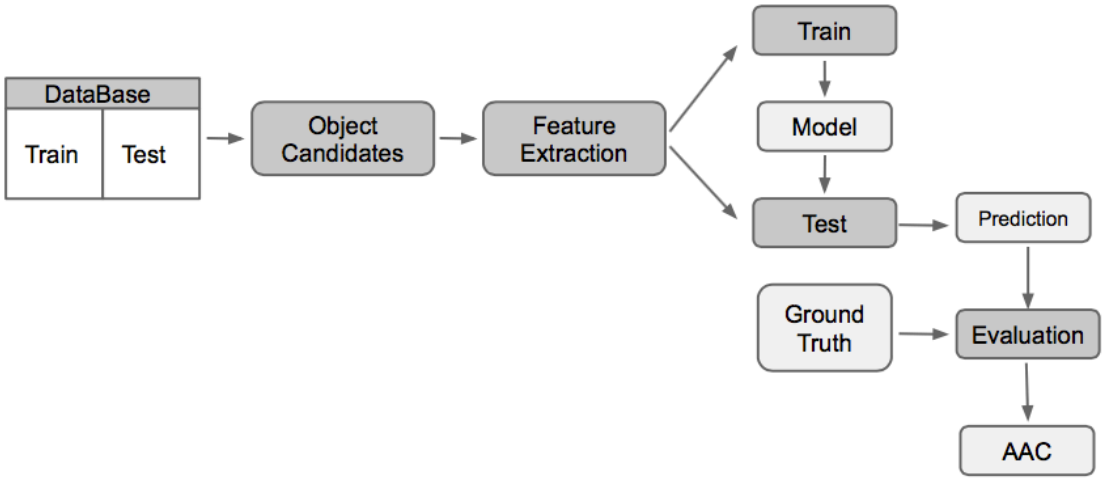


Figure 8: Pipeline of the experiments carried out in this thesis.

Figure 8 illustrates the pipeline of the experiments. The goal of this research is achieving an **efficient semantic segmentation** in terms of the **number of object proposals** analysed and the **accuracy obtained**. Firstly we have a dataset partition for training and another for testing the performance of the algorithm. **PASCAL [14]** already provides a train and validation partitions to accomplish this role, with their corresponding ground truths that allow both training and assessment.

In the **training** phase, we aim at generating an algorithm that semantically segments our images. For every image of the training set, we generate a set of object candidates. Two algorithms will be used to obtain object candidates: **CPMC [5]**, based on the idea of generating regions that define objects of the image, and **UCM [3]** which generates a hierarchical partition, whose regions will be considered object candidates. The second step is extracting the features of these regions. For the first experiments in this work, **O2P [4]** descriptors will be used. Features extracted from the **SDS [6]** architecture will also be tested at the final stage of the experimentation.

The next step is training a linear **support vector regressor (SVR) [4]** for each object class category to predict the overlap between an object candidate and the object that defines each class category. In the **training phase** we aim at obtaining a **model** that determines each object class category, so we will obtain as many models as object class categories we are dealing with. With the **Ground-Truth object segments** for every class category and the **object candidates with higher overlap with the Ground-Truth**, we train an SVR for each class category.

Once all the SVRs are trained, we will be able to obtain for every object proposal a vector whose elements are related to the **likelihood of that object to belong to each object class category**. These vector elements are the soft-decisions obtained from every SVR classifiers, and capture the **confidence values** of belonging to each category.

Once the algorithm is trained, it is time to **test** it. For testing first we **obtain the object candidates** for those images from the test dataset and **extract their features**. The algorithm outputs the **predicted confidence values** of each object candidate with the different object class categories, and predicts a **semantic segmentation** from the different confidence values and the **estimation of objects contained in an image**, which for PASCAL VOC dataset is **2.2 objects per image**. The segmentation obtained is assessed using the **AAC metric**.

3.3.2. Implementation

This research has used as starting point the source code of **João Carreira [15]** in MATLAB, which segments images using **150 CPMC** object candidates and **O2P features [4]**. The code also assesses the segmentation accuracy with **AAC**. For the purpose of measuring **efficiency in terms of the number of regions analysed** using **different types of object candidates** and **descriptors**, the following changes were required:

- Enable to **set the number of object candidates,** so we can later compare **efficiency** and assess **AAC vs. number of object candidates**.
- Enable the algorithm to use **UCM regions** as object candidates and extract **their features**.
- By default CPMC object candidates are sorted by their likelihood of being an object. This work has researched several alternative approaches for UCM, trying to find **which regions of the hierarchical partition are more likely objects**, so the calculation of different orders of these regions following different criterions should be coded. Also the code needed to be adapted to **efficiently calculate AAC vs. number of object candidates of different orders** of the same regions.
- **Extract SDS [6]** descriptors for UCM different configurations and CPMC and **train the algorithm** using such descriptors. The code to extract SDS descriptors has to be adapted to extract **CPMC** and **UCM** regions, due that it was originally designed for **MCG [11]** regions.

The core of this research has been testing **different methods for sorting UCM regions and assessing AAC depending on the number of regions analysed**. Notice that the number of regions analysed is the budget of regions introduced into the segmentation algorithm. If a budget of 5 regions is considered, it means that we introduce 5 regions into the segmentation algorithm, and the algorithm decides which regions to select in order to better semantically segment the image, considering that the average number of objects of an image of the PASCAL VOC data set is of 2,2 objects, a pretrained value kept in our work to allow a fair comparison with the CPMC-based solution of [15].

First of all **some general features of the UCM** hierarchical partitions structure should be highlighted:

- A UCM tree has a total of **(2*(number of leaves) -1)** nodes.
- Each region of the partition is represented by an **index**. The highest index is the root node, and the lowest indexes correspond to the leaves.
- The structure that stores the hierarchical partition contains two important files

1. *a file of the image size that contains the **contours** of the image and the **weight** of these contours.*

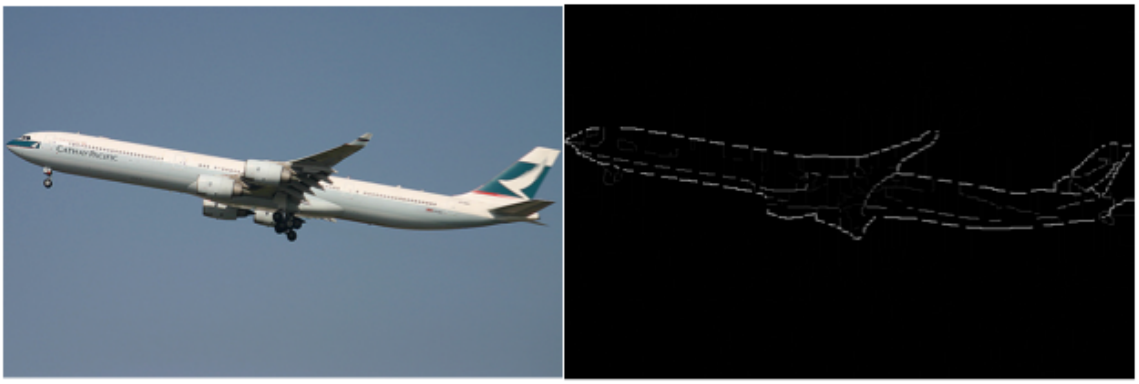


Figure 9: The picture 2008_003876 of the PASCAL dataset of a plane and its corresponding ucm, which shows the boundaries of the partition with different weights.

2. *a **merging sequence**, i.e. the sequence of fusions between nodes that defines the hierarchical partition.*

	1	2	3
1	20	12	77
2	45	49	78
3	26	70	79
4	62	54	80
5	46	78	81
6	57	67	82
7	33	37	83
8	28	21	84
9	19	65	85
10	30	25	86
11	14	68	87
12	44	53	88
13	18	16	89
14	81	82	90
15	40	88	91

Figure 10: Merging sequence of the first fusions of 2008_003876 picture. For each row the two first columns are the indexes of the nodes that merge, and the last column is the index of the node generated.

- The UCM hierarchical partition can be represented with a **dendrogram** that not only represents the fusions between regions, but also the **costs** of each fusion, which correspond to the **height of the node in the dendrogram**. Notice that the regions are created depending on their fusion cost value, so that the costs of the created regions always **increase**. The range of the costs' values is of [0,1]. For instance, a dendrogram of the UCM hierarchical partition of an image of PASCAL data set is depicted in Figure 11 from below. For an easier visualization leaves costs are zero, but as our tree is **pruned** as it is explained below, the actual **leaves costs are not zero**.

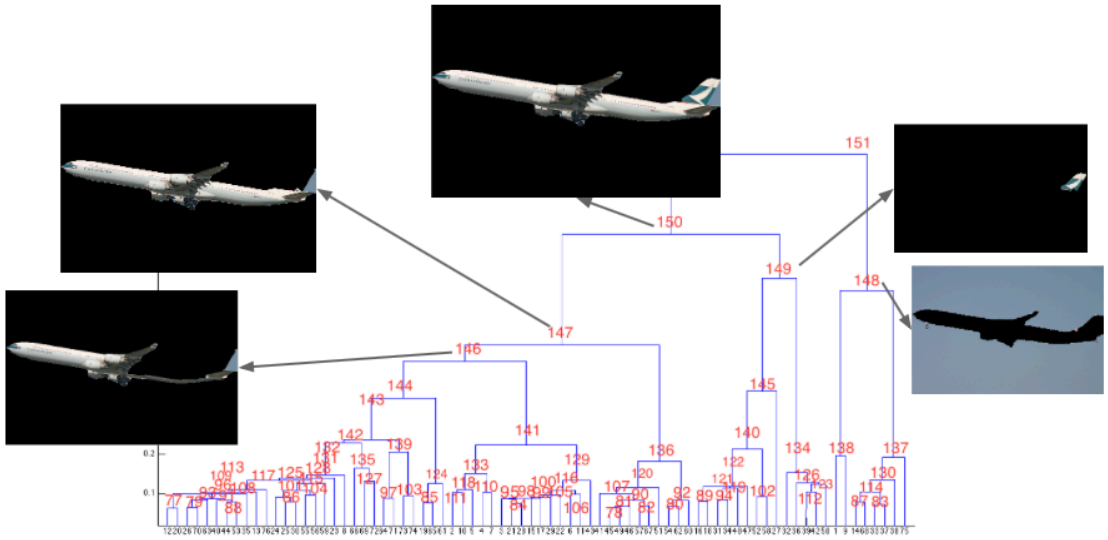


Figure 11: Dendrogram of 2008_003876 picture, whose y axis represents the costs, and the x axis the indexes of the leaves. Also masks of the regions associated to different nodes are shown.

Now **some specific traits of the UCM** hierarchical partitions used for this research should be highlighted in order to later better understand the configurations:

- First a **fully-grown** tree, i.e. the complete tree that UCM algorithm creates, is computed with an approximated number of 1500 nodes.
- Afterwards the tree is **pruned**, i.e. substitutes some subtrees for leaves depending on the number of nodes that we want to use. In particular we have selected a tree of **151 nodes (75 leaves)** in order to compare results to 150 CPMC object candidates. Notice that a tree of 150 nodes is impossible to generate because the total amount of nodes must be an odd number (2*(number of leaves) -1).

Different configurations of UCM refer to different strategies of picking the different regions of the partition in order to later segment the image. The criterions of the order of the different configurations considered in this research have been the following.

3.3.2.1. Class-agnostic exploration

Hierarchical partitions are generated by **iteratively clustering** regions of pixels with similar characteristics, starting from a fine partition. The merging sequence is then determined by the similarity between regions. If two adjacent regions are really dissimilar, it is said that a **strong boundary** separates them, and that the **cost** associated to such pair of regions is high. This information is stored in a file as stated in 3.3.2.

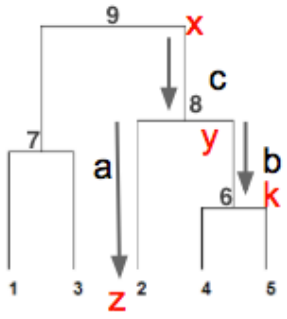
The first approach to sort the regions of the hierarchical partition is based on the assumption that an object is **homogeneous** in comparison to the background of the image. For this reason it makes sense to think that **fusions that add content** to an homogeneous object may have **similar costs**, whereas the fusion cost of the **merging between the object and the background** will be **higher** because they are more dissimilar.

Configurations based on indexes: The indexes of the nodes of a hierarchical partition determine when each region has been **created**. The first regions created are associated to the lowest fusion costs and to the lowest indexes. Consecutive indexes are associated

to regions that share similar costs. If the descendants of a node differ in the number of index mean that though they are adjacent, they have been created in different moments of the hierarchical partition generation, so that their costs are also different. If they are adjacent but their costs are different it means that probably they are dissimilar regions, because their creation was characterized by different boundaries costs, so the regions can be associated to **different objects**. This argumentation has lead to think that **objects can be found in regions associated to indexes that differ from the indexes of their adjacent regions**.

In order to find these regions, the basic idea followed in the different configurations proposed by this thesis, is a top-down approach that **descends through the nodes related to higher indexes**, which in general are the ones more similar to their ancestors, but consider their sibling as object candidate because they are the regions associated to more **distinct indexes**. The different configurations derived from this idea are further explained in the **Appendix I**.

Configurations based on costs: A second research line addressed in this thesis is the exploration of the UCM based on the costs of the merges instead of the index of the merging sequence. The indexes give an insight of which regions are associated to higher costs compared to other regions, but not of the costs values themselves. We can use the costs values to further explore the same intuitions that guided the study based on indexes. First of all a clear notation should be established. The dendrogram in Figure 12 introduces the terminology used:



Consider this tree of 9 nodes. We are going to analyse the node with index number 8:

x : cost of the parent node
y : cost of the node
z,k : costs of the descendants
a :y-z difference of fusion cost between node and one of its descendants
b :y-k difference of fusion cost between node and one of its descendants
c :x-y difference of fusion cost between father and node to analyse

Figure 12: Nomenclature of the different costs and their relations depicted in a dendrogram.

A major part of the configurations calculate derivatives of the costs related to a node. If node 8 in the previous picture is the one we want to analyse, it is interesting to know the differences between its father and descendants costs. A first approach is the following set of **first derivatives**.

$$c = x - y$$

$$a = y - z$$

$$b = y - k$$

Notice that we are not normalizing by any value these derivatives, but we could normalize either by the first operand or the second one. If it is considered that normalization is not required is because it is meaningful for us the distance of a single fusion, so we would normalize all these derivatives by a unit.

These first derivatives give insight of the differences of costs between consecutive regions in the partition. Another interesting approach is the **second derivative**, which takes into consideration all three levels of the node in the hierarchy. By computing the

second derivative an insight of how the differences between costs change can be apprehended. More possibilities could be contemplated, but for this research the following second derivatives have been considered:

$$c - min(a, b)$$

$$c - max(a, b)$$

In the Appendix I there is a table with all the configurations considered, not only the ones that consider the derivatives criterion, but also a set of configurations that mixes configurations based on indexes with configurations based on costs.

All configurations **sort the regions depending on the values obtained with these derivatives or other functions**. In concrete, in the appendix all the configurations are defined by the function that is being maximized.

3.3.2.2. Class-dependent exploration

This last approach differs from the two previous because it is not class-agnostic. The criterion to select which regions to analyse first considers the outputs of the **SVR** for each object class category, i.e. the **predicted overlap** with every object of the different categories.

As the scope of this work is developing an efficient strategy, it is interesting that the decision of the classifier for a region gives us information of **what region to analyse next**, so that a sequence of **smartly selected regions** is defined.

Xavier Giró developed an efficient strategy to explore a hierarchical partition during his PhD thesis using **Bag of Words Descriptors [1].** The strategy consisted of a top-down approach that uses a detector and a classifier in order to recognize objects in images. The detector detects which objects are **contained** in every region, so that en efficient exploration can be performed. At the same time a classifier on the region analysed determines whether the region itself **is the object** we are seeking or not.

In the framework of the problem that we are dealing with in this work, we are trying to recognize the 20 object classes from Pascal. A first approach would be using the **SVR outputs** for each region to decide how to explore the tree. The proposed top-down exploration algorithm is the following:

1. The algorithm starts always at the **root node**. It is not necessary to know the confidence vector for this node, because we are always going to start with it.

2. Consider its **two descendants** regions, and obtain the **SVR's** outputs for both. For any of the classes we are considering, the **one that has a higher confidence value is the one we are considering in order to descend**. We **store its sibling into a queue**, because we are going to analyse it later.

3. Now we store into the queue the two descendants of the node analysed in the previous step.

4. We check which element of the queue has a **higher confidence value** for any of the classes. The algorithm is going to descend through this node if the class that scores the maximum punctuation is a **different** one than the previous class considered, and in case the previous class considered is the same there are two options:

- The node analysed previously **was its ancestor**. In this case we check the score, and if it is lower than in the previous iteration, the algorithm **does not**

descend through this node in order to avoid exploring a branch when the object at issue **has already been found** on that branch. The algorithm checks the nodes in the queue until a **proper node** is found. If no proper node is found, the one with higher overlap is analysed.

- The node analysed previously **was not its ancestor**. In this case the algorithm **descends** through that node.

5. We **keep repeating the 3 and 4 steps** until all nodes have been analysed.

The following Figure 13 depicts the flow chart of the algorithm previously explained.

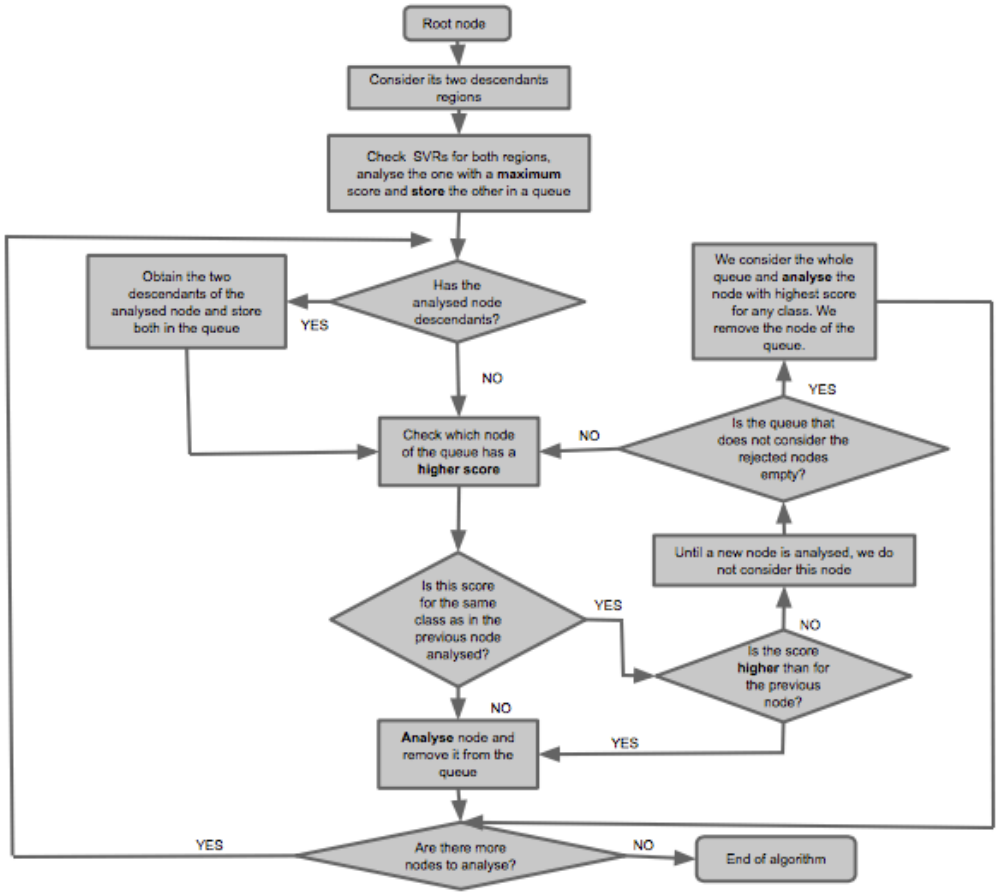


Figure 13: Flow chart of the class-dependent exploration algorithm.

4. Results

In this section the best results obtained from all the configurations tested are illustrated. All the plots depict **the AAC obtained when using a budget of a certain number of regions**. At the **Appendix I** there is an explanation and also different plots for each configuration tested using the different criterions.

4.1. Results with O2P descriptors

4.1.1. Comparison of naive UCM and CPMC

In Figure 14 and Table 1 we are going to analyse the results obtained with **O2P descriptors** when **no smart exploration** is performed.

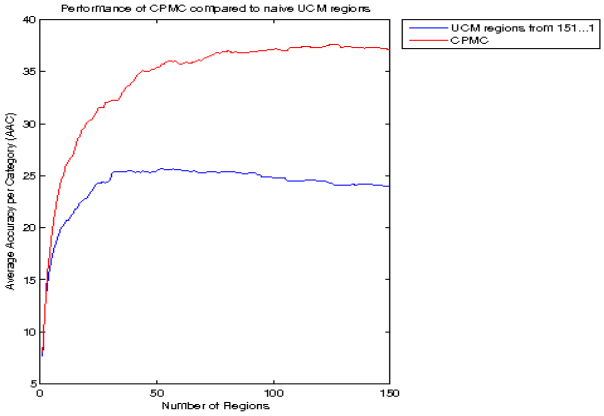


Figure 14: Plot of Accuracy vs. number of regions with CPMC and naive UCM, whose regions are sorted according their indexes, starting by the higher index and descending until the first region.

AAC	5 regions	15 regions	Maximum
CPMC	19.4995	27.6783	35.5921
UCM	17.1496	21.8057	25.6645

Table 1: Comparison of AAC between CPMC and UCM for different regions budget.

The first interesting fact perceived from Figure 14 from above is the difference between the trends of the UCM and CPMC curves. The CPMC curve is **mainly increasing** whereas the UCM curve **decreases after around 25 regions**. This is plausible because of the intrinsic properties of the regions generated by a hierarchical partition. The different regions linked to the nodes of a hierarchical partition belong to **different levels of the partition associated to different regions sizes**. As the budget of regions available increases the **SVRs output better scores for small regions although they are not the actual object** of the image, so that an **over segmentation** can affect the classifier decisions leading to a decreasing accuracy in the final segmentation. This fact is not present with CPMC regions because they do not present the hierarchical properties.

The configuration of UCM plotted in the Figure 14 corresponds to a **naive order of UCM**. It picks the UCM regions according to their indexes in a descending manner. A first conclusion about this curve is that the regions of UCM that really make a difference in terms of accuracy are the ones associated to **higher indexes**, i.e. the ones that belong to the **highest levels** of the hierarchy. This is highly related to the regions size, since those

nodes are linked to the biggest regions, and it is reasonable to think that **if the budget of regions is very small** it is advisable to select the regions associated to the highest nodes. Notice that in the CPMC configuration the first zones are also the ones that increase more dramatically the accuracy, but this is because the regions are sorted depending on the likelihood that the regions are objects in the image.

As the scope of this project is to develop an efficient strategy to segment images, we are going to concentrate on the **lower amount** of number of regions. For the top 5 and top 15 zones, CPMC outperforms the results obtained with the naive UCM configuration as depicted in Table 1. It is also quite outstanding that **CPMC** maximum accuracy is much higher to the one obtained with **UCM**.

4.1.2. Results obtained with smart explorations of UCM

Many configurations of UCM regions have been tested based on the different criterion exposed in this thesis in order to guide an **efficient exploration through the tree partition**. The best results achieved with O2P descriptors are depicted in Figure 15 and Table 2.

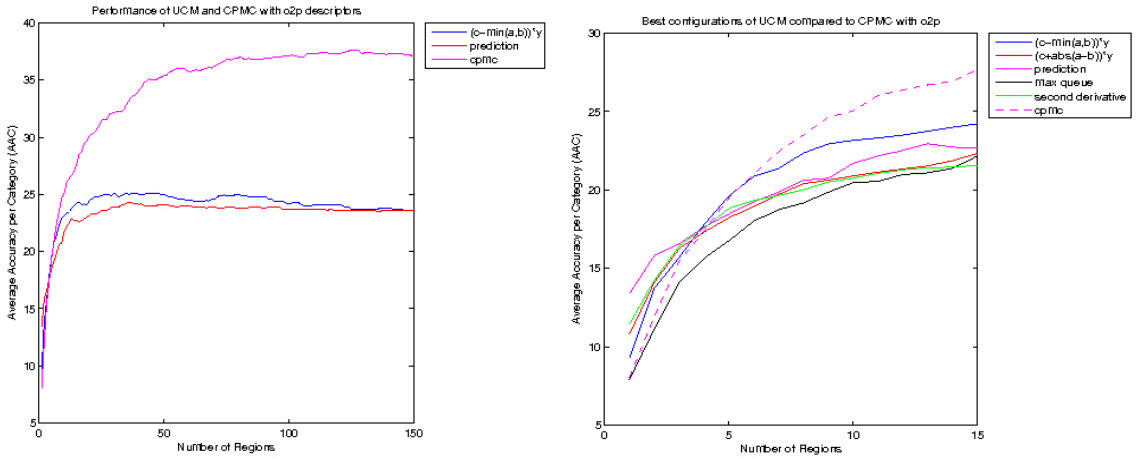


Figure 15: The plot of the left shows the two best configurations of UCM compared to CPMC in a 150 regions sweep. The plot of the right shows a sweep of 15 regions of the set of best configurations of UCM compared to CPMC.

Configuration	3 regions	5 regions	15 regions	Maximum	Method
Max queue	14.1210	16.7512	22.1611	26.4423	UCM index
$(c - min\ (a, b)) * y$	**15.6977**	**19.6139**	**24.2172**	25.1349	UCM cost
$(c\ +\ \|a - b\|) * y$	16.2680	18.2270	22.3265	25.4995	UCM cost
Second derivative according nodes	16.4387	18.8192	21.5547	25.3733	UCM index and cost
SVR	**16.5685**	18.4662	22.6758	24.2867	Top-down prediction
CPMC	**15.4528**	**19.4995**	**27.6783**	37.5921	Object Candidates

Table 2: Comparison of AAC between different configurations of UCM and CPMC for different regions budget using O2P descriptors.

4.1.2.1. Results of configurations based on UCM indexes

One of the best results is obtained by the configuration **max queue** that only considers the **indexes of the merging sequence** (**(-)** in Figure 15). This configuration is a **top-down approach** that follows the argumentation previously in this report explained, that objects in a hierarchical partition might **be found in regions associated to indexes that differ from the indexes of their adjacent regions**. The exploration guided through the tree starts at the root node and descends through the nodes with highest indexes, that are the ones that have been created later and are consequently more alike to their ancestors regions. However, we analyse their siblings, because they **have been created earlier and are consequently more different compared to their ancestors**. Once a leave is reached, the next node to descend through is the one through which the algorithm has not descended yet and is associated to a higher index, so it has been created later, and analyse its sibling who has been created earlier compared to its ancestor. Regions of the **higher levels** of the hierarchy and **more different to their adjacent regions** are the first selected.

4.1.2.2. Results of configurations based on UCM costs

The best configuration of UCM is the one that maximizes a **second derivative** that captures how the differences of costs change considering the three levels of the hierarchy related to a node (**(-)** in Figure 15). The three levels are the node's ancestor, the node itself and its descendants. If we represent a node by the index *i,* the configuration seeks those regions that maximize the following:

$$max_i\ ((c_i - min\ (a_i, b_i)) * y_i) = \ (max_i(c_i) - max_i(min\ (a_i, b_i))) * max_i(y_i) =$$

$$= (max_i(c_i)\ +\ min_i(min\ (a_i, b_i))) * max_i(y_i)\quad \forall_i$$

This second derivative demonstrates that best results are achieved by regions whose cost is similar to the cost of one of its descendants but really dissimilar to its ancestor's cost. This supports the theory that **an object can be found in nodes whose costs compared to their descendants are similar** because an **homogeneous region** was being created, but whose **father's cost is really dissimilar** because it is **associated to the fusion with the background of the object**. This has lead to think about configurations that **maximize the difference between the two descendants' costs**, since configurations like $max_i\ (max\ (a_i, b_i))$ and its variants (can be seen on the appendix) yield also good results. Maybe implementing $max_i\ (max\ (a_i, b_i))$ and $min_i(min\ (a_i, b_i))$ is the combination that we are seeking. Following this idea $max_i\ (c_i\ + + |a_i - b_i|) * y_i$ has been tested with very good results (**(-)** in Figure 15 on the right), though $max_i\ (c_i - min\ (a_i, b_i)) * y_i$ still is the best.

4.1.2.3. Results of configurations based on UCM indexes and costs

Other configurations have resulted quite successful, such as the one explained in the appendix that calculates the **second derivative of the costs with respect to the indexing of the nodes** (**(-)** in Figure 15). The indexes are related to the regions creation, and the regions creation depends on the costs values, so that the regions costs with respect to their indexes are always increasing. However, the costs values increments can change, and this second derivative detects those nodes **whose costs are much different in comparison to those nodes created in similar moments during the hierarchical partition generation**. This criterion performs well because regions, mainly

of the nodes associated to higher indexes, are created consecutively due that there remain only a few last regions to merge, so the same argument followed in $max_i\ (c_i - min\ (a_i, b_i))$ (**(-)** in Figure 15) is valid for this configuration.

4.1.2.4. Results of top-down prediction configurations

Another interesting criterion that obtains very good results especially when using a few regions, is the one based on the **per-class prediction** (**(-)** in Figure 15 on the right and **(-)** on the left). The criterion follows a **top-down approach** and descends through those nodes that score a higher overlap prediction with any of the 20 PASCAL classes.

The main issue here is that the **SVR's are trained to detect whether a region belongs to a class**, but not to detect whether that region **contains a certain object**, so it is already quite surprising that the configuration works well, and gives us a clue that following this research line and using a detector in order to determine which objects are contained in a region could be really promising in order to guide an efficient exploration of the partition.

4.1.2.5. Final comparison of CPMC and UCM using O2P descriptors

Finally it is quite outstanding that **CPMC configuration outperforms all UCM configurations from 5 regions and above**. CPMC object candidates are created from the merging and filtering of regions that the parametric min-cuts produce, resulting in a set of **accurate segments**. UCM segments conversely are the **straight regions obtained from the hierarchical partitions**, which are not created with the purpose of becoming object candidates but regions that share similar characteristics in the image.

4.2. Results with SDS descriptors

The best configurations obtained with UCM and O2P descriptors have also been tested using **SDS deep learning descriptors [6]**. The results are depicted in Figure 16 and **Table 3**.

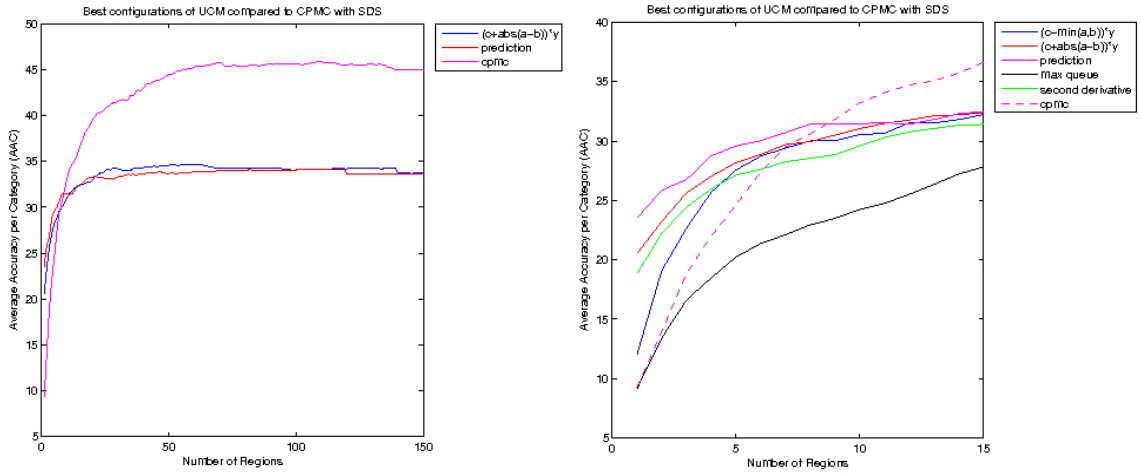


Figure 16: The plot of the left shows the two best configurations of UCM with SDS compared to CPMC with SDS for a sweep of 150 regions. The plot of the right shows a set of the best configurations using SDS for a 15 regions sweep.

Configuration	3 regions	5 regions	15 regions	Maximum	Method
Max queue	16.6154	20.2263	27.8129	35.6149	UCM index
$(c - min\ (a, b)) * y$	22.6603	27.5581	32.2131	34.3101	UCM cost
$(c\ +\ \|a - b\|) * y$	**25.6244**	**28.1885**	32.3963	34.6870	UCM cost
Second derivative according nodes creation	24.4298	27.1347	31.4007	34.4783	UCM index and cost
Prediction	**26.7601**	**29.5846**	32.4771	34.2708	Top-down prediction
CPMC	18.8319	24.6160	**36.6310**	**45.8890**	Object Candidates

Table 3: Comparison of AAC between different configurations of UCM and CPMC for different regions budget using SDS descriptors.

SDS descriptors **perform much better** than O2P, all configurations yield better results using these descriptors. The best results obtained with these descriptors are discussed in the following sections.

4.2.1. Results of top-down prediction configurations

The results reflect that the best configuration for an efficient exploration with SDS descriptors is the one that considers the **per-class prediction** (**(-)** in Figure 16 on the right and **(-)** on the left), which drastically **outperforms CPMC for a few regions budget**. This supports even more the improvement of SDS descriptors compared to O2P, due that using the classifiers decisions to explore the tree partition results quite efficient compared to other techniques only based on costs of the tree partition. Another proof is that the **curves of UCM** using SDS **do not decrease once they saturate**, because the classifier does not make wrong decisions even when small regions associated to low levels of the hierarchy are considered.

4.2.2. Results of configurations based on UCM costs

The second best configuration in this case is $max_i\ (c_i\ +\ |a_i - b_i|) * y_i$, (**(-)** in Figure 16 on the right and **(-)** on the left) which reinforces the intuition grasped while experimenting with O2P, that regions that best fit objects in the image could be the ones whose cost is different from their ancestors' costs, and whose descendants' costs are dissimilar between each other, being one similar to the node's cost and the other pretty dissimilar. An explanation of why the nodes whose descendants have these characteristics yield good results could be that **real objects are not homogeneous, they posses parts that are homogeneous but probably parts that are not**. The fusion of two nodes, one being the most homogeneous part of the object with a node that is the less homogeneous part of the object, would have a cost of these characteristics, due that the first region would have a similar cost to the node's cost, and the second region, the one that is more dissimilar, would have a much different cost.

4.3. Visualization of regions selected

Figure 17 and Figure 18 below allow a better visualization of the object proposals selected for each UCM configuration. Notice that the confidence values in order to descend through the partition for the *prediction configuration* are the ones obtained with **SDS** descriptors.

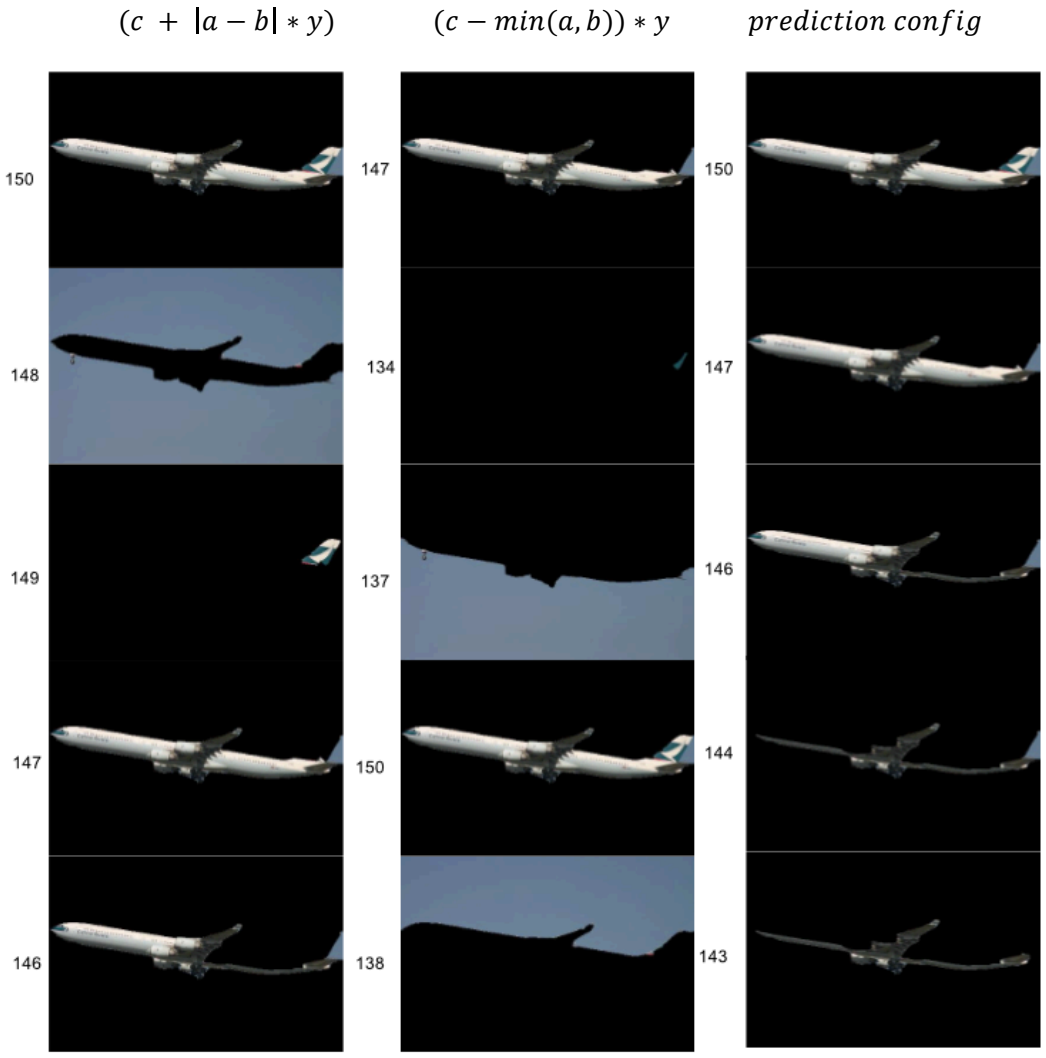


Figure 17: Regions and their associated indexes selected when using different configurations of UCM for picture 2008_003876.

This example in Figure 17 corresponds to an image whose contours match **nearly complete objects**, as in this case the plane of the picture. Regions 150 or 147 are really good object candidates, and the three configurations find one of these regions. In fact the three configurations select in the first place a region that is really **appropriate**.

On the other hand, partitions of other images such as the one for 2007_000042 are not that ideal, as depicted in Figure 18.

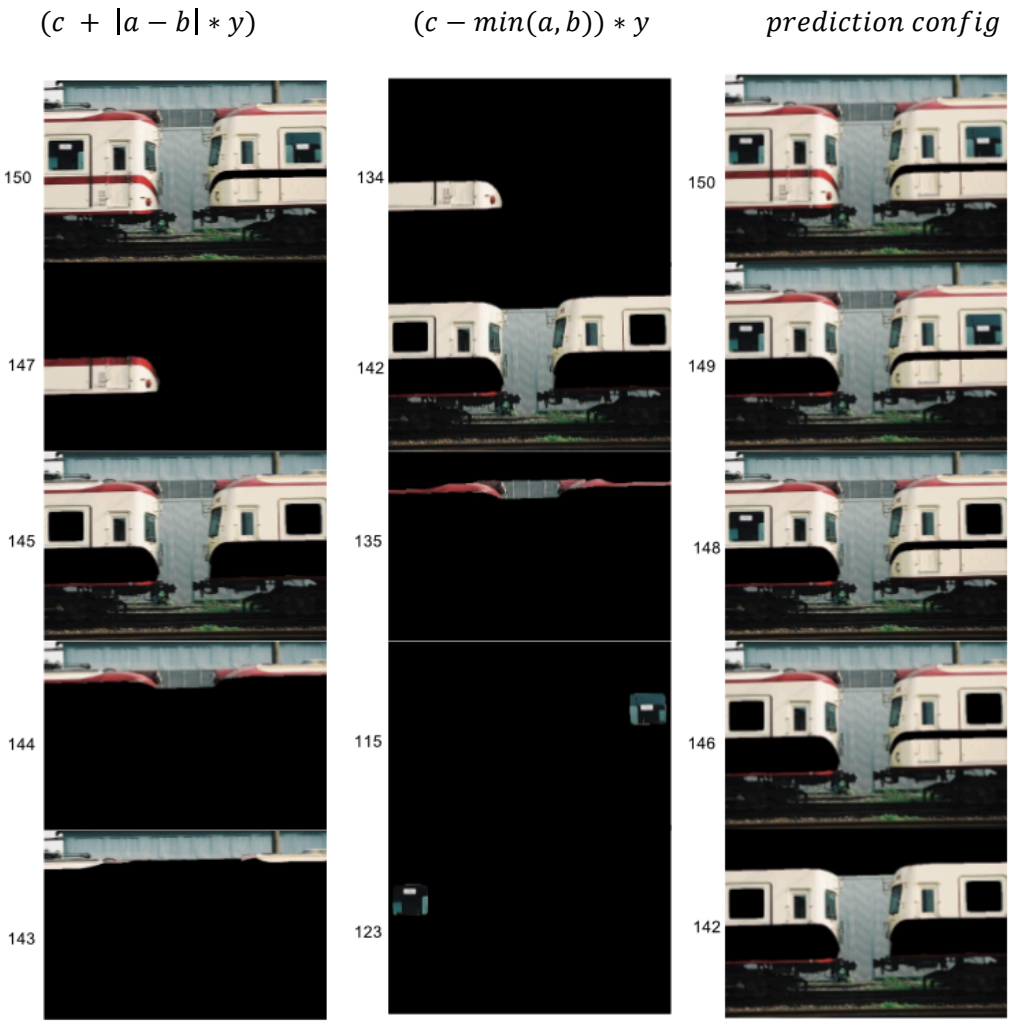


Figure 18: Regions and their associated indexes selected when using different configurations of UCM for picture 2007_000042.

The whole **dendrogram** and **ucm** of image 2007_000042 are in the **Appendix II**. In this case a **strong contour** separates into two the main objects in the image, the trains, and for this reason the object candidates generated are not very useful. **None of these object candidates would yield good results** because they all include lots of pixels that belong to the **background**, or some **part of the trains is missing**. Maybe the best regions would be 142 and 145, and the three configurations find at least one of such candidates.

4.4. Comparison of computation time

Finally, a comparison in terms of **time of computation** is important as the scope of the project is finding an **efficient strategy**. The times of computation stated in the article **[5]** and **[13]** for CPMC and UCM hierarchical partitions are illustrated in Table 4.

	CPMC 150 regions	**UCM hierarchical partition**
Computation Time	250 s/ image	24.4 ± 3.6 / image

Table 4: Computation time of the set of object candidates of CPMC and the hierarchical partition.

So computing the **CPMC** regions **is much slower** compared to the creation of a **UCM** hierarchical partition. Moreover, if we reduce the computation time of the prediction by using just a few regions budget, the total computation time is reduced. An approximate estimation based on the experimentation performed during this project is in Table 5.

Computation Time	**Prediction with 5 regions**	**Prediction with 15 regions**	**Prediction with 150 regions**
Set of 1111 images	1 min 50 s	2 min 30 s	3 min 30 s
Image	≈0.001647 s / image	≈0.00225 s/ image	≈0.00315 s/ image

Table 5: Computation time of the prediction of the final segmentation.

The second row is just an approximation, because fixed times independent of the number of images are not known.

5. Budget

The budget of the project consists of the wages of a junior engineer and two senior engineers, as stated below.

	Number	Wage	Hours / week	Total weeks	Total
Junior Engineer	1	8 € / hour	30 h	20	4800 €
Senior Engineer	2	20 € / hour	10 h	20	8000 €

Table 6: Total personal costs.

The project has been performed with Matlab. Its license cost and amortization are stated below:

Number of Licenses	Price / Year	Months of project	Amortization/ Month	Total Amortization	Total
1	500 €	5 months	41.67 € / month	208.33 €	291.67 €

Table 7: Total licenses costs

So the total cost of the project is of **13.091,67 €.**

6. Conclusions and future development:

The initial aim of this thesis was finding an efficient strategy to semantically segment images using hierarchical partitions, trying to achieve the maximum accuracy possible when only disposing of a few regions budget.

Although **UCM segments are not that accurate in comparison with CPMC** object candidates, the intrinsic **multiscale information** of the hierarchical partitions provides a wide range of possibilities to **efficiently explore the partition** and detect those nodes that more probably are objects in the image at issue.

Results have shown that for the PASCAL data set the nodes of the best regions are associated **to the nodes of the higher levels of the hierarchy**, due their size. Moreover, **adjacent regions that have been created in different moments** of the hierarchical partition generation, and therefore associated to **dissimilar indexes**, yield better results. Following the same idea, regions associated to **similar costs compared to one of their descendants** but very **dissimilar costs compared to their ancestors**, prove good performance.

These findings support the idea that a region that is a good candidate for being an object is generated with a **series of fusions of regions that have similar costs**, since if we are assuming an object is homogeneous, or at least more homogeneous than comparing it with the background of the image, the costs of adding homogeneous parts to an object should be similar because the boundaries that define the fusions have similar weights. Furthermore, a region is potentially a good object candidate if **the merging with an adjacent region is characterized by a strong boundary**, and therefore both regions share a high cost fusion, because of the lack of homogeneity between the object and its background.

An attribute of these regions that has not been explicitly considered and could be interesting for further researching is the **diversity** between regions. Two regions may be good object candidates but if they are redundant with each other and we only dispose of a few regions budget, it would be more desirable to **discard duplicative object proposals**.

A configuration that has obtained indeed promising results, is the one that descends through the tree depending on the **estimated overlap of the regions with the objects pursued**. Though the SVRs have been trained to predict whether a region **is** an object and not to predict the objects **it contains,** the results have been good enough to think that applying a **detector** of the objects contained in an image in order to discard branches of a tree, and also implementing a **classifier** to detect whether an object belongs to a certain class category, is the **next step to follow** this research.

Another improvement would be using **MCG object candidates**, which are generated from the regions of the UCM partition. The computation time of such candidates is minimal compared to CPMC, and the hierarchical partition of the UCM could allow the efficient exploration addressed in this research. Also, the candidates are much more accurate than the straight regions obtained directly from the UCM, due that the object proposals are generated from the combination of different regions of the partition.

Bibliography:

[1] Giró i Nieto, X. (2012). Part-based object retrieval with binary partition trees.

[2] Salembier, P., & Garrido, L. (2000). Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *Image Processing, IEEE Transactions on*, *9*(4), 561-576.

[3] Arbelaez, P. (2006, June). Boundary extraction in natural images using ultrametric contour maps. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on* (pp. 182-182). IEEE.

[4] Carreira, J., Caseiro, R., Batista, J., & Sminchisescu, C. (2012). Semantic segmentation with second-order pooling. In *Computer Vision–ECCV 2012* (pp. 430-443). Springer Berlin Heidelberg.

[5] Carreira, J., & Sminchisescu, C. (2012). Cpmc: Automatic object segmentation using constrained parametric min-cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *34*(7), 1312-1328.

[6] Hariharan, B., Arbeláez, P., Girshick, R., & Malik, J. (2014). Simultaneous detection and segmentation. In *Computer Vision–ECCV 2014* (pp. 297-312). Springer International Publishing.

[7] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (Vol. 1, pp. I-511). IEEE.

[8] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[9] Itti, L., Koch, C., & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (11), 1254-1259.

[10] Van de Sande, K. E., Uijlings, J. R., Gevers, T., & Smeulders, A. W. (2011, November). Segmentation as selective search for object recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (pp. 1879-1886). IEEE.

[11] Arbelaez, P., Pont-Tuset, J., Barron, J., Marques, F., & Malik, J. (2014, June). Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on* (pp. 328-335). IEEE.

[12] Lindeberg, T. (2012). Scale invariant feature transform. *Scholarpedia*, *7*(5), 10491

[13] Carreira, J., Caseiro, R., Batista, J., & Sminchisescu, C. (2015). Free-form region description with second-order pooling..

[14] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, *88*(2), 303-338.

[15] Carreira, J., Li, F., & Sminchisescu, C. (2012). Object recognition by sequential figure-ground ranking. *International journal of computer vision*, *98*(3), 243-262.

[16] Lavoué, G. (2012). Combination of bag-of-words descriptors for robust partial shape retrieval. *The Visual Computer*, *28*(9), 931-942.

[17] Fontdevila-Bosh, E. (2015). Region-oriented Convolutional Networks for Object Retrieval.

Appendix I: Different configurations of Average Accuracy per Category vs. number of regions

This appendix contains the explanations and plots of the different configurations tested. For each configuration an explanation, a code, and different plots are exposed. By default all configurations use O2P descriptors.

Configurations CPMC

Default	The first object candidate is the one that more likely is an object of the image, up to 150 object candidates.

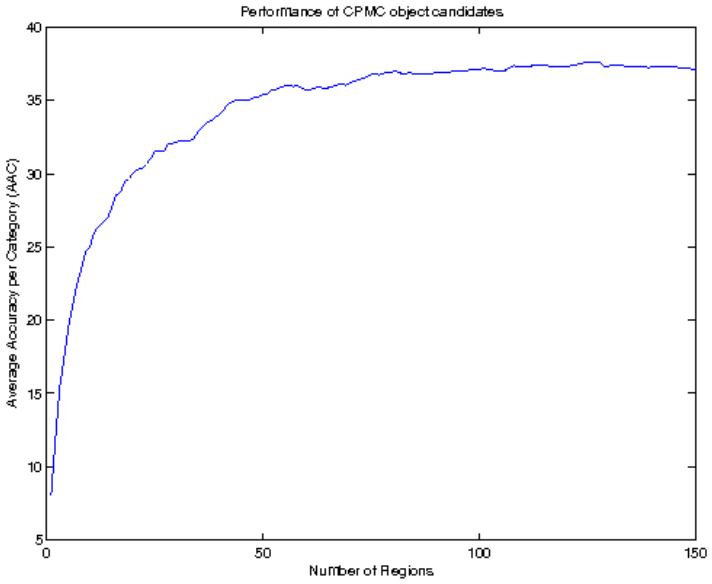


Figure 19: Plot of CPMC performance for 150 regions.

Configurations UCM

0. *Baseline*

Random	Random permutation of the 151 regions

1. *Based on indexes*

Name	Explanation	Code
Increasing indexes	The masks from 1 to 151	0
Decreasing indexes	The masks from 151 to 1	1
FIFO	Starting from the root node, descending through the highest node and analysing its sibling. At the same time	2

	storing in a FIFO queue the node analysed. When a leave is reached, pull first element in the queue and descend through it. ***Notes**: Top-down approach*	
LIFO	Starting from the root node, descending through the highest node and analysing its sibling. At the same time storing in a LIFO queue the node analysed. When a leave is reached, pull last element in the queue and descend through it. ***Notes:** Top-down approach*	3
MAX queue	Starting from the root node, descending through the highest node and analysing its sibling. At the same time storing in a queue the node analysed. When a leave is reached, pull maximum element in the queue and descend through it. ***Notes:** Top-down approach*	4

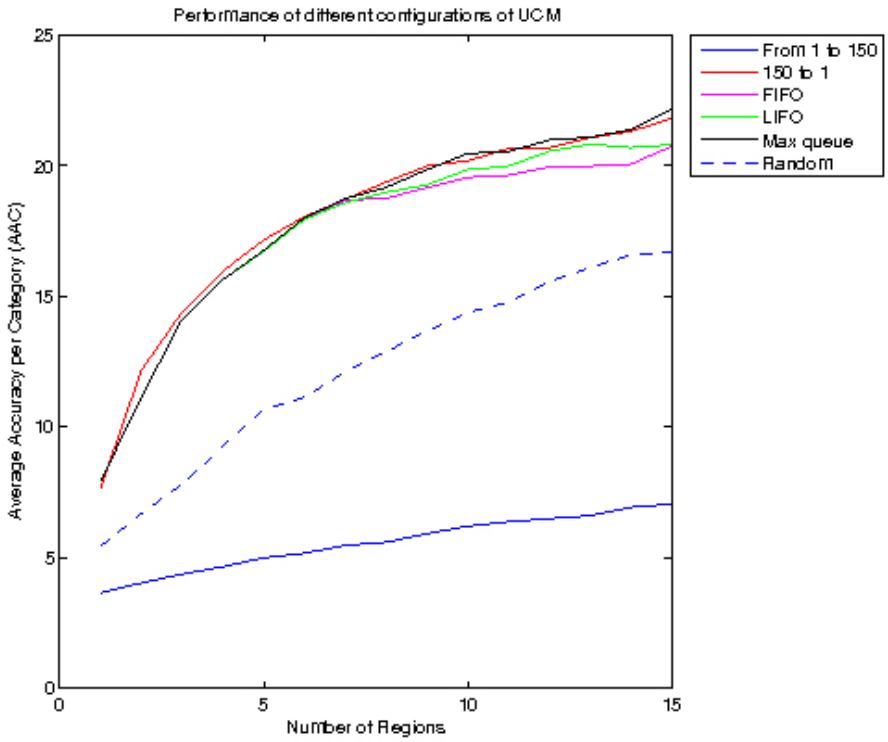


Figure 20: Configurations from 0 to 4, and random configuration. Plot with 15 regions.

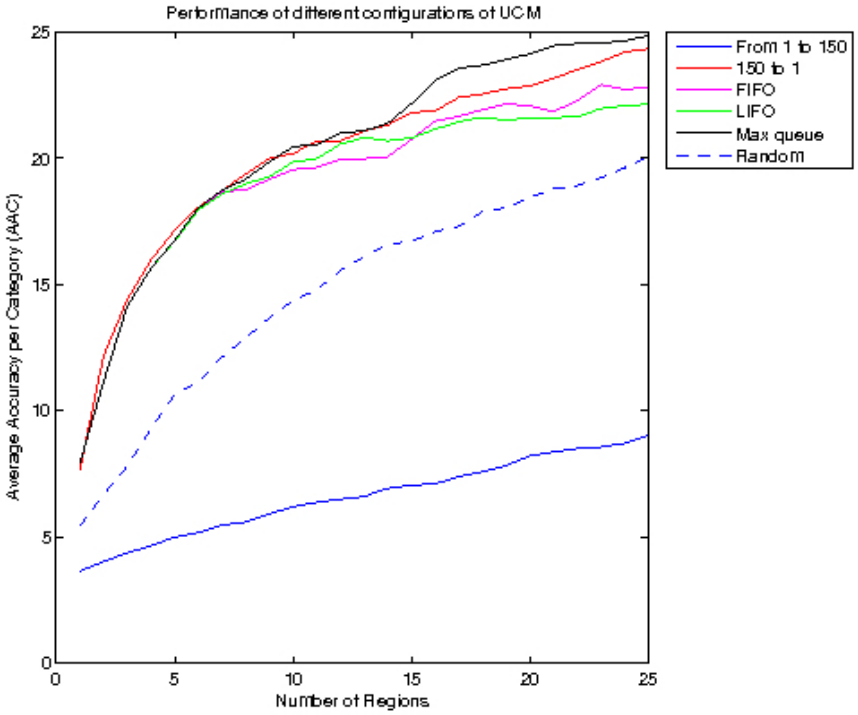


Figure 21: Configurations from 0 to 4, and random configuration. Plot with 25 regions.

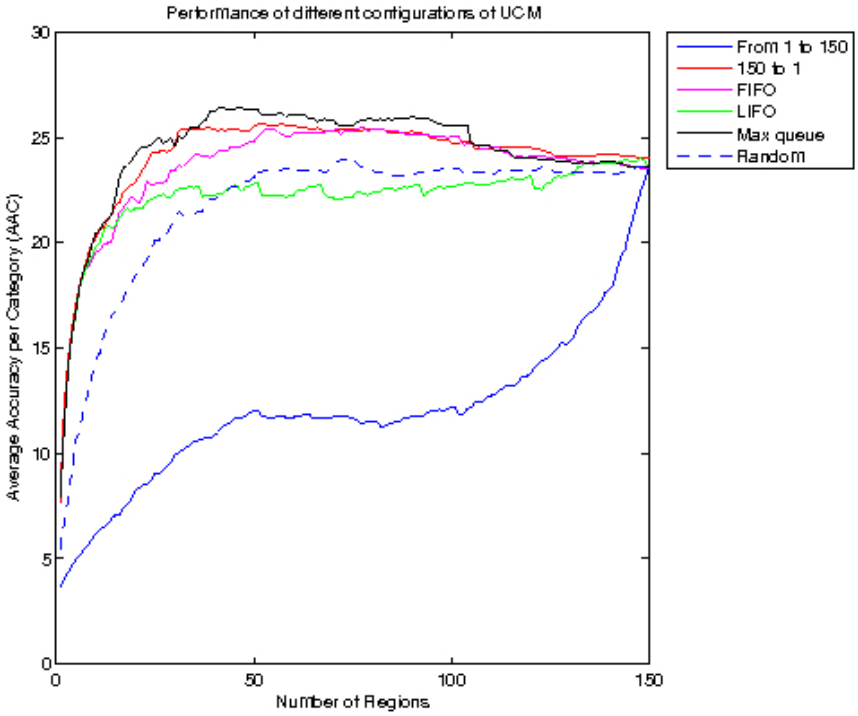


Figure 22: Configurations from 0 to 4, and random configuration. Plot with 150 regions.

2. *Based on Costs*

Name	**Explanation**	**Code**
$max_i(c_i)$	Maximize c, i.e. the difference between the cost of the father and the cost of the node.	5
$max_i(c_i - (a_i + b_i))$	Maximizing the difference between the father' cost and the cost of the node, and minimizing the differences of the node and its siblings' costs.	6
$max_i\left(\frac{c_i}{a_i \times b_i}\right)$	Maximizing the difference between the father' cost and the cost of the node, and minimizing the differences of the node and its siblings' costs.	7
$max_i(max(a_i, b_i))$	Maximizing the maximum distance between the cost of the node and the cost one of its descendants.	8
$max_i\left(\frac{max(a_i, b_i)}{c_i}\right)$	Maximizing the maximum distance between the cost of the node and the cost of one of its descendants and minimizing the difference between the cost of the father and the cost of the node.	9
$max_i(max(a_i, b_i) \times c_i)$	Maximizing the maximum distance between the cost of the node and the cost of one of its descendants and maximizing the difference between the cost of the father and the cost of the node.	10
$max_i(min(a_i, b_i))$	Maximizing the minimum distance between the cost of the node and the cost of one of its descendants.	11
$max_i\left(\frac{x_i}{y_i}\right)$	Maximizing the ratio between the cost of the father and the cost of the node.	12
$max_i\left(\frac{x_i \times k_i \times z_i}{y_i^3}\right)$	Maximizing the ratio between the cost of the father and the cost of the node, multiplied by the ratio of the cost of a descendant' cost and the cost of the node, and multiplied by the same factor for the other sibling.	13
$max_i(c_i - min(a_i, b_i))$	Maximizing the difference between the cost of the father and the cost of the node and minimizing the minimum difference between	14

	the node and either of its siblings.	
$max_i\left(\frac{c_i - min(a_i, b_i)}{y_i}\right)$	Maximizing the difference between the cost of the father and the cost of the node and minimizing the minimum difference between the node and either of its siblings, divided by the node cost.	15
$max_i((c_i - min(a_i, b_i)) \times y_i)$	Maximizing the difference between the cost of the father and the cost of the node and minimizing the minimum difference between the node and either of its siblings, multiplied by the node cost.	16
$max_i\left(\left(\frac{c_i}{y_i} - min\left(\frac{a_i}{z_i}, \frac{b_i}{k_i}\right)\right)\right)$	Second derivative, as in previous configuration, normalizing by second operand of every first derivative.	17
$max_i\left(\left(\frac{c_i}{y_i} - min\left(\frac{a_i}{z_i}, \frac{b_i}{k_i}\right)\right) \times y_i\right)$	Second derivative, as in previous configuration, normalizing by second operand of every first derivative, multiplied by the cost of the node.	18
$max_i\left(min\left(\frac{a_i}{z_i}, \frac{b_i}{k_i}\right)\right)$	Maximizing the minimum first derivative normalized by the second operand.	19
$max_i\left(max\left(\frac{a_i}{z_i}, \frac{b_i}{k_i}\right)\right)$	Maximizing the maximum first derivative normalized by the second operand.	20
$max_i(max(a_i, b_i) \times y_i)$	Maximizing the maximum first derivative multiplied by the node cost.	21
$max_i(c_i + \lvert a_i - b_i \rvert)$	Maximizing the sum of the first derivative with the father and the absolute difference of the first derivatives with the siblings.	22
$max_i((c_i + \lvert a_i - b_i \rvert) \times y_i)$	Maximizing the sum of the first derivative with the father and the absolute difference of the first derivatives with the siblings, multiplied by the cost of the node.	23
$max_i((\lvert a_i - b_i \rvert) \times y_i)$	Maximizing the absolute difference of the first derivatives with the siblings, multiplied by the cost of the node	24

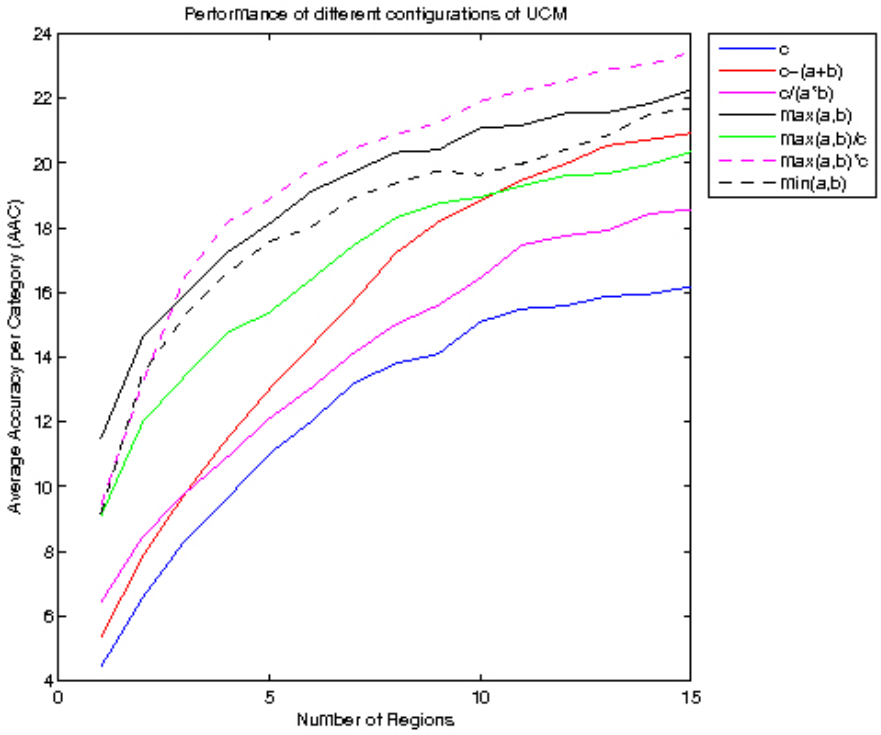


Figure 23: Configurations from 5 to 11. Plot of 15 regions.

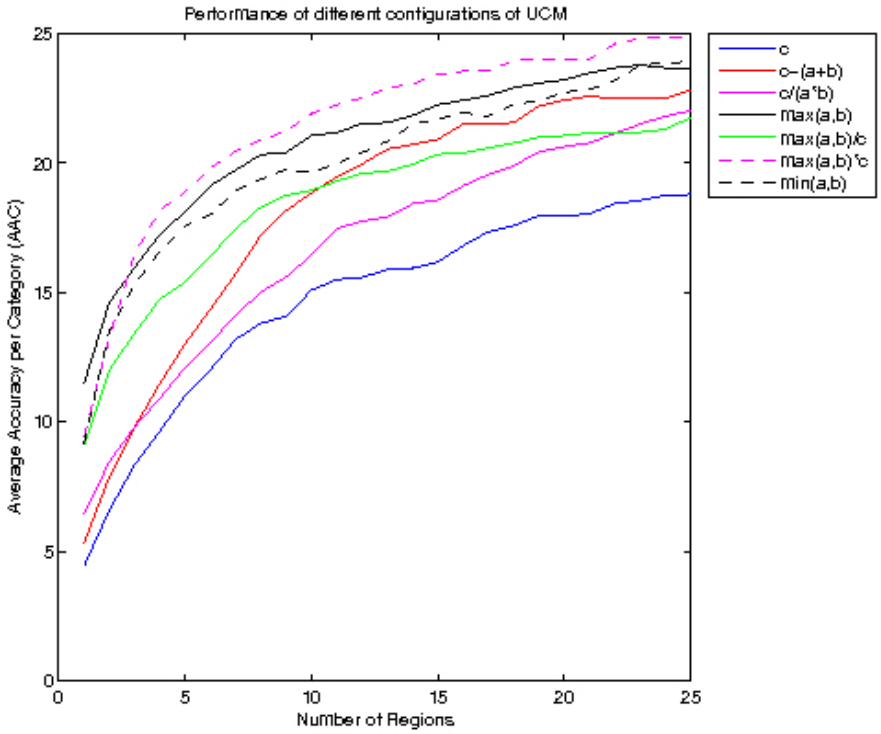


Figure 24: Configurations from 5 to 11. Plot of 25 regions.

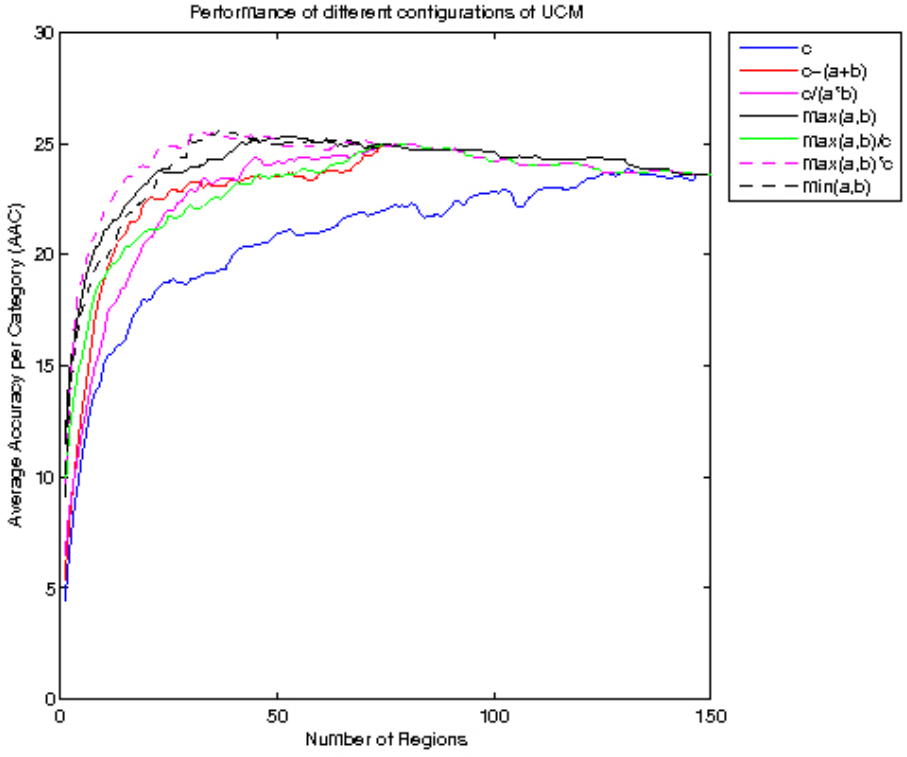


Figure 25: Configurations from 5 to 11. Plot of 150 regions.

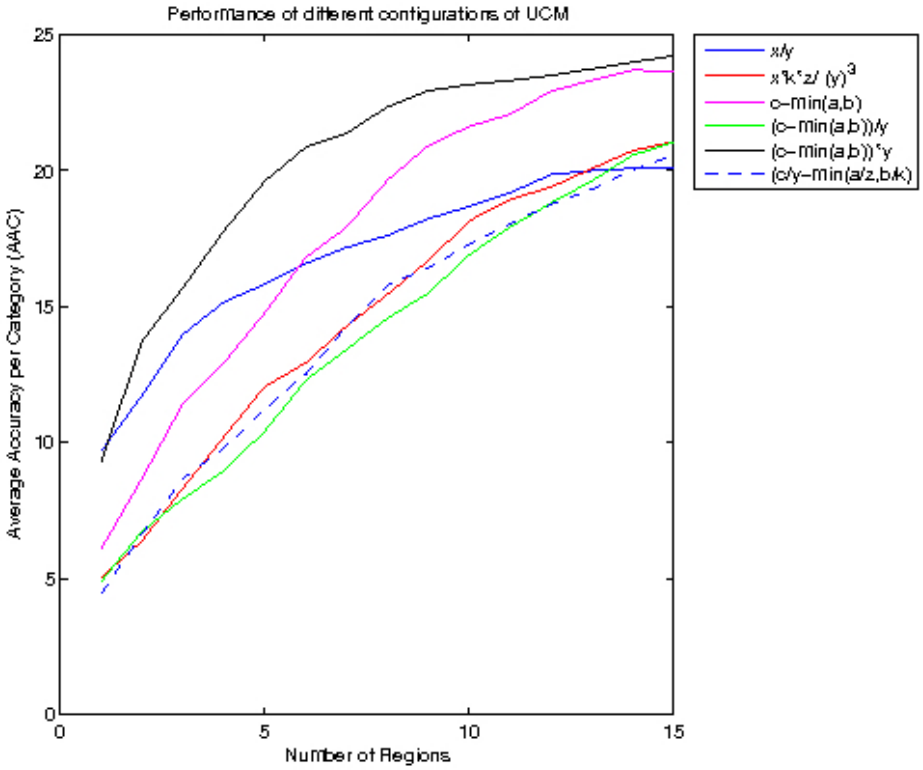


Figure 26: Configurations from 12 to 17. Plot of 15 regions.

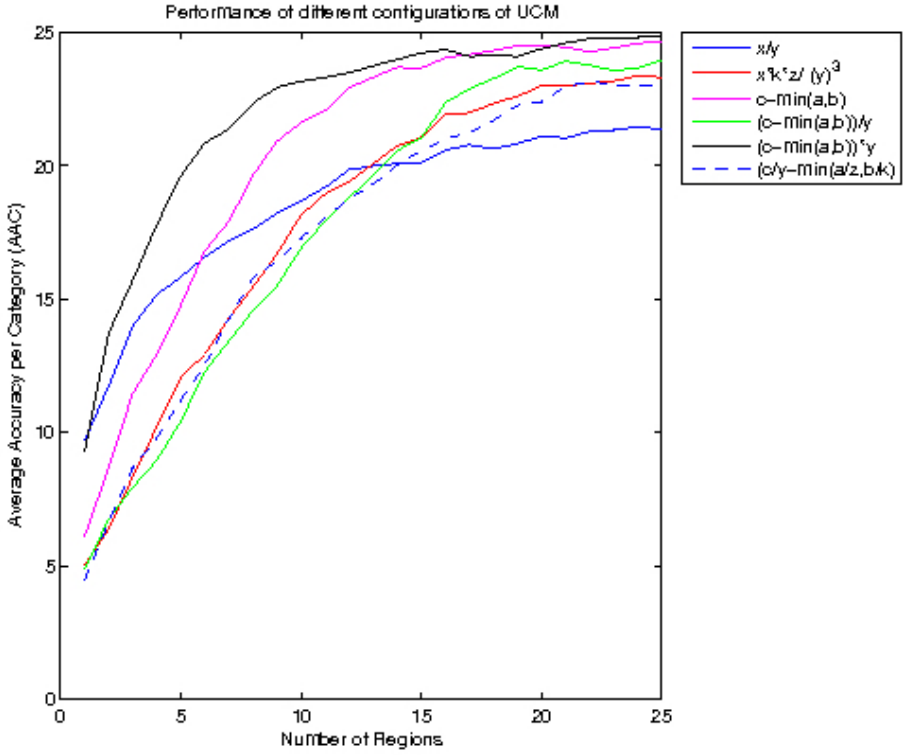


Figure 27: Configurations from 12 to 17. Plot of 25 regions.

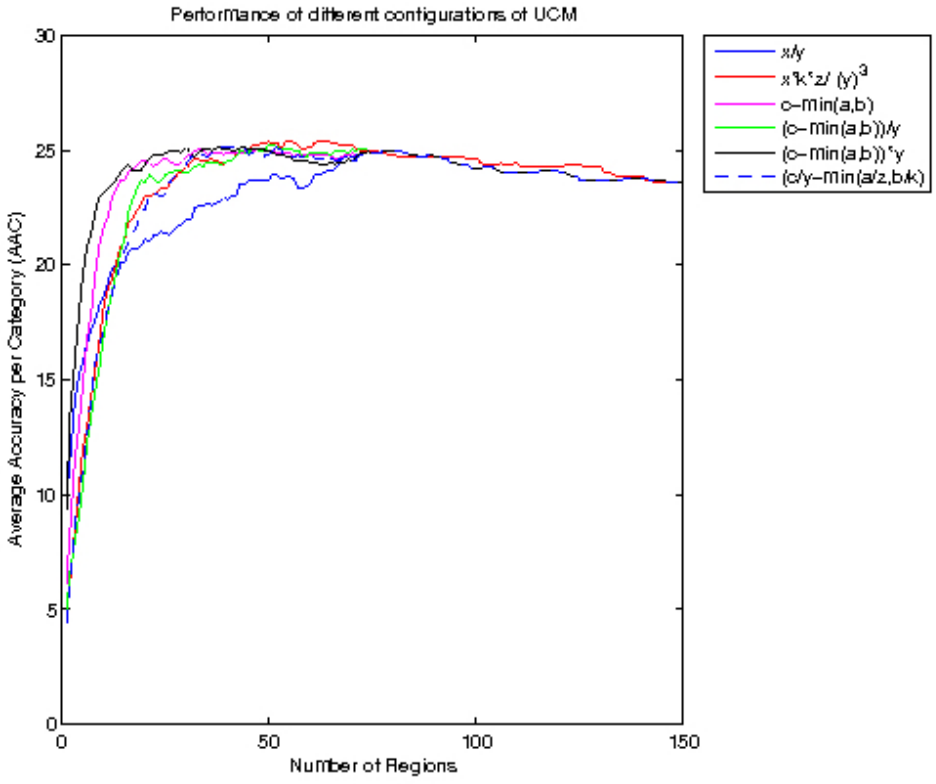


Figure 28: Configurations from 12 to 17. Plot of 150 regions.

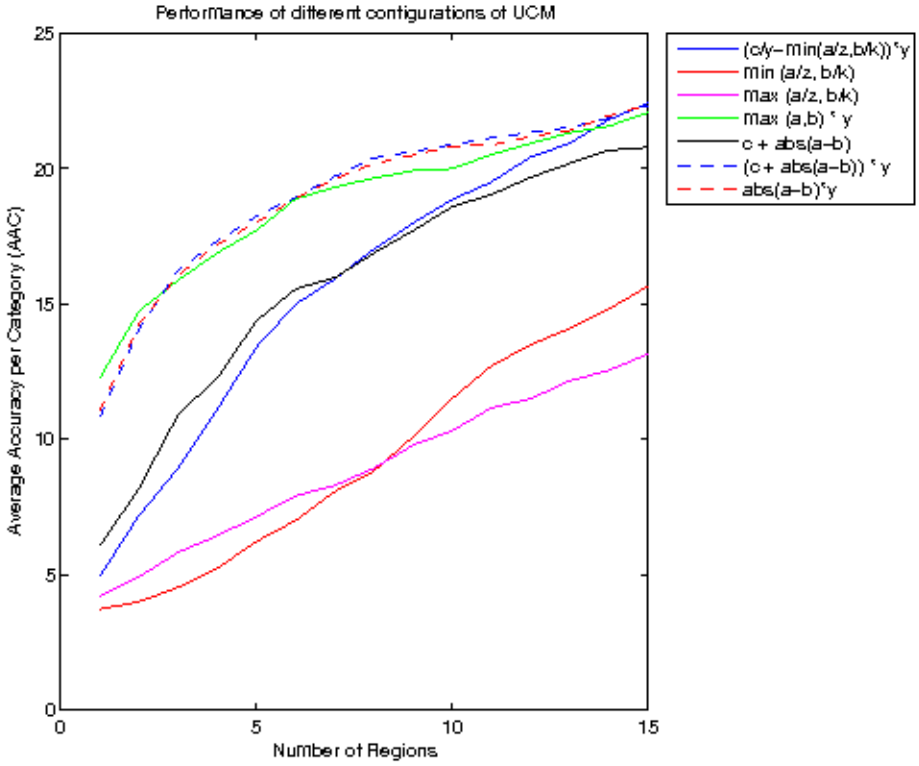


Figure 29: Configurations from 18 to 24. Plot of 15 regions.

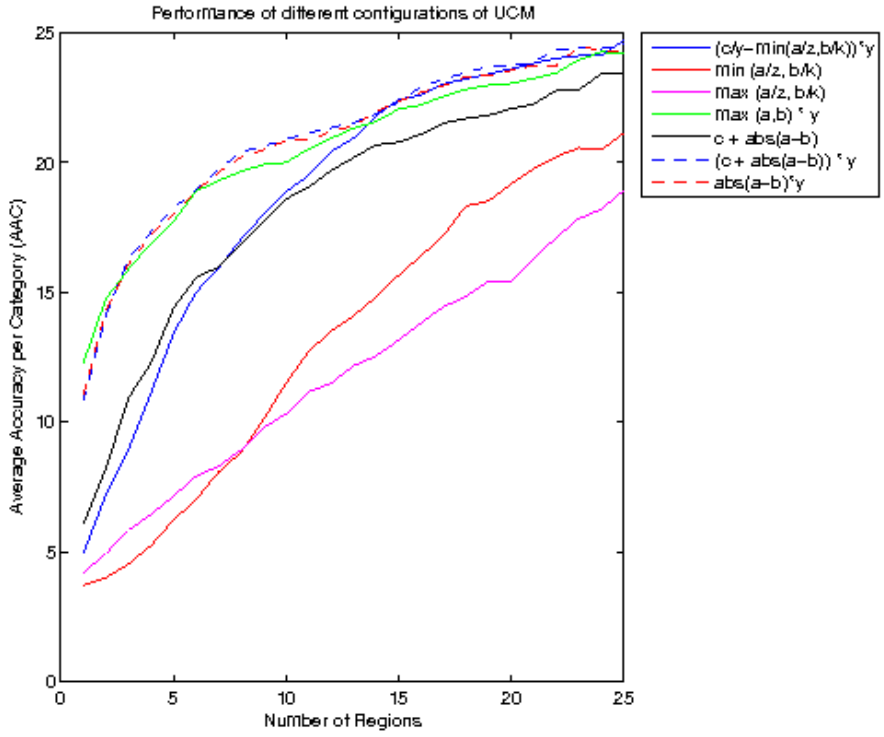


Figure 30: Configurations from 18 to 24. Plot of 25 regions.

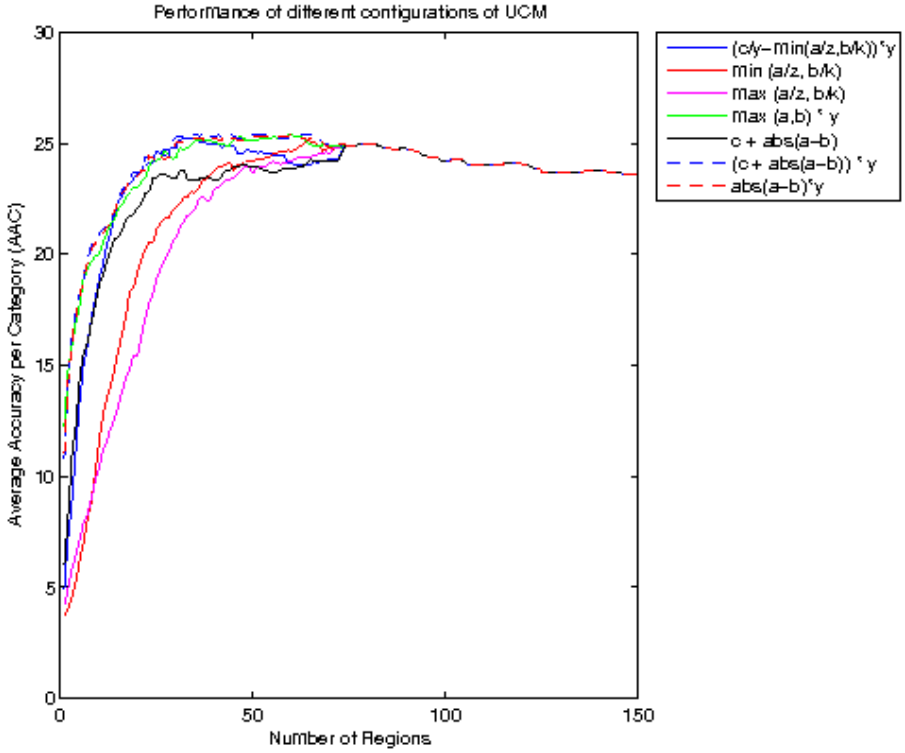


Figure 31: Configurations from 18 to 24. Plot of 150 regions.

The best configurations of all that consider costs are plotted in the plots below. These configurations are 16, 23, 21 and 10.

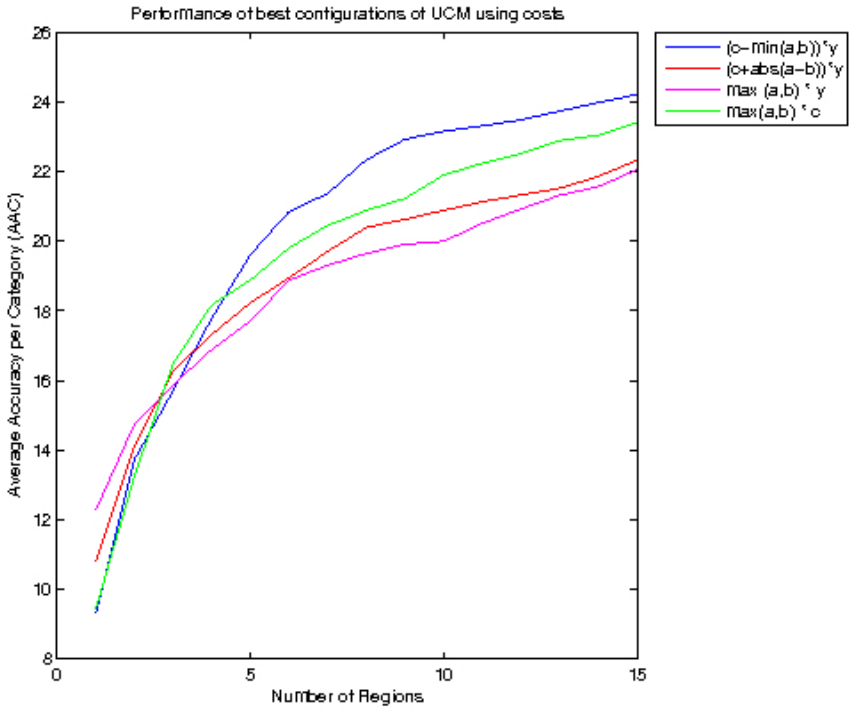


Figure 32: Best configurations that consider costs. Plot of 15 regions.

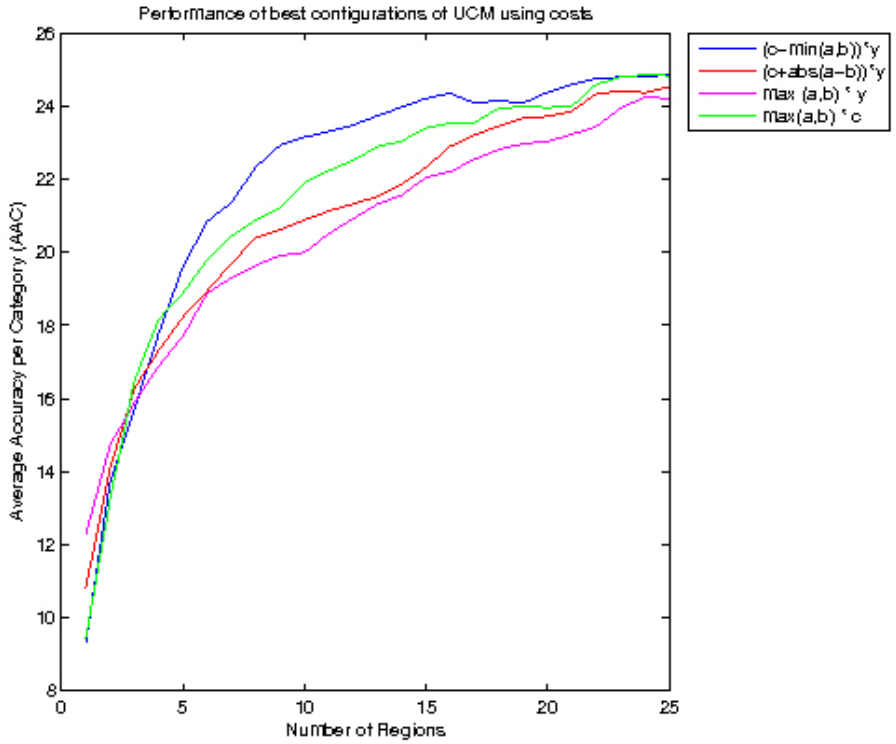


Figure 33: Best configurations that consider costs. Plot of 25 regions.

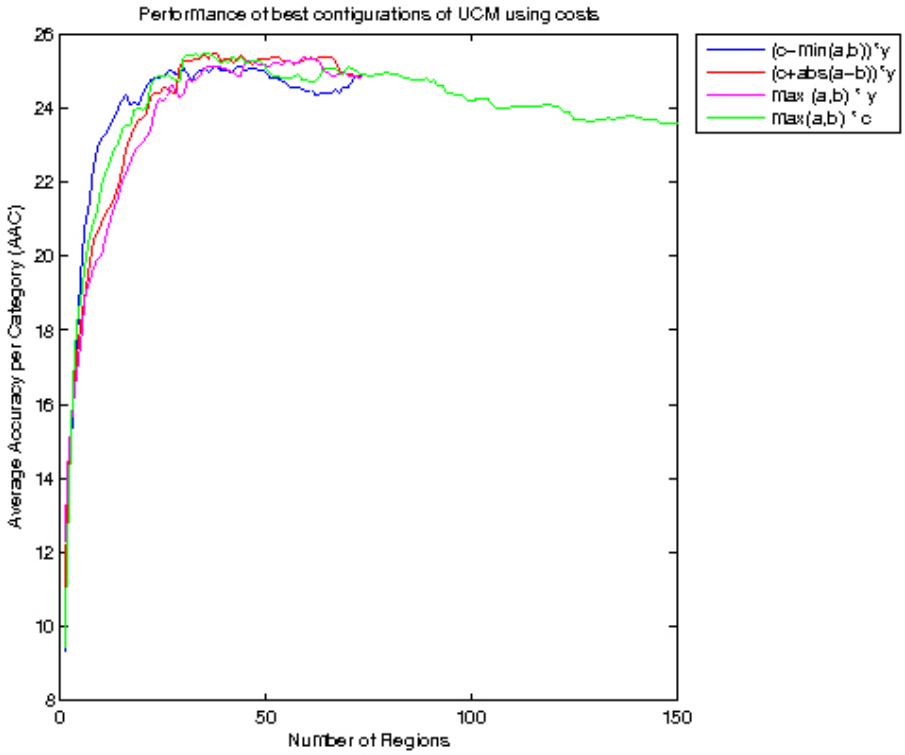


Figure 34: Best configurations that consider costs. Plot of 150 regions.

3. *Mix of indexes and costs*

Once the previous configurations have been tested, a mixture of both criteria can be considered.

Top-down approach with $(c - min(a,b)) * y$ only when a leave is reached	This approach mixes both criteria exposed previously, costs and indexes. From root node, the algorithm decides to descend through the descendant whose index is maximum. When a leave node is reached, the algorithm descend through the sibling whose $(c - min(a,b)) * y$ is maximum.	25
First derivative of the consecutive fusions' costs	The first derivative of the costs of the consecutives regions generated in the hierarchical partition is computed.	26
Second derivative of the consecutive fusions' costs	The second derivative of the costs respect to their indexes, so that the differences between costs of regions **merged consecutively** are analysed. This does not mean that consecutive merged regions have a hierarchical relation.	27

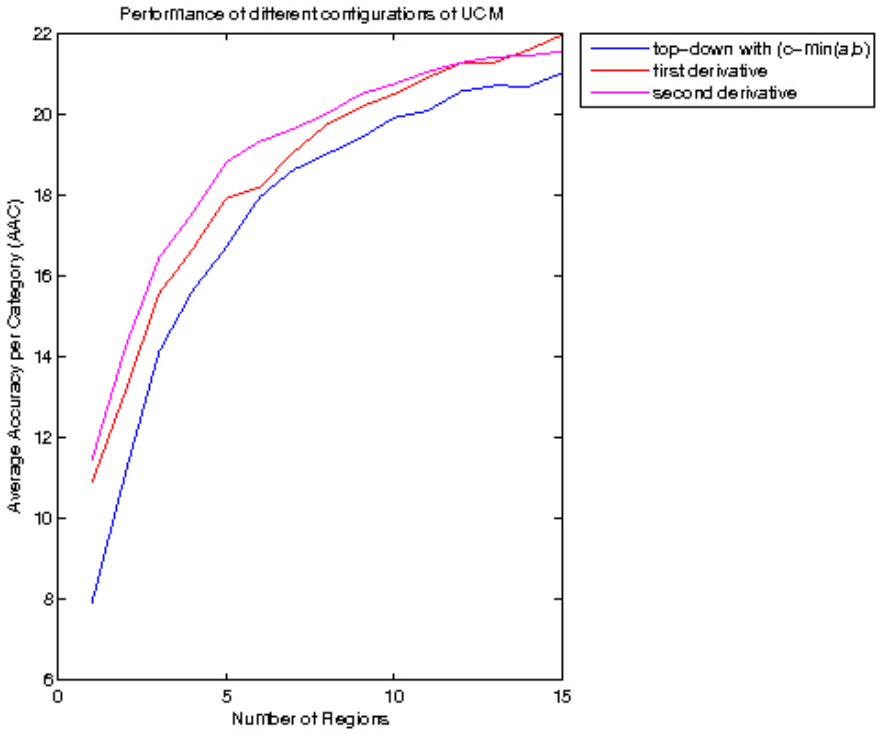


Figure 35: Configurations from 25 to 27. Plot of 15 regions.

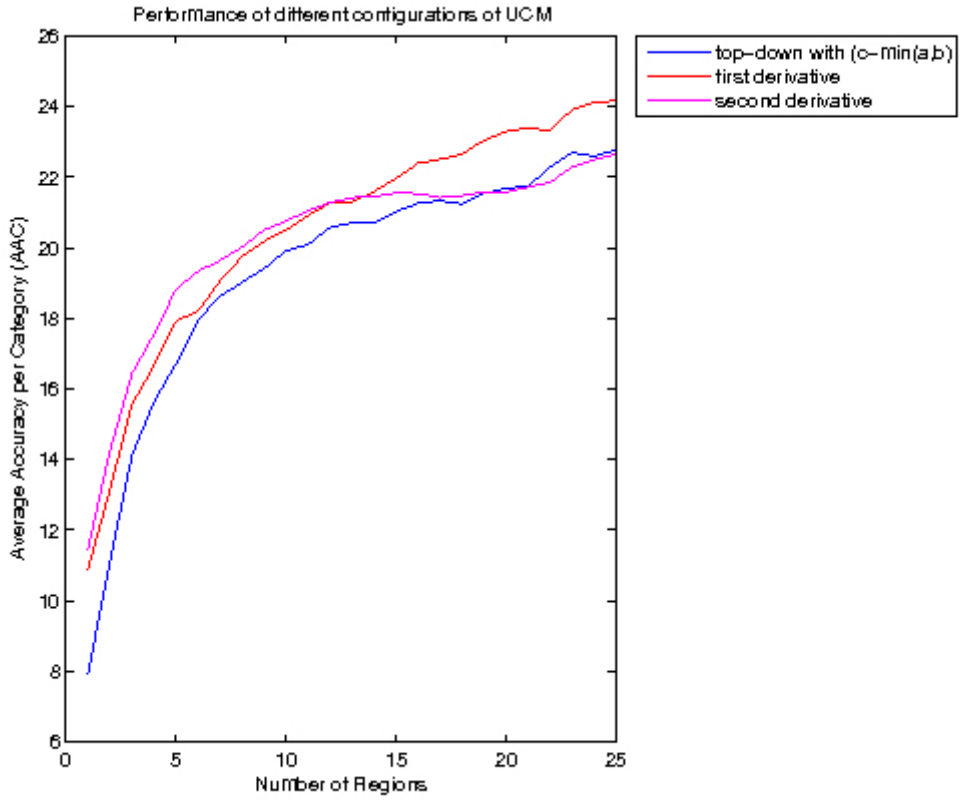


Figure 36: Configurations from 25 to 27. Plot of 25 regions.

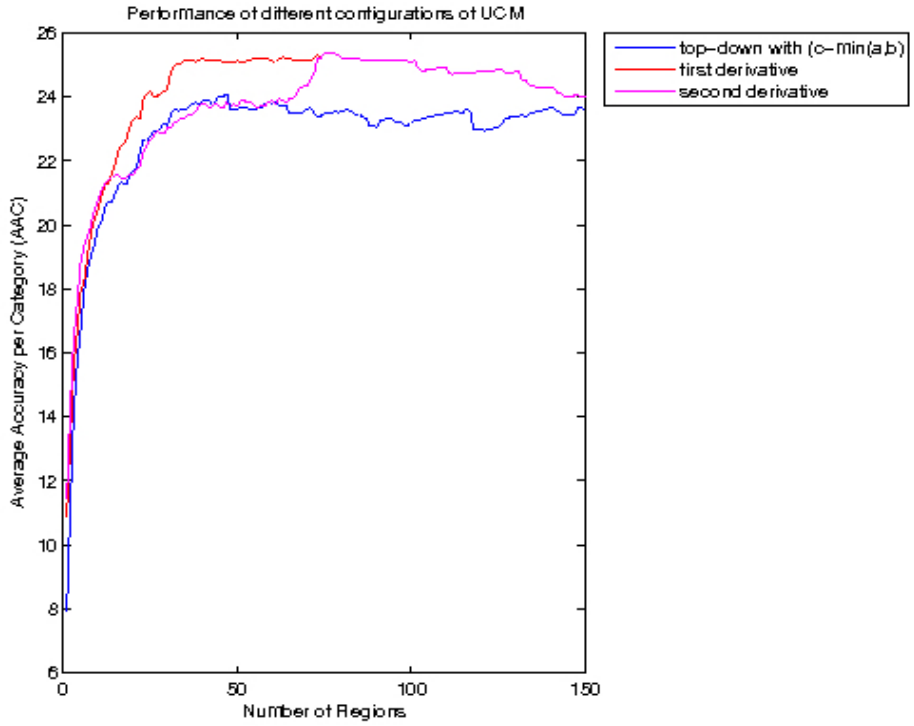


Figure 37: Configurations from 25 to 27. Plot of 150 regions.

The second derivative of the consecutive fusions' costs of the image 2008_003876 of PASCAL VOC and indexes of the regions analysed:

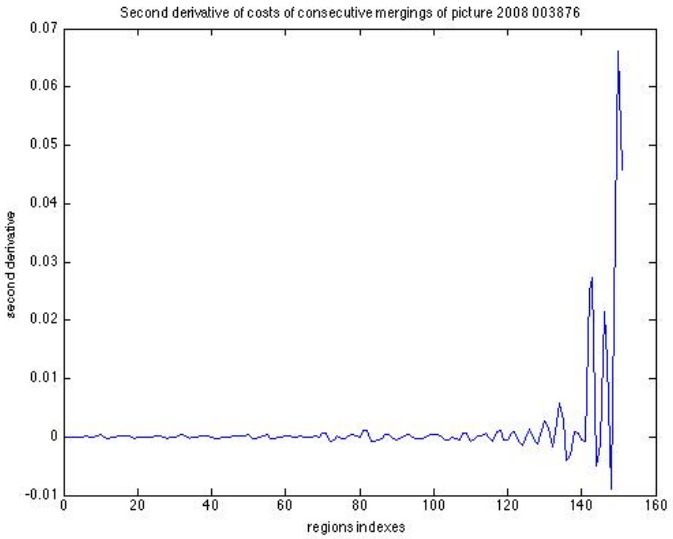


Figure 38: Second derivative of regions' costs respect to their regions' indexes.

```
150   151   149   143   142   146   147   134   135   130   131   126    81   118    82
```

Figure 39: Selected regions' indexes according this criterion.

3. By Prediction

Prediction	Top-down approach, descend through the branch which has better prediction for any of the classes	28

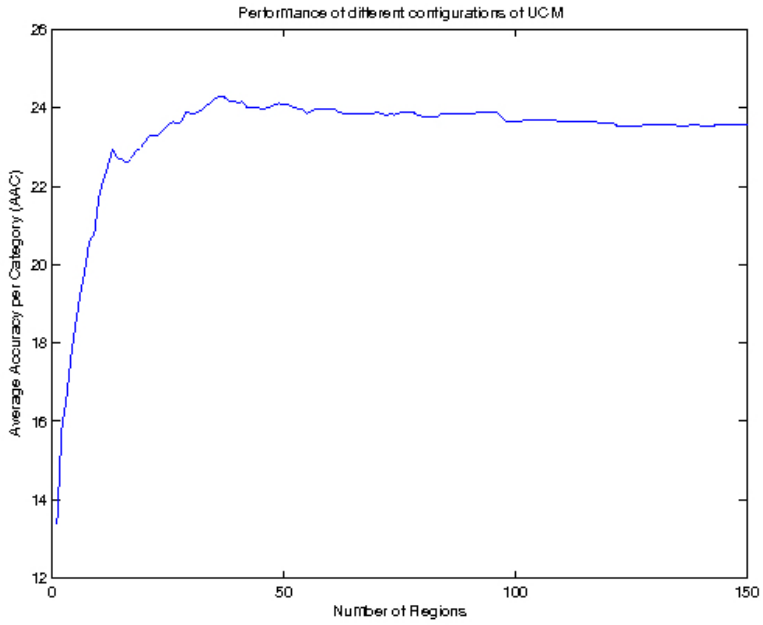


Figure 40: Configuration 28. Plot of 150 regions.

Now a set of plots of the best configurations of UCM compared to CPMC using O2P will be plotted. The best configurations using UCM are 16, 23, 28, 4 and 27.

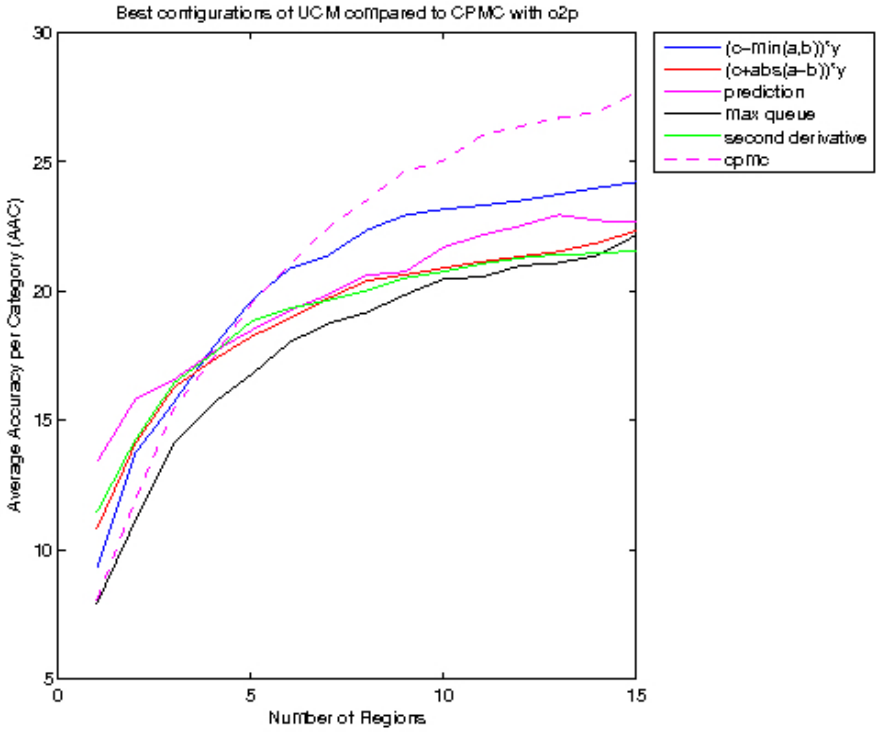


Figure 41: Best configurations of UCM compared to CPMC. Plot of 15 regions.

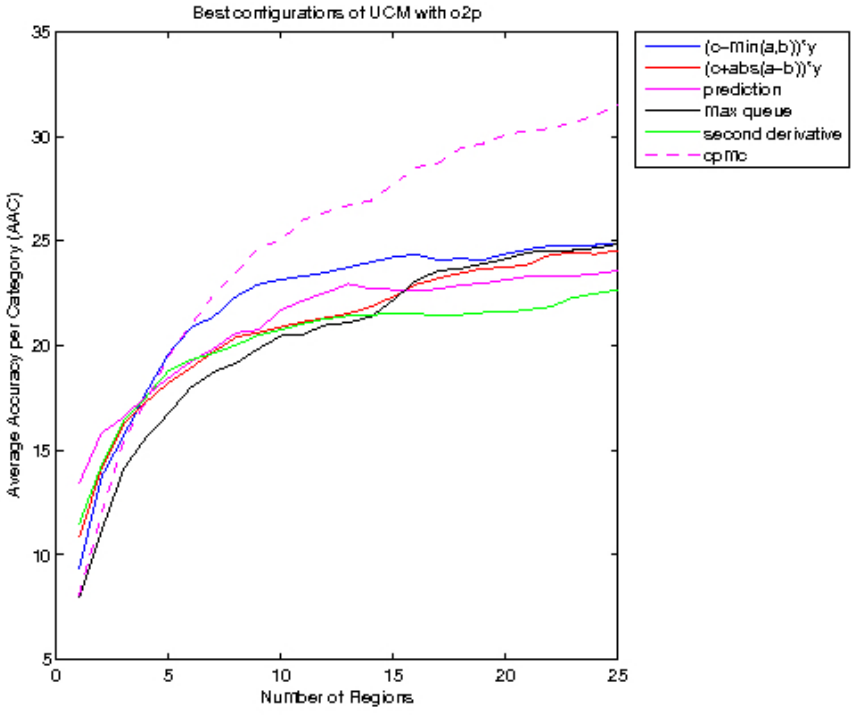


Figure 42: Best configurations of UCM compared to CPMC. Plot of 50 regions.

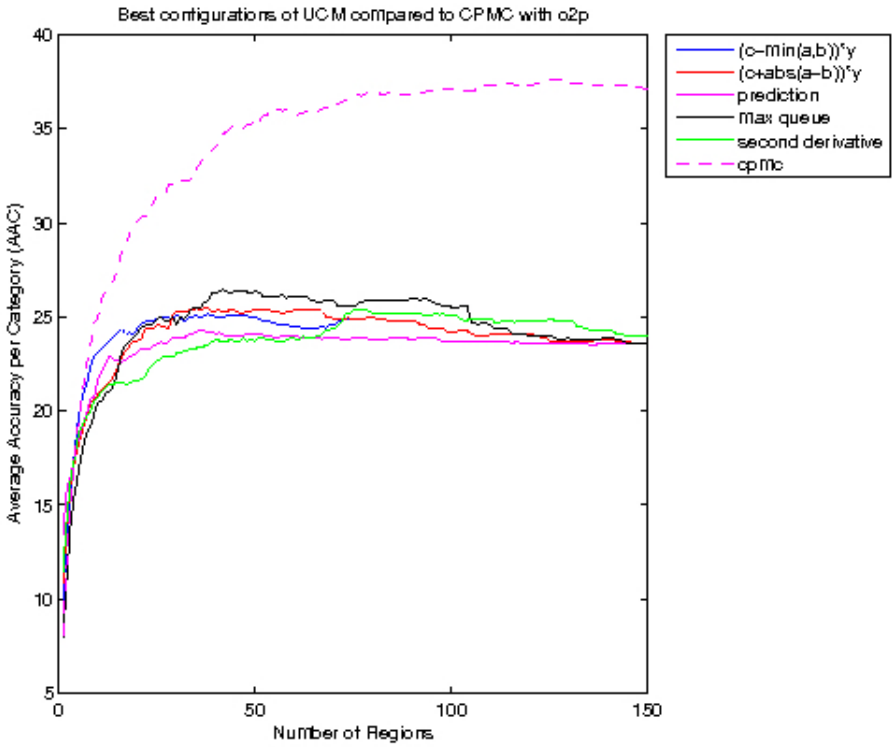


Figure 43: Best configurations of UCM compared to CPMC. Plot of 150 regions.

Now only the two best configurations of UCM compared to CPMC using o2p will be plotted. These two configurations are 16 and 28.

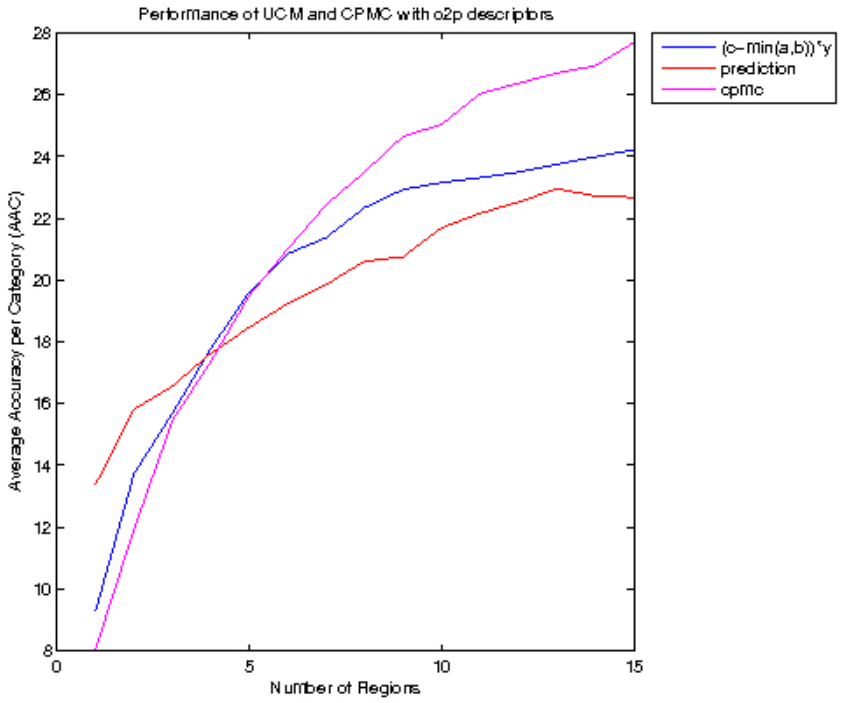


Figure 44: Two best configurations of UCM compared to CPMC. Plot of 15 regions.

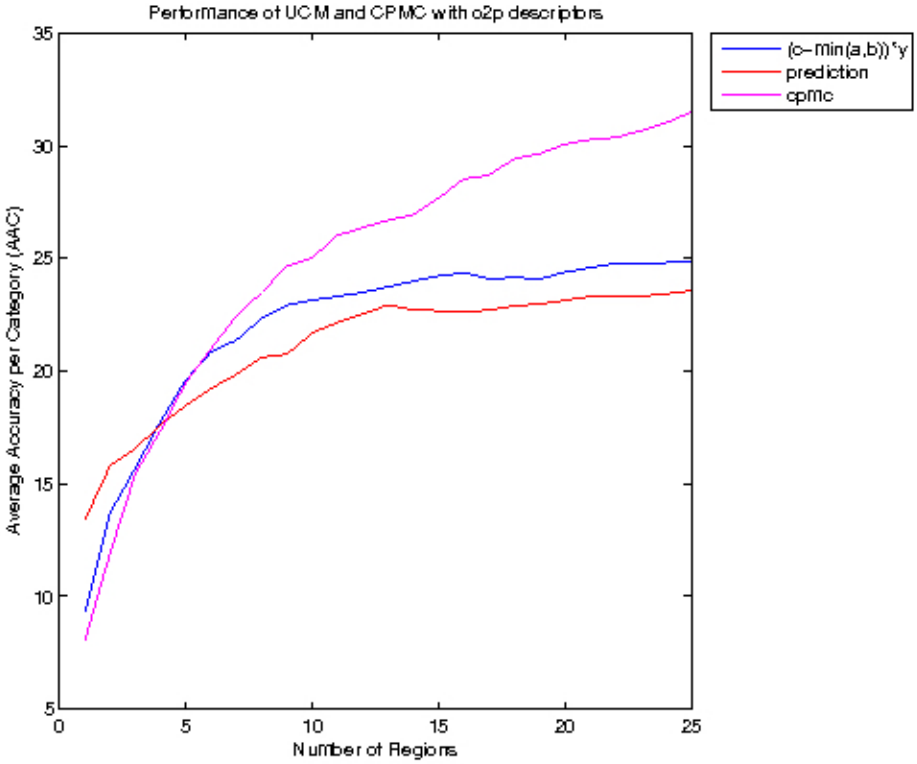


Figure 45: Plot of two best configuration of UCM compared to CPMC. Plot of 25 regions.

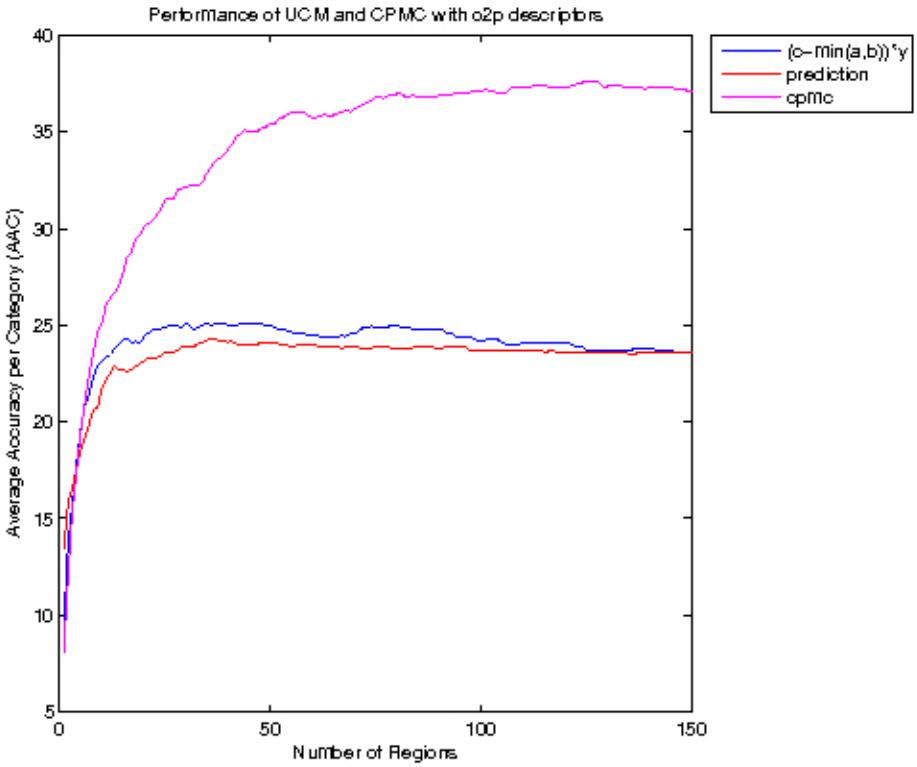


Figure 46: Plot of the two best configurations with UCM compared to CPMC. Plot of 150 regions.

- **SDS descriptors**

In the plots from below the best configurations of UCM compared to CPMC using SDS descriptors will be plotted. These configurations are the same as when using o2p.

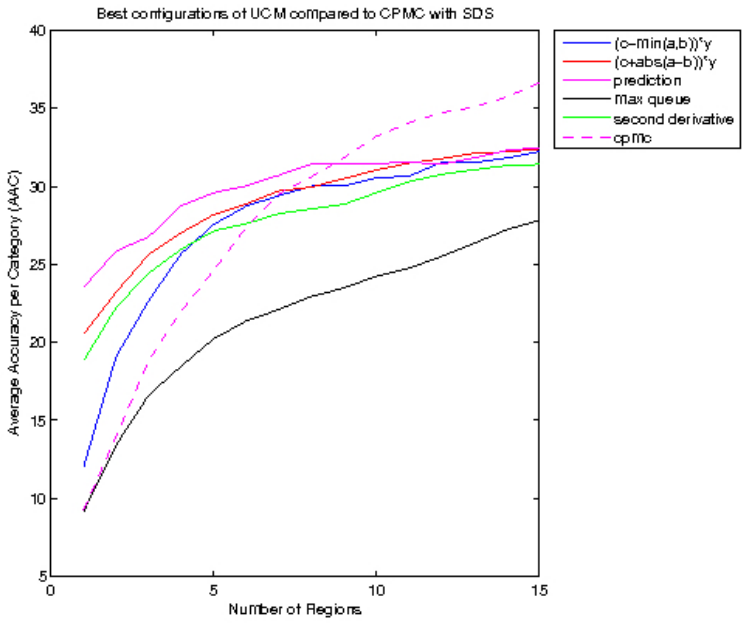


Figure 47: Best configurations of UCM compared to CPMC using SDS descriptors. Plot of 15 regions.

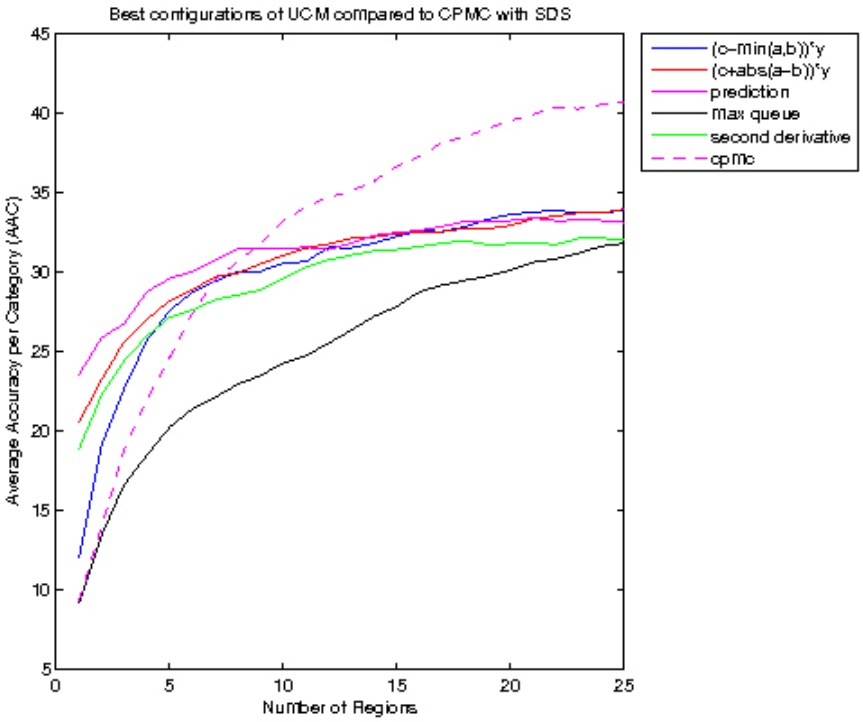


Figure 48: Plot of best configurations of UCM compared to CPMC using SDS descriptors. Plot of 25 regions.

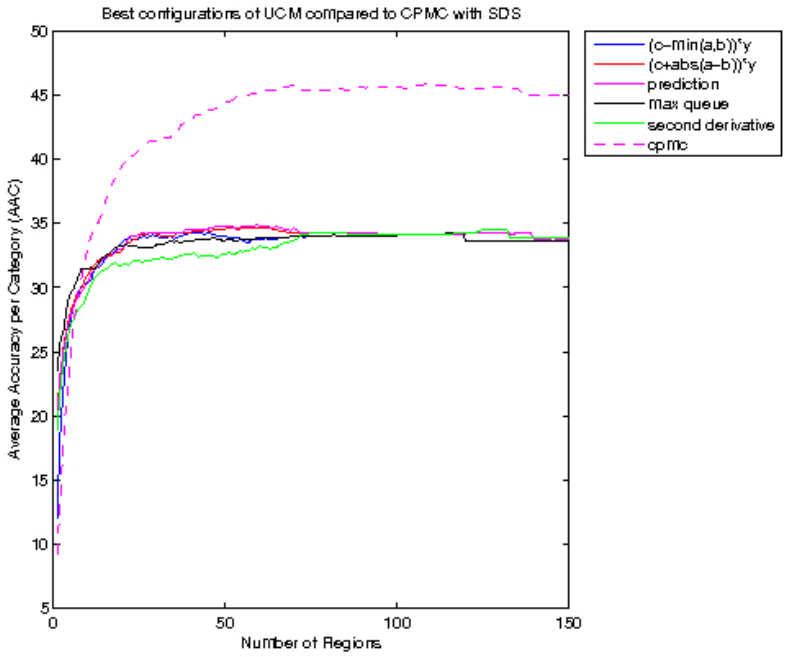


Figure 49: Best configurations of UCM compared to CPMC using SDS descriptors. Plot of 150 regions.

Now only the two best configurations of UCM compared to CPMC using o2p will be plotted. These two configurations are 23 and 28.

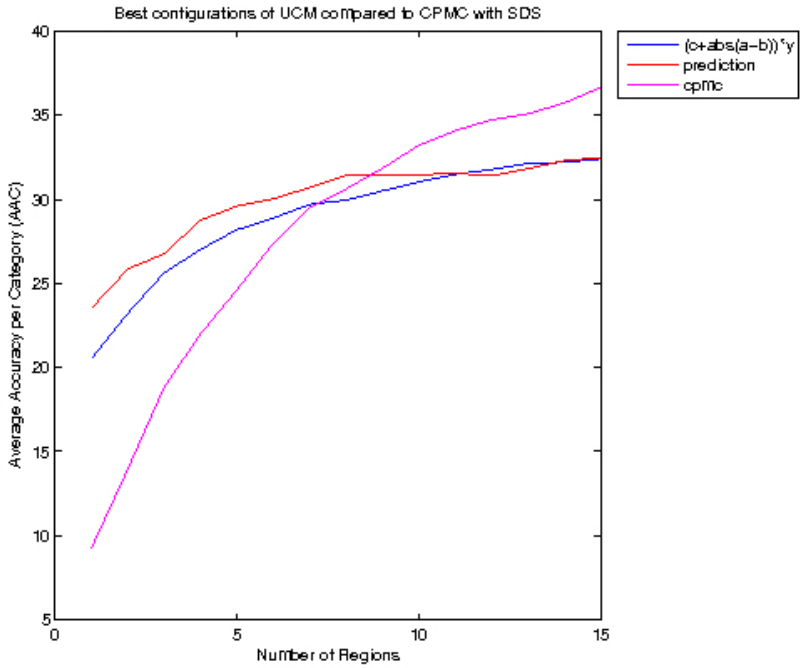


Figure 50: Two best configurations of UCM compared to CPMC using SDS descriptors. Plot of 15 regions.

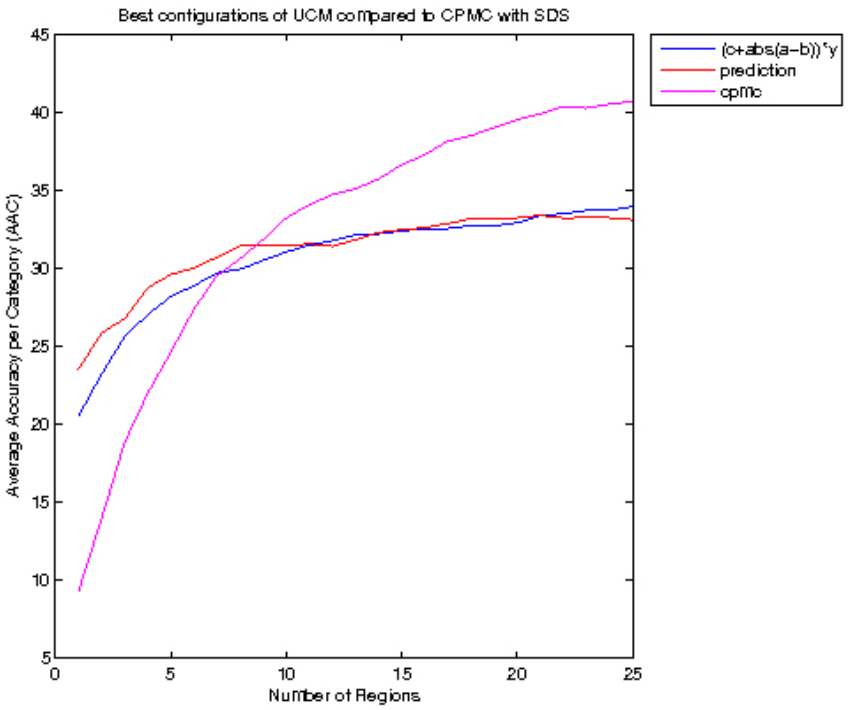


Figure 51: Two best configurations of UCM compared to CPMC using SDS descriptors. Plot of 25 regions.

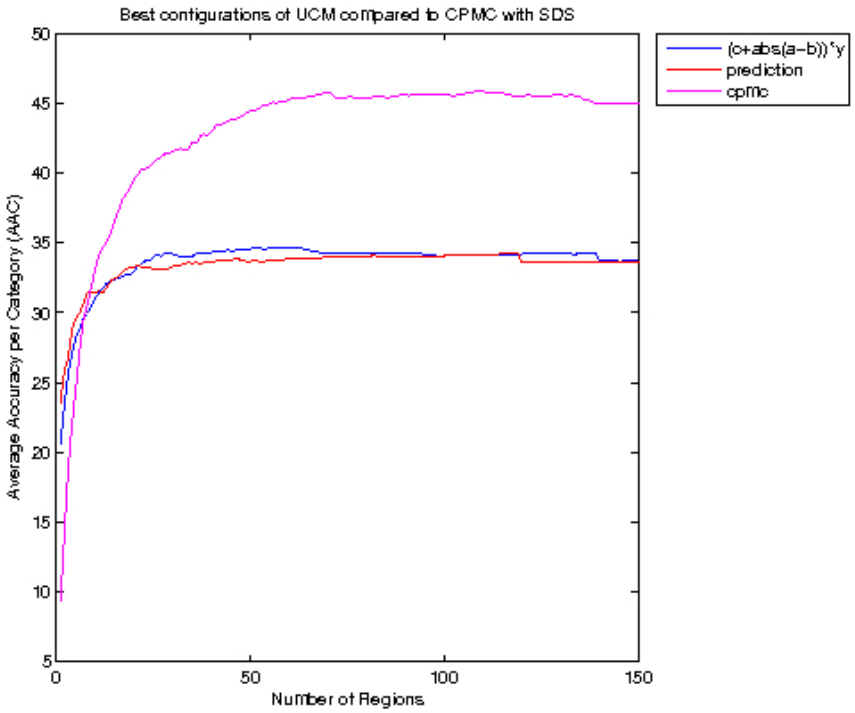


Figure 52: Two best configurations of UCM compared to CPMC using SDS descriptors. Plot of 150 regions.

Appendix II: Example of an ucm and a dendrogram

In the report there's an example of a picture of the PASCAL VOC data set ucm partition and dendrogram with a set of masks associated to different nodes of the partition. In the following example a picture of the same data set whose partition generation is not that ideal illustrated. Notice that in this case the trains of the image have a strong contour that confuses the algorithm and separates the trains into two different parts really separated in the tree as depicted in the dendrogram.

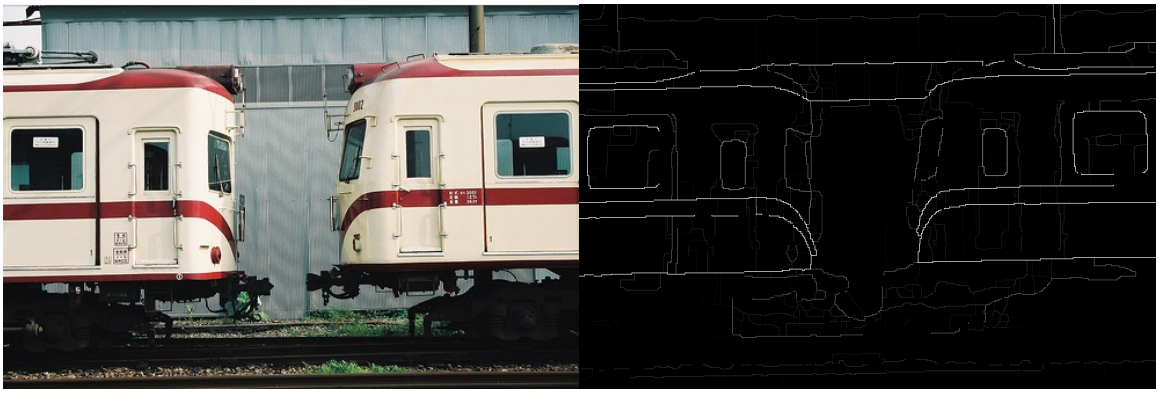


Figure 53: The picture of the left corresponds to a picture of the PASCAL VOC dataset, and the picture of the right is its ucm partition.

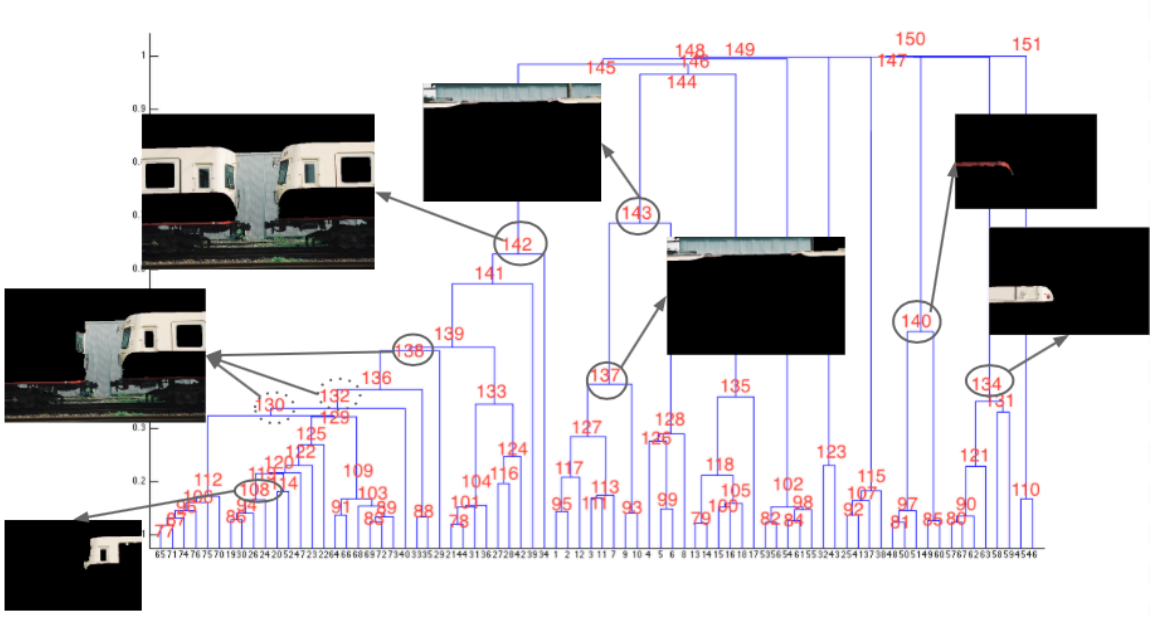


Figure 54: Dendrogram of the tree partition generated from this same picture.

Appendix III: Extended Abstract for WiCV in CVPR

Efficient Intra-image Search for Object Recognition

Míriam Bellver
Universitat Politecnica de Catalunya (UPC)
Jordi Girona 1-3, 08034 Barcelona
miriam.bellver@alu-etsetb.upc.edu

1. Motivation

The motivation of this work is the exploration of hierarchical image partitions for an efficient object recognition within an image. While many efforts have been focused on efficient image search in large scale databases, few works have addressed the problem of locating and recognizing object in the large amount of pixels contained in an image. My work considers as an input a multiscale and hierarchical partition of an image that defines a set of regions as candidate locations to contain an object. Given this partition, two problems are addressed in my research: (a) a bottom-up feature extraction to describe the regions in the hierarchy and, (b) a top-down intra-image search to detect and recognize objects at a local scale. This work is an extension on my advisor's Phd thesis [5].

2. Related Work

The traditional local analysis of an image is based on scanning with a sliding window different locations and scales of an image. This exhaustive approach was combined with an efficient feature extraction in the classic work by Viola and Jones [14] and is also at the core of popular convolutional neural networks (convnets) [11, 10].

An alternative to exhaustive search is using class-agnostic image processing to estimate the most feasible locations in an image to contain an object. A first family of solutions is based on the saliency maps [9], which typically assign to each pixel a likelihood value that predicts the user attention. These solutions though do not directly provide a region of support for the object.

A different approach aims at reducing the amount of possible locations for an object by clustering the pixels with a segmentation algorithm. In these cases, a reduced set of regions are automatically defined with precise boundaries. However, flat segmentations typically focus their analysis at a certain spatial scale, which is not rich enough when the size of the object is unknown. As an alternative, hierarchical segmentations [12, 1] provide a nested set of regions that capture a broad range of scales.

A last group of techniques are based on generating a ranked list of object candidates in the image, whether as bounding boxes [13] or accurate segments [4, 2]. These techniques try to model the generic appearance of an object so that class-specific detectors are trained on them. For example, selective search boxes [13] were trained on convnets in [6], CPMC regions were used to extract O2P features in [3], or MCG boxes and segments [2] were also processed by convnets in [7].

My work aims at comparing the performance of hierarchical partitions when compared with object candidate approaches in terms of efficiency, both during feature extraction and intra-image search.

3. Bottom-up feature extraction

Hierarchical partitions are excellent structures to propagate features from the finer to the coarser regions. Each node in the hierarchy represents not only a region but the smaller region it contains, that is, the ones defined in the hierarchy below.

We aim at exploring two approaches for efficient feature extraction. The first one is based on the O2P features proposed in [3]. They are based on an average or max pooling of the SIFT features densely generated from a region.

In addition, we also aim at extending the efficient feature extraction on convnets for spatial pyramids presented in [8], where the arbitrary partition used would be replaced by UCMs [1].

4. Top-down intra-image search

The top-down intra-image search is performed by training and assessing in each node two different classifiers. The first classifier will aim at estimating if an instance of the modeled class is *contained* in the region under analysis. This classifier will drive an efficient search as all regions in the discarded sub-trees will be discarded. A second classifier will actually aim at assessing whether the region under analysis actually represents an instance of the modeled class.

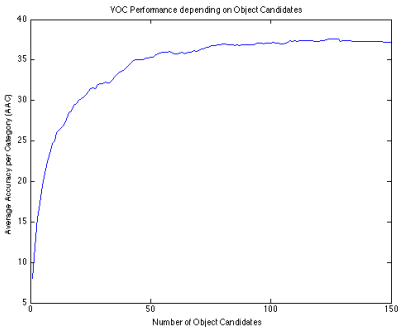


Figure 1. Accuracy per Category in CPMC depending on number of object candidates.

Given that in many cases the regions defined in the hierarchical partitions do not perfectly match the object but a part of them, when a high confidence is achieved on a region, unions with neighbouring regions will also be assessed for detection.

5. Experiments

In the interest of assessing the strategy defined through this abstract, a series of experiments are programmed to compare its performance to state of the art techniques with the benchmark defined by PASCAL VOC project and based in the Matlab implementation published in [3]. This will enable us to compare the Average Accuracy per Category (AAC) obtained by different algorithms using different numbers of object candidates, since this is the key to assess efficiency.

The first experiment in Figure 1 tests how the number of object candidates used to segment images with CPMC affects the accuracy obtained. The results prove that though increasing the number of object candidates raises the quality of the semantic segmentation, the quality is mainly achieved with the first candidates of the sorted list given by CPMC.

6. Conclusions

Further experiments will compare CPMC to UCM partitions. First a naive order of the UCM regions will be tested, and then UCM regions will be smartly sorted considering which branches should be discarded in order to drive an efficient search through the tree.

Our expectations are that a top-down intra-image search in hierarchical partitions could be beneficial in terms of efficiency without substantially affecting the accuracy.

References

[1] P. Arbelaez. Boundary extraction in natural images using ultrametric contour maps. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pages 182–182. IEEE, 2006.

[2] P. Arbelaez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 328–335. IEEE, 2014.

[3] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *Computer Vision–ECCV 2012*, pages 430–443. Springer, 2012.

[4] J. Carreira and C. Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1312–1328, 2012.

[5] X. Giró i Nieto et al. Part-based object retrieval with binary partition trees. 2012.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

[7] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *Computer Vision–ECCV 2014*, pages 297–312. Springer, 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *arXiv preprint arXiv:1406.4729*, 2014.

[9] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[12] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *Image Processing, IEEE Transactions on*, 9(4):561–576, 2000.

[13] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders. Segmentation as selective search for object recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1879–1886. IEEE, 2011.

[14] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

Appendix IV: Post in Bitsearch blog about Semantic labelling of CPMC object candidates

This blog reviews how the work of Joao Carreira et al in "Semantic Segmentation with Second-Order Pooling" (ECCV 2012) uses object candidates to generate a semantic segmentation of an image. In order to differentiate between *object segmentation* and *semantic segmentation*, we present the present example in Figure 1 extracted from the examples provided by PASCAL.

Figure 1: Object segmentation and class segmentation that are different

Figure 1 shows the object segmentation of an image that contains two buses. As the class category for both objects is the same, when performing object segmentation we obtain two different objects, but if we perform a class or semantic segmentation, the result is two objects tagged by the same category, for this reason colored with the same color.

Some works aiming at the automatic segmentation of objects from images have adopted a pipeline based on object candidates. These candidates correspond to segments in the image which may potentially represent a semantic object. The output of these algorithms is typically a list of a predefined amount of object regions, ranked according to the confidence of representing a semantic object. For example, Constrained Parametric Min-Cuts - CPMC (Carreira 2010) or Multiscale Combinatorial Grouping-MCG (Arbelaez 2014) are two examples of such techniques. Figure 2 shows an example of the CPMC object candidates proposed by Carreira.

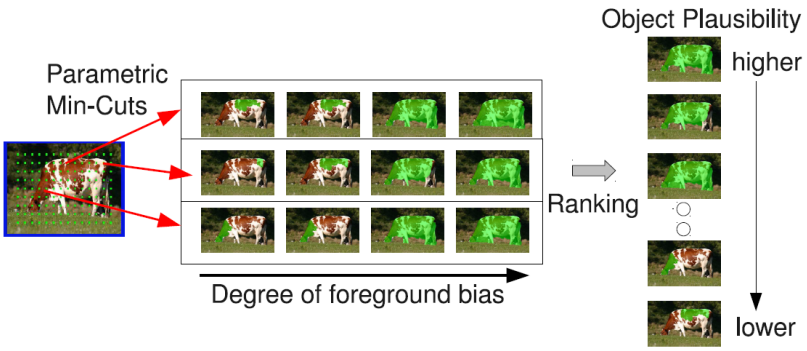


Figure 2: Ranked list of object candidates with CPMC

The process of generating a class segmentation from the object candidates of an image is explained in detail by Joao Carreira in Object Recognition by Sequential Figure-Ground

Ranking. The pipeline of this process in presented in Figure 3.

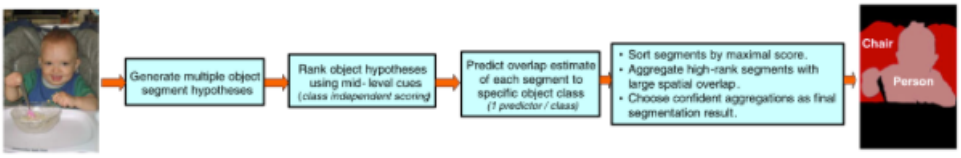


Figure 3: Semantic segmentation pipeline

First of all, a set of object candidates are generated, which is explained in detail in our previous post "Object Candidates with Constrained Parametric Min-Cuts". Next step is ranking object candidates by means of a class independent scoring system, which detects segments with object-like regularities. After that, a confidence score for each class is computed over each object candidate. This prediction is generated after training a regressor with those object candidates from the training set with a high overlap with the labelled regions in the ground truth. The object candidates are ranked based on their maximum score obtained on any of the 20 classes. Finally, a confidence threshold is learned from the training dataset to maximize the final evaluation metric. This threshold is applied on each test image to keep only those object candidates with a higher score. Finally, the class segmentation is obtained by painting the kept candidates with their highest class label in increasing order. Figure 4 shows two different object candidates generated from a very tricky image depicting two buses. For each object candidate, the likelihood that this candidate belongs to a certain class is predicted. The object candidates are ranked based on their segment score and, by learning a threshold on that score, only the top candidates would be kept.

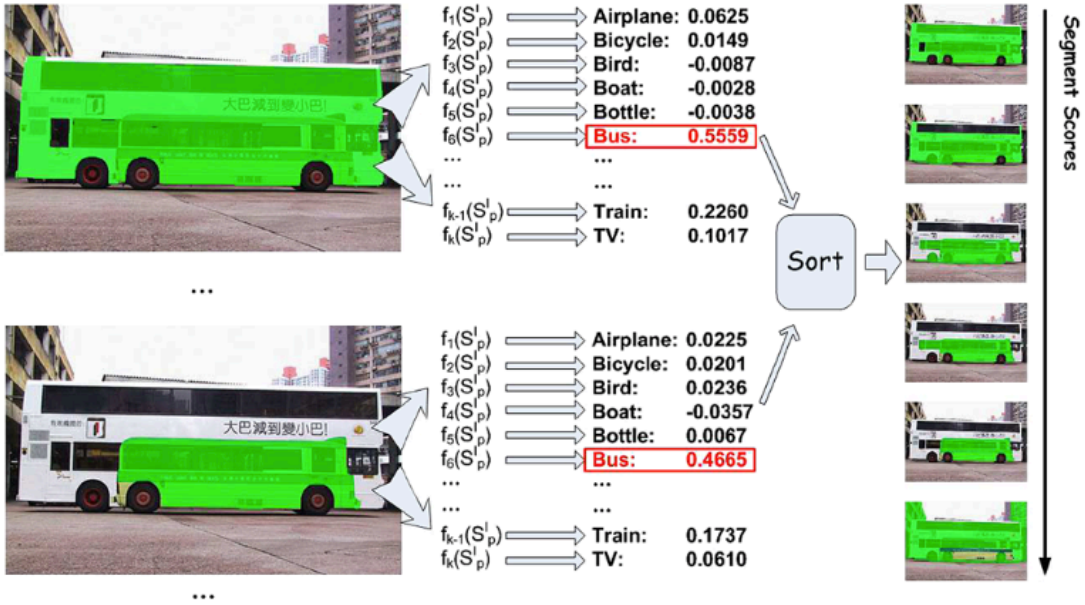


Figure 4: Sorting object candidates

Glossary

Ground-Truth: Annotations of the training set of a dataset.

Handcrafted features: Features that describe images using techniques developed by humans. In contrast learned features are features learned using machine learning.

Min-Cut: In graph theory, a minimum cut of a graph is a cut whose cut set has the smallest number of edges or smallest sum of weights possible.

LSUN: Large-scale Scene Understanding Challenge.

WiCV: Women in Computer Vision Workshop.

CVPR: Premier annual Computer Vision event.

CNN: Convolutional Neural Network. A CNN is similar to a neural network, i.e. a family of statistical models inspired by biological neural networks that are configured to recognize patterns or to classify data through a learning process. CNN architectures are adapted for images, so that certain characteristics of images are encoded in the same architecture.

Fine-tuning: Fine-tuning is the technique that fine-tunes the CNN weights using the in-domain data.

SVM: Support Vector Machines are supervised learning methods that can be applied to classification or regression.

SVR: Support Vector Regression is the version of SVM for regression.

Dendrogram: It is a tree diagram used to illustrate the arrangement of the clusters produced by hierarchical clustering.

Pruned tree: A tree whose size has been reduced by removing sections of the tree.

Fully-grown tree: A tree that is not pruned.

Leave: A leave of a tree partition is a node that has no descendants.

Root node: The root node is the node from which all the tree partition is developed.

Top - down approach: A top-down approach is an approach that descends through the partition starting at the root node.

LIFO: Last In First Out.

FIFO: First In First Out.