Departament de Teoria
del Senyal i Comunicacions

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Part-based Object Retrieval with Binary Partition Trees

PhD Thesis Dissertation

by

XAVIER GIRÓ I NIETO

Submitted to the Universitat Politècnica de Catalunya (UPC)
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

April 2012

Supervised by Professor FERRAN MARQUÉS ACOSTA
and Professor SHIH-FU CHANG

Phd program on Signal Theory and Communications

Contents

Abstract

This thesis addresses the problem of visual object retrieval, where a user formulates a query to an image database by providing one or multiple examples of an object of interest. The presented techniques aim both at finding those images in the database that contain the object as well as locating the object in the image and segmenting it from the background.

Every considered image, both the ones used as queries and the ones contained in the target database, is represented as a Binary Partition Tree (BPT), the hierarchy of regions previously proposed by Salembier and Garrido (2000). This data structure offers multiple opportunities and challenges when applied to the object retrieval problem.

One application of BPTs appears during the formulation of the query, when the user must interactively segment the query object from the background. Firstly, the BPT can assist in adjusting an initial marker, such as a scribble or bounding box, to the object contours. Secondly, BPT can also define a navigation path for the user to adjust an initial selection to the appropriate scale.

The hierarchical structure of the BPT is also exploited to extract a new type of visual words named *Hierarchical Bag of Regions (HBoR)*. Each region defined in the BPT is characterized with a feature vector that combines a soft quantization on a visual codebook with an efficient bottom-up computation through the BPT. These features allow the definition of a novel feature space, the *Parts Space*, where each object is located according to the parts that compose it.

HBoR features have been applied to two scenarios for object retrieval, both of them solved by considering the decomposition of the objects in parts. In the first scenario, the query is formulated with a single object exemplar which is to be matched with each BPT in the target database. The matching problem is solved in two stages: an initial top-down one that assumes that the hierarchy from the query is respected in the target BPT, and a second bottom-up one that relaxes this condition and considers region merges which are not in the target BPT.

The second scenario where HBoR features are applied considers a query composed of several visual objects. In this case, the provided exemplars are considered as a training set to build a model of the query concept. This model is composed of two levels, a first one where each part is modelled and detected separately, and a second one that characterises the

combinations of parts that describe the complete object. The analysis process exploits the hierarchical nature of the BPT by using a novel classifier that drives an efficient top-down analysis of the target BPTs.

Chapter 1

Introduction

1.1 Problem statement

The increasing amount of multimedia digital content has raised new challenges for its analysis and management. Audiovisual data is currently stored in large databases where navigation and search can no longer depend only on manual annotations. As a response to this need, the signal processing, computer vision and pattern recognition communities have proposed several techniques to retrieve content trying to satisfy user needs and expectations [18] [57] [81] [85].

The basic retrieval problem is formulated as how to generate a list of items from a target database ranked according to their relevance to a user query. This thesis focuses on those cases where queries correspond to visual objects extracted from an image and where the relevance is estimated considering only the visual data contained in the database. This scenario is referred as the *instance search* problem and can be formulated in two different situations, depending how the query is formulated: with one or multiple examples.

The *query by example* [27] approach considers that the user has provided one single instance of the object that is to be found. This situation is solved by trying to match the provided object with every potential "object" in the database to generate the list of those candidate objects ranked according to a measure of these matchings. Figure 1.1 represents this case.

A more complex situation occurs when the query is formulated with *multiple* objects, as shown in Figure 1.2. This problem could be solved by splitting it into several queries by example, but doing that would multiply the search time for the amount of examples. Additionally, the common patterns within the set of queries could not be exploited by the system. When a single query is formulated by multiple objects, there is a chance of abstracting what is common between all these instances and building a model of the query. This model is later compared with every object in the target dataset to generate the ranked list of results.

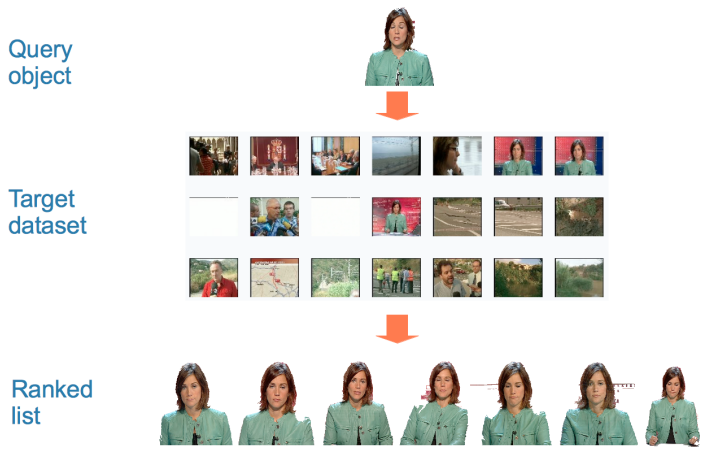


Figure 1.1: Query By Example

Figure 1.2: Query By Multiple Objects

Some authors consider that the multiple objects of a query represent a single concept, so this configuration is referred as *query by concept* [19] [99].

The two presented problems can be very easily expanded to another very popular topic in computer vision: the detection/recognition of objects. This problem consists of deciding whether a certain observation corresponds to an instance of a certain semantic class. For example, *text* and *faces* are two very popular semantic classes with multiple solutions for their automatic detection and recognition. The only difference between the retrieval and detection/recognition scenarios is that in the second case a certain minimum detection threshold is set. Such threshold, once applied on a ranked list, provides a decision boundary for detection/recognition. Figure 1.3 shows a case in which the detection threshold is set in a way that the first six retrieved regions are considered valid detections of the query object. This decision generates one false positive and two false negatives.

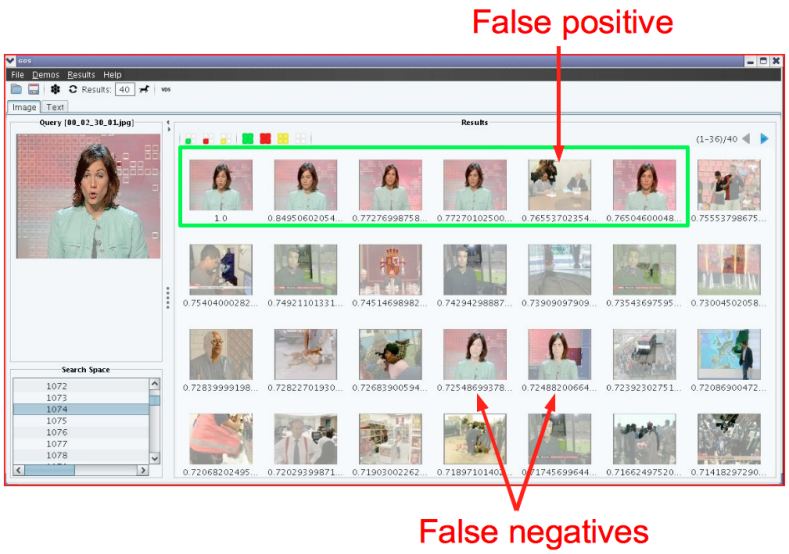


Figure 1.3: Retrieval vs Detection

1.2 Challenges

The *instance search* problem presents several challenges, but the main one is the information loss between the user idea and the formulated query. The search expectations from the user typically are associated to certain semantics which are expressed as visual content. This content can be analysed and quantified by a computer according to its perceptual properties, but the result of this analysis might be insufficient or absolutely uncorrelated to the user's will. This problem is referred to by the scientific community as the *semantic gap* and it represents the distance between the low level perceptual data extracted in terms of visual descriptors, and a high level semantic interpretation of the visual content.

The presented work also faces a second difficulty in terms of diversity. While the first generation of computer vision applications focused on solving specific domain problems, such as face or text retrieval/detection, it is a general trend in the multimedia retrieval community to develop generic solutions valid for any domain. In this way, not only a semantic gap between query concept and visual signal exists, but both semantics and perceptual data can be *diverse*. That is, the presented approach should make no difference when searching for *anchorwomen* or *traffic signals* in the database, the core algorithm and workflow must be the same. Moreover, the way these diverse semantics are to be represented will also vary, whether due to changes in the acquisition context (point of view, illumination, ..) or more drastic differences, such as the *anchorwoman* concept being represented by images with different people wearing different clothes and working in different studios.

In the generic framework addressed by this work, the adage "a picture is worth a thousand word" can be interpreted as an image containing several diverse types of semantic information.

Different areas of the image can represent different semantics, and the combination of these areas can still represent additional semantics. In order to extract all possible semantics from an image it is necessary to analyse it at a local scale, as well as consider these local data in a context, as these local data are surrounded with other local data that mutually influence their semantic interpretation. The first applications of pattern recognition on images used to consider them as a whole, extracting features at a global scale. However, detecting objects within an image requires considering the local scale and, to do so, it is necessary to define what localizations inside the image are to be studied. This approach poses the problem of how to define the parts of the image that will be analysed, a third challenge that is also addressed in this thesis.

1.3 Evaluation

This thesis presents a set of techniques that aim at solving the presented challenges. The performance of each solution has been assessed with a common experiment, whose results are provided in the *Evaluation* sections of every chapter. The establishment of a shared benchmark allows the comparison of the results in order to assess the gain obtained with every proposed technique.

The evaluation as been performed at both the *global* and *local* scales, as presented in Section 1.3.1. The reported experiments have used the datasets introduced in Section 1.3.2 and the metrics defined in Section 1.3.3, following the cross-validation set up described in Section 1.3.4.

1.3.1 Instance Search and Segmentation

This thesis proposes a collection of techniques that try to solve the *Instance Search* task. This problem processes a user query defined at a local scale to generate a ranked list of images according to a certain similarity criterion, as exemplified by Figure 1.4. The first aspect that must be evaluated is the relevance of the retrieved images, considering them from a *global* scale.

A second type of measures focus in the area of support for the retrieved object. The localization might be rough, for example, with a bounding box, or more precisely, at the pixel level. Figure 1.5 presents a screenshot of the graphical user interface developed to exploit the techniques presented in this thesis [34]. The query object corresponds to the *head* of an anchorwoman and is shown in the top left part of the screen. The grid of thumbnails display the ranked list of retrieved images, but also the image parts that have been automatically selected as the best match for the query object inside every target image.

The evaluation of results at the *global* and *local* scales correspond to the two stages of the search process. Given an image from the target database, the first task is to solve is an

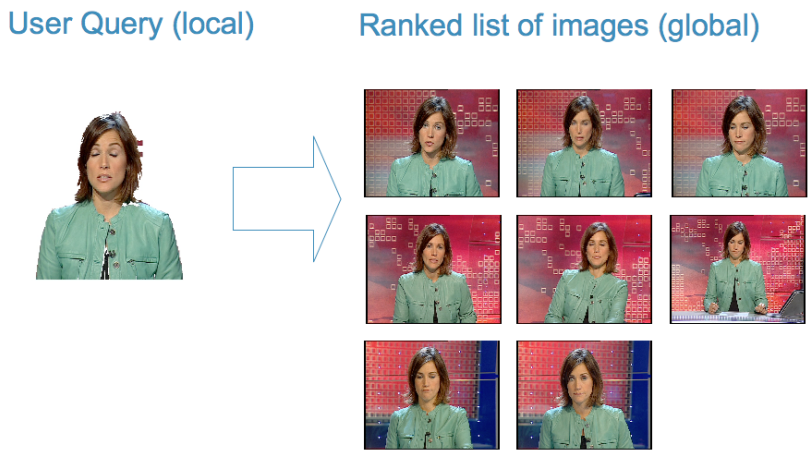


Figure 1.4: Region-based Image Retrieval.

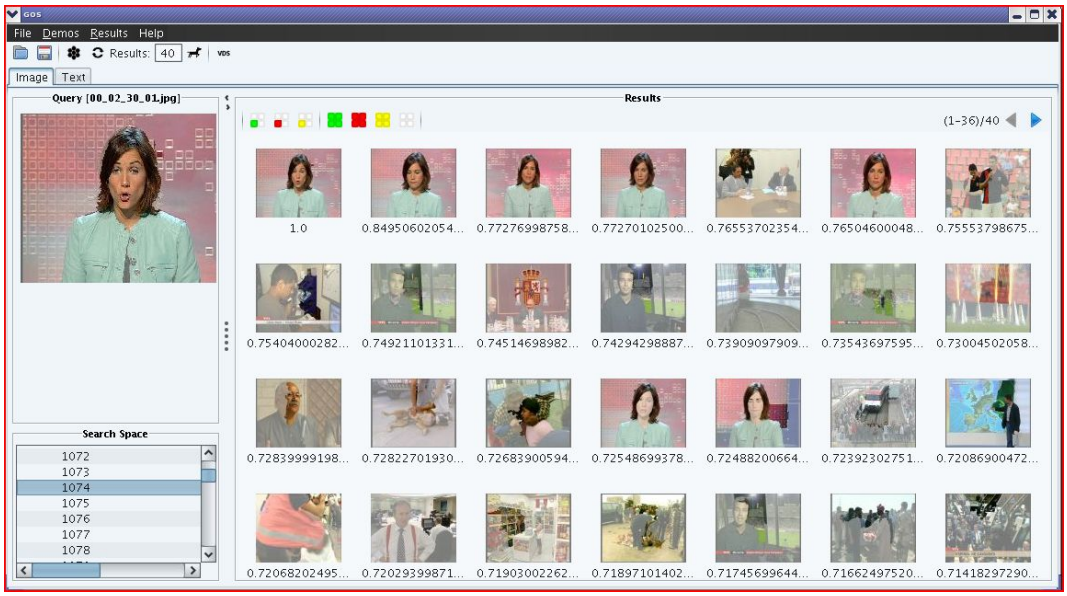


Figure 1.5: Region-based Object Retrieval and Segmentation.

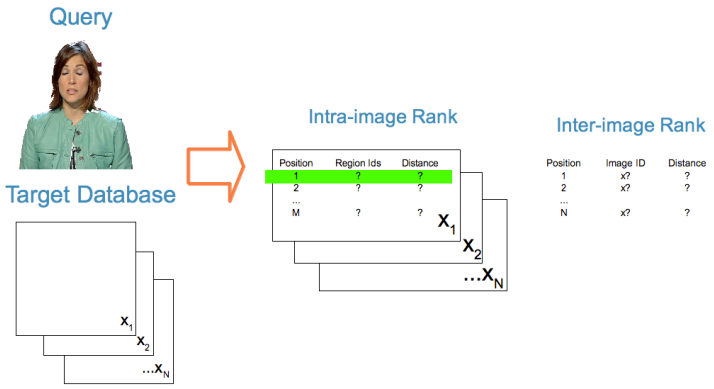


Figure 1.6: Intra- and inter-image searches.

intra-image rank, that aims at finding the local portion of the image that best matches the query object. If the target image is considered as a set of image parts, the problem becomes also a retrieval problem, but among these image parts. Once this local search is solved, the similarity score obtained for the best match is the one associated to the whole image. In the second stage, this score is used to build the ranked list of images that conform the *inter-image* rank. This thesis focuses in the *intra-image* rank as it represents the most challenging task from an image processing perspective. The reader is referred to [31] for details on the author's contributions on *inter-image* search.

The *instance search* task is additionally evaluated from a last point of view: processing time. The results provided throughout the thesis also include the time invested in the search. This value provides an estimation of the computational effort required by each technique. Additionally, in the case where a model of the object is built, the time invested in its construction has also been measured.

1.3.2 Datasets

This thesis has considered two datasets with different characteristics that test the performance of the presented solutions in different scenarios. All datasets contain a collection of images whose objects have been manually annotated at the local scale to provide a ground truth. The goal of the pattern recognition techniques is to reproduce the manual annotation, but from a completely automatic fashion.

ETHZ Shape Classes

The first dataset considered is the ETHZ Shape Classes [26], that contains 255 test images for five diverse shape-based categories (Apple logos, Bottles, Giraffes, Mugs and Swans). Every image in the dataset belongs to one, and only one category; but a single image can contain multiple instances of an object class. Apart from the original images, the dataset also includes

a manually generated contour for every instance, in addition to the coordinates of a bounding box around every object. The dataset provided by its authors was pre-processed in order to adapt it to the particularities of this thesis. This pre-processing is described in Chapter 12, included as an Annex. The ETHZ dataset has been adopted in the experiments of Part II.

CCMA News Bulletin

The second dataset contains automatically 247 extracted keyframes from a news bulletin from the Catalan Broadcasting Corporation (CCMA). The content of the bulletin is diverse, with stories coming from news agencies as well as produced by the same CCMA. Most stories are introduced by an anchorwoman, who is shown in 11 keframes from a frontal view on a static background. This *anchorwoman* object can be decomposed into distinctive parts with little variations between shots. This object has been adopted as the study case for the experiments in Part III.

1.3.3 Metrics

The evaluation of the search tasks requires numerical measures capable to express the quality of the results. This subsection presents a set of popular metrics broadly adopted among the scientific community, both at the global and local scales.

Image Retrieval

The retrieval problem aims at organizing a set of data according to a query posed by a user. Search engines are the most popular scenario for this challenge, where very large amounts of data are to be mined in order to provide the user with a ranked list of results. This ranked list must sort the results in descending order according to their relevance to the query. Typically this problem is solved by comparing the features extracted from the query with those associated to the content kept in the database. Each type of feature will have a similarity metric that will allow the comparison of the query with the elements in the database and, by doing it, establishing a criteria to rank the results.

Evaluating a retrieval system requires a set of queries and a dataset whose elements are labelled as relevant or non-relevant for each of the queries. In general, the query document should not be included in the target database.

In addition to the relevance, many retrieval systems not only try to provide relevant results but also provide diverse results. Diversity is another desirable quality in many scenarios and that is measured differently from relevance. In the present work, though, the relevance will be the only criterion applied as it is the most common in the state of the art evaluation campaigns.

Precision and Recall at k The two basic metrics for information retrieval are Precision and Recall. The combination of the two metrics provides valuable information to evaluate the performance of a retrieval algorithm. In both cases their computation requires defining how many of the first elements in the ranked list are to be assessed (k). By considering different values for k, different pairs of precision-recall values can be generated and plotted in a graph.

The *Precision* or specificity is a measure of the ability of a system to retrieve only relevant instances. It measures the exactness or fidelity of the system among of the first k retrieved documents. It is defined in Equation 1.1.

The *Recall* or sensitivity is a measure of the ability of a system to retrieve the relevant instances contained in the dataset. It is used for evaluating the completeness of results among the first k retrieved documents considered that the whole database contains a total of m relevant documents. It is defined in Equation 1.2.

$$p(k) = \frac{|relevant \cap retrieved(k)|}{k} \tag{1.1}$$

$$r(k) = \frac{|relevant \cap retrieved(k)|}{m} \tag{1.2}$$

Average Precision Given a database with m relevant documents for the query, the perfect system would be the one that would provide with a maximum precision of 1.0 for $k \in [1, m]$, and whose recall would also be 1.0 for $k \in [m, \infty]$. Such a system would be characterized by a squared-shape precision-recall curved of total area 1.0. Based on this observation, a one-dimensional measure is widely used to assess the quality of a precision-recall curve. This measure approximates the area under the precision-recall curve and it is name *Average Precision (AP)*.

AP is measured by averaging the precisions obtained at those k positions where relevant documents are located in the ranked list of results. Its expression can be found in Equation 1.3, where $rel(k)$ is a binary function whose value is 1 where the document at position k is relevant for the query, and 0 otherwise.

$$AP = \frac{1}{M} \sum_{k=0}^{N} p(k) rel(k) \tag{1.3}$$

Mean Average Precision (MAP) The evaluation of a retrieval system is normally assessed on a set of queries Q. The AP obtained for each ranked list can be averaged to obtain a measure that describes the overall performance of the retrieval solution. This value is called the *Mean Average Precision (MAP)* and it is presented in Equation 1.4.

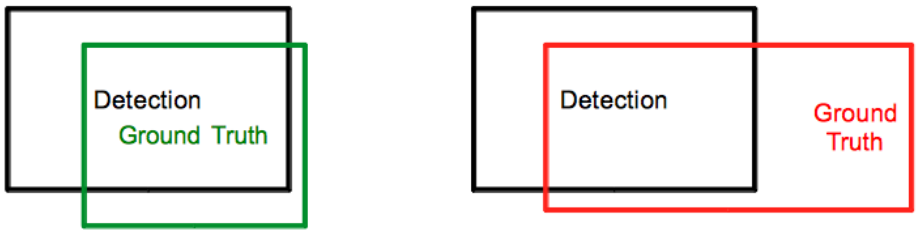


Figure 1.7: Pascal criterion.

$$MAP = \frac{1}{|Q|} \sum_{i \in Q} AP_i \tag{1.4}$$

F-measure F-measure presented in Equation 1.5 also considers both precision and recall providing a single measurement for a system. The importance of the precision or the recall can be adjusted through a β parameter, where β values smaller than 1 prioritize precision, and those over 1 give more relevance to recall. If not β is specified, the intermediate case of $\beta = 1$ is considered.

$$r(k) = (1 + \beta^2) \frac{p(k)r(k)}{(\beta^2 p(k)) + r(k)} \tag{1.5}$$

Object detection

The goal of the *object detection* problem is to determine if an object of a certain category is represented in an image and, if so, provide a rough estimation of its location. The location is expressed through the bounding box and a detection is considered correct if the bounding boxes of the detected object and the annotated ground truth overlap on a ratio higher of 0.5 when compared to the union of two boxes. Figure 1.7 shows an example of both a correct and incorrect detection. This criterion is known as the Pascal criterion, as it is the one used in the Pascal Visual Object Classes Challenge [24].

In our work, the Pascal Criterion has only been assessed on those images which are known to contain at least one instance of an object of the same category as the query. This means that if an image contains multiple objects of the query category, any of them would be a good match for the detection. Notice that during the evaluation of the object detection, only those images annotated as belonging to the same class as the query are considered. The evaluation at the global scale between relevant and non-relevant images is already assessed when measuring results in terms of image retrieval.

Figure 1.8: Correct detections but inexact segmentations.

Object segmentation

The object detection based on the Pascal Criterion considers a bounding box as the region of support for the objects. Although in many application this degree of accuracy may be sufficient, some others may require the extraction of the object with a precise segmentation from the background. The evaluation of such requirement needs a more accurate measure that will consider what pixels of the detected object actually correspond to the object ground truth. For example, the cases shown in Figure 1.8 present good results in terms of object detection, but inexact segmentations of the bottles.

There exist multiple methods to evaluate the quality of the segmentations, but this work considers the quality at the pixel scale. Given the region of the detected object and a ground truth mask, four possibilities may occur: detected pixels that correspond to the ground truth mask will be true positives, while those that do not correspond are false positives; on the other hand, those pixels considered as background which actually belong to the object are false negatives, while the correct ones are true negatives. These measures can be used to compute the precision and recall of the object segmentation, and can also be combined in a single measure known as the *Jaccard Similarity Index*. Equation 1.6 presents its formulation, where A represented the set of pixels of the detected object and B the set of pixels of the ground truth.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1.6}$$

Notice that the presented problem is very similar to the one presented in the image retrieval case. The only difference is in the terminology, where *relevant* documents are now the object pixels in the ground truth mask, while *retrieved* documents are now those pixels that belong to the masks of the segmented objects.

Analogously to the assessment of object detections only on relevant images, the quality of the segmentation will only be measured in this work on the correct detections. This approach follows the criterion adopted in this thesis, where the provided measures must be considered incremental with respect to the previous one.

1.3.4 Data Partition and Cross Validation

The evaluation of a pattern recognition system requires a previous processing of the database by dividing it in two sets: the training and the test set. The training dataset can be used to tune the classification system, while the evaluation metrics are performed only on the labels predicted for the test dataset.

In some of the retrieval scenarios considered in this thesis it is not necessary to define a training set, because there is no learning from the system. However, the division between training and test has been kept for coherency with the rest of the experiments that do require a training set. This way, there cannot be any bias when comparing the results, as the test dataset is identical for each configuration.

It is a common practice to generate different partitions between training and test to run the experiment several times and finally average the results obtained. This option is called cross-validation. Whenever a new division is created, a new trial is defined. The final results are provided by averaging the obtained results and computing the mean and standard deviation of such results.

The results presented in this thesis have been generated by averaging the results of 5 cross validation trials. In each trial, the complete dataset has been randomly split in two equal parts, so the size of the training and test sets are equal.

1.4 Solution Overview

This thesis follows the structure of a classic pattern recognition problem: the input data is pre-processed to map it into a feature space where the actual analysis will be performed. This pre-process generates feature vectors that contains a condensed version of every data object, in our case, images. The data transformation must synthesize the original data, but at the same time keep the information relevant to solve the posterior analysis problem. In this work, the data pre-processing is presented in the first two parts.

Part I presents how images and objects are segmented to define parts. This process allows the local analysis of the data using a completely automatic process that generates a hierarchy of regions. The hierarchical image representation presents several opportunities for object retrieval and the overall contribution of this work is how to exploit these data structures with the state of the art techniques for object retrieval. In this part, different strategies for the interactive segmentation of objects are presented in the framework of partition trees.

Part II focuses on the visual features that are extracted from each of image part. Firstly, an overview of the classic region-based features is provided with special attention to the visual descriptors proposed by the MPEG-7 standard. This work studies how these features can be used to define visual codebooks and how these codebooks offer a chance to solve of some of the

problems introduced during the creation of the hierarchies of regions. The main contribution of this part is the novel concept of Hierarchical Bag of Regions, a novel type of features that represents a subtree through a bottom-up expansion of the soft mapping onto a visual codebook.

Part III applies the proposed Hierarchical Bag of Regions in the main tasks targeted by this thesis. The *query by example* problem is formulated as a matching between the hierarchy of regions of the example object and the hierarchies of regions of the images in the target database. The presented solution considers both a top-down fast matching as well as an exhaustive but more costly bottom-up approach. The *query by concept* case is formulated as a part-detection problem, where the object classes are modelled from their parts. The detection algorithm finds candidates for each part in a first stage and later combines them to construct the complete object.

This thesis ends with final conclusions on the opportunities of hierarchies of regions in the field of image retrieval in classification. The reader must be aware that this work does not include a specific chapter for the state of the art because, given the diversity of topics considered, each chapter includes its own references according to its content.

Part I

Image and Object Representation

Summary of Part I

Whenever developing an analysis solution, the first choice to take is the basic work unit that will be processed. Although the input data unit considered in this thesis is an image, its visual and semantic heterogeneity is not suitable for an object analysis. Part I of this thesis presents the adopted solution for decreasing the global default scale offered by the images to a more local one.

Chapter 2 introduces a region-based multiscale representation of the image under the form of a *Partition Tree*. This representation can be automatically generated through an iterative algorithm applied on an initial segmentation of the image. This process is conducted by an automatic perceptual analysis of the pixels, whose result may or may not match the semantic contents in the image.

The introduction of semantic information is discussed in Chapter 3, where the user is introduced in the loop to provide the human interpretation of the input data. The user interaction is aimed at defining what parts of the image represent the objects through an annotation process. This interactive segmentation can benefit from the hierarchical image partition presented in Chapter 2. This chapter discusses how different markers (points, bounding boxes and scribbles) are mapped into the Partition Tree and how the user can used this image representation to navigate through multiple local scales in the image.

The results of user interaction on the Partition Tree is analysed in Chapter 4 to build a hierarchical representation of the objects under the form of *Object Trees*. Such structures are used in future chapters to formulate the queries and generate annotations that will be compared with a test dataset of Partition Trees.

Chapter 2

Hierarchical Image Partitions

The input data of the proposed system are images. Analysing the complete image to detect the objects it contains is a very challenging approach, as the pixels representing the objects will be jointly considered with those associated to the background. For this reason, this chapter presents a solution to study the image as a collection of parts.

2.1 Image Parts

There exist different options that consider images as a collection of parts. The different solutions vary depending on the required processing effort and the precision obtained if compared with the actual semantic partition.

A first option is to apply an arbitrary partition of the image independent from the content. This partition simplifies the feature extraction process as there is no need to firstly decide which portions of the image are to be considered. On the other hand, this simplicity also requires considering several localizations where no semantics are represented, which generates a useless computation overhead. For example, sliding windows are a popular case of this strategy, normally applied at multiple scales. The broadly adopted Viola-Jones object and face detectors [97] prove that this approach, despite this drawback, can be efficiently implemented and used in real-time applications.

A second and very popular approach at the time of writing this thesis is defining regions of interest around certain interest points (or keypoints). These are precisely located in the image and their surrounding semantics are robust to local and global perturbations in the image, such as changes of perspective, location and scale. The neighbourhood of the points is considered to extract a visual feature, such as the broadly adopted SIFT [56] or SURF [5] descriptors. These visual features capture the spatial frequencies around each of these points. One of their advantages is the simplicity of the metric that assesses their similarity: the cosine or Euclidean distances.

However, both sliding windows and interest points miss capturing a very important cue for the local analysis: shape. In both cases the local analysis will indifferently consider together pixels from the object and the background. This mixture may cause problems when learning and classifying a certain pattern. Note that, while the object pixels are supposed to be informative for the model, using the background pixels will introduce perceptual data to the model that is not really representative of the object. This problem can be solved with a third method for defining image parts: segments.

Segments are groups of pixels in an image which are spatially connected. There exist several image processing algorithms capable of defining segments that present a certain visual homogeneity in two basic families of segmentation algorithms: contour- and region-based [14] [77] ones. Contour-based algorithms define regions by firstly detecting the high frequency edges in an image and use them to define segments. Region-based algorithms start finding the homogeneous areas of an image and make them grow until a certain stop criterion is reached.

When a set of regions cover the complete image, this set is called a *partition*. A partition is the type of image representation that results from a *segmentation* process. There are multiple algorithms to perform this operation and their goal is always to group pixels in regions according to certain criteria, normally based on the relations of color and neighbouring. A good segmentation algorithm will define a partition whose region boundaries correspond to the different semantic entities represented in the image. If this is true, the subsequent analysis algorithms can focus on these regions.

One basic limitation of considering a single image partition for its semantic analysis comes from the diversity of scales where the semantics can be present. Not only in many cases semantics are represented at the global or local scale, but in several situations semantic entities contain other semantic entities. For example, composite visual objects such as *people* have clearly separate visual parts with their own semantics, such as *head* and *body*, and each of these parts could be further decomposed semantically as *face* and *hair* for the *head* or *trunk* and *legs* for the *body*. If each of these semantic entities is to be represented by a segment in the image, it is not enough to consider a single partition at a given scale, multiple scales must be analysed.

One option for capturing the multi-scale diversity is to apply the same segmentation algorithm on the same image with different tunings. Ravinovich et al [69] worked with different segmentations of the image by applying the normalized cuts algorithm [79] with different parameters. The set of segmentations obtained were evaluated in terms of stability in front of perturbations and only those segmentations with high stability were considered in the further stages of the analysis. Another option is to directly apply different segmentations algorithms on the same image, as proposed by Vieux et al in [94]. The main drawback of these solutions are in terms of efficiency because, although dealing with several segmentation increases the

chances of capturing the semantic contents, it will also generate several segmentations that will not be used or that will be redundant with each other. A typical segmentation algorithm is highly demanding in terms of computation, so it is advisable to use it with special care.

The concept of *hierarchical partitions* provides a compromise between computation and multi-scale semantics. This type of image representation defines a set of segments based on an initial partition at the fine scale. The segments in this partition are iteratively merged with other neighboring segments to define new segments at larger scales. The creation of such hierarchy ends when all regions have been merged into a single one that represents the complete image. The present work is based on this type of segmentations and, for this reason, they are carefully analysed in the following section.

2.2 Region-based Hierarchical Partitions

Our framework considers a set of regions R defined as combinations among the elements of an initial set of regions R^0. R^0 is defined with an image partition and is also contained in R. The combinations are generated by iteratively merging two or more neighbouring regions which have not been previously merged. All these combinations can be represented by a graph, by assigning a vertex to each region in R and edges connecting each region with their composing ones. In fact, the resulting graph is a tree, as it satisfies the condition that there exists a single path between all possible pairs of nodes.

Partition Tree A Partition Tree (PT) is defined by a pair $T = (R, E)$, where R is the regions set, E is the set of directed edges from the composing regions to the resulting union.

Partition Trees are a specific case of an Attributed Relational Graph (ARG) [91] in which the edge represents the topographic attribute *inclusion*.

The generation of PTs defines certain relations among the regions defined in the set R. These relations will be referred in this work following a terminology based on *family* links. So, when a region b is defined as the union of another region a with other regions, region a is told to be a *child* of region b in the PT and belong to the set of b's children, C^b. On the other hand, region b can also be referred as the *parent* of region a which, in the context of a tree, is always unique. The parent of a region r will be referred as P^r. In the PT example of Figure 2.1, the nodes representing the *hair* and *face* of the anchorwoman are the children of the *head* node, so the *head* node is the parent of the *hair* and *face* nodes.

The concepts of parents and children can be further bottom-up or top-down extended through the PT to define two new terms: *ancestors* and *descendants*. Given a region r in the PT, the vertices above define the set of its *ancestors* (A^r) while the elements contained in the

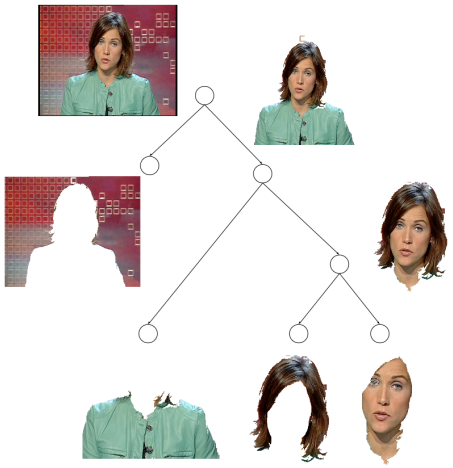


Figure 2.1: Binary Partition Tree

subtree below are called its *descendants* (D^r). The union of ancestors and descendants is the region's *family* (F^r).

2.3 Binary Partition Trees

The *Binary Partition Trees (BPT)* is the specific type of Partition Tree whose merged nodes have two, and only two, children. It was proposed by Garrido and Salembier [75] as a tool for region-based image representation. The present work uses this image segmentation as the input data for every presented technique, so the BPT can be considered the basic foundation over which this thesis is developed.

The BPT simplifies the creation of the PT as it does not need to estimate how many regions are to be merged at every iteration. The only decision to be taken at every step of its creation is which of the two neighbouring regions will be fused. The used implementation is based on a criteria that takes into account the color similarity between the two neighboring regions and the complexity of their contour [96].

The amount of nodes in the BPT, $|R|$, can be directly computed from the amount of regions in the initial partition $|R_0|$, as presented in Equation 2.1.

$$|R| = 2|R_0| - 1 \tag{2.1}$$

Figure 2.1 shows an example of a BPT defined on an initial partition of four regions. Their iterative merging defines three additional regions. In this example, the nodes in the initial partition represent the *background*, *body*, *hair* and *face* semantic entities. The merges in the tree define the *head*, *anchorwoman* and *full image* nodes.

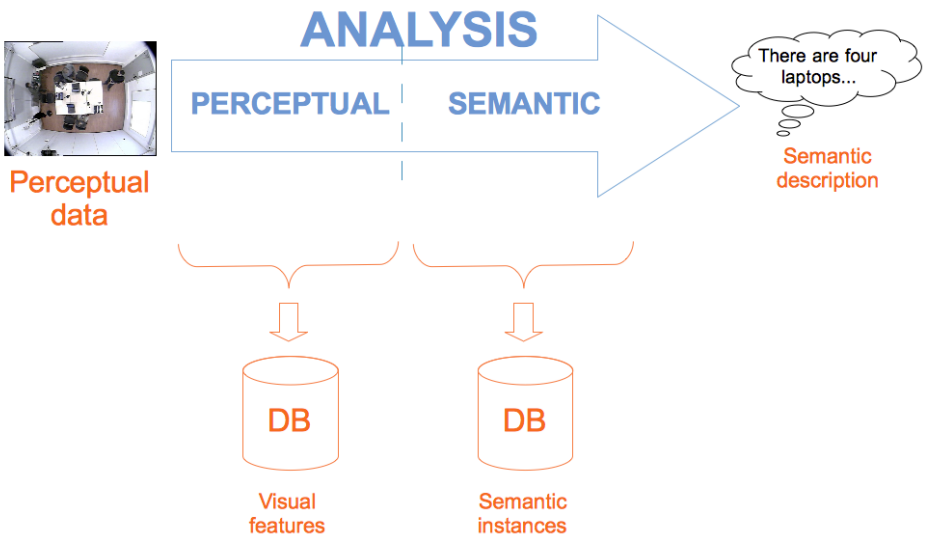


Figure 2.2: Perceptual and Semantic stages of the image analysis.

2.4 Limitations of Partition Trees

The definition of image segments is nowadays a costly process in terms of computation. This limitation forces a careful usage of such techniques. In the context of object retrieval from image databases, the response time of the system is normally an critical factor. In order to speed up the search process, the image segmentation and posterior feature extraction must be performed during the ingestion of the images in the database, that is, before any user query is processed. Given the diversity of semantic entities that could be contained in a general-purpose image database, the segmentation and feature extraction algorithms cannot be optimized for any specific semantic class and must be generic for any type of object. The most common architecture divides image analysis in two different stages: a first and generic one based on the perceptual (visual) characteristics, and a second one that implies a semantic interpretation of the image parts dependent on the object of interest, as displayed in Figure 2.2.

As no previous knowledge of the object of interest is available at segmentation time, the criteria that drive this process must fully rely on perceptual judgements. Defining image parts based only on perceptual criteria may lead to segments that do not correspond to the semantic objects in the image. Semantic segments do not necessary imply homogeneity from the visual point of view and, if this is not the case, it is difficult for a perceptual and generic segmentation algorithm to correctly define the contours of the semantic objects. Given an object of interest, hierarchical partitions can drive to three different situations: object match, object split or a merge among the object and background parts.

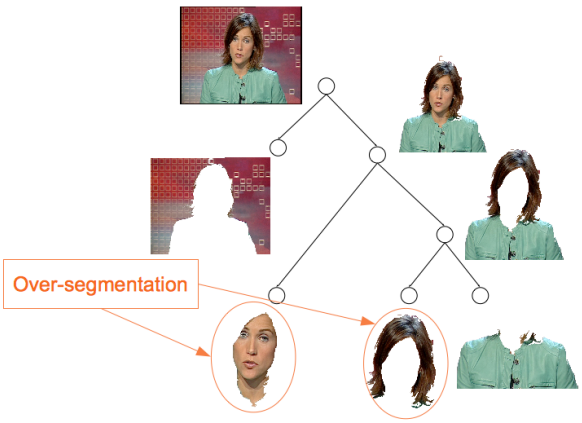


Figure 2.3: Split problem in a BPT.

2.4.1 Object Match

The ideal case corresponds to that one where one region in the initial partition R_0 exactly corresponds to the object of interest. For example, this situation normally occurs when the object presents very homogeneous visual properties which are, at the same time, very heterogeneous if compared with the background features.

However, in several cases, semantic objects are not visually homogeneous and they are decomposed in multiple image parts from the initial partition R_0. This situation corresponds to an *over-segmentation* of the object in the initial partition R_0. Partition Trees can overcome this problem if, during their creation, these object parts are merged to define a region that precisely represents the semantic object.

2.4.2 Object Split

The over-segmentation problem will only be solved if the PT fuses all object parts into a single node. This is not guaranteed, and it is also possible that the decision criteria merges a part of the object with a part of the background. As a result, the object will appear split in different subtrees and the full semantic object will not be represented by any node in the PT. For example, Figure 2.3 describes a case in which the semantic object *head* is not represented by any of the BPT nodes, but the BPT correctly combines the over-segmented parts of the *anchor* in a single BPT node.

2.4.3 Object and Background Merge

Opposite to the over-segmentation case, the initial partition can present an under-segmentation problem. In this situation the semantic parts of the object appear in R_0 already fused with a part of the background. This scenario is more difficult to handle than the over-segmentation,

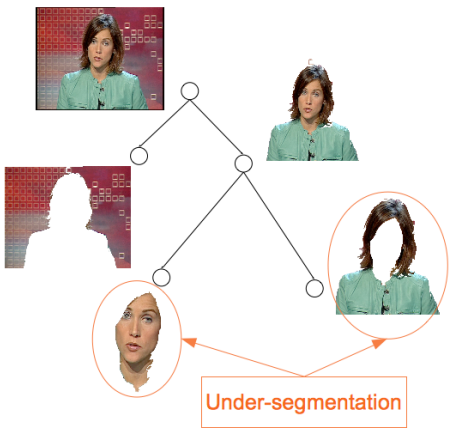


Figure 2.4: Merge problem in a BPT.

as the visual features of the object parts will unavoidably be fused with those belonging to the background.

The PT does not provide any solution for this problem, but depending on the nature of the visual features, a partial detection of the object is achievable. Figure 2.4 shows an example where the semantic object *head* is not represented by any node in the BPT nor by any combination of them because the *hair* part has been merged with the *body* part at the initial partition.

2.5 Summary

This Chapter has presented the adopted image representation, a hierarchy of regions in the form of a Binary Partition Tree (BPT). Such structure is generated through an automatic segmentation process that provides the basic units for latter analysis. The hierarchical nature of the BPT naturally allows a multiscale analysis of the image.

Despite its multiple qualities, the perceptual criteria used during the creation of the BPT may not match the semantic interpretation of the image. In these cases, the objects might never be represented by a single region in the BPT, but by a set of them. The main contributions of this thesis aim at solving the analysis problems that these situations present. Part II proposes novel features that exploit the hierarchical nature of the BPT, while Part III presents efficient analysis strategies for BPTs. Before these two main parts, the following Chapters 3 and 4 explain how BPTs are combined with user interaction to generate valuable data for the semantic analysis of the image.

Chapter 3

Interactive Segmentation

The content-based retrieval of objects requires the definition of the query in visual terms. This thesis assigns this role to a human user, that must somehow express the mental query through a visual representation. This information could be provided synthetically by formulating the query by a sketch [44] [23] [82] or choosing from a predetermined palette of patterns [27]. Another option, the one considered in this work, is to outline the query object in an existing image [41].

The delimitation of an object in an image can be a difficult task for a human depending on the visual characteristics of the object. Basic graphical user interfaces offer simple markers and shapes to indicate the boundaries of the object in an image; such as points, lines, rectangles or polygons [73] [88]. However, these rough annotations may introduce important errors with respect to the actual boundaries of the object if such boundaries do not exactly match the shape pattern of the marker. This problem is addressed by the *interactive segmentation* techniques [58], that support the segmentation effort of the user with image processing techniques.

Systems offering precise local annotations can be classified into region-based or contour-based approaches. Region-based annotations [67] [63] let the user select among a set of segments from an automatically generated partition of the image, while contour-based solutions [71] [101] aim at generating a curve that adjusts to the pixels located at the border between object and background. Four methodologies based on the region-based family are proposed to interactively generate a segmentation of the instance. In all of them, the success of the interaction is tightly dependent on the goodness of the segmentation.

The considered solutions are based on the three types of markers presented in Figure 3.1: points, bounding boxes and scribbles. State of the art human-computer interfaces, such as mouses or touch screens, can easily capture these markers on images.

This chapter focuses on the exploitation of these markers together with hierarchical partitions, which corresponds to the first contribution of this thesis. The details of their integration

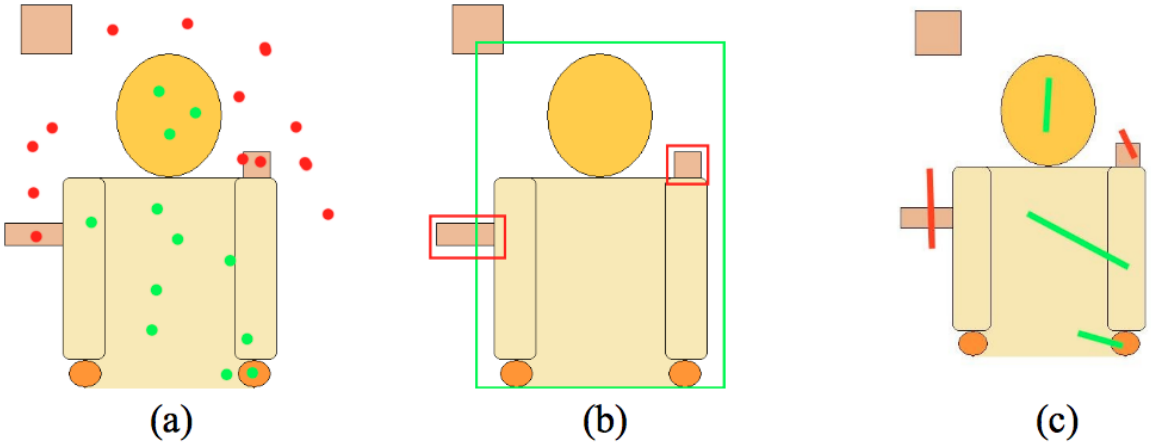


Figure 3.1: Three types of markers: a) Points, b) Bounding boxes and c) Scribbles.

in a software tool and system architecture are included as annexes in Chapters 10 and 11, which are extracts from publications [30] [32], respectively.

3.1 Partition-based techniques

3.1.1 Point

The most basic option for interactive segmentation on a partition is the usage of point markers. These are generated with a click on the image, which is mapped into a region of the initial partition. This action will swap the state of the region from selected to unselected, or viceversa. The main drawback of this proposal is that, on oversegmented images, the user might need to click several times before completing the whole selection of the object. Nevertheless, this option has been proved very intuitive for users and can also be a good complement to another of the presented techniques.

3.1.2 Bounding Boxes

This selection scheme requires the user to draw a bounding box around the instance, so that the algorithm will automatically select the regions from the partition which are included in the rectangle. After the initial selection, the user can toggle the selected regions by clicking on the image. Every click on the image defines a point marker, indicating the region in the partition whose state will be switched. Figure 3.2 contains a screenshot of the interface in the case of interactive segmentation with a rectangle marker. This strategy has been proved as very intuitive for users, who are very familiar with drawing rectangles and clicking.

The main challenge of this technique is to decide which regions from the initial partition are to be selected by using the box marker. Figure 3.3 shows an example of different approaches

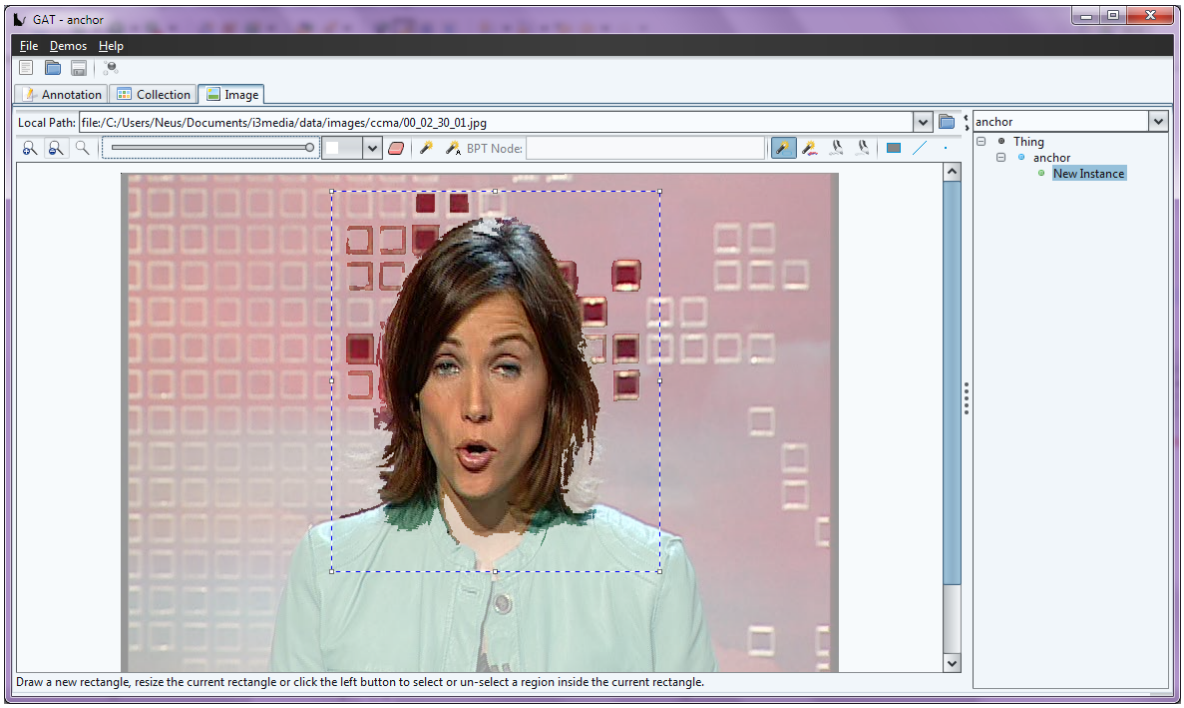


Figure 3.2: Rectangle marker and selected regions.

applied on a challenging example of the segmentation of a bottle on a complex background. The rectangular marker has been taken from the manual object annotation provided by the ETHZ dataset [26]. The first solution, marked as a), shows the selection when considering those regions which are completely included in the rectangle. The result in this case is not very promising because many parts of the object have not been selected and a portion of the background has been because it is completely included in the rectangle.

The missing regions can be selected by introducing two modifications to the initial selection by bounding box. The first one is an automatic expansion of the bounding box. Figure 3.3 b) shows the results of automatically increasing the box dimensions a 0.01% . The second solution is a tolerance of pixels outside the selection rectangle for the region. Figure 3.3 c) shows the result of repeating the experiment with a tolerance of 5%. These strategies have been applied in the ETHZ dataset to generate a valid ground truth for the experiments run in this thesis. The arguments and details are described in Chapter 12 in the Annexes.

3.1.3 Masks

The usage of masks for interactive application is not a common practice, but they are often provided in evaluation campaigns to define the ground truth objects in a dataset. For example, these data were provided in the Instance Search task of the TRECVID 2010 campaign [66] or in the several editions of the Pascal Video Object Challenge. Despite not being really an

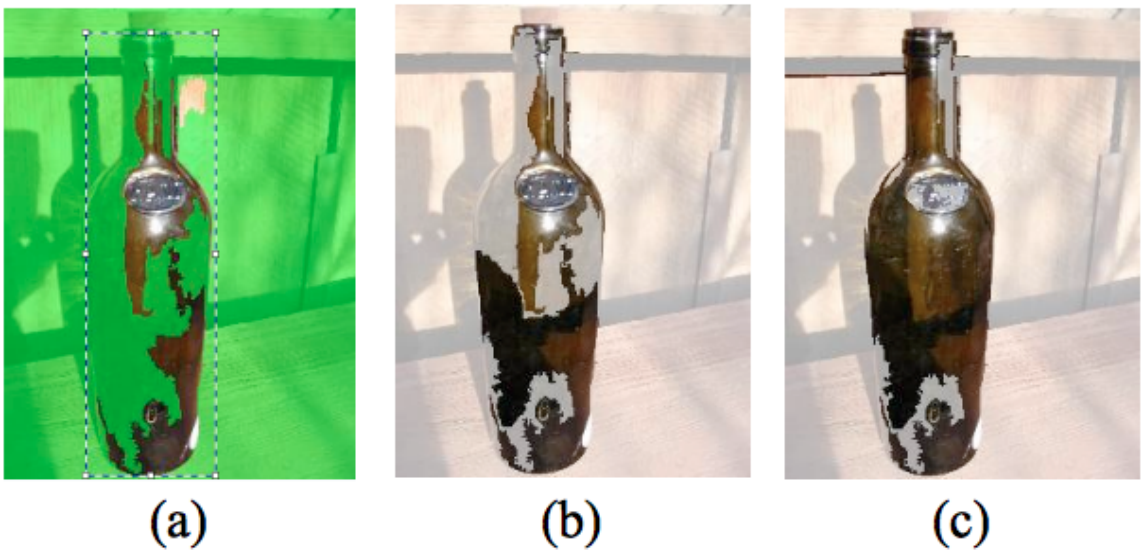


Figure 3.3: a) Completely included, b) Expansion, c) Expansion & tolerance.

interactive scenario, this situation is discussed in this section due to its similarity to the rest of presented cases. The problem is basically how to map the provided object mask into a set of regions in the partition.

A first approach is selecting any region which overlaps with the mask in one or more pixels. Empirical experiments with the TRECVID 2010 dataset [66] show that this is not a good strategy because the provided masks are often in contact with many regions from the background, as shown in Figure 3.4.

In order to avoid this over-selection of regions, the opposite solution would be to only select those regions in the partition which are completely included in the mask. Figure 3.5 proves that this solution is too drastic because now too many parts from the object are missing. The reason is exactly the same as the previous case, many foreground regions may also include a small portion in the ground truth background.

Given that selecting all regions with a minimal overlap of one pixel produces too many selections, but requiring a full overlap drives to too few selections, the proposed solution defines a new parameter for the algorithm to determine the minimum portion of a region overlapping the mask to be selected. This way the compromise between under-selection and over-selection can be leveraged. Figure 3.6 includes the results obtained if the minimum proportion of pixels in an image part to be selected must be 0.8.

3.1.4 Bounding Box and Mask markers on the ETHZ dataset

The annotation of objects in the dataset ETHZ introduced in Section 1.3.2 is provided under the form of both bounding box and mask markers. These data have been used to build a

Figure 3.4: Overselection due to including any region that intersects with the mask.

Figure 3.5: Underselection due to only including the regions completely overlapped the mask.

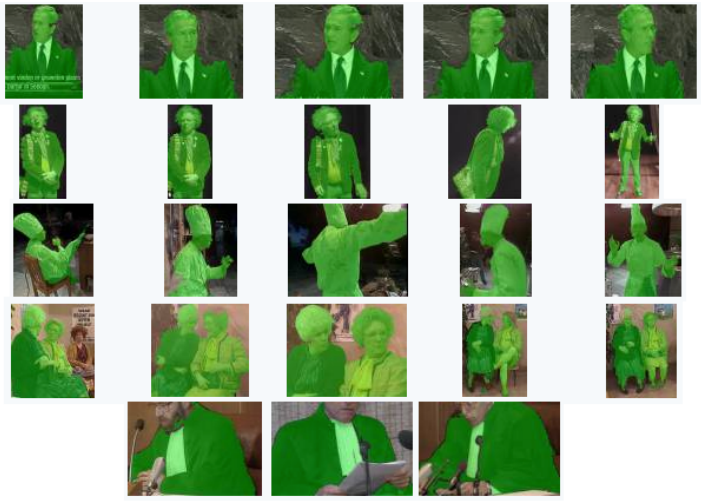


Figure 3.6: Better selection considering regions with more than 80 % overlapping with the mask.

ground truth of object annotations on image partitions. These ground truth data has been used all through this thesis to evaluate the proposed solutions so, instead of comparing results on the pixel-wise annotations provided in the dataset, retrieved objects are compared with the ground truth objects defined on the same image partition. This way, the provided results are not distorted by the problems that might be introduced during the automatic segmentation of the images. The scope of this thesis is not the generation of accurate object segmentations but, given a hierarchical partition, use it as a data support for object retrieval.

The quality of newly generated ground truth has been studied by computing the Jaccard Index and the bounding box occupation between the provided ground truth and each type of partition masks: the one generated by mapping the ground truth mask and the one generated with the ground truth bounding box. The results shown in Figure 3.7 clearly state that those objects with a larger occupation in the bounding box (apples, bottles, mugs) offer a better accuracy than those more complex objects, such as the giraffes and the swans.

The generation of ground truth partition annotations from the bounding boxes and masks provided in the ETHZ dataset required an adaptation process. The applied modifications and detailed description of the resulting dataset is described in Chapter 12 contained in the Annexes.

3.2 Partition Tree-based techniques

The interactive segmentation techniques presented in Section 3.1 make use of an initial partition of the image. The three techniques presented in this Section take advantage of the hierarchical region-based structure represented in a Partition Tree (PT).

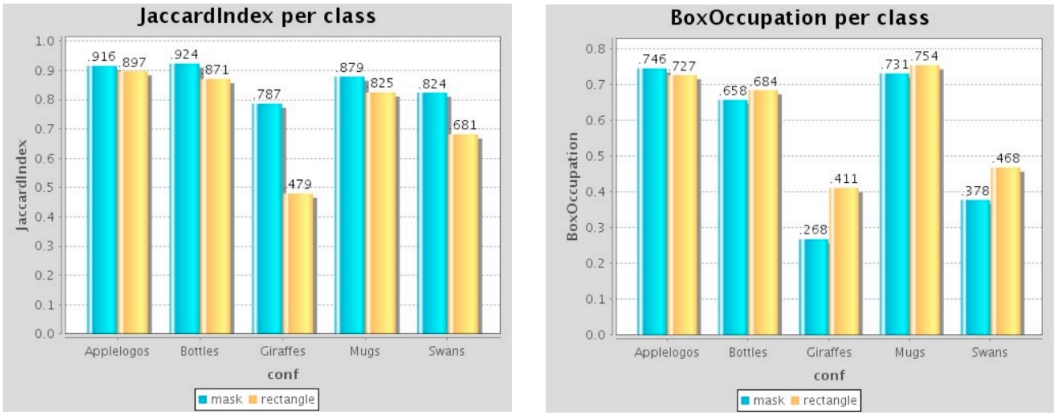


Figure 3.7: Jaccard index and box occupation for every concept.

3.2.1 Label Propagation

The first application of PTs for interactive segmentations is the propagation of labels through its structure. In this case, the user interaction requires drawing scribbles on the image specifying if these markers refer to the object or on the background. Every time a new scribble is added, the label is assigned to the PT leaves underneath and may be expanded through the tree structure. There exist different approaches for this task.

The previous works on the topic required an initial labelling of both *foreground* and *background* regions. The original work by Garrido and Salembier [76] expanded each *foreground* label to its ancestors as long as none of the ancestors's descendants was labelled as *background*. The original implementation was improved by Adamek [2] with a most efficient algorithm. However, both solutions required the two types of labels and, so, at least two user interactions.

This thesis proposes a simpler propagation rule: expand the *foreground* labels to its parent only if the sub-tree defined by the sibling contains one or more *foreground* labels and no *background* labels. The main advantage of this approach is that it only requires one type of user interaction, leaving the *background* ones as optional to correct possible mistakes during the propagation of the *foreground* ones. If it is necessary to obtain labels for the complete image, as in the frameworks considered in [76] and [2], all unlabelled regions may be considered *foreground*.

The example presented in 3.8 uses a single object scribble (green) to selection two leaves from a BPT: the *body* and *face* of the *anchorwoman*. The node representing the *hair* is automatically selected because the subtree defined by the *body* node's sibling contains the *face* node.

Similarly to the rectangle and points scheme, the selection can be refined through successive iterations adding background scribbles that will unselect the partition regions under the scribble. Figure 3.9 shows a first step (a) where an object scribble (green) is drawn over a face.

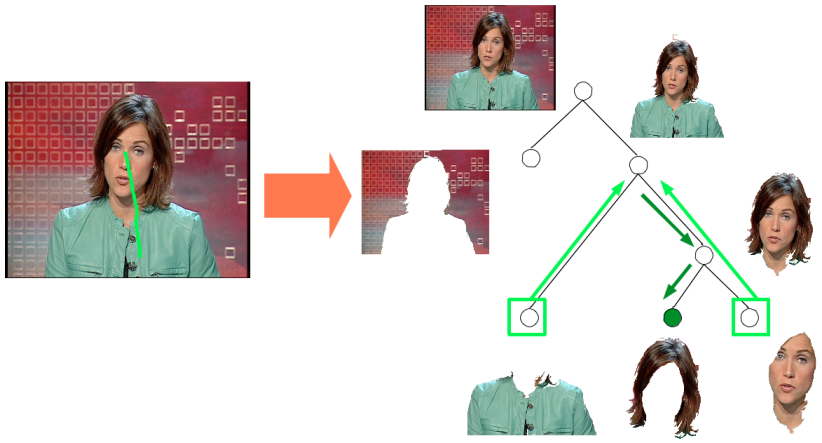


Figure 3.8: Automatic expansion through PT.

(a) (b) (c)

Figure 3.9: Sequential segmentation with scribbles.

Step (b) shows how the label propagation has erroneously selected some regions belonging to the background, so a background (red) scribble is drawn over them to finally obtain a better segmentation in step (c).

3.2.2 Navigation

PT structures can also be used as a navigation path for the user. An initial selection of a branch determines a set of regions that can be intuitively explored by the user as an expansion or contraction of the currently selected segment. Two modes of the interaction mechanism have been defined, depending on if an initial click is needed to start navigation.

The *clickable* mode starts with a left-click on the area of support of the object. With this action, the user is implicitly selecting one branch from the PT, as every pixel in the image corresponds to one, and only one, branch in the PT. After this first user interaction, the interface highlights the region associated to the PT leaf so that the user can evaluate if the proposed region correctly depicts the object. If this is not the case, the selected PT node can be modified by rotating the mouse wheel, moving upwards or downwards in the branch at every wheel rotation. Every new move will expand or contract the selection depending on the direction of the rotation. Figure 3.10 shows the selected regions associated to each

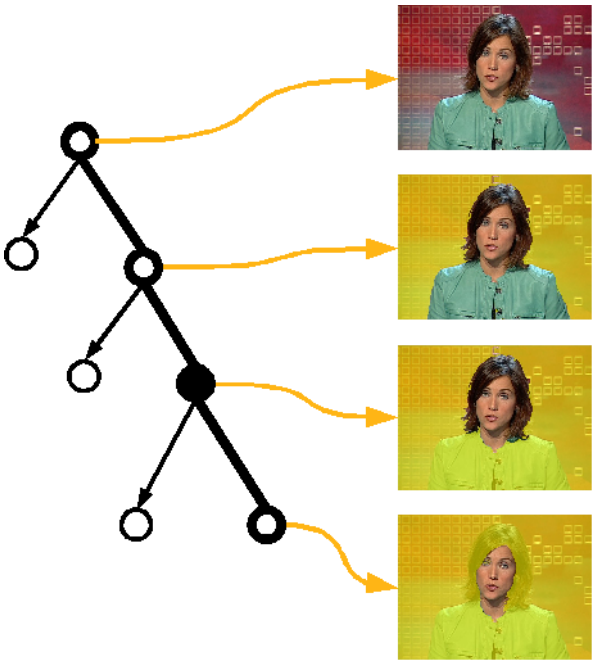


Figure 3.10: Mouse wheel navigation through a BPT branch.

node in the PT, which are shown on the graphical interface as a transparent segment over a semitransparent mask.

The selection path is defined between the PT root, where the whole image is selected, and a PT leaf, where a region at the initial partition is shown. A second left-click will save the currently selected node and allow choosing regions from other PT branches before the final validation with a right click. Notice that this scheme allows the local annotation of non-connected components. The sense of the rotation on the wheel determines whether next selection corresponds to the parent or child node. This is typically the case when the system proposes an object at a certain spatial scale but the user wants to analyze the image at a different scale (e.g.: the system selects the head while the user is willing to annotate the face or the complete TV anchor).

The *clickless* mode is based on the same principles as the clickable case but it requires less interaction from the user side. The multi-scale navigation and multiple branch selection are shared features among them, but in this mode PT branches are selected by just placing the cursor over a region, with no need of an initial click. This means that, whenever the cursor is over the image panel, a region is highlighted. Some users have reported that this option too confusing due to the high activity on the panel.

3.3 Summary

This chapter has presented the opportunities offered by the Binary Partition Trees in terms of interactive segmentation. This process introduces the user in the analysis loops, as it can indicate the precise location of the objects of interest. The generated segmentation of the object is exploited by the system to formulate the user query in the instance search problem.

Traditional markers available in most graphical user interfaces such as points, rectangles and scribbles are mapped into leaf or intermediate nodes of the BPTs. Special attention has been given to the mapping of masks into BPTs, as it is a common practice in retrieval benchmarks to provide the ground truth data in this form. This is the case of one of the datasets considered througout this thesis, the ETHZ dataset, whose ground truth masks have been mapped into nodes of the generated BPTs. Further experiments will always be referred to the BPT mappings in order to minimize the influence of inaccurate partitions during the assessment of the techniques presented in this thesis.

The following Chapter 4 explains how the manual segmentation of the object is used to generate a hierarchical decomposition of it. The resulting hierarchy will be used in Parts II and III for the automatic visual analysis of BPTs.

Chapter 4

Object Tree

In the instance search problem, the input data is the query region that describes the object of interest for the user. The particularity of this type of queries is that the content corresponds to a local segment of the image query. In the context of hierarchical partitions considered in this work, a local segment of an image corresponds to a set of regions on a Binary Partition Tree (BPT). As a consequence, the input data is not only a region but the hierarchy of regions defined by the BPT nodes selected during the query. The user operation for selecting these regions is called *interactive segmentation* and it has been explored in Chapter 3.

The query object might exactly correspond to one single node in the target BPT but, there is no guarantee of that and, often, this will not be the case due to the split problem. So, in general, whenever a user formulates the query, a set of BPT nodes will be selected. In the present work, a further restriction will be imposed, as it is considered that all these BPT nodes are adjacent. For this reason, the presented techniques have only been explored for the case of connected components. Figure 4.1 shows a visual selection of a connected object that it is mapped into three different subtrees in the target BPT.

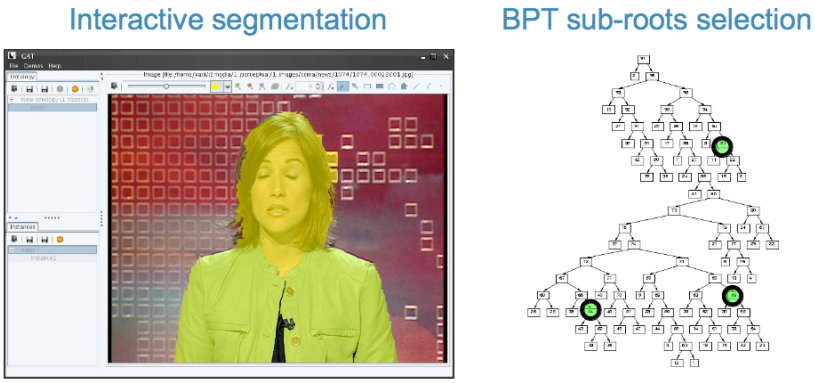


Figure 4.1: Selection of sub-BPT roots.

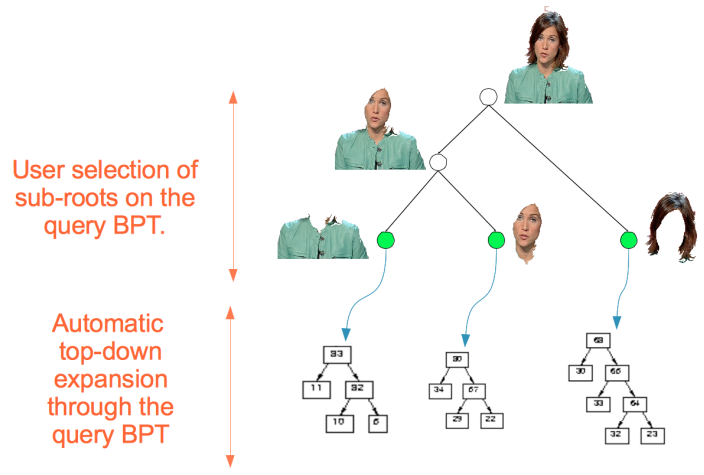


Figure 4.2: Object Tree.

The manually selected sub-BPT roots are the input data for the definition of the *Object Tree (OT)*, the proposed data model to handle the object query in the context of a hierarchy of regions. The OT is generated throughout two stages that take into consideration both the manual user interaction and the automatic segmentation algorithm used for BPT creation. The result is a hierarchy of regions whose nodes represent the parts and sub-parts into which the query is visually decomposed.

The proposed *Object Tree* framework is similar to the data model of Jaimes and Chang [41] made in the framework of learning visual detectors as a multilevel problem, organized in a *Definition Hierarchy*. Figure 4.3 shows the four levels into which the object was decomposed. An *object* was modelled by a structured set of adjoining *object-parts* which, at the same time, could be divided again into *perceptual-areas*. The main difference between *object-parts* and *perceptual-areas* was that the first ones were related to a higher-level semantic interpretation (eg. grass) while the latter referred to a low-level perceptual category (eg. green). The *Definition Hierarchy* included an additional *scene* level above the *object* level that considered a structured set of objects. This top level corresponded to the *global* scale of the complete image. Users of this system were requested to manually create a *definition hierarchy* before starting the annotation task that would generate a training dataset for a multi-level *Visual Object Detector*.

The main difference between the *Definition Hierarchy* and the proposed *Object Tree* is the fusion of the *object-part* and *perceptual-part* levels. In the OT solution, the *region* level corresponds to the regions in the initial partition. These *regions* are automatically combined to define BPT nodes, that would correspond to the *perceptual-object-parts*. In the OT framework, the user does not need to define any hierarchy, only select the complete object. The underlying BPT structure is exploited to define the *perceptual-object-parts* that, in our case, are simply referred as *Object Parts*. These *Object Parts* may or may not have a semantic

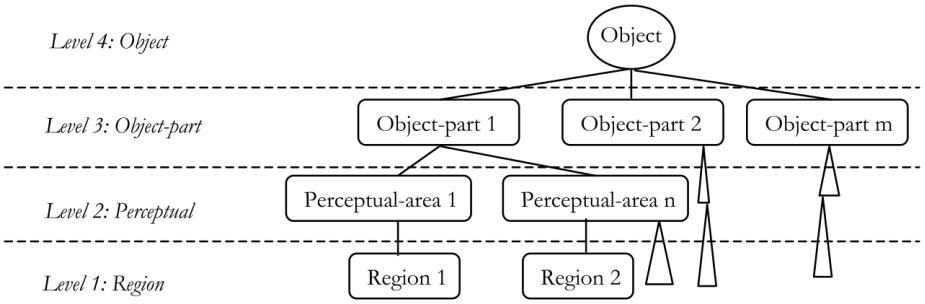


Figure 4.3: Definition Hierarchy proposed by Jaimes and Chang (2001).

meaning, a property that is irrelevant for the later analysis.

The construction of the Query Object is performed in two stages: one considering the user-selected sub-BPT roots, and one taking into account the structure of BPT from the query image.

In the first stage, the manually selected sub-BPT roots are sorted according to their size and adjacency. The algorithm starts by finding the node associated to the largest region and defines from it a leaf node for the OT. At this point, this single node also defines the OT root. Afterwards, an iterative algorithm finds the largest of the remaining sub-BPT roots that is connected to the OT root. When determined, a new node in the OT is created by merging the previous root node and the new one. By doing so, a new root is defined as the union of regions associated to the previous root and the new node. The algorithm is run repeatedly until no more sub-BPT roots remain to be merged, or until a manually set threshold of maximum amount of query parts is reached.

The second stage for the creation of the OT considers the underlying BPT defined over the query image. The hierarchical structure of the selected sub-BPTs can be replicated on the OT to determine more sub-parts for the query. Figure 4.4 describes how the three nodes manually selected in green on the query BPT become the leaves of the OT during the first stage of its creation and are later automatically expanded through the image BPT.

The mechanism by which the BPT is constructed may generate several small regions which are not very relevant in visual terms, but whose consideration might have a significant impact to the search algorithm in terms of required computation effort. The impact of these tiny regions can be limited by introducing a minimum threshold for the size of one region to be considered a part of the OT. Figure 4.5 shows a case where the filtering of four regions from the query BPT results into a non-binary structure in the BOT. Notice that the pixels associated to the discarded part are not ignored, as they are still present in every ancestor node. This modification on the original algorithm might produce a hierarchical structure which is not binary at all its levels. For this reason, the proposed structure is named as *Object Tree* instead of *Binary Object Tree*.

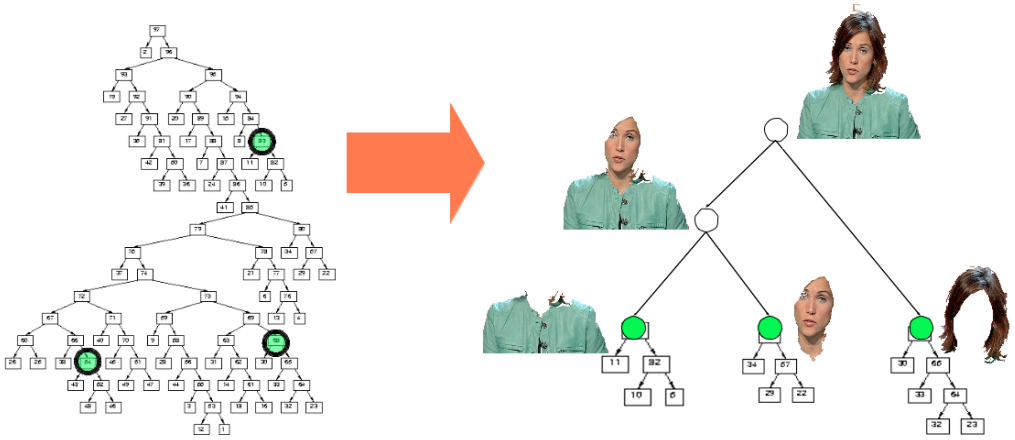


Figure 4.4: Second stage of BOT creation.

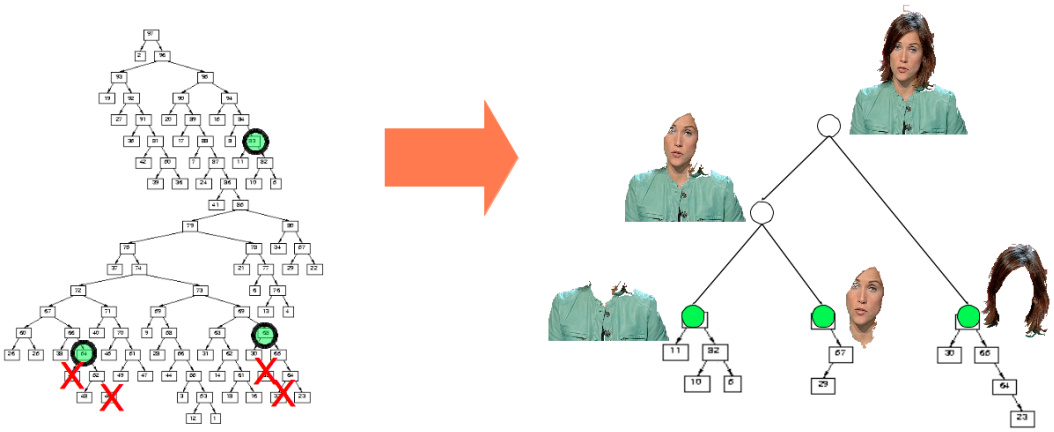


Figure 4.5: Query parts are discarded by their size.

4.1 Summary

The user interaction described in Chapter 3 has been used in the Chapter as an input data to construct a hierarchical representation of the query objects. Those objects that appear split in several sub-trees of the BPT are reconfigured to build a novel *Object Tree (OT)*. The OT is built in two stages, a first one that iteratively merges the object parts to construct a hierarchy of object parts, and a second one that just copies the decomposition proposed by the BPT.

The hierarchical nature of the OT will exploited in Part II for a bottom-up computation of the object features, as well as in Part III when formulating the instance search problem as a matching of the OT representing the query object with each of the BPTs contained in the target database.

Part II

Feature Extraction

Summary of Part II

The second part of this thesis focuses on the features that describe each of the visual parts introduced in Part I. These features represent the characteristics of the image and object parts, a quantification that is automatically achieved through image processing algorithms. Each of the generated feature vectors corresponds to an observation that will be later processed by a pattern recognition system. This posterior analysis is left for the subsequent Part III.

Part II starts in Chapter 5 with an overview of the most common features used to represent regions: color, texture, shape and localization. The MPEG-7 standard includes proposals of visual descriptors that characterize each of these cues, solutions that have been adopted in this work. The chapter introduces the concept of *recursive* features, which exploit the hierarchical structure of a Partition Tree (PT) to efficiently extract the visual features following a bottom-up approach. The chapter concludes with a first retrieval experiment that obtains one-part objects with one single visual descriptor.

Chapter 6 presents how region-based descriptors are refined by mapping them into codebooks. In particular, this work proposes a soft-mapping of regions into a codebook, an operation which can be interpreted as finding the coordinates of every region into a *Parts Space* defined by a codebook. The hierarchical structure of the PT is exploited to propose the *Hierarchical Bag of Regions (HBoR)* as a novel type of features that combine the multi-scale properties of PTs with the quantized nature of codebooks. The experiments presented evaluate different options for codebook creation as well as the advantages of the HBoR when compared with a classic region-based mapping.

Based on the results obtained in Chapters 5 and 6, the final Chapter 7 in this part focuses on how to combine the separate analysis obtained from every visual descriptor in a single result. Both region- and codebook-based features are considered to evaluate a fusion strategy based on the Weibull distribution. This strategy normalizes every similarity metric after a mapping into a Weibull distribution. Experiments indicate that the proposed approach succeeds into combining the information captured by every type of feature in a way that each of them contribute to the final similarity measure.

Chapter 5

Region Features

An image or part of an image can be characterized in multiple ways, each of them describing its visual appearance considering different cues. Humans are capable of quantizing concepts such as *color*, *texture*, *shape* or *left* with linguistic terms such as *green*, *stripped*, *round* or *above*. Analogously, image processing algorithms can also quantize the attributes of a region with a visual descriptor. Pattern recognition techniques use these visual features as the input data to infer valuable information for the semantic interpretation of the content.

Visual descriptors can be computed over the whole container and/or over its segments. In the first case, they are normally referred to *global* as its final value takes into account pixels from the whole content. Global features provide a high-scale representation of the complete image and have been widely adopted for scene analysis [109]. On the other hand, *local* features are computed only over a portion or segment of the content, which requires a previous definition of the segments of interest. Local descriptors offer a higher resolution description of the content at the cost of an individual computation for each segment of interest. In some cases, a type of visual descriptor can be computed both from the whole image or from one of its parts.

As presented in Part I, the basic analysis units of this work are regions which have been automatically defined through a segmentation process. Each of these regions presents certain perceptual homogeneity, and each object part ideally presents this homogenity, too. This quality is not present in other popular solutions such as arbitrary segmentation (eg. block-based or spatial pyramid) where it is highly probable that those regions of support located at the object contour will also contain pixels belonging to the background. A similar problem appears in the elliptical regions of support for the popular SIFT and SURF descriptors, whose feature vectors may also be influenced by pixels belonging to the background. It is true that considering background pixels is a method for introducing context information in the data analysis, an important additional information when dealing with local objects. In the previous work by Carson et al [14] it was observed that local features performed better than global

ones when retrieving images queried with distinctive objects, such as tigers, cheetahs and zebras. In the other hand, other object queries with highly characteristic backgrounds (eg. airplane) obtained higher precision when addressed with global histograms. In this later case, the context contained the distinctive features that allowed a successful retrieval. However, the aim of this work is both image retrieval and accurate object segmentation, as previously introduced in Section 1.3.1. This second goal can only be achieved if background data are separated from the object features, so that pixels can be individually classified as object or background. This approach also allows the recognition of objects when they appear in new contexts that do not introduce any prior about the presence or non-presence of the object. The contextual information could be introduced at a higher and separate level, as proposed in [60].

Visual features can be divided in two big families: *generic* and *domain-specific*. The generic family offers solutions which can be used in a wide range of applications. On the other hand, domain-specific descriptors have been developed and optimized for a certain type of application, such as face [95] or text [52] extraction. Following the generic approach considered when defining the image parts in Chapter 2, this work focuses in generic purpose descriptors with a wide application.

This chapter studies the generic visual descriptors that can be computed in the framework of the hierarchical partitions introduced in Chapter 2. The study introduces the generic descriptors defined at the MPEG-7 standard in Section 5.1, which have been adopted all through this thesis as visual features. Section 5.2 presents the concept of *recursive* descriptors depending on their possibilities of efficient computation through a hierarchical partition. Finally Section 5.3 presents the results of retrieval experiments performed by considering three different visual descriptors extracted from the regions of a Binary Partition Trees.

5.1 MPEG-7 Descriptors

In 2001, MPEG-7 [11] made an effort to collect and standardize a set of visual descriptors organized in four categories: color, 2D/3D shape, texture and motion. The standard also defined a exchange data format (both XML and binary), opening the door to an interaction of databases in terms of metadata. It must be noted that MPEG-7 only standardized the formulation and data format for these descriptors. It was left to the scientific community the research to find the best solutions for their computation and comparison, although default techniques were proposed, too. The detailed algorithms for their computation can be found in [12] [11] and references therein.

Previous works have applied MPEG-7 descriptors for image retrieval and classification. Ojala et al [65] compared the MPEG-7 color descriptors at the global scale, concluding that among them, those that also consider some sort of local spatial distribution information

outperform those that do not. Molina et al [59] achieved their best image classification rates when combining global and local descriptors.

This work has adopted a subset of MPEG-7 descriptors for two reasons. The first one is their richness and completeness for representing still regions, which allows a good characterization of the visual segments defined in the PTs. Secondly, they are popular and accepted in the scientific community, which facilitates repetitiveness of the reported experiments.

This section provides an overview of the main attributes considered by the MPEG-7 standard for still images and presents the three visual descriptors adopted in this thesis.

5.1.1 Attributes

The MPEG-7 proposes four different types of attributes in which the generic descriptors can be grouped: color, texture, shape and localization.

Color descriptors are widely used because they are intuitive, simple and robust to viewing angle changes and rotations. They are based on the homogeneity of the pixel values expressed in the components of a typically three-dimensional feature space. MPEG-7 specifies four different descriptors for still images to satisfy a wide range of application domains. The Dominant Colors Descriptor contains the mean and variances of up to the eight most representative colors in a segment, the Scalable Color Descriptor contains a multiscale Haar transform of the color histogram, the Color Structure Descriptor contains information of the spatial distribution of colors and how compact they are distributed, and the Color Layout Descriptor provides a compact representation of the color distribution based on a Discrete Cosine Transform. While the two first are both valid for global or local scale, the two later require some adaptation to be used in the local scale.

Texture features refer to some basic pixel primitives that are repeated in a connected region. MPEG-7 has standardized three texture descriptors which are extracted from the luminance component of the images. The Homogeneous Texture Descriptor describes the energy of a set of frequency channels, the Texture Browsing Descriptor characterizes regularity, coarseness and directionality of the texture, and the Edge Histogram Descriptor contains a set of histograms of five types of edges, one histogram for each of the blocks resulting from a 4x4 partition of the image. Both of them require some adaptation as they are designed to be extracted from rectangular regions of support.

Shape Descriptors possess important semantic information, a proof of it is that humans are capable of recognizing objects by just seeing its shape. However, all this semantic richness can only be extracted with a prior good segmentation. For this reason, all shape descriptors are local. MPEG-7 proposes two types of 2D shape descriptors: the Region-based Shape Descriptor captures the distribution of pixels within a region, and the Contour-based Shape Descriptor specifies the closed contour of an image segment. Both cases are designed to be

exploited at the local scale.

Localization descriptors are local descriptors that provide information about the position of the segment referred to the container's origin of coordinates. These descriptors offer the necessary data to compute the spatial relations between these segments and represent spatial structures.

5.1.2 Descriptors

The MPEG-7 standard defines a collection of visual descriptors that cover the described attributes. Each descriptor in the standard is accompanied with indications about how to quantize every value contained in the feature vectors. This quantization step provides a maximum resolution of a descriptor in a manner that can match the perceptual response of the human visual system. This way, although two regions may not present exactly the same visual properties, they may be considered identical in terms of human perception. All results presented in this thesis have been generated taken into account this quantization step.

Among the broad offer of visual descriptors proposed in MPEG-7, this work has focused in three local descriptors to represent the color, texture and shape of considered regions. This choice allows a rich characterization of every region considering three complementary visual cues. The selected descriptors are the *Dominant Color Descriptor*, *Edge Histogram Descriptor* and *Contour-based Shape Descriptor*. This triplet covers the main cues that describe local segments and keep the amount of features small to avoid unnecessary computation time. The choice also considered the performance obtained in a preliminary evaluation in terms of precision and computational time among the different options on the available software. It must be highlighted that it is not the goal of this thesis to design nor evaluate better visual features but to study how typical descriptors can be efficiently exploited together with hierarchical region representations. This research approach is similar to the one that has motivated the release of several datasets that, in addition to the images and annotations, they also include the global and local features as well as the scores of a set of concept detectors [86] [43].

Dominant Colors Descriptor (DCD)

The DCD is a compact color descriptor specially designed for similarity retrieval and browsing. It can support up to eight colors but, unlike the predefined bins used in other histogram techniques, these dominant colors are computed for each image. A clustering algorithm partitions the color space and selects a maximum of eight dominant colors. DCD describes, for each cluster, the dominant color, the cluster percentage of pixels in the image and, optionally, the variance of the pixel values. Finally, a last parameter expresses the overall spatial homogeneity of the dominant colors.

Edge Histogram Descriptor (EHD)

The EHD provides information about the spatial distribution of the edges in an image. The image is split into 4x4 sub-images and an edge histogram for each sub-image is built. The histograms are created by further subdividing the sub-images in around 1100 image-blocks and assigning a label to each of them according to its edge orientation. Five categories are defined: vertical, horizontal, 45 degrees, 135 degrees or non-directional. EHD also includes a global and a semi-global edge histograms by accumulating the local histograms for the whole image or for sub-blocks. As a result, a total of 5 global bins + 65 semi-global bins + 80 local bins are available for matching. Previous experiments have shown that EHD is useful in the retrieval of natural images with nonuniform textures and clip art images, as well as for sketch retrieval. MPEG-7 defined EHD only at the global scale, so the adaptation proposed by Ventura [93] was adopted to represent the regions in the PT.

Contour-based Shape Descriptor (CSS)

Contour-based Shape Descriptor is suitable for those objects whose discriminative data is contained at the shape of its silhouette. It is robust to non-rigid deformations, changes in orientation, scaling, mirroring and perspective transformations. It is based on a Curvature Scale-Space (CSS) representation of the contours. CSS decomposes the contour into convex and concave sections and expresses how prominent they are, how long compared to the full contour and their position. This is done in a multi-resolution approach, by applying a smoothing process to the contour at different scales. The CSS curve is obtained by plotting the convex to concave change points on the 2-D plane, with the normalized distance along the contour on the X axis and the amount of smoothing on the Y axis.

5.2 Recursive Features

The computation of region descriptors can benefit from the hierarchical structure of a PT. The relations of inclusion between children and parent nodes determine certain dependencies between the visual features associated to each of these nodes. In the cases presented in this section, the hierarchical structure is exploited for an efficient computation of the visual descriptors. For some descriptors, the consuming process of individually processing the pixels associated to every node can be substituted for a fast bottom-up propagation of the descriptor values. In other words, the computation of the descriptors associated to a PT node is obtained by *recursively* extracting the features from its children. In these cases, only the regions at the PT leaves need to be processed from scratch, which represent almost half of the total amount of regions. Moreover, leaf regions are usually smaller than other nodes in the PT, which further reduces the computation effort.

A visual feature $x(r)$ belongs to this set of *recursive features* X_e if its value on a certain region r can be computed from the features previously extracted from the set of region's children C^r:

$$x(r) \in X_e \quad \Leftrightarrow \quad x(r) = f\left(x(c_1)...x(c_{|C^r|}) \ : \ c_i \in C^r\right) \tag{5.1}$$

Recursive features present a second property that makes them very interesting during analysis. They contain information obtained from its descendants, a property that allows considering the descriptor not only representing the PT node but all the sub-tree below. An analysis algorithm following a top-down exploration through the PT could discard complete sub-trees with this information. This strategy speeds up the analysis process as not every single PT node needs to be considered.

5.2.1 Types of Recursion

The bottom-up propagation of the features from the children to its parent node in the PT can rely on different operations. These are the most popular in the framework of region-based expandable features:

Addition: The propagation is performed by directly summing the descriptors values, as shown in Equation 5.2. The most popular example of this type of descriptor are histograms, whether based on texture or on color. The computation of the region *area* is also an example of such approach.

$$x(r) = \sum_{c \in C^r} x(c) \tag{5.2}$$

Weighted average: The propagation of the mean and variance descriptors are performed by weighting and normalizing the children features according to their area in pixels. Examples of this type of combination are the descriptors computed from the addition of individual values associated to each pixel, for example color and localization. Equation 5.3 shows how the mean value can be computed in this context.

$$\mu_x(r) = \frac{1}{\sum\limits_{c \in C^r} x_{area}(c)} \sum_{c \in C^r} x_{area}(c)\mu_x(c) \tag{5.3}$$

Max-Min In other cases the descriptor value corresponds to a certain maximum or minimum value associated to a region. The resulting descriptor is easily computable by reapplying the operation on each descriptor associated to the region's children, as shown in Equation 5.4. For example, the computation of the non-oriented bonding box of a PT

node can benefit from such approach, where the maximum and minimum coordinates of the region are propagated.

$$w_e(r) = MAX(w_e(C^r)) \tag{5.4}$$

$$w_e^{min}(r) = MIN(w_e(C^r)) \tag{5.5}$$

5.3 Evaluation of the Baseline Approach

The three MPEG-7 region descriptors presented in section 5.1.2 have been tested separately with the ETHZ dataset. The reader is referred to the previous Section 1.3 to learn about the configuration of the experiments and the presented metrics, a set up that is replicated in every further *Evaluation* section reported in this thesis.

This first experiment aims at quantifying the richness of each descriptor to characterize an object. Each of the instances in the test set has been used to define a query region, which has been compared with every region in the BPTs of the test dataset. The ranked list has been generated according to the similarity measure obtained for the best match between the query region and one of the regions in the target BPT.

The image retrieval results are presented in Figure 5.1, while the local analysis (detection and segmentation) is reflected in Figure 5.2 and Figure 5.3 to indicate the detection and segmentation performance of the descriptors. In the three figures, these graphs provide an initial insight of the main challenges encountered in this work. Every figure contains, on the left, the individual results for each category and, on the right, the averaged results obtained by combining all possible queries. Notice that on the average results, those semantic classes with more instances have a larger impact on the final value, as the averaging is not category-based but query-based. In the retrieval case (Figure 5.1), and in order to establish a baseline reference, it has been included the results obtained by a random retrieval (*random*).

An analysis of the three figures allows making some observations about the individual performances of the descriptors. The first observation is that none of the descriptors clearly outperforms the rest. While, in average, texture is the feature that provides the best results, this is not true for all categories. So the first conclusion it that the performance of a descriptor is dependent on the visual category that is being considered. This drives into a scenario where the results obtained for each descriptor may need to be combined, an option that will be addressed in Chapter 7.

Overall performance is also clearly dependant on the category. While in most cases the use of one separate region-based descriptor outperforms the random result, this is not certain for every case. Specially in the case of the *Swans* category, results are poor, with no significant gain with respect to the random approach.

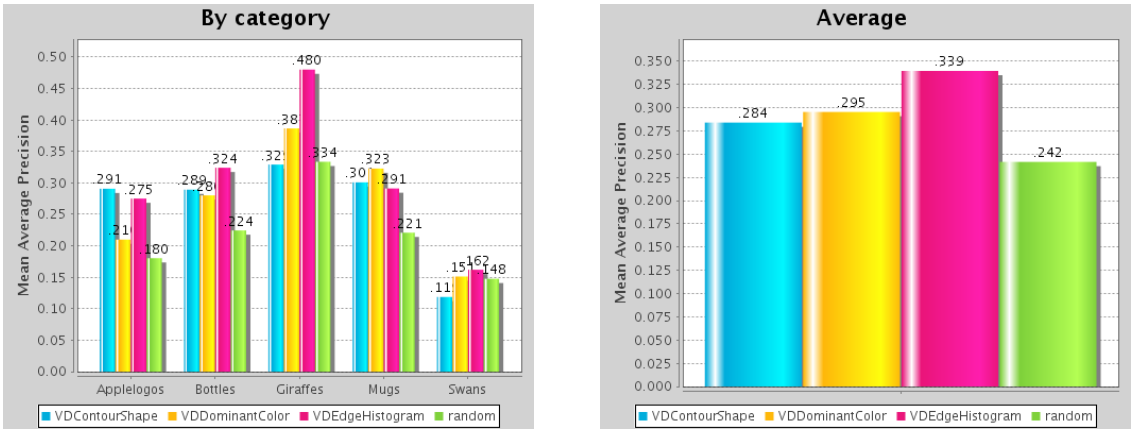


Figure 5.1: Image retrieval with separate descriptors.

Detection and segmentation results in Figure 5.2 and Figure 5.3 suggest a high dependency of the quality of the segmentation performed at an earlier stage. Especially in the very sensitive case of the Countour Shape descriptor, the *Apple logo* category presents significantly better results in terms of detection and segmentation. This outstanding result of the shape descriptor in this category is explained by two reasons. Firstly, the shape in a commercial logo is rigid and designed to be easily identified by consumers, that is, it is not expected to be mistaken by any other shape. In our experiments, this means that there are little chances to find an apple-shaped region in an image that does not belong to the *Apple logo* category. The second reason is that, in many of the instance objects, the logo is very homogenous in color, which helps its segmentation into a single region. So, while in the other categories most of the instances are defined by multiple disjoint sub-trees in the image BPT, these logos usually appear represented by a single BPT node. This circumstance makes matching with the query much more feasible than for more complex objects, whose parts may have split in several subtrees of the target BPTs. These situations are not handled with the baseline approach studied in this chapter, but are the main research topic of Part III.

As it can be seen in Figure 5.1, the texture descriptor obtains the best results for image retrieval, and offers also competitive results for detection and segmentation if compared with the color and shape ones. This descriptor is not entirely recursive because its computation must be performed from scratch for every BPT node and cannot be computed bottom-up. Nevertheless, its histogram nature makes its concept very similar to recursive as the bin values obtained through a BPT branch tend to be incremental, even if approximately. This property makes this descriptor more robust to object splits and, despite not providing the best results in the local analysis of detection and segmentation, it tends to perform better than color and shape, especially in those categories with more object splits.

The difference of performance of the texture descriptor between image retrieval and its

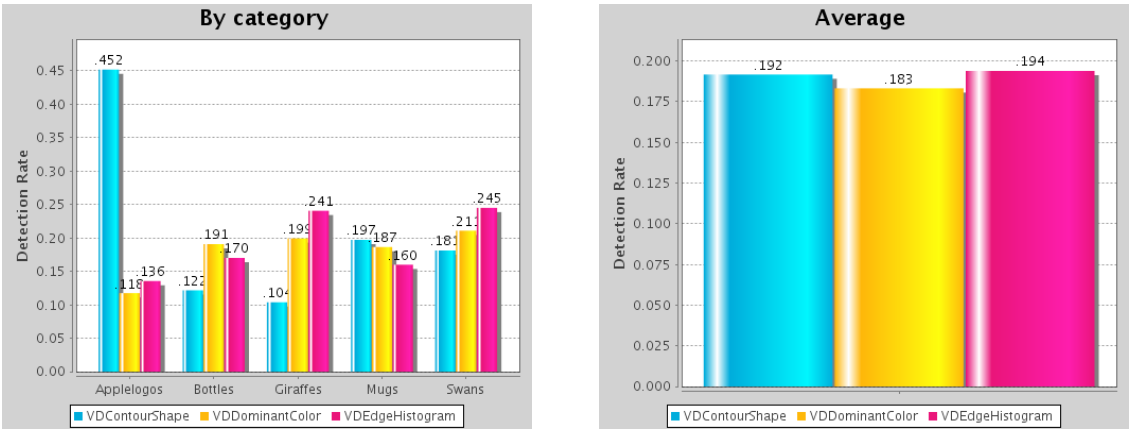


Figure 5.2: Object detection with separate descriptors.

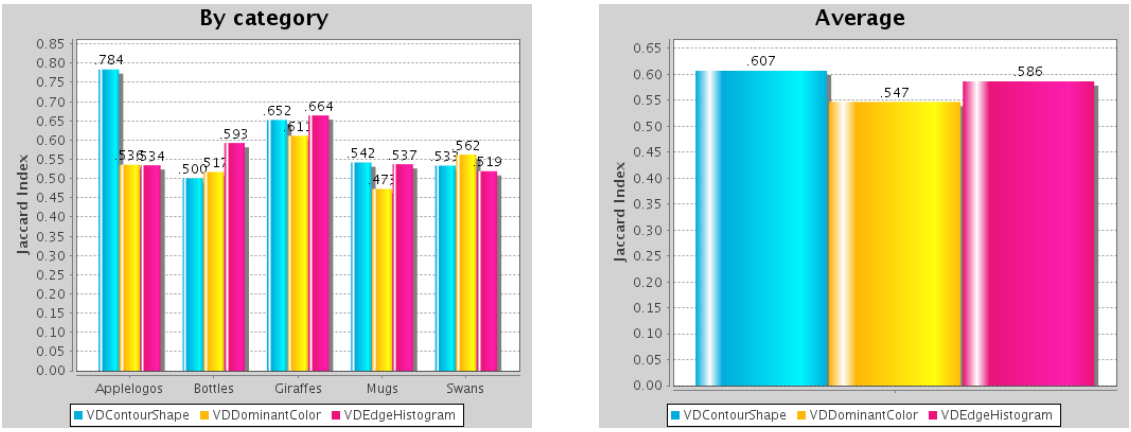


Figure 5.3: Object segmentation with separate descriptors.

local detection indicates that an important part of the obtained image hits might have been referred to the background instead of to the object. In the best case of the *Giraffe* class, their spotty pattern on the fur is the result of thousands of years on natural evolution that have created this natural tool for camouflage. It is reasonable to think that, the same way this pattern has confused their predators, it may have also confused the presented algorithm in the reverse: considering the texture of trees and bushes as belonging to a giraffe.

5.4 Summary

This chapter has introduced the region-based visual features used in this work, which are based on the visual descriptors proposed in the MPEG-7 standard. Visual descriptors that represent the color, shape and texture of every region have been described to characterize every node in the Partition Tree. This hierarchical PT structure can be exploited for a fast

extraction of visual features through a bottom-up propagation process. This property is desirable given that, in general, the computation of visual descriptors is high demanding in terms of processing power. Finally this chapter has presented the first baseline experiments, obtained by defining an object query with a single region and comparing the visual descriptors of this region with those associated to every PT node in the target database. Results suggest that there is no visual descriptor significantly better than the rest.

Chapter 6

Codebooks

Visual descriptors represent the visual attributes of the regions with a very high level of accuracy. This level of detail is valuable when comparing regions from a visual perspective, but this precision is usually much higher than the semantic interpretation humans assign to visual cues. For example, while a typical 8-bits per component RGB space can discern between more than sixteen million colours, most people would have more than enough with one hundred linguistic terms to describe colors from a semantic perspective. This observation inspires this chapter, that proposes expressing the visual descriptors associated to the regions taking as a reference a reduced set of visual primitives.

In natural language, adjectives describe a set of visual features that are considered perceptually similar among themselves, but dissimilar enough from other sets labelled with different adjectives. The selection of an appropriate adjective term to describe a visual property can be understood as a quantization process performed by the human brain. In the pattern recognition terminology, this process corresponds to a partition of the feature space so that an observation is assigned to a natural language adjective. In the signal processing community, this process is known as *vector quantization* and it is defined as a mapping from a high dimensional into a lower dimensional discrete space. The dimensionality reduction compresses the input data, and this operation can be beneficial in terms of data storage and processing requirements. However, any quantization process must also control the information loss so that the result does not degrade the performance of the desired results.

A good quantization process is capable of finding a good compromise between information loss and functionality with respect to the final application. For example, the digital gamma correction in video defines smaller quantization intervals in darker values than in lighter because the human visual system is more sensitive to the first than to the later. This way, the obtained bits are more informative to the end user than if a uniform quantization had been applied.

The proposal in this work does not directly apply the quantization on the numerical values

of the visual descriptors, but it is inspired by a vector quantization approach that transforms every feature vector in relation to an auxiliary dataset known as *codebook*. The codebook contains a set of prototype regions chosen after a clustering process on a training dataset. These prototype regions are the visual primitives that, if correctly chosen, will represent the different sets of visual features that can be found in a certain problem.

These visual prototypes, or *codewords* if considered as composing elements in a codebook, are similar to the *perceptual-areas* proposed by Jaimes and Chang [41] that were previously introduced in Chapter 4. They were defined as a decomposition of the object parts into low level perceptual categories. This work considers a closed set of these *perceptual-areas* defined in a codebook so that they can be referred by a diversity of semantic objects or their subparts. As an example, a *green patch* could be a codeword used to represent several semantically diverse objects, such as grass, a traffic light, a frog...

The use of codebooks in the context of image retrieval has been broadly adopted in the literature mostly associated to the Bag of Features (BoF) model by Sivic and Zissermann [80]. They proposed to extract SIFT descriptors from elliptical regions around interest points to build a visual codebook through the K-means clustering algorithm [38]. This codebook allows representing every image by extracting their SIFT descriptors and assigning each of them to a prototype in the codebook. This vector quantization is synthesized in a histogram that characterizes the contents of the image. Given that the resulting histogram does not keep any information about the spatial relationship between the points, this approach is also referred to as a *bag of features* model of the image, because all image parts (SIFT descriptors) are considered as if they were kept in a bag, with any order. In fact posterior work has proposed solutions for adding the spatial information to the model, for example, through a hierarchy of part models [89] [62] or spatial pyramids [51] [8].

The concept of *Bag of Features (BoF)* literally refers to working with a set of unsorted observations from the document that is being analysed. However, the enormous popularity of the Bag of SIFT Features model proposed by Sivic and Zissermann [80] has often associated this term with two additional properties: i) working with interest points, and ii) creating a *signature* from the extracted features, such as a histogram [69]. Other authors [35] [60] [37], though, have stuck to the literal meaning of the term and consider the *Bag of...* as a collection of unstructured observations, with no further consideration on how they are further processed. This second interpretation is applied in the current work, as the proposed techniques deal with sets of regions that represent images and parts of images, exploring different possibilities for their posterior analysis.

The *Bag of Features* model is inspired in the *Bag of Words (BoW)* model proposed by the text retrieval community, a common model used for the classification and clustering of text documents. In this community, it is common to represent documents with the *Term Frequency-Inverse Document Frequency (TF-IDF)* descriptor. This descriptor is also based on

a feature vector referred to a trained vocabulary that contains all relevant and discriminative words in the target dataset. Typically, this vocabulary does not contain semantically poor terms, such as articles or prepositions, which are very frequent but little informative about the contents of a text document. Every value in the TF-IDF feature vector corresponds to a term in the vocabulary. The corresponding value combines the degree of occurrence of a term in a document (TF) with the overall occurrence of the term in the dataset, the IDF. This way, the most frequent terms in a document which are not frequent through the database obtain highest values to characterize their relevance when characterizing the document. Given the similar approaches between the text and visual domains, this principle was also ported to the image retrieval case to improve the baseline results obtained with the classic bag of features technique [42].

The use of codebooks is not exclusive for point-based image parts, but has also been explored on region-based representations. Gokalp and Aksoy [35] used a codebook to generate a *Bag of Regions (BoR)* representation of an image. Each considered region was assigned a discrete label that related it to the most similar region type in the codebook.

Mylonas et al [60] used in their work a region-based codebook, which was referred as *visual dictionary (thesaurus)*. This codebook was built through a hierarchical clustering of all the segmented regions in a training dataset. In this case, the centroids of the clusters were referred as *region types*, an entity located between low-level descriptors and high-level semantic concepts. The rest of feature vectors contained in a cluster were referred as *synonyms*, by applying an analogy with the linguistic terminology. The analysis on new images is performed by segmenting the image and computing the Euclidean distance between every region in the partition and each region type. However, only the smallest distance is considered. Finally, a *model vector* describes the complete image by concatenating the smallest distance found for every visual type (codeword) among the regions in the image. The obtained model vector is further combined with strategies to exploit the topological context between regions for high-level concept detection.

Instead of defining a feature vector for the complete image, Socher and Fei-Fei [87] used visual words to represent each of the regions defined by a segmentation. In particular, they defined a separate codebook for each of the four visual descriptors considered: color, position, texture and shape. Each region is represented by a *visual word* that contained a sequence of assignments to the most similar codeword in each of the four codebooks. The sizes of the codebooks varied depending on the type of visual descriptor. Their work included a study about codebook creation, where they compared two options: creating a single codebook of concatenated features or creating separate codebooks by clustering each feature separately. The two options were assessed in terms of purity and frequency, and the results clearly indicated that working with separate codebook gives much more flexibility.

A hybrid combination of regions and points of interest was proposed by Ravinovich et al

[69], who work with a shortlist of image segmentations and characterize every resulting region with a histogram of SIFT features. The same approach was adopted by Yanai and Barnard [105], who refers to it as *region-based bag-of-features* representation.

An even richer combination of region- and point-based descriptors was proposed by Vieux et al [94], who represented each region of an image segmentation with a set of three frequency histograms. Two of these histograms were built from region-based descriptors (color and texture) and the third one represented regions with the histogram of SURF features. The three histograms were concatenated to build a feature vector for region classification.

This chapter is inspired by different aspects of the reported work, by trying to adopt them in the context of a hierarchical image partition. In particular, the use of codebooks [80] is adopted, but instead of using SIFT features they will be based on region ones, as in [35]. However, instead of a hard assignment between every region and a codeword, a soft quantization will be used, as in [35] [60]. Separate codebooks for every visual descriptor will be considered, as suggested by [87]. Finally, a multi-scale approach like [89] [62] [51] [8] is adopted, but based on a content-dependent segmentation represented by Partition Trees.

6.1 Hierarchical Bags of Regions

This section proposes a solution to exploit both the quantization nature of codebooks and the hierarchical region structure of Partition Trees. The adopted approach allows that, in the ideal case where the sub-parts of an object correspond to regions in the codebook, a region that contains only the object sub-parts would be mapped to a distinctive area of the feature space. If this requirement is satisfied, it will be feasible to define metrics that will compare two regions depending on the sub-parts that compose them. This desirable feature space is named the *Parts Space*.

A bidimensional Parts Space is depicted in Figure 6.1, where the two codewords correspond to the two sub-parts of the *anchorwoman* object, a plausible assumption given the visual dissimilarity between them. The two dimensions of this Parts Space are represented in every axis, and the location of the shown regions depends on the best similarity between any of their sub-parts and each of the two codewords. Some of the considered regions represent the *anchorwoman* as a whole, some others sub-parts of the object, and a third type correspond to parts of the background. In this space, those regions containing a *head* are represented by a feature vector close to (1,0), while those containing a *body* with a complementary vector (0,1). When considering regions that merge two classes of sub-parts, the resulting feature vector in the Parts Space is approximately (1, 1). On the other hand, the region representing background parts are coded with vectors close to (0,0). These four types of feature vectors are located in different areas of the feature space, so they can be effectively discriminated with simple and fast distance metrics, such as the Euclidean distance (Equation 6.1) or the

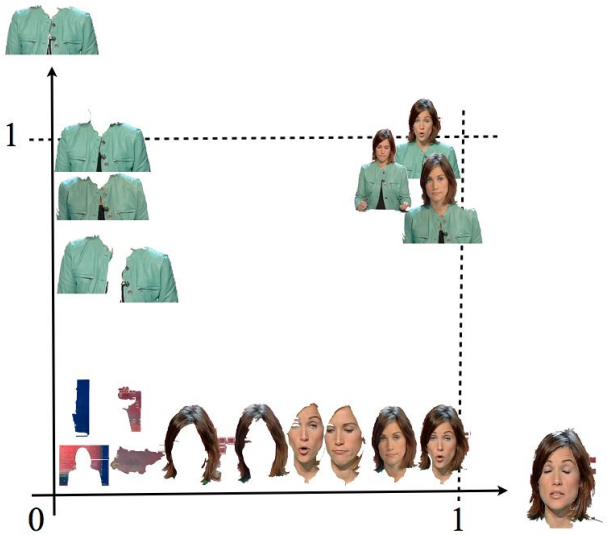


Figure 6.1: Parts Space defined by the two sub-parts of an object.

cosine distance (Equation 6.2).

$$d(x,y) = \sqrt{\sum_{i=0}^{N}(x_i - y_i)^2} \tag{6.1}$$

$$d(x,y) = \|x\|\|y\|cos\theta = \frac{x \cdot y}{\|x\|\|y\|} \tag{6.2}$$

.

The Parts Space is also valid to separate between those cases where object and background parts have been merged in single region. These situations can be addressed by introducing codewords that do not belong to the object but from the background annotations or, in a multi-class problem, from other classes of objects. Figure 6.2 shows another Parts Space where one codeword corresponds to the object and the other to the background. In this case, the regions containing the object and parts of the background could also be discerned.

Each component in the feature vector is obtained by computing the descriptor-specific distances between every observation and the codewords. During this step, any perceptual interpretation of the descriptor values is performed, so the resulting feature vectors do not require additional perceptual corrections.

The design of this proposed Parts Space requires the combination of two different approaches that are presented in the next two sections: a possibilistic quantization and a max pooling of the region features through the Partition Tree.

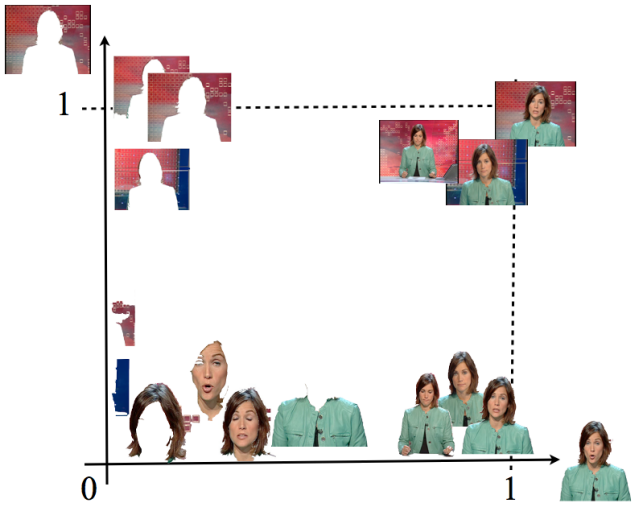


Figure 6.2: Parts Space defined by an object and a background part.

6.1.1 Possibilistic Quantization

The use of codebooks adopted in this thesis differs from the classical *hard* vector quantization where every observation is assigned to a single codeword [22]. In this work, each component of the feature vector corresponds to the degree of similarity between the observation and the corresponding codeword. This type of solutions belongs to the *soft* quantization paradigm, where the components of the vector take values between 0 and 1 [7] [3]. For example, a *probabilistic* feature vector [3] would fall into the category of *soft* quantization, as each component would indicate the probability of the observation of belonging to the same class as each of the codeword. In the probabilistic case, it is required that all the values in the feature vector sum to one.

In addition to the *hard* and *probabilistic* quantization, there exists a third option, the *possibilistic* one [7]. This approach evaluates the possibility of every observation to belong to the same class as each codeword. The main difference of working with possibilistic feature vectors is that their components do not need to sum one. The possibilistic interpretation of a feature vector composed of similarity measures suits better than a probabilistic one. For example, in case that the observation were not similar to any of the codewords, it is desirable that its associated feature vector was completely null. If a probabilistic approach were adopted, the obtained similarity values would be originally very low, but the posterior normalization stage may increase their value. This step would introduce a distortion on the interpretation of the feature vector in the parts space. For example, in the 2D Parts Space presented in Figures 6.1 and 6.2, all feature vectors would be projected in the line crossing

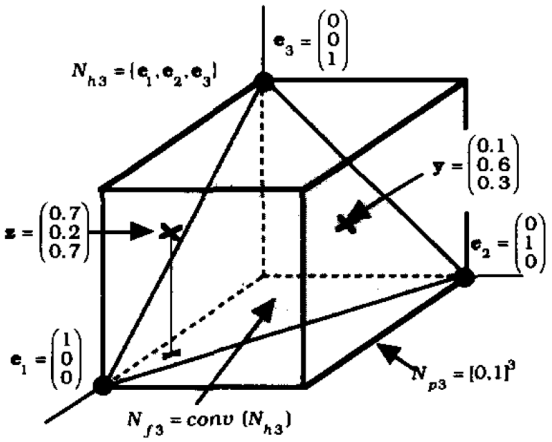


Figure 6.3: Hard, probabilistic and possibilistic feature vectors. (Bezdek, 1995)

(1,0) and (0,1), that represents all cases whose vector components sum up to 1.

The *hard*, *soft* and *possibilistic* approaches are depicted in Figure 6.3, which contains a scheme extracted from [7]. This example would correspond to a codebook of size three, one codeword for each dimension in an Euclidean space. The *hard* quantization case would map any observation to one of the three e_i vectors located at the corners of the cube. These vectors correspond to the three basis of a canonical Euclidean space of dimension 3, named N_{h3}. A *probabilistic* analysis would describe each observation with a vector in the plane defined by the three e_i, which is the convex hull N_{f3}. For example, vector y belongs to this plane. Finally, the space considered by a *possibilistic* solution corresponds to the whole unit cube N_{p3}, and vector x is an element in this space.

When applied to the visual descriptors in a PT, this possibility-based quantization will produce a feature vector for every region, as shown in Figure 6.4. In this example, a codebook of four regions has been used to represent the content of every considered image part. When a PT node is very similar to a codeword, a maximum 1 value is obtained on the corresponding vector component, while a 0 value represents the opposite.

The adoption of quantized feature vectors allows the introduction of certain indexing solutions and allow the fast computation of similarity measures. However, the possibilistic solution does not satisfy the requirements of a Parts Space because its vector components represent the region-to-codeword similarity instead of their inclusion. This missing requirement can be solved by applying a max pooling approach, presented in the next section.

6.1.2 Max Pooling

Associating the feature vectors to the regions in the PT does not satisfy the requirements expected by the Parts Space. If going back to the example in Figure 6.1, the *head* and *body*

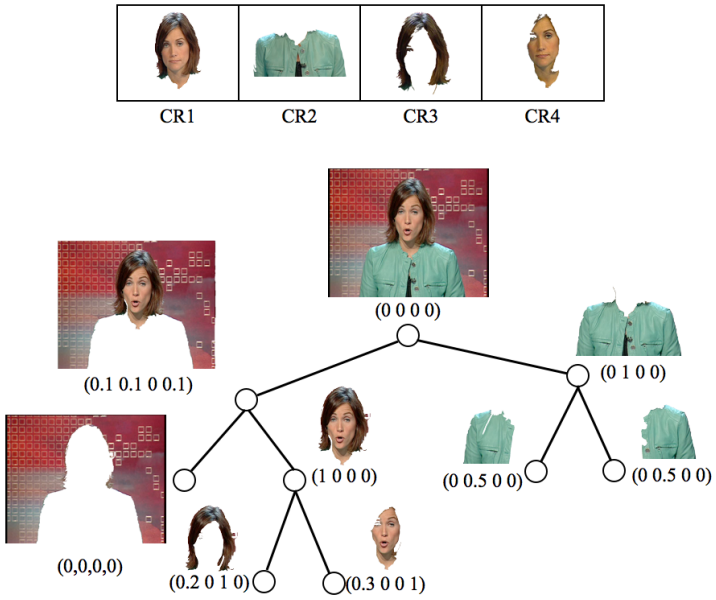


Figure 6.4: Feature vector for regions in a Partition Tree.

parts would be discriminatingly represented by (0,1) and (1,0) feature vectors, respectively. On the other hand, when trying to map the region feature of the complete *anchorwoman* to each of the two codewords, the result would place the full objects near to the (0,0), together with the rest of the clutter parts. Instead of this situation, the *Parts Space* should represent the *anchorwoman* with a vector close to (1, 1), so that they could be easily discriminated from the clutter parts. Notice that the reasoning is valid for any type of quantization (hard, probabilistic and possibilistic).

The proposed solution combines the possibilistic quantization with the hierarchical region-based representation provided by the Partition Trees. The basic idea is to characterize PT nodes not only with its associated region, but also with regions in the sub-tree it defines. Analogously to how the bag of SIFT features considers images as an unstructured collection of points, this solution proposes to consider every node of the PT as a bag of the regions in its associated subtree. Given that this consideration is applied at every level of the PT, this scheme has been named *Hierarchical Bag of Regions (HBoR)*.

The computation of the HBoR features combines the possibilistic feature vectors of the regions defined by the PT node, with a bottom-up MAX operation that it is iteratively applied at every level. This strategy is known as *pooling* [9], as it combines the features associated to nearby locations in a new feature that is expected to preserve the important information while discarding irrelevant detail . There exist different methods for combining spatial features, but according to Boureau et al [9], the *max* or *average* operations seem to suit most applications. In the proposed soft-possibilistic approach, the MAX option has been chosen as it perfectly

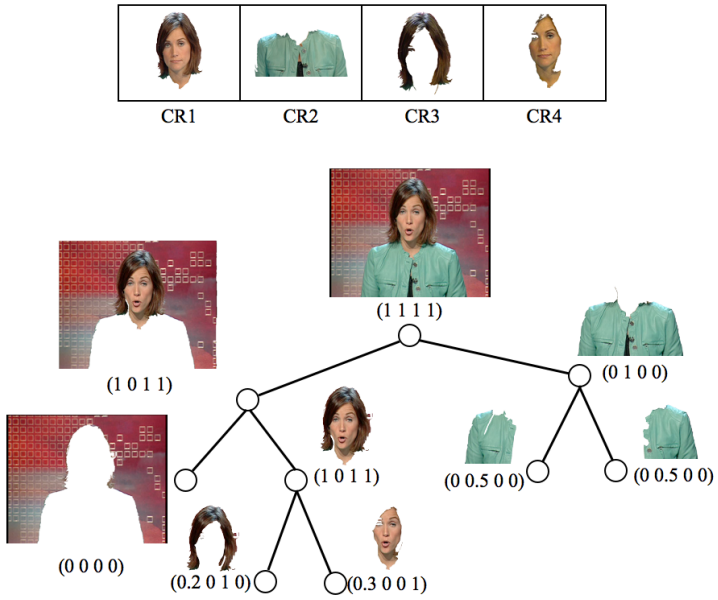


Figure 6.5: Possibilistic Features in Hierarchical Bag of Regions.

satisfies the requirements of the Parts Space.

Following the same example previously presented in Figure 6.4, Figure 6.5 combines the similarity scores of every PT node with the ones from their children through a MAX operator. For example, the node describing the *head* part is represented by a $(1, 0, 0, 0)$ vector in the region-based quantization of Figure 6.4. This vector presents null values at the third and fourth components, which correspond to the *hair* and *face* codewords. Nevertheless, the HBoR solution shown in Figure 6.5 codes this same region with $(1, 0, 1, 1)$, which indicates that these *hair* and *face* codewords are subparts of the the *head* region. This second approach does satisfy the requirements expected in the Parts Space.

6.1.3 Split Resiliency

The proposed HBoR presents several qualities that make it suitable to deal with the object split that could occur during the PT creation, as described in Section 2.4. HBoR features satisfy two properties: (a) a recursive and fast computation during their extraction that allows considering multiple regions which are not in the PT, and (b) an inclusive nature that helps detecting objects even when they are not represented by a single PT node. These two qualities are described and discussed in this section.

Fast extraction (bottom-up)

The extraction of HBoR from a PT requires a previous computation of the visual descriptors from the regions represented by the PT nodes, a condition that is reasonable in the retrieval domain considered in this work. Once these data are available, the population of the PT with HBoR features follows a bottom-up scheme that firstly computes the possibilistic feature vector of every region to later combines it with the ones from their children through a MAX operation.

In some cases, it may be interesting to consider combinations of PT nodes which are not represented in the PT. These cases are common when trying to deal with object *splits*; this is, over-segmented objects that have been merged with parts of the background during the creation of the PT. A possible approach to deal with these situations is assessing combinations of PT nodes which were not considered in the PT. Studying these new combinations requires the extraction of their visual features.

Given that the amount of non-PT combinations to analyse may be considerably large, being able to quickly extract their features is a very desirable property. Approximate versions of HBoR can satisfy these requirement as they can be rapidly computed by fusing the HBoR features of the composing PT nodes. This rough HBoR would ignore the region descriptors of their union because their extraction would require a costly processing effort that would make the analysis infeasible in terms of computation time. The main advantage of this rough HBoR is precisely that their light computation allows assessing several unions of PT nodes that could not be considered if their visual descriptors had to be extracted from scratch.

Figure 6.6 shows two possible configurations of the PT from the same image, where the object is represented by one or two image parts. This example considers the same codebook used in Figure 6.5, where the codewords correspond to the object semantic parts. In the split situation (on the right), an approximate HBoR for the non-PT region is rapidly computed by combining the image parts marked with a box. The resulting HBoR vector is in fact identical to the non-split case (on the left) because the codebook is composed by sub-parts of the object; that is, there exist no codeword representing the complete object. This example shows how rough HBoR features seem a valid tool to deal with split situations with a minimal effort in terms of feature extraction.

Efficient analysis (top-down)

HBoR features are good representatives of the image parts included in the sub-tree. The inclusive nature, associated to the *bag of* paradigm, allows a rough analysis of the regions contained in the sub-tree from the HBoR feature. If the codebook is representative enough, its components can determine if the underlying tree contains the parts of the object which is being recognized. For example, in Figure 6.5 the root node of the PT obtains the highest

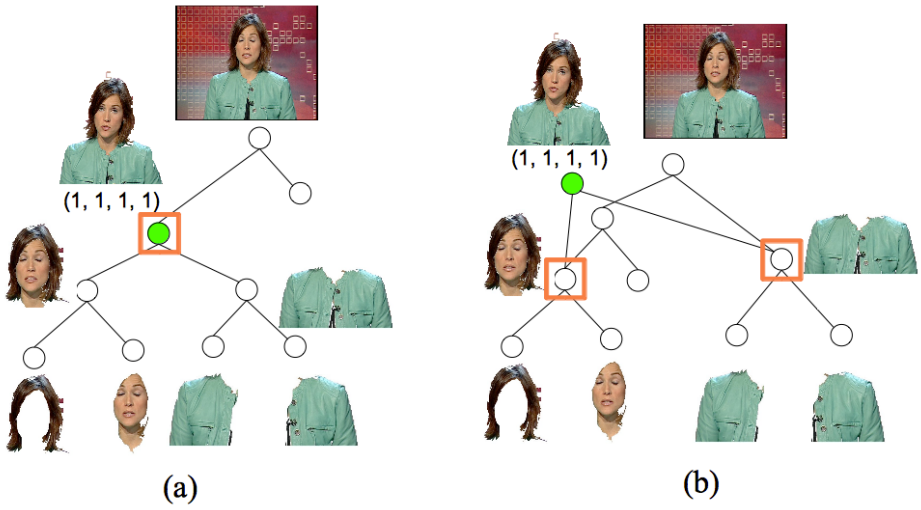


Figure 6.6: Identical HBoRs in (a) non-split object and (b) split object

score for every codeword. The feature vector indicates that the tree contains all the parts of the object, but it does not provide any information about their location not even if they are connected. This information would also be true if the object were split in several sub-trees, so the decision about the inclusion of the object parts would be resilient to this type of problems.

Even if the feature vector at the root is not conclusive to determine that the object is in the image, it is very conclusive to determine that the object is not included in the image. So, a zero-valued feature vector at the root would indicate the analysis algorithm to discard the rest of the nodes in the PT, saving a large amount of computation effort. Moreover, given the hierarchical structure of the PT, this reasoning can be applied at every node, so full sub-trees can be discarded when following a top-down analysis. This property can save an important amount of processing effort as not every node in the PT needs to be assessed.

6.2 Codebook Creation

There exist several options for the definition of the Parts Space, basically dependent on how the codebooks are generated. The first step before the creation of the codebook is to obtain a representative dataset that will be processed to select what elements will be chosen as prototypes in the codebook. The goal of this process is to identify which regions can be selected without losing much representativeness in front of the non-quantized alternative. This problem corresponds to the one targeted by the *clustering* algorithms, widely studied by the pattern recognition community [1].

Automatic clustering algorithms provide solutions for this type of problems on two basic perspectives: supervised or non-supervised. In the supervised case, the amount of clusters

is predefined so the algorithm iteratively assigns elements of the training set to every cluster to later re-estimate the configuration of the cluster until a certain convergence criterion is reached. The K-means and Mean Shift algorithms are broadly adopted solutions among the scientific community to deal with this problem. The main drawback of supervised methods when building codebooks is the definition of its size. Choosing a too small value may produce a codebook which is not representative enough of the visual richness of the dataset. On the other hand, using a codebook which is too large will drive into unnecessary computation, which will turn the solution as inefficient.

Unsupervised clustering could provide an answer to the limitations presented by the supervised case, but they also pose some challenges, normally related to stability. In order to focus the research on the best configuration for feature design, the construction of the codebooks has been based on the K-means supervised solution [38]. The exposed limitations have been overcome by assessing the presented techniques for different codebook sizes, from very small to very large.

6.2.1 Choice of Regions

Given a clustering algorithm and an annotated dataset, the next question is how regions from the training dataset are to be used to build the codebook. The different options can be grouped in two big families: those which build the codebook upon regions representing foreground objects, and others which also take into account the regions that belong to the background.

Foreground Solutions

Solutions that only consider annotated foreground regions of the training dataset can be further divided into two options. The first one builds a specific codebook for every considered semantic class, while a second one combines the regions associated to any class of annotated object into a unified codebook.

Using *specific-class* codebooks is the equivalent of using *specific* visual descriptors in the quantized domain. They will provide very precise information for the category whose image parts are considered, but may not be very informative if tried to be used in other domains. Their main advantage is the size of the generated codebook, which is the smallest among all options. This property will drive to smaller feature vectors that will speed up the analysis and require less storage resources. However, if the final application must deal with multiple object classes, this approach may not be efficient because it will require extracting and processing a different feature vector for each codebook defined on every class. Moreover, such design would not allow sharing codewords between classes, even if they are very similar.

A more generic but somehow still controlled solution are the *foreground* codebooks. These

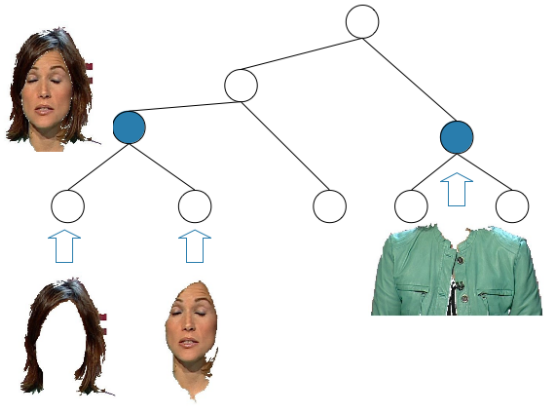


Figure 6.7: Sub-PT roots.

are inter-class codebooks learned by considering all the foreground parts in the training dataset. Their main advantage is that, by considering all object classes simultaneously, it is possible to exploit the redundancy that one may encounter if a separate codebook was to be generated for every class. In this scheme, a single codeword can effectively describe parts of different object classes. Reusing codewords makes this option more efficient than using as many class-specific codewords as semantic classes.

The usage of foreground annotations poses an additional question when determining which regions to consider in the framework of a PT. A local annotation of an object corresponds to a selection of one or several subtrees from the PT. The more visually heterogeneous is the annotated object, the more chances that more subtrees will be needed to describe the complete object. For example, Figure 6.7 shows and example where the four PT leaves that represent the *anchorwoman* object correspond to two PT subtrees. The creation of the codebook may consider the subtree roots, the leaves, or both, taking into account that the later case means using every node in the subtree.

Our work assumes that the training dataset is complete enough to represent every possible split of the considered objects. So, if the training set contains instances of the object split in several parts, then the test set will also present splits of the same order. This solution reduces the amount of image parts to consider during codebook creation, taking as a risk that if a new type of split appears in the test set, the resulting subparts may not be discriminatingly represented with the resulting codebook.

Foreground and Background Solutions

The second strategy in the choice of regions is to build a *generic* codebook by considering both foreground and background regions from the training dataset. This is the most flexible solution as no previous assumption is made about the semantic categories of interest. All

regions in the dataset are considered, which normally drives into a very diverse codebook. Working with *generic* codebooks allows re-usability and might even require less storage resources when the amount of classes of interests increases. In fact, this is exactly the same discussion related to generic or specific visual descriptors presented in Chapter 5.

In most situations, the portion of foreground regions represented by annotated objects is a much smaller fraction in comparison to the background set. If a supervised clustering method like K-means is used to create the codebook, it is also probable that the final portion of foreground codewords will also be much smaller than the one coming from the background. Given the higher relevance of the first ones, it may be advisable to adjust the codebook creation to ensure their presence in the final codebook. A possible solution is to generate separate codebooks from each set and later fuse them into a single codebook. In the presented work, this approach has been applied in equal parts: one half of the codewords coming from the foreground and the other half from the background.

Another common problem during codebook creation is the important computation requirements that the clustering algorithm may require. This situation rapidly arises when considering regions from the background, as in the case of *generic* codebooks. Assuming a random data distribution among background regions, the computation effort can be leveraged with a previous random subsampling of the background regions [64]. This strategy has also been applied in the results presented in this work.

In the classic vector quantization, it is important for a codebook to contain codewords belonging from both the foreground and the background. This way, the assignment of a feature to a histogram bin somehow implies the labelling of the image part to a foreground or background class. In the proposed solutions, the *generic* case satisfies this condition by nature, as it includes background parts in its construction. The *foreground* option can be equivalent if the visual diversity among categories is large. If this is the case, the object parts from different classes can play the same role as the background parts, attracting those parts that do not belong to the object. If the data distribution of the rest of object classes is similar to the background case, this is, very diverse, the *foreground* option will also accomplish the property in a more efficient way than the *generic* case.

6.3 Evaluation of Possibilistic Features

The proposal of possibilistic features has been compared with the region-based features results reported in Section 5.3. The same ETHZ dataset and data partitions have been used under different configurations according to the discussions presented in this chapter. Notice that in this experiment it is necessary to work with a training dataset different from the one used for testing. The training data is used to create the codebooks, so that the region features of the test PTs can be mapped onto the codebooks created for each of the five trials. Results are

	Retrieval	Detection	Segmentation
Shape	-5.5 %	+3.64 %	-1.48%
Color	+1.02%	+13.67%	+13.71%
Texture	-12.09%	-13.92 %	+1.54%

Table 6.1: Region features quantized on codebooks of 120 words compared to region features

comparable to the experiments in Section 5.3, as these ones were generated on the same test data and partitions.

In these experiments, the K-means clustering algorithm has been adopted to build the codebooks. Four sizes have been tested: 15, 30, 60 and 120 codewords, which correspond with the four points in every plotted curve. Results are represented in graphs where the performance measures are dependent of the codebook size (NWords). The zero-size case in the retrieval graph corresponds to the random baseline results.

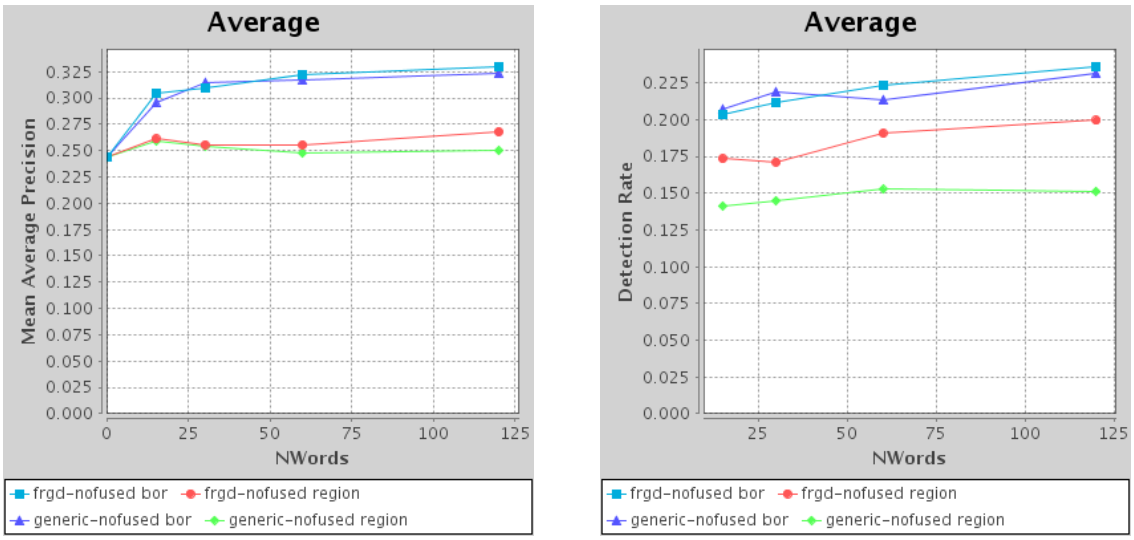
Each of the three visual descriptors has been tested separately, so Figure 6.8 presents the results with the Contour Shape descriptor, Figure 6.9 with the Dominant Colors descriptor and Figure 6.10 with the Edge Histogram. Four different graphs are shown: the image retrieval labelled as (a), the object detection labelled as (b), the object segmentation as (c) and the search time as (d). Each of these figures provides results considering a codebook based only on foreground regions (frgd) or on all regions (generic); and also compares the performance between the features computed with a soft-mapping on the region features (region) or on the HBoR features (bor). The values have been obtained averaging the measures for every instance in the test dataset, ignoring the object class they belong to.

These results provide answers with respect to the multiple configurations that can be considered when working with codebooks:

Remark 1. *Little accuracy loss due to the introduction of codebooks*

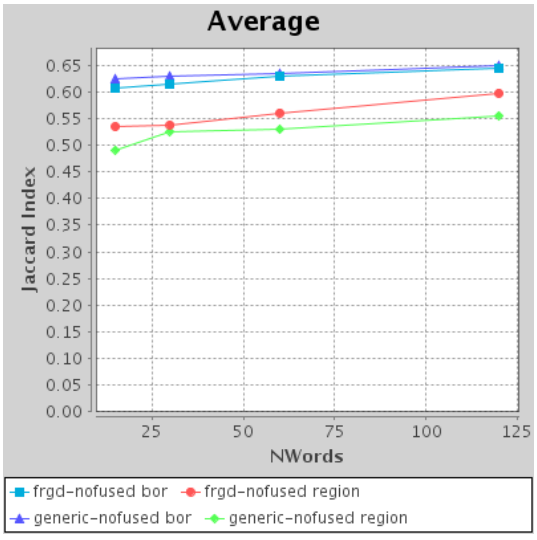
The first question is what the impact is of changing the direct similarity assessment between the visual descriptor, presented in *Average* graphs from Figures 5.1, 5.2 and 5.3, and a comparison based on the feature vectors obtained by the soft-mapping on the codebooks, which corresponds to the curves labelled with *region* in Figures 6.8, 6.9 and 6.10. Table 6.1 quantifies the changes in performance of the largest considered codebook with respect to the non-quantized approach. Results show that there is little impact in the Shape case, that the results of color slightly improve but that the performance with texture suffers a little decrease, too. On average then, there is no significant loss due to the introduction of codebooks as long as their size is large enough. This loss is expected as any quantization process like the one applied during the codebook creation introduces a simplification with respect to the original feature, even if the feature vector is created through a soft-mapping.

Remark 2. *HBoR improves accuracy over region-based codebooks for shape and texture*

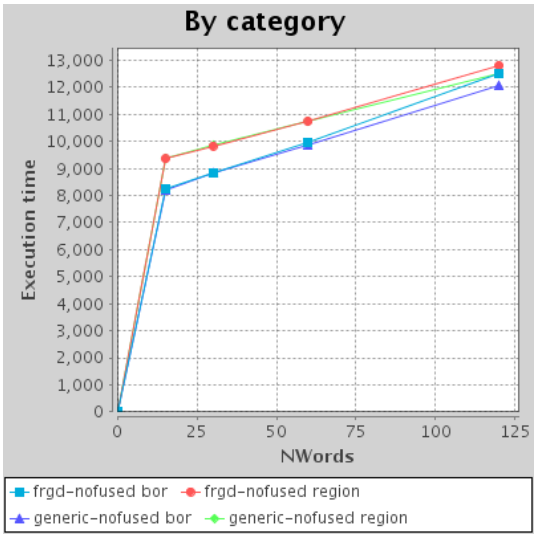


(a) Image Retrieval

(b) Object Detection

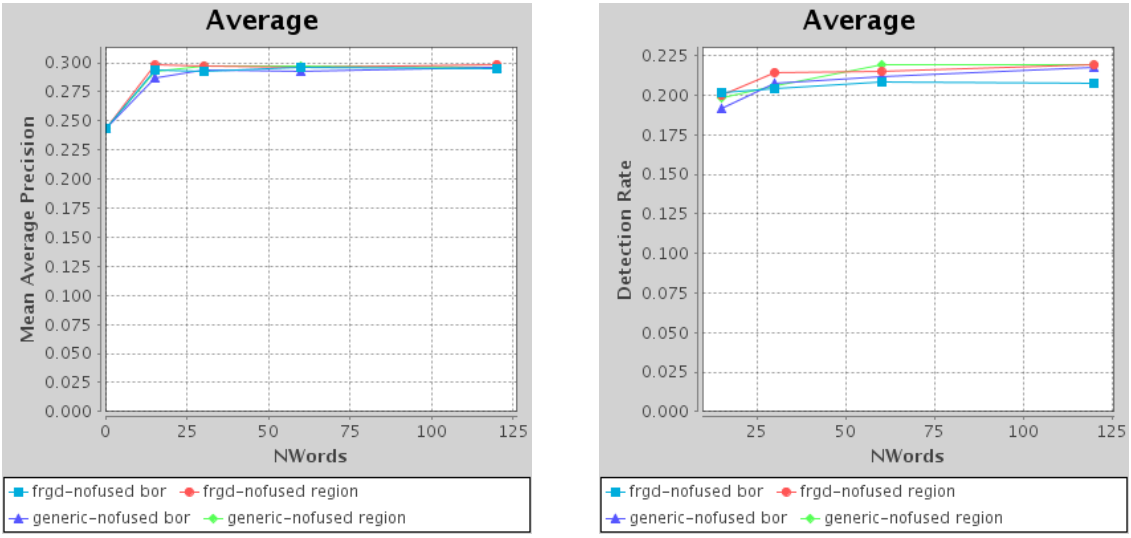


(c) Object Segmentation



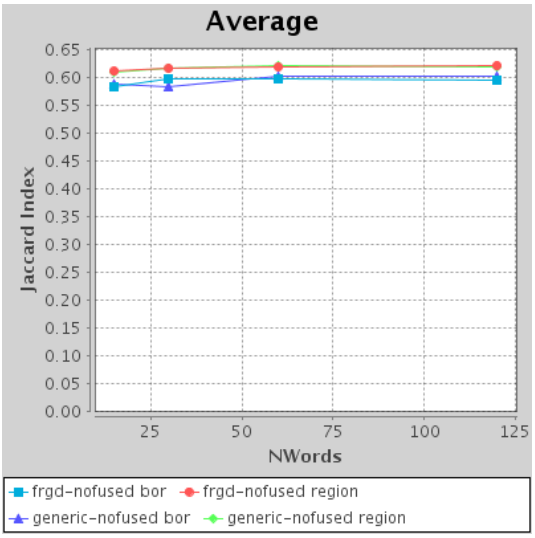
(d) Search time

Figure 6.8: Performance with a codebook of the Contour Shape descriptor.

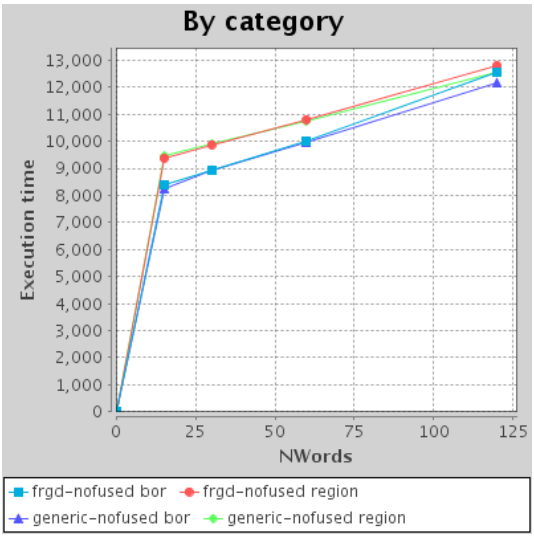


(a) Image Retrieval

(b) Object Detection

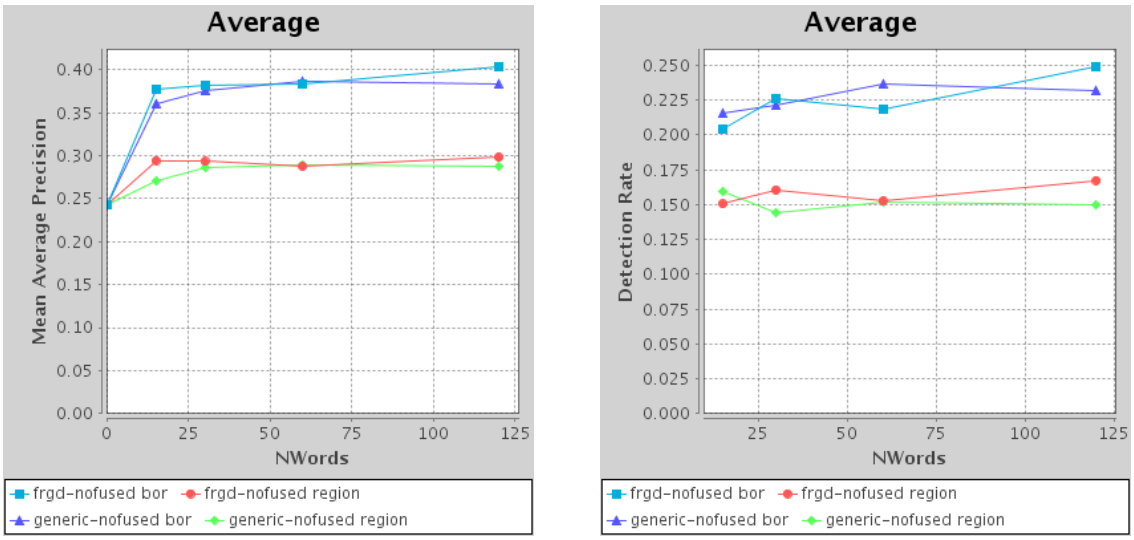


(c) Object Segmentation



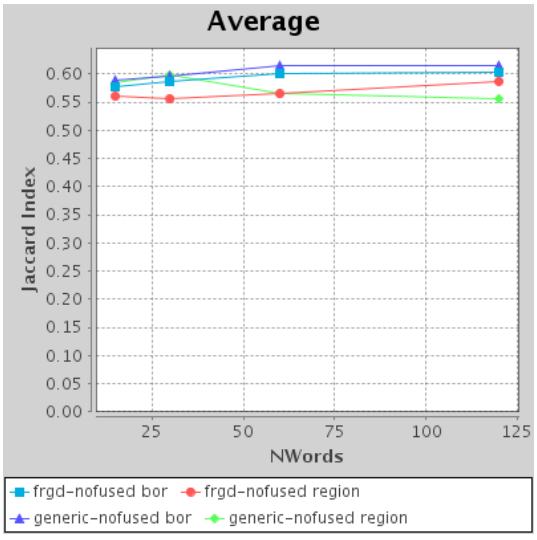
(d) Search time

Figure 6.9: Performance with a codebook of the Dominant Colors descriptor.

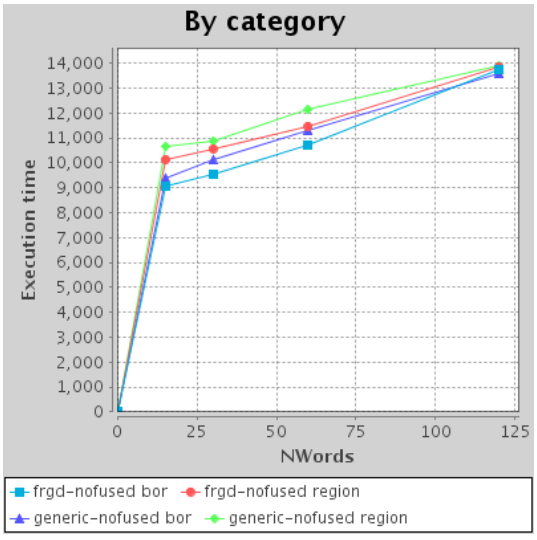


(a) Image Retrieval

(b) Object Detection



(c) Object Segmentation



(d) Search time

Figure 6.10: Performance with a codebook of the Edge Histogram descriptor.

	Retrieval	Detection	Segmentation
Shape	+16.20 %	+22.92 %	+5.33 %
Color	0 %	+13.66 %	+8.96 %
Texture	+19.17 %	+28.35 %	+3.07 %

Table 6.2: HBoR features on codebooks of 120 words compared to region-based features

Once studied the impact of mapping region features onto a codebook, the next observation focuses on the proposed Hierarchical Bag of Regions (HBoR) with respect to only using the region feature vector. The curves in the graphs show that this approach is clearly beneficial in the case of the shape (Figure 6.8) and texture (Figure 6.10), and it has hardly impact in the case of color (Figure 6.9). This may be due to the replication of functionality between the Dominant Colors descriptors and the HBoR: both represent the colors present in the subtree. The main difference is that the Dominant Colors descriptor limits the amount of distinctive colors to eight, while the HBoR have no bound.

Remark 3. *HBoR improves accuracy over region-based descriptors*

If HBoR features are compared with the non quantized, the problems with the shape and texture descriptors reported in Table 6.1 disappear and the obtained results improve the initial figures in every measure, as show in Table 6.2 . These results prove that the proposed scheme is overcoming some of the problems analysed in this chapter, even when the considered objects are not clearly described as the union of semantically distinctive parts, like the majority of instances in the ETHZ dataset. This experiment also validates the fast bottom-up extraction of HBoR features for combinations of PT nodes, as several of the considered query objects do not correspond to any node in the PT, but a combination of them. This configuration is tested because the extraction of visual descriptors is a computationally intense task with a significant impact on usability when performed at query time, as it increases the time response of the system. Given the satisfactory results obtained, this approach is considered as validated and will be applied in future experiments.

Remark 4. *Little gain when adding background parts to a multiclass codebook*

The graphs also offer an interesting result in terms of what image parts, only foreground or foreground plus background parts, are to be considered when building the codebooks. According to the experiments, there is no significant impact in terms of retrieval and object description. This is because the parts of one object class act as background parts for the rest of categories. This result is important during the codebook creation because the larger computation effort that requires considering background regions seems unnecessary in this context. Given that most visual databases aim at solving multi-class problems, this result suggests that constructing codebooks with regions associated to multi-class objects is enough to obtain satisfactory codebooks.

The presented experiments have not considered the case of specific-class codebooks. This decision has been taken because in the large repository framework considered by this work is intended to use a single codebook for all categories. This scenario requires an extensive and representative amount of training data to use their image parts to create this multi-category codebook. This adopted strategy aims at defining a vocabulary of multi-purpose visual primitives that should be rich enough to represent a great diversity of visual concepts in the repository.

6.4 Summary

This chapter has explored the use of codebooks in the context of Partition Trees. The initial Section 6.1 has presented the Parts Space as a new feature space suitable for the application of pattern recognition techniques to the problem of multi-part object detection and segmentation. This Parts Space relies on the combination of two principles, the possibilistic vector quantization presented in Section 6.1.1 and the application of a max pooling strategy on the Partition Tree, as proposed in Section 6.1.2. These novel feature vectors are name *Hierarchical Bag of Regions (HBOR)* and represent the first main contribution of this thesis. Section 6.2 has discussed the different options when choosing which regions are to be considered to construct the codebooks when local annotations of multiple object categories are available. Finally, Section 6.3 has compared the impact of adopting a codebook-based solution when compared with directly working with the region visual descriptors. The obtained results point that the introduction of region-based codebooks hardly affects the performance despite its quantization nature. Moreover, the HBoR clearly outperforms the other approaches thanks to the hierarchical structure provided by the PT. Experiments also indicate that, in the case of diversity of multiple object classes, effective codebook can be built by only considering the regions associated to the foregorund object parts.

Chapter 7

Fusion of Visual Modalities

The diversity of visual attributes presented in Chapter 5 is a proof of the variability of interpretations of visual information. This multiplicity of features must be handled at some point of the analysis to provide a final result instead of a separate output for each different feature. Every visual descriptor has normally associated one or several similarity metrics that provide a core function to evaluate what is similar or dissimilar to a certain query or model of an object. Previous work [6] has shown that combining different sources of information in an appropriate way improves the performance of a retrieval system. However, deciding how to combine the visual distances obtained from every descriptor is a challenging task. This problem is not exclusive to the visual domain and it is also commonly formulated in any pattern recognition context with different types of data sources.

Feature combination can be handled with two basic strategies [84]. The first one, named *early fusion*, considers combining the multiple features into a single value and generate a single similarity score with it. On the other hand, one can also obtain a similarity score for each type of feature and combine the separate results into a final one. This second option is normally referred as *late fusion*. Past work in the field of multimedia semantics has targeted the issue of what option is the best, with results that basically state that the answer depends on the visual class. Given the different similarity metrics associated of the visual descriptors (color, texture and shape) considered in this thesis, the most appropriate scheme is the *late* one.

In the two previous chapters, two types of features have been explored: those extracted directly from the regions and a second type that results from a soft-mapping of the first type onto codebooks. In both cases, the early fusion requires solving two problems. Firstly, ensuring that the values for each result have the same interpretation for every feature. This requirement is normally addressed by a normalization process. Secondly, the normalized values must be fused by a certain combination rule, such as a weighted sum, product, max/min, etc. In the present work, the normalization problem has been the main issue of study, and

the obtained results have been combined with common solutions. Previous work by Belkin [6] pointed out that averaging is the best option in general, while another publication by Tax [90] indicates that the product rule tends to provide the best performance when combining classifier scores.

7.1 Fusion of Region Features

The metrics associated to every descriptor try to quantify visual similarity as a human would from a perceptual point of view. Ideally, the numerical scale generated by the similarity measure should match the mental scale for the *similarity* concept that would rule a human perception. It is desirable that every similarity measure would use the same subjective scale, so that a value obtained with one visual descriptor would correspond to the same subjective interpretation if obtained with another visual descriptor. Unfortunately, this is not normally the case, as every metric is defined by different authors and generally addressing different purposes. As a result, in general, the obtained values are not comparable. This is also the case for the MPEG-7 based descriptors considered in this work.

Figure 7.1 exemplifies a desirable solution of a normalization process, where a different α distance associated to each of the considered MPEG-7 descriptors is mapped into a β value whose subjective interpretation in term of similarity is harmonized. In addition, the zero (identical) and mean (different) points are also forced for coherence with the original distance metric. The notion of "different" is associated to the mean (μ) of the distance because empirical problems have shown that the notion of "completely different" is already satisfied at this value (and probably earlier).

One method of solving the diversity of criteria would be to design a standardized and broad set of experiments with several users who would subjectively evaluate the visual similarity between a collection of shape, color and texture patterns. This solution is demanding in terms of setting up the experiments and obtaining a representative amount of users. A second possibility is the application of a statistical analysis on a dataset annotated from a semantic point of view. This type of data is publicly available and, with the appropriate processing, is already a valuable source of information even from the perceptual point of view. This second option has been adopted in this work because, in multimedia databases, semantic annotations can be naturally obtained during the normal workflow of the system. Whether through perceptual or semantic annotations, the outcome of the process is a set of parameters that normalize the similarity metric associated to every visual descriptor.

Normalization algorithms of region-based features typically learn their parameters after analysing a representative set of similarity values. There are two basic configurations depending on the reference of such similarity value. The first option is to compute the mutual similarities among a collection of elements, the second configuration is to choose a single ele-

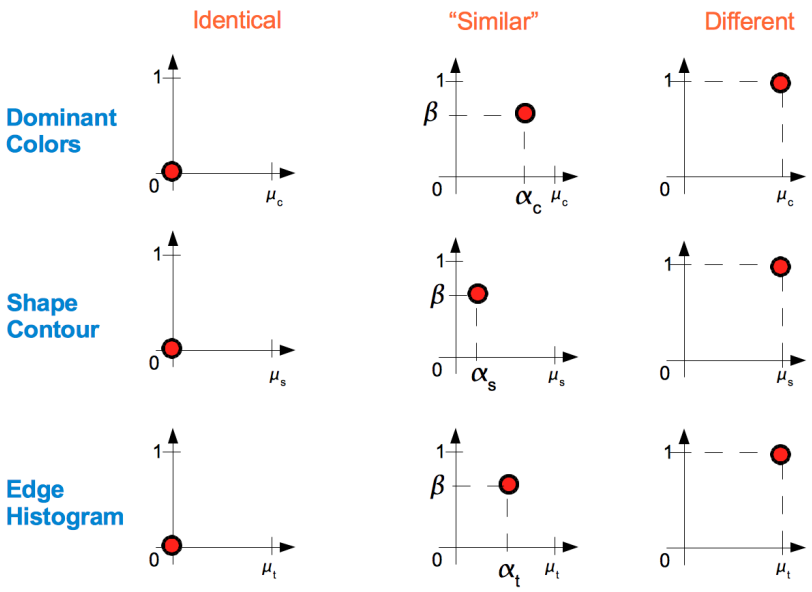


Figure 7.1: Expected behaviour of a normalization process

ment as a common reference and calculate the distance between all elements and this one. The first configuration is the one available in a supervised learning framework, where a training dataset is processed to adjust the system for future unobserved inputs. The second situation is the traditional one in a retrieval system, where a query becomes the reference from which every element in the database is compared to. The generated ranked list plays a similar role to the training set of the first case, with the particularity that each of its elements has also been assigned a position in the ranked list.

7.1.1 Linear normalization

All considered normalization solutions start by transforming a distance value d into a score x that takes values between $[0, 1]$. Comparisons between identical elements correspond to a 0 minimum distance, but the maximum possible distance can vary depending on the visual descriptor. This variability can be harmonized by learning the maximum possible distance d_{max} from the training dataset and using it to obtain the score x, as shown in Equation 7.1.

$$x = 1 - \frac{d}{d_{max}} \tag{7.1}$$

It has been argued that the presence of outliers might seriously damage the performance of such normalization procedure, and that better results would be obtained using a Gaussian normalization [72]. Equation 7.2 uses the mean (μ_x) and standard deviation (σ_x) of the similarities to map the 99% of them into the range of $[0, 1]$. This normalization is complemented by mapping all outliers to the extreme values. This proposal is very similar to the *z-scores*,

where instead of dividing by $3\sigma_x$, the divisor is defined as σ_x.

$$\bar{x} = \frac{1}{2}\left(1 + \frac{x - \mu_x}{3\sigma_x}\right) \tag{7.2}$$

The normalization of the MPEG-7 distances used in this thesis has started with a linear transformation inspired by both equations 7.1 and 7.2. A linear normalization based on the mean of distances (μ) has been adopted because, after a visual exploration of the MPEG-7 distances, it has been considered that all distances over the mean must be equally interpreted as *different*. As a result, Equation 7.3 has mapped the 99% of distances into the $[0, 1]$ range, while the remaining outliers are assigned to the extreme values.

$$x = 1 - \frac{d}{\mu} \tag{7.3}$$

Distributions of linearly normalized scores

The mapping of distances into a range of $[0, 1]$ similarity scores facilitates the comparison between different visual descriptors. An experiment has been run on the ETHZ dataset to generate a set of representative scores. Each instance in the training dataset has been used to define a query, which has been assessed with the rest of the PTs in the training dataset. The score obtained from each comparison corresponds to the best match between the query region and one of the 99 regions in the PT. The process has been repeated for each of the three considered visual descriptors: color, texture and shape.

Figure 7.2 shows the three pairs of resulting histograms in three rows, one associated to each visual descriptor. The left column corresponds to the scores of those retrieved regions considered *relevant* for the query, and the right column the scores when the retrieved region was considered *irrelevant*. A retrieved region was labelled as *relevant* if, when compared to the ground truth object segmentation, its bounding box satisfies the Pascal Criterion and its segmentation mask achieves a Jaccard Index of at least 0.7. These two metrics have been previously defined in Section 1.3.3. The plotted data corresponds to the training dataset of one of the five cross-validation trials considered in all presented experiments. Details on the data partition can be found in Section 1.3.4.

The obtained figures clearly show that every visual descriptor presents different distribution shapes. In some cases there are peak values at the highest and lowest bin, which basically indicates the behaviour of a distance metric that collapses all distances over/below a certain threshold in the same value.

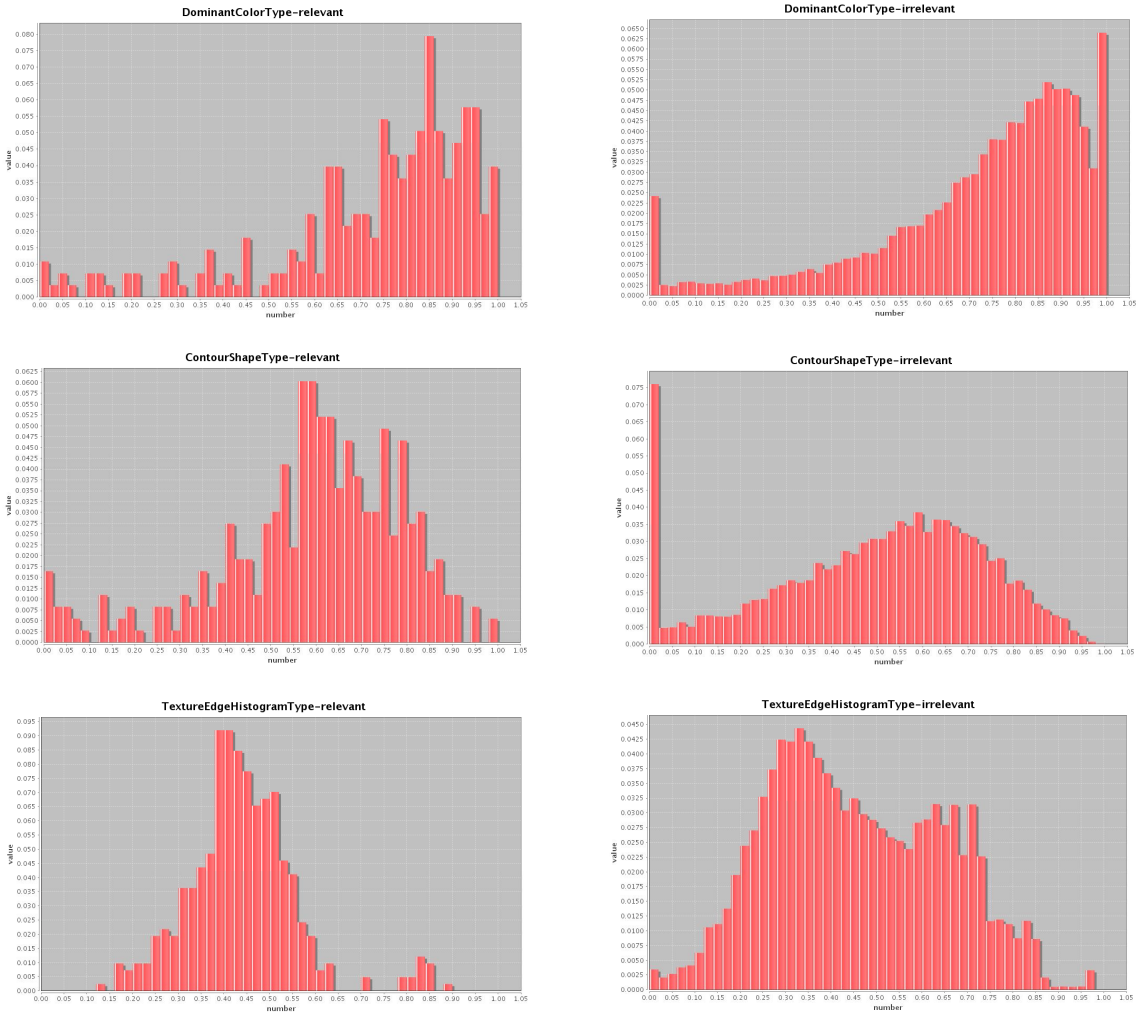


Figure 7.2: Histograms of the relevant (left) and non-relevant (righ) scores.

7.1.2 SVM normalization

A first tested approach for score normalization addresses the situation as a classification problem. If it is possible to draw a sample of similarity scores that successfully capture the concept of *similar* and another set of scores that represent a *non similar*, the labelled set of scores can be used to train a classifier that will estimate the degree of normalized *similarity* of a new score.

The challenge of this approach remains into the choice of the which similarity scores are to be labelled as *similar* and *non similar*. The histograms depicted in Figure 7.2 correspond to the scores for *relevant* and *non-relevant* hits which, although correlated, do not allow a precise labelling in terms of similarity given the visual diversity of the considered objects.

The adopted solution is inspired on previous proposals of pseudo-relevance feedback [104], where the top elements of an obtained ranked list are considered *relevant* for the query, while the last one are labelled as *non relevant*. The pseudo-relevance feedback approach uses these labels to train a classifier that recalculates the relevance scores of the hits. In our problem, the concept of *relevance* is substituted by the *similarity* one, so the first hits are labelled *similar* and the last ones *non similar*. In our experiments, the top and last 5 hits of every ranked list were used to train an SVM classifier. The output of such classifier, one for each visual descriptor, corresponds to the normalized value valid for fusion.

7.1.3 Weibull normalization

Recent research by Scheirer et al. [78] bases the feature normalization on the Extreme Value Theory (EVT). This theory provides an alternative to the Central Limit Theorem as, instead of focusing on the median values, its interest relies on the extreme values of a distribution. This approach matches the classical visual retrieval problem, since the first hits in the ranked list can be associated to the extreme values of a distribution. EVT states that their related scores will follow a Weibull distribution, a curve that is determined by two parameters k and λ and defined by the probability density function (pdf) in Equation 7.4.

$$f(x) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{(k-1)} e^{-\left(\frac{x}{\lambda}\right)^k} \tag{7.4}$$

According to the EVT, the best similarity scores obtained for any descriptor will follow a Weibull distribution. If this assumption is accepted, these values can be normalized through the Cumulative Distribution Function (CDF) of the Weibull distribution, shown in Equation 7.5. Considering the non-normalized score as the input x, this curve will map any value to a comparable range $[0, 1]$. Scheirer names these type of normalized scores as *w-scores*.

$$\bar{x} = F(x) = 1 - e^{-\left(\frac{x}{\lambda}\right)^k} \tag{7.5}$$

In the instance search problem studied in this thesis, the similarity scores correspond to the most similar node in the PT with respect to the region so, in other words, it is the best result after an intra-image region retrieval task. The scores obtained with the rest of nodes from the target PT are discarded, so this scenario clearly sets the problem into the high extreme of the data distribution that would be obtained if all scores were analysed. Considering only the best scores of each intra-image search supports the assumption that the Extreme Value Theory is applicable on them.

Scheirer observed that, given a query, it is common to obtain a much larger amount of samples that do not match the query (irrelevant) than those that do match the query (relevant). Moreover, the score distributions for match and non-match hits will probably be different. So they proposed to focus on the non-match hits, learn their Weibull parameters and use the CDF of these non-match dataset to normalize all scores. This consideration completely applies to the histograms in Figure 7.2, that show how non-relevant distributions, on the right, are less noisy than the relevant ones because many more samples were obtained in these experiments with the ETHZ dataset.

Once the histograms of the non-relevant scores have been visualized, the next step is trying to estimate the parameters of the Weibull distribution that fit to the measured data. Given the mathematical framework for normalization, its application requires finding the parameters of the Weibull distribution that match its Probability Density Function (PDF) to the obtained histograms. The applied solution was based on the Least Squares algorithm, which allows the estimation of the parameters of a line. The Weibull CDF does not present a linear behaviour, so it is necessary to transform it according to Equations 7.6 to convert the expression into a linear form.

$$\begin{aligned} F(x) &= 1 - e^{-\left(\frac{x}{\lambda}\right)^k} \\ \ln\left(1 - F(x)\right) &= -\left(\tfrac{x}{\lambda}\right)^k \\ \ln\left(ln\left(1 - F(x)\right)\right) &= -k\ln\left(\tfrac{x}{\lambda}\right) \\ \ln\left(ln\left(1 - F(x)\right)\right) &= -k\ln(x) - \ln(\lambda) \end{aligned} \tag{7.6}$$

The resulting formula can be identified with a linear equation where estimating a and b is equivalent to estimating the Weibull parameters, as shown in Equation 7.7.

$$\begin{aligned} \ln\left(ln\left(1 - F(x)\right)\right) = -k\ln(x) - \ln(\lambda) &\equiv y = a + bx \\ k &= b \\ \lambda &= e^{-\frac{a}{k}} \end{aligned} \tag{7.7}$$

The estimation of parameters a and b corresponds to fitting a line to the points generated by every observation. This type of problem is called linear regression and has a simple and closed solution applying a least squares estimation [46]. The solution minimizes the square error accumulated between each observation y_i and its linearly estimated value obtained by

Trial	$\hat{\mu}$	$\hat{k}$	$\hat{\lambda}$
Dominant Colors	1.047	1.951	1.090
Shape CSS	0.502	1.989	0.656
Edge Histogram	9.524	2.433	0.501

Table 7.1: Estimated parameters for normalizing similarities (trial 1)

combining x_i with estimated parameters $\hat{a}$ and $\hat{b}$.

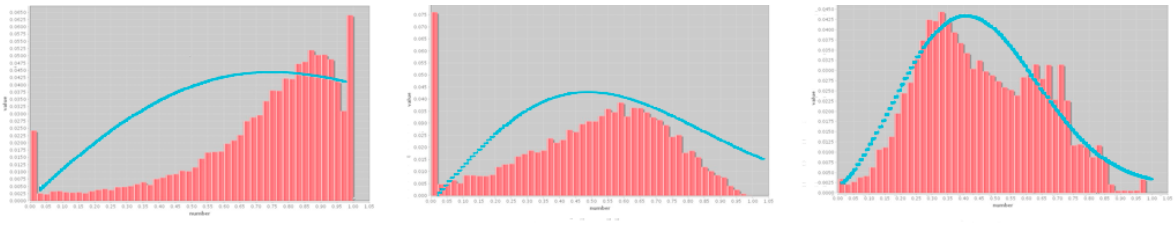
A closed form for the estimation of the two linear parameters a and b is presented in Equations 7.8, where N corresponds to the total amount of considered observations.

$$\begin{aligned} \hat{b} &= \frac{\sum_{i=1}^{N} x_i y_i - \frac{1}{N}\left(\sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i\right)}{\sum_{i=1}^{N} x_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N} x_i\right)^2} \\ \hat{a} &= \frac{1}{N}\sum_{i=1}^{N} x_i - \frac{\hat{b}}{N}\sum_{i=1}^{N} y_i = \bar{y} - \hat{b}\bar{x} \end{aligned} \tag{7.8}$$

The estimation of the Weibull parameters has been tested on the same collection of data used for comparing the histograms of relevant and non-relevant hits. A first analysis of the obtained parameters can be performed based on the estimated values shown in Table 7.1. This table clearly shows how the mean distance $\hat{\mu}$ differs among descriptors. This value has been used for a first normalization of the distances, according to Equation 7.3. The estimated Weibull parameters $\hat{k}$ and $\hat{\lambda}$ also differ among descriptors, as already observed.

Figure 7.3 compares the actual histogram of non-relevant matches with the Weibul PDF plotted with the parameters from Table 7.1. All the plotted data corresponds to one of the cross validation trials considered in the experiments with the ETHZ dataset. The obtained curves fit the envelope of the histogram with different accuracy. While the cases for *shape* and *texture* successfully follow the histogram, the *color* curves present different behaviour in terms of convexity/concavity. A similar comparison is depicted in the graphs of Figure 7.4, where the actual and synthesized CDFs have been drawn. As these graphs have been generated with the same data used in Figure 7.3, also the main divergence between the two type

Despite the mediocre results for the *color* case, the Weibull fitting was adopted as an strategy for normalization as it provides solid statistical grounds, adjusts well to the *shape* and *texture* scores and previous works [78] [48] have reported about its successful application.



Color Shape Texture

Figure 7.3: Real histogram (red) and estimated Weibul PDF (blue)

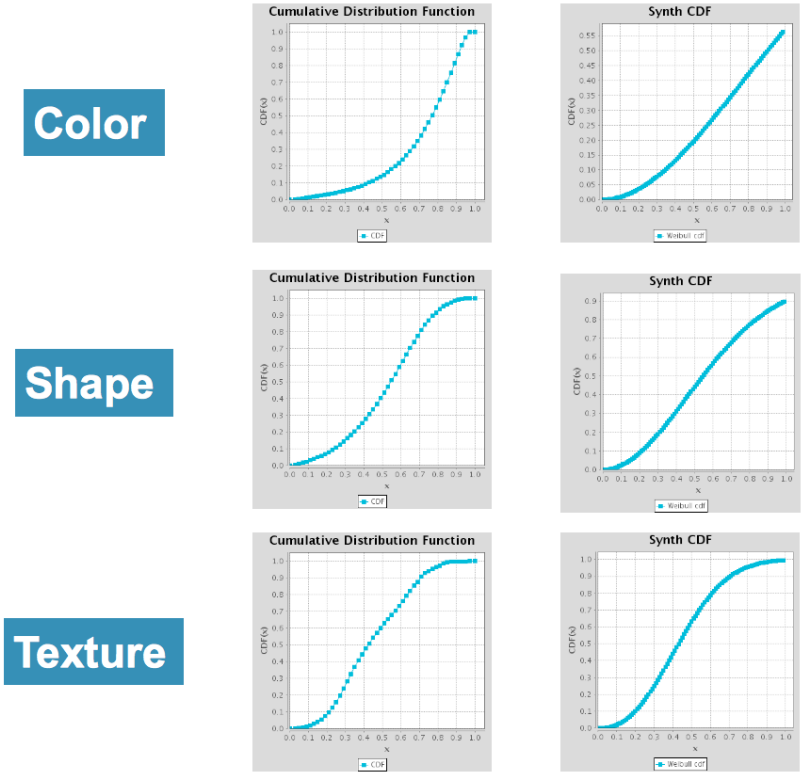


Figure 7.4: Actual and Weibull synthesized CDFs

	Max	Min	Product
z-scores	24.06%	32.33%	2.85 %
SVM	16.34%	20.68%	11.95%
w-scores	14.84%	29.87%	5.74%

Table 7.2: Gain with average fusion in the image retrieval experiment

7.1.4 Evaluation of early fusion techniques on region-based descriptors

The experiments previously presented in Section 5.3 were repeated, considering now three different fusion techniques: the modified z-scores from Equation 7.2, the probability scores obtained with an SVM classifier presented in Section 7.1.2 and the w-scores introduced in Section 7.1.3.

The experiments results of the experiments are presented in Figures 7.5, 7.6 and 7.7, both by individual categories as well as averaged values among all the considered instances.

Remark 1. *Average is the best fusion rule*

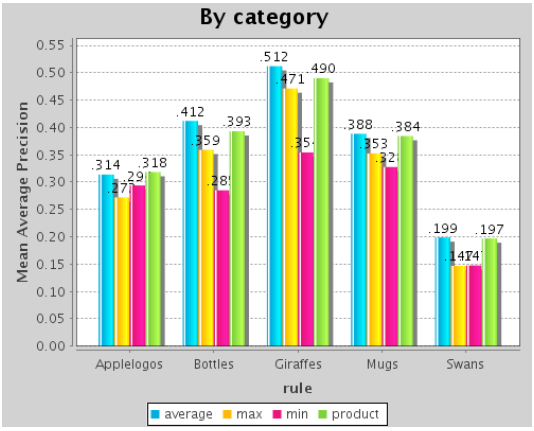
All graphs showing the aggregated results (on the right) and most categories in all tasks present their best results for the *average* fusion rule. Table 7.2 shows the gain of the *average* fusion with respect to the other considered options for the task of image retrieval. The table indicates that averaging normalized scores is the best rule for fusion, independently from the normalization technique. In addition, the second best option is the *product*, followed by the *max* rule. These conclusions agree with the ones achieved in [6] and [90].

Remark 2. *z-scores and w-scores outperform any other configuration*

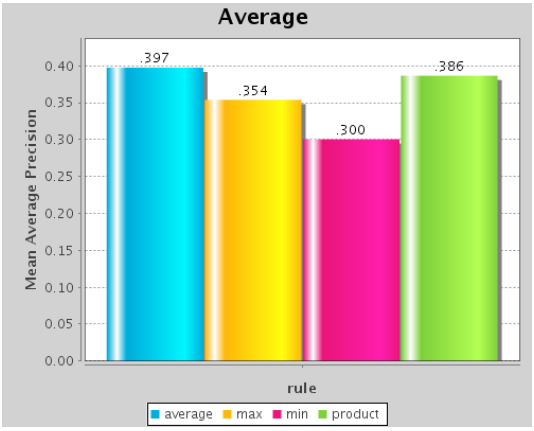
A second observation is that the z-scores and w-scores fusions generate the best results among the considered solutions. Table 7.3 presents the results obtained for each task, together with the values previously obtained in Section 5.3 for separate descriptors. In general, the results for the z-scores are slightly better than those for w-scores when considering non-quantized descriptors.

7.2 Fusion of Visual Words

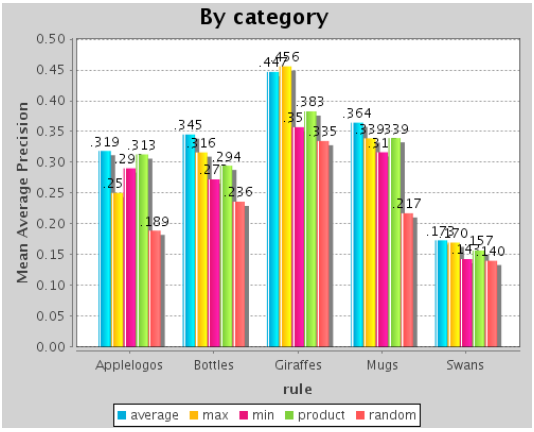
In the context of codebooks introduced in Chapter 6, the assessment of the similarity between two regions is achieved by the comparison of their quantized feature vectors through a distance metric, such as the Euclidean distance (Equation 6.1) or the cosine distance (Equation 6.2). In the framework the possibilistic quantization presented in 6.1.1, the contents of the feature vectors correspond to the similarity scores between the represented regions and the codewords. The introduction of different types of visual descriptors poses for the codebooks case the same



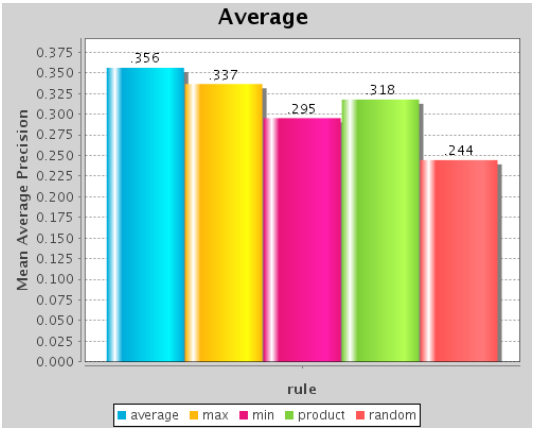
(a) Z-scores by category



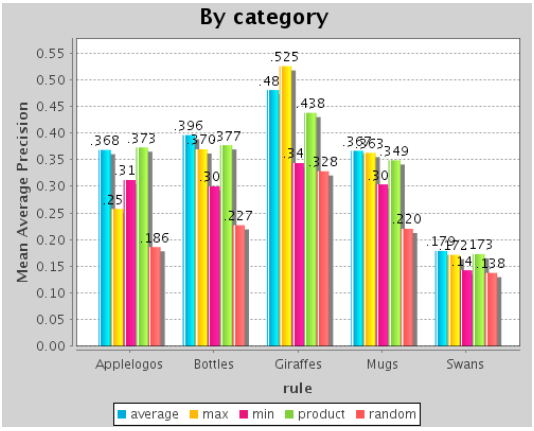
(b) Z-scores (averaged)



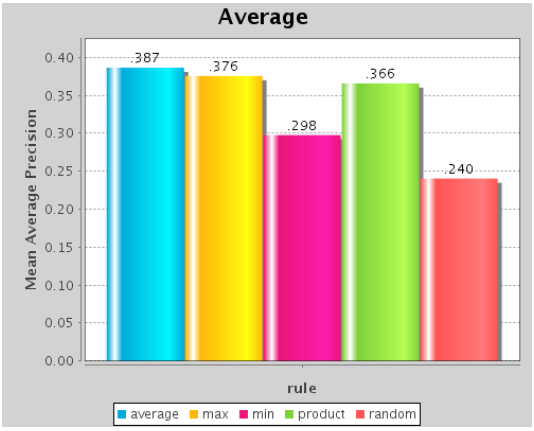
(c) SVM by category



(d) SVM (averaged)

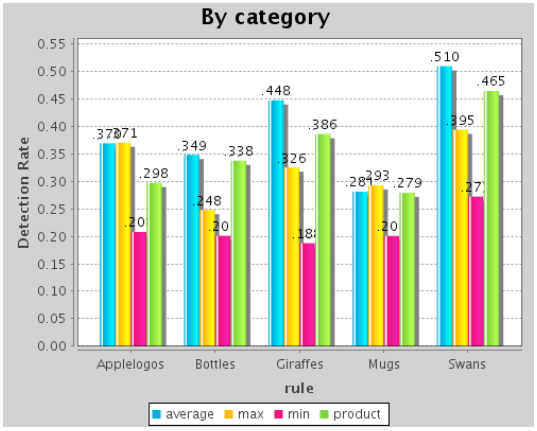


(e) W-scores by category

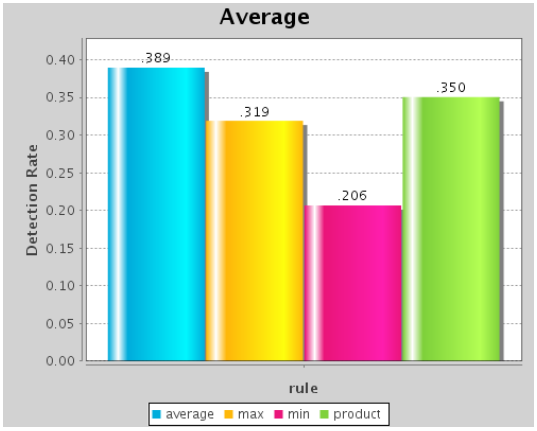


(f) W-scores (averaged)

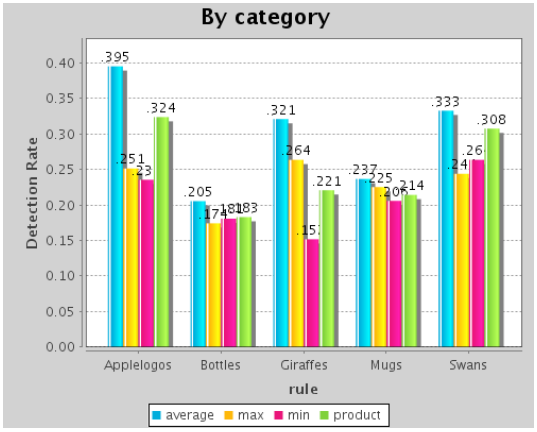
Figure 7.5: Image Retrieval by fusing normalized scores.



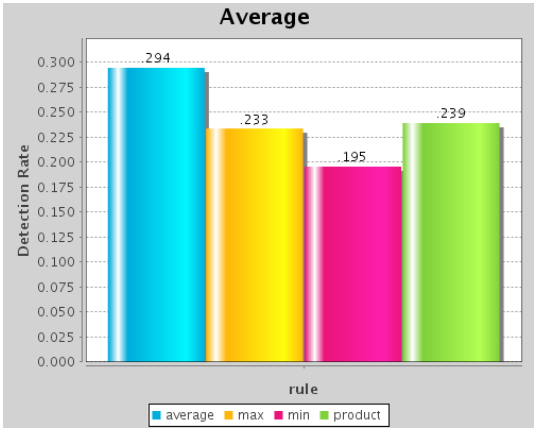
(a) Z-scores by category



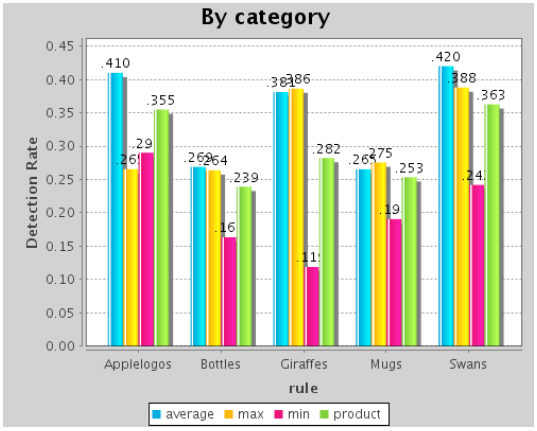
(b) Z-scores (averaged)



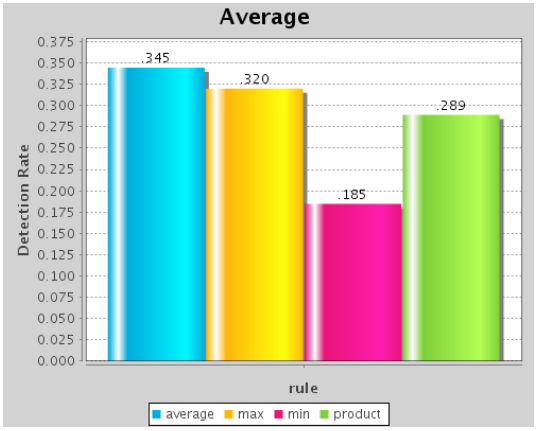
(c) SVM by category



(d) SVM (averaged)

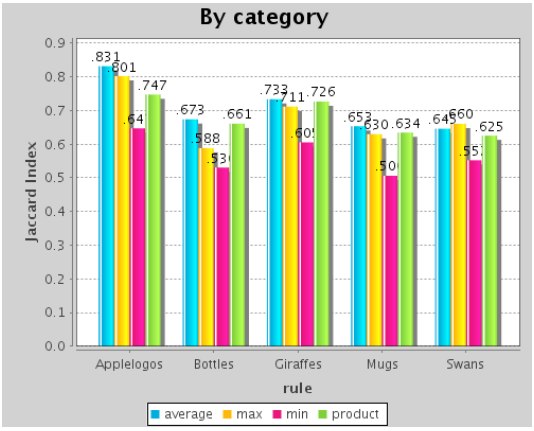


(e) W-scores by category

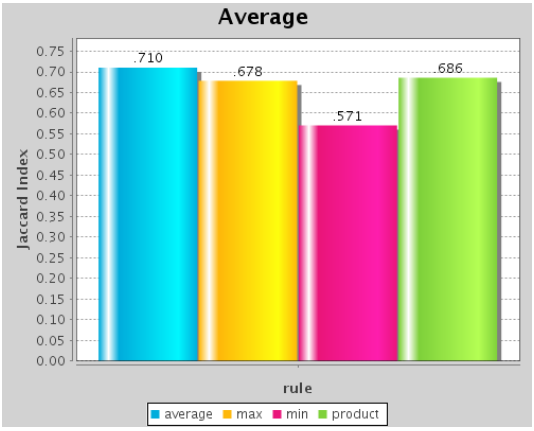


(f) W-scores (averaged)

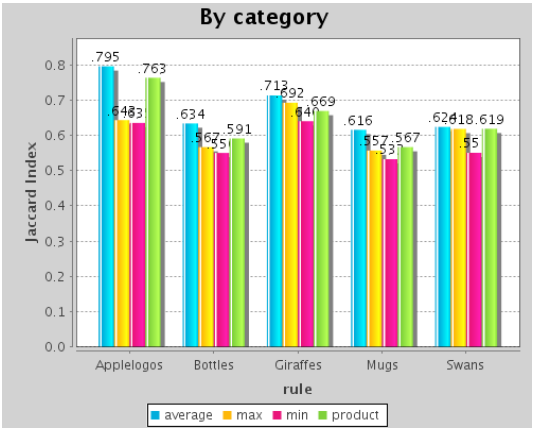
Figure 7.6: Object Detection by fusing normalized scores.



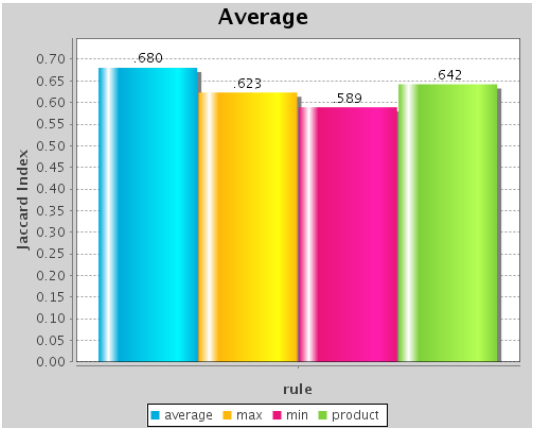
(a) Z-scores by category



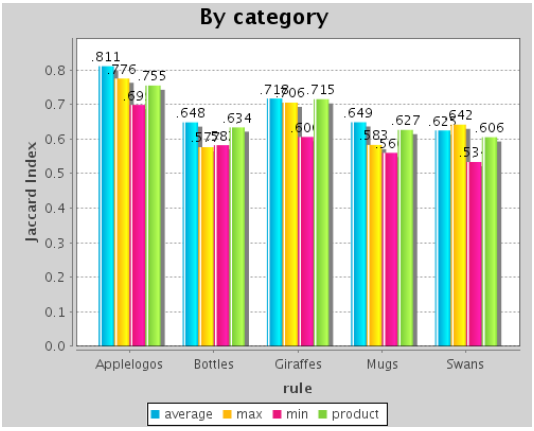
(b) Z-scores (averaged)



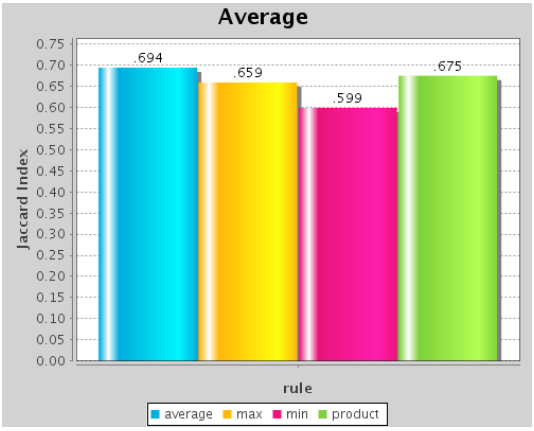
(c) SVM by category



(d) SVM (averaged)



(e) W-scores by category



(f) W-scores (averaged)

Figure 7.7: Object Segmentation by fusing normalized scores.

	Retrieval	Detection	Segmentation
Color	0.295	0.183	0.540
Shape	0.284	0.192	0.607
Texture	0.339	0.194	0.586
z-scores	0.397	0.389	0.710
SVM	0.356	0.294	0.680
w-scores	0.387	0.345	0.694

Table 7.3: Single feature and fusion by average for region-based descriptors

heterogeneity problems in terms of scaling and interpretation as in the region-based features discussed in the previous Section 7.1.

A first step for dealing with this heterogeneity is normalizing the similarity scores in the word descriptors. By doing that, the similarity assessments obtained with different visual descriptors become numerically comparable. However, an open issue remains related to the moment when these normalized scores are to be fused. The use of codebooks allows two solutions to this question, which are presented in Section 7.2.1 and compared in Section 7.2.2.

7.2.1 Fusion strategies

The usage of multiple visual descriptors when dealing with codebooks can be addressed with two different schemes: (a) create a single codebook based on an early fusion of the normalized scores or, (b) create a separate codebook for each visual descriptor and fuse the distances obtained on each of them.

Early fusion of normalized scores

The first solution creates a single codebook that combines all considered descriptors. This codebook is built with a clustering algorithm that computes the distance between elements with normalized distances. That is, every step in the algorithm that compares two candidate codewords computes the normalized scores for each considered descriptor and fuse them to obtain a similarity measure. This scheme generates a single codebook, so that each observation is represented with a single feature vector. This components of the visual words are the normalized and fused scores between the region's descriptors and the ones of each codeword.

Building a single codebook drives to very precise codewords, as they concentrate all types of information of the codewords: color, texture and shape. If considering an analogy with natural language, this would mean using words like red-dotted-circle or blue-stripped-square. Figure 7.8 shows an example of a codebook built by combining the color, texture and shape features.

Figure 7.8: A single codebook with fused features.

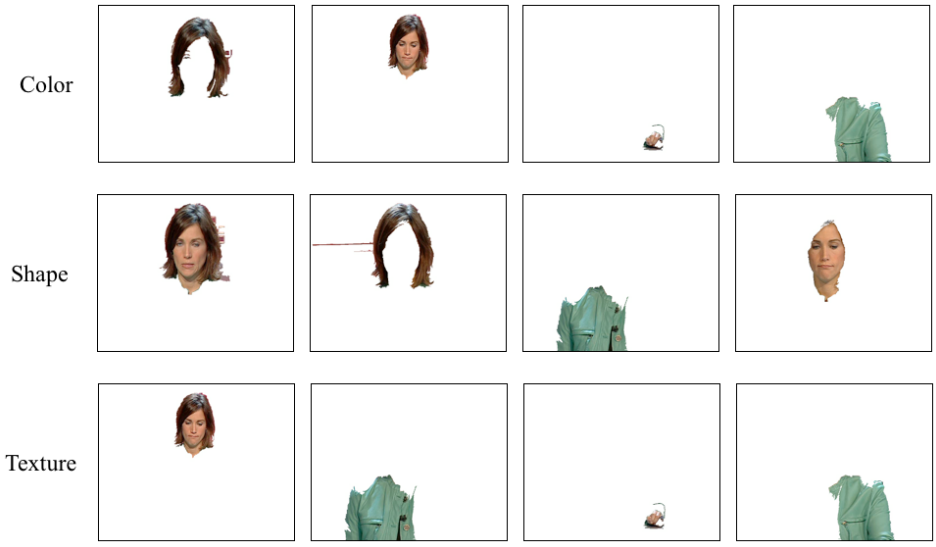


Figure 7.9: A codebook for every considered visual feature.

Late fusion of distances

The second solution requires building a codebook for each considered visual descriptor; this is: one codebook for color, one for texture and one for shape. This solution reduces the precision of every codeword but is much more flexible in terms of possible combinations. This is exactly the strategy adopted by natural language, where color, texture and shape adjectives can be freely combined. Figure 7.9 shows three different codebooks, each of them obtained by applying the same clustering algorithm on a single visual feature.

Working with different codebooks will produce a set of feature vectors associated to every considered region, one for each codebook. The computation of the similarity distance between two regions will in this case generate as many distance values as codebooks. Given that the contents of the word descriptors are already normalized, it can be assumed that each of the distance values will also be normalized and comparable. This allows the fusion of distances into a single one, again by applying one fusion rule (average, product, max...).

7.2.2 Evaluation of fusion schemes for codebooks

The presented strategies for the early fusion of features in the context of codebooks have been tested with the same experiments conducted in Section 6.3. In that section, the region and HBoR features were compared, as well as the impact of using a codebook built exclusively on foreground regions with respect to another one that would consider also the background. The combination of these two variables together with the working with a single fused codebook or three separate ones generate the eight plots contained in each of the graphs of Figure 7.10 (fusion with z-scores) and Figure 7.11 (fusion with w-scores). The only considered fusion rule was the *average* given its superior performance in Section 7.1.4.

Remark 5. *W-scores outperform z-scores when considering HBoR*

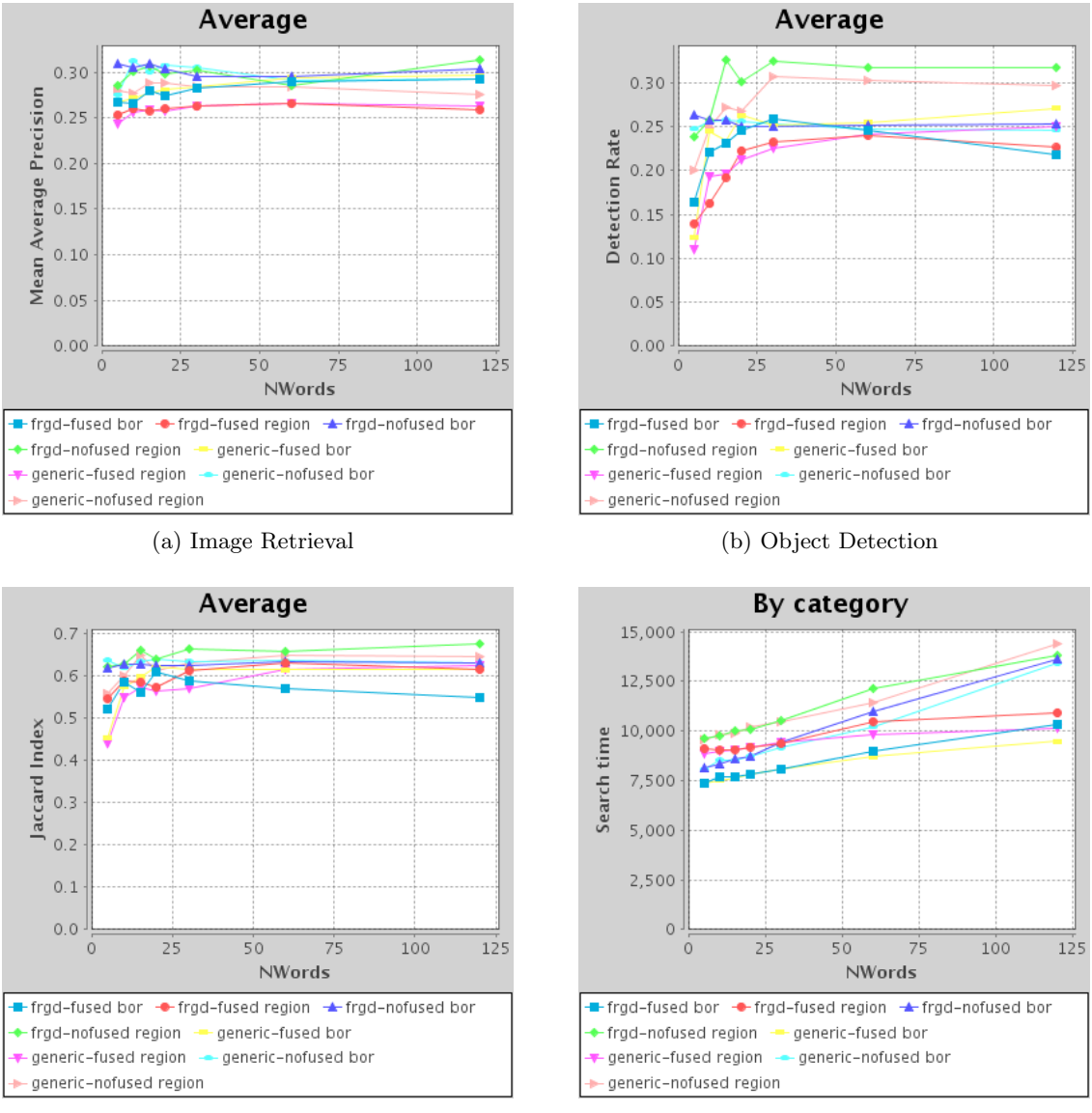
The values of the *image retrieval* experiments associated to the HBoR features when fusing w-scores (Figure 7.11) are around 30% better than the ones achieved with z-scores. This different behaviour with HBoR features does not occur with region-based features, neither when considering codebooks nor when directly using region-based descriptors. This gain must be due to the different curves used for normalization, a line for z-scores and a Weibull curve for w-scores. In the Weibull case, the concave behaviour of the CDF curve penalizes the non-normalized values on the lower range (see Figure 7.4). This shape requires a non-normalized value to be over a certain minimum level to have a relevant impact after normalization, a requirement that is more relaxed for the linear case of z-scores. Given that the HBoR features are based on a max pooling, setting a higher standard of quality provides more reliable values during their bottom-up propagation through the PT.

The remaining of this thesis focuses solely on normalization solutions with w-scores. This decision has been taken based on these results together with the theoretical grounds presented in Section 7.1.3.

Remark 6. *Working with separate codebooks tends to provide higher accuracy*

The main issue analysed in this Section is comparing the option of working with separate codebooks (*no fused* in the graphs) with the one of working with a single codebook (*fused* in the graphs). In terms of accuracy, the clearest differences appear in the (a) *Image Retrieval* and (b) *Object Detection* graphs. For example, the best performance is obtained when codebooks made of foreground regions from the objects (*frgd* in the graphs) and are exploited by HBoR features (*bor* in the graphs). This configuration corresponds to the blue triangle (*frgd-nofused bor)*, whose plot is for all codebook sizes over its corresponding pair *frgd-fused bor)*, painted with the blue square.

In all four combinations of *frgd / generic* and *bor / region*, the lines for separate codebooks are above the ones for fused codebooks. However, this observation is not conclusive regarding which of the two options is better: the *NWords* in the X-axis refer to the size of



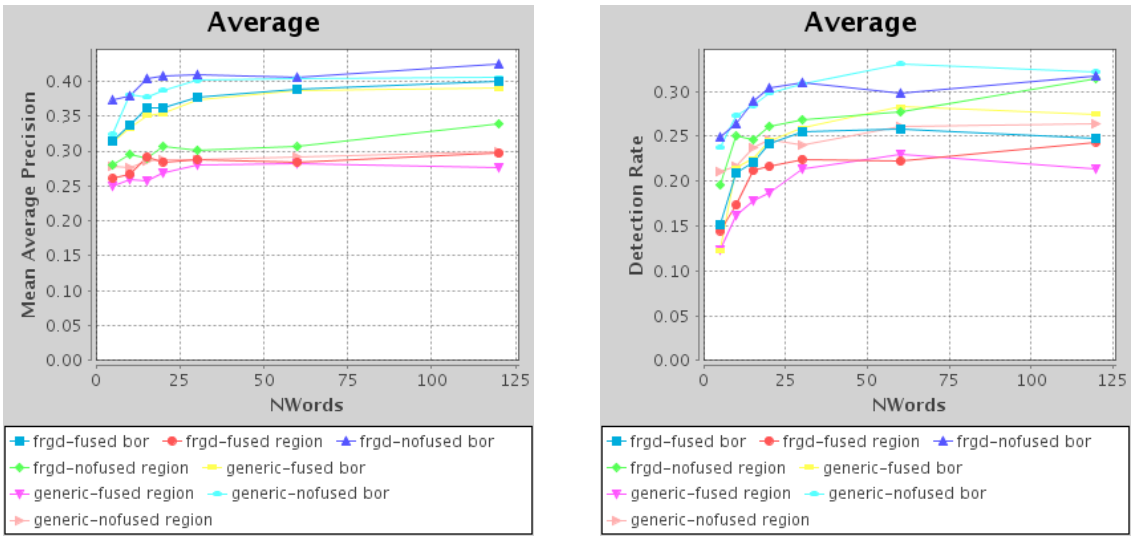
(a) Image Retrieval

(b) Object Detection

(c) Object Segmentation

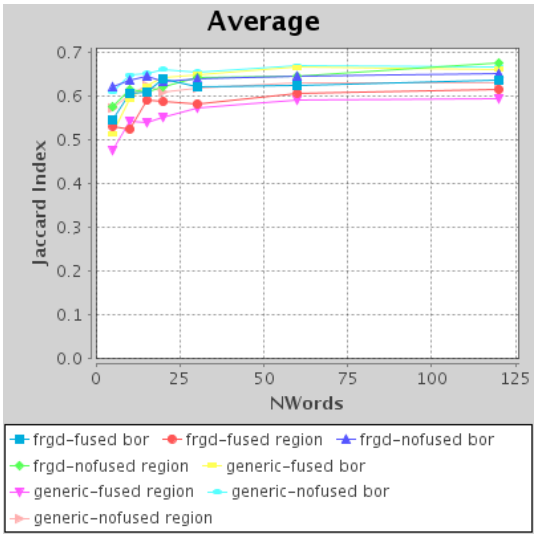
(d) Search time

Figure 7.10: Fusion of multiple visual words with z-scores.

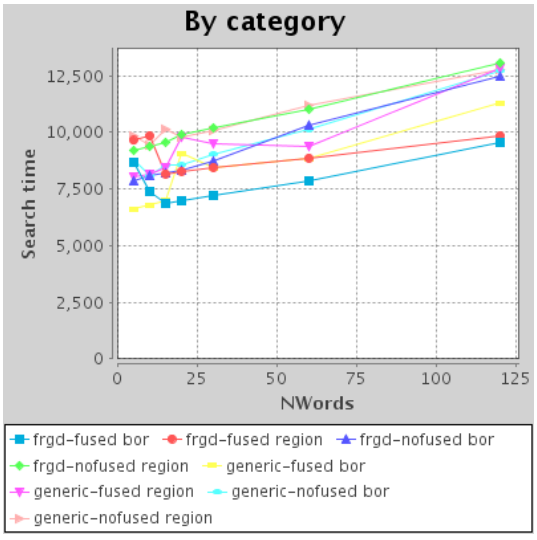


(a) Image Retrieval

(b) Object Detection



(c) Object Segmentation



(d) Search time

Figure 7.11: Fusion of multiple visual words with w-scores.

	3x5 vs 15		3x10 vs 30		3x20 vs 60	
	HBoR	Region-based	HBoR	Region-based	HBoR	Region-based
Foreground	3.05 %	-3.78 %	0.27 %	2.78 %	4.88 %	8.10 %
Generic	-7.41%	7.75 %	1.87 %	-1.78 %	0.00 %	1.41 %

Table 7.4: Relative gain in MAP of separate codebooks compared to a w-fused codebook

	3x5 vs 15		3x10 vs 30		3x20 vs 60	
	HBoR	Region-based	HBoR	Region-based	HBoR	Region-based
Foreground	-14.34 %	-13.99 %	-10.05 %	-11.86 %	-2.93 %	-9.27 %
Generic	-10.95 %	-10.18 %	8.98 %	4.48 %	12.45 %	2.43 %

Table 7.5: Relative gain in search time of separate codebooks compared to a w-fused codebook

every considered codebook and, in these experiments, the *nofused* options use three different codebooks, one for every visual descriptor. So, given an *Nwords* value, the total amount of codewords for the *nofused* case triplicates the plotted value. This is not the case of the *fused* case, which uses a single codebook.

The comparison between the two options must consider the same amount of codewords, so that the *nofused* configuration of N codewords must be compared with the *fused* solution of $3N$ codewords. Table 7.4 includes the gain in MAP when considering the three combinations of codebook sizes where the separate codebooks are compared to the w-fused solution. The obtained values indicate a slighter preference for the *nonfused* solution because most configurations show positive gains. However, in 3 of the 12 cases the performance decreased, and in a fourth one the result was identical.

The results of the experiments have also been analysed in terms of search time. Table 7.5 presents the gain obtained by working with three separate codebooks instead of a single and fused one with w-scores. This table is not conclusive either about the preference for one or another configuration. In 8 of the 12 configurations the fused solution performs faster, but in the remaining 4 the separate codebooks show a better result. Given the obtained figures, it is not possible to conclude if one of the configurations is better than the other in terms of search time. In fact, this outcome seems reasonable as the search time is basically dependent on the computation effort, which is equivalent in both cases.

7.3 Summary

This chapter concludes Part II, dedicated to the feature vectors associated to every node in the PT. In this final chapter the region-based visual descriptors introduced and separately tested in Chapter 5 have been combined with a late fusion strategy. Three different normalization

techniques have been tested (*z-scores*, *SVM* and *w-scores*) combined with four different rules for fusion (average, max, min and product). The *average* fusion rule has provided the best results in each of the three normalization strategies, generating performance values better than the ones obtained in Chapter 5 for separate descriptors.

In the second part of this chapter, normalized distances have been applied on visual words with two different configurations: using a single codebook of fused visual descriptors or working with three separate codebook, one for each visual descriptor. The experiments have shown little difference between the two options, but have validated the hierarchical bag of subtrees features proposed in Chapter 6, as the best results have been reached under this configuration.

The three considered normalization strategies (*z-scores*, *SVM* and *w-scores*) were tested both for non-quantized features and visual words. For non-quantized features, *z-scores* slightly outperformed w-scores, both of them providing far better results than a fusion based on an SVM classifier. In the case of visual words, though, *w-scores* clearly provided better values. Given that the HBoR visual words are the novel topic proposed in this thesis, *w-scores* have been adopted for normalization in the remaining of the thesis. This technique is based on normalizing the similarity metrics by fitting the similarity measure to a Weibull Cumulative Distribution Function. This curve is estimated by learning the Weibull parameters from a training dataset.

This chapter concludes the study of cases where the query is composed by a single part, that is, a single node in the Object Tree. The next Part III in the document will use the best configurations found in the present Part *part:FeaturesExtraction* to address more complex pattern recognition problems. In particular, a Chapter 8 will consider queries formulated as an Object Tree with multiple subparts, and a Chapter 9 will propose a modelling of an object from a set of such Object Trees belonging to the same class.

Part III

Pattern Recognition

Summary of Part III

This third part of the thesis applies the image representation introduced in Part I and the features proposed in Part II to a part-based analysis of the images. The next two chapters are inspired by the fact that several objects are not represented by a single node in the Partition Trees (PTs), but by multiple. This situation is addressed by adopting pattern recognition techniques that consider combinations of PT nodes. In both cases the hierarchical structure of the PT is exploited to design an efficient solution.

This part is structured in two chapters that cover the two main retrieval scenarios described in Chapter 1: *query by example* and *query by concept*. The *query by example* problem considers that a single exemplar of the object describes the user query, while a query composed of multiple instance is considered a *query by concept* and requires a previous analysis of the query to abstract the patterns it describes.

Chapter 8 formulates the pattern recognition problem as a matching between the parts of a query object and the image parts contained in the target dataset. Two types of solutions are proposed: a first one assumes that the hierarchical decomposition of the object is also replicated in the target dataset, while a second approach only requires the most basic sub-parts of the object to be represented in the target PT. Both schemes are sequentially applied, using the best results of the first one as a baseline to efficiently limit the amount of combinations to be considered by the second one.

Chapter 9 considers queries composed of several examples. The proposed solution tries to abstract the multiple-instance query by learning a model out of it, a model which will be evaluated on each target PT during the search process. A part-based solution is also adopted, this time considering a model that initially identifies which nodes from the target PT correspond to parts of the object and, in a later stage, combines the detected parts to build the complete object. Analogously to the *query by example* case, this *query by concept* solution also proposes strategies for an efficient analysis of the PTs.

Chapter 8

Partition Tree Matching

The previous Part II has focused on the definition of region-based visual features on hierarchical partition and their application to the instance search problem. The queries considered in the reported experiments were defined by collapsing all object parts into a single region. The instance search problem was solved by comparing the visual features extracted from this region with those associated to the regions in the target dataset. This type of queries will be referred as *atomic queries* because they are defined on a single part that cannot be further decomposed. In the context of the Object Tree (OT) introduced in Chapter 4, *atomic* queries are those that only consider the OT root node. The present chapter will take one step further and exploit the hierarchy represented by the OT to define the *Query Tree (QT)*, a new data structure to support *composite* queries. A *Query Tree* replicates the topology of the OT and adds a weight to each node. The replication can be total or partial, depending on the maximum amount of QT nodes considered for the matching.

There exist two problems when using *atomic* queries. The first one is that the match is only based on the visual descriptors associated to the query region and the target PT nodes. Region features can richly characterize an image segment but, in many cases, they lack structural information about how this region is internally composed. For example, this is the case of the CSS Shape and Dominant Colors descriptors considered in this work. Partition Trees (PTs), already used for image segmentation in Part I and feature extraction in Part II, provide a hierarchical decomposition that can be also exploited during matching. This chapter proposes a reformulation of the matching process, considering it no longer as a region-to-region matching but as a subtree-to-subtree problem between a QT and a portion of the target PT, as presented in Figure 8.1. The comparison between partition hierarchies requires the definition of a metric capable of assessing their similarity. Section 8.2 presents a solution that weights the importance of every QT node depending on their occupation and decomposition level.

A second limitation inherent to any *atomic* query is that a good match is only possible when the object is represented by a single node in the target PT. As previously seen in Section

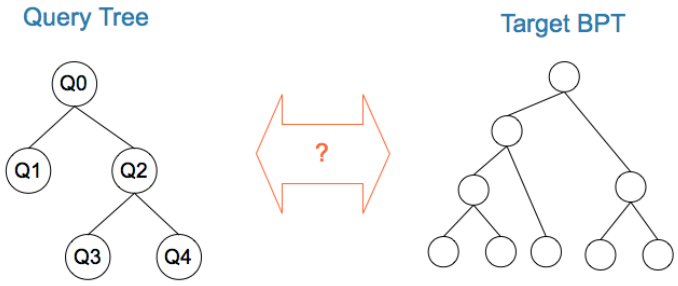


Figure 8.1: Subtree matching between the Object Tree defined by the query and a target PT

2.4, this requirement is not satisfied in several cases, as many objects are split into several subtrees. When the split occurs in the query image, the user interaction solves it by selecting the subtree roots that will be fused to build the QT. This interactive segmentation has been presented in Chapter 3. However, if the object split occurs in the target PTs, no good match will be possible because they will not be structured under a single subtree, even if the object parts are represented in the target PT. The solution to this situation requires breaking the hierarchical structure defined during the creation of the PT. Section 8.3 presents an algorithm to drive this type of analysis while avoiding an explosive and unmanageable amount of possible combinations.

8.1 Related Work

The problem of matching the parts of a query image/object to the parts of a target database has been broadly studied in previous works. There exist solutions that focus on interest points or on regions, and some drive a hierarchical approach while other consider flat visual representations.

Cohen, Vinet, Sander (1989)

The first reference found to the addressed problem is a work from 1989 by Cohen et al in [20]. Their goal was to segment a pair of images in stereo vision, considering that the mutual information would help in both cases. The input data to their system were a set of region hierarchies of the stereoscopic images, among which the algorithm had to choose one for each image and select the best scale. In this case, though, the coarsest level of the hierarchies does not correspond to the complete image, but to some intermediate scale. The problem addressed in this thesis is also the matching of two hierarchies, but instead of considering the matches between two PTs that represent complete images, one of the two hierarchies corresponds to the query object.

The adopted matching scheme follows a top-down approach with a minimum similarity

threshold to accept a matching. The algorithm works with two sets of candidate regions for each of the two images, L and R. These sets are initialized with the root nodes from all considered hierarchies. In each iteration, the similarity between all the regions in L and all the regions in R is assessed. There is a restriction about what regions can be matched to based in the location, as in stereo images the location of the potential matchings can be roughly predicted. Then, all those matchings with a similarity score above the threshold are matched and removed from the L and R sets. The iteration ends by adding all descendants from the regions in L and R to their respective region sets. This incremental addition of regions to the set does not guarantee that the best matching for the region is found, as a sub-optimal pair found in a prior iteration would already remove a region before considering all possibilities. The authors justify this incremental addition and gradual removal of matched regions to avoid a combinatorial explosion. In this thesis, all combinations are considered, which represents a higher computation effort. However, we only consider one PT per image, instead of the multiple hierarchies that are contemplated by Cohen et al.

Smith and Chang (1996)

The VisualSEEk by Smith and Chang [82] addressed the problem of matching multiple regions by jointly considering their visual content and spatial location. The query can be automatically determined by detecting the salient regions in the query image, or manually defined by the user, whether drawing a sketch or selecting the regions of interest.

Their work defines two type of parameters: *intrinsic* and *extrinsic*. The *intrinsic* parameters refer to the individual features a associated to every region (color, size and location), while the *extrinsic* parameters are extracted from combinations of regions (relative spatial location) or retrieved images after a search based on the *intrinsic* parameters. The *intrinsic* parameters are indexed to be able to rapidly obtain a short list of image candidates for a query. The indexing of the *intrinsic* features is possible because they are not dependent from the query. The search process is divided in two stages: a first and quick one where image candidates are retrieved from the *intrinsic* parameters of the query regions, and a second one that assesses the *extrinsic* parameters, whose evaluation implies a higher complexity. The solution presented in this thesis does not consider any indexing strategy as it focuses in the *intra-image* search, so this analysis in two stages would not be applicable under the current configuration. In fact, the hierarchical structure of the PT combined with the HBoR features can be considered a tree-based index that already combines the visual and spatial location of the regions.

Smith and Chang also proposed *color sets* as a quantization of the color space. This is a type of possibilistic quantization but only considering binary values; that is, a region can just *contain* or *not contain* one of the quantized colors, there is no in-between. However, a region can be associated to several colors in the set. This quantization is also used to

extract the salient regions of the query images, define the bins in the color histograms of the regions and design an indexing algorithm for fast retrieval. This quantization of the color space to later define a histogram is very similar to the word signatures popularized by Sivic and Zisserman for interest points [80]. This thesis has also considered a quantized solution when working with codebooks, but instead of a arbitrary partition of the HSV color space, as adopted in VisualSEEk, we have adopted the common solution of learning the codewords after a clustering process.

VisualSEEk adopted a *late fusion* strategy in terms of visual descriptors, so a query with a single region runs an individual search for each considered feature. The amount of top hits to be considered is a parameter to be defined by the user, a value that may differ for each visual descriptor. The obtained ranked lists are combined with an intersection to obtain a set of common images. Finally, a global distance metric is computed for each retrieved image to generate the final ranked list. This final distance fuses the similarity values obtained for color, position, area and spatial extent, being this last feature a rough estimation of the shape based on the dimensions of the bounding box.

The case of multiple regions in the query is handled similarly to the multiple features problem: a separate search for every region is run and only the images with candidates for all query regions are kept. The relative location of the regions is assessed with *2-D strings* [17], a descriptor that represents the ordering of the regions when they are projected on the X and Y axis. Only those images whose 2-D strings match the one associated to the query are included in the final ranked list. The score associated to each candidate image is obtained with a weighted sum of the match for every query part. The use of 2D strings allows a fast evaluation of the relative location of the regions, and it is robust to scale variations. It is not invariant to rotation though, and for this reason the authors proposed working with an additional 2D string rotated 45 degrees. This approach to multiple query regions allows working with non-connected parts of the query, a situation that is not contemplated in this thesis.

Li, Wang and Wiederhold (2000)

The *Integrated Region Matching (IRM)* system proposed by Li, Wang and Widerhold [54] also focuses on developing a similarity measure to compare images from its composing regions. The images are segmented with a K-means algorithm, so the amount of regions per image must be previously set. However the authors report that experimental results are insensitive to the amount of regions.

The algorithm explores every region in each of the two images compared and matches them to one or multiple regions from the other image. A matching is defined when the similarity score between the region features is below a certain threshold. Finally, the overall similarity distance between the two images is obtained as a weighted sum of the individual distances

associated to all the created matchings. The considered weights, named *significance credits*, indicate the importance of the matching for determining the similarity between images. These significance credits of the matchings are estimated with an iterative algorithm that assigns higher values to those matchings with smaller distances. The iterative nature ensures that the summation of the weights is equal for every pair of compared images.

Moreover, each region is also assigned a significance that complements the significances of the matchings. The significance of the region is based on the relative area of the region compared to the whole image. This way, the algorithm prioritizes larger regions. This criterion, named by the authors as *area percentage scheme*, is also adapted in this chapter of the thesis. Li et al also propose a weighting strategy that considers the location of the regions to slightly down-weight those regions located around the boundaries of the image.

Wang, Rui and Sun (2003)

The IRM scheme inspired Wang et al [102] to propose another solution to compare two images according to the regions they contain. They proposed the *Constrained-based Region Matching (CRM)*, which enriches the IRM solution by introducing the spatial relations between the regions. In their paper, the term *constrain* refers to a *feature*.

The authors adapt the *intrinsic* and *extrinsic* terminology used by Smith and Chang to *first-order* and *second-order* constraints, using the ordinal to refer to the number of regions involved. So, in this case, only binary relations were considered. The *first-order* constraints (features) used were color, shape and position, while the *second-order* constraints evaluated the relative orientation, inclusion and sizes of the two regions being compared.

The main contribution of the paper was a new method to estimate the weights in the IRM formulation. This method that includes the relational features under a novel probabilistic formulation. This formulation considers the normalized similarity scores as good estimators of the *probability* of the two matched regions to be "similar". This interpretation allows a reformulation of the expression that estimated the weight of a match. Instead of purely depending on the visual appearance of the regions, it also considers the relations they establish with the rest of the regions in their respective images, being the impact of such relation also weighted by the visual similarity between the regions they are matched to.

Lowe (2004)

The well-known work by Lowe on SIFT descriptors [56] also includes an algorithm for keypoint matching. This algorithm was tested by comparing each test image with a training dataset of objects. The targeted problem did not consider any local segmentation, so both the query and the labels from the training dataset were defined at a global scale.

A first contribution from the author was a mechanism to identify those keypoints from

the test image that had no match, that is, that correspond to the background or that were not generated in the training images. The author discarded using a threshold on the similarity score because some keypoints were considered more discriminative than others. So the adopted solution defined a threshold on the ratio of the closest neighbor to that of the second-closest neighbor. If the ratio was small, the match was considered significant, but if it was large it would be flagged as irrelevant and discarded. Lowe's approach was presented in a context of image categorization with a training dataset of 32 images, where the second-closest neighbor is forced to belong to a different class from the first. However, in a large dataset this criterion may not apply, as not all images will be labelled and it is more probable that the second-closest neighbor will indeed be very similar to the keypoint.

The experiments reported by Lowe considered that reliable matches were possible with as few as 3 points. The clustering of points to define a single object was achieved through the Hough transform. Additionally, there still exist a final step that verified that the geometric distribution of the clustered points was coherent with the matched points in the training dataset. The overall strategy differs from the one that will be proposed in this chapter, where the matching of the object parts is guided by the PT. Instead of an individual matching of points that are later analysed for consistency, the matching solution proposed in this thesis only considers combinations of parts in the target images that can satisfy the hierarchical structure defined by the query. One drawback of our approach with respect to Lowe's is the inability to deal with object occlusions, especially if the occluding object splits the object of interest in non-adjacent regions.

Lazebnik (2006)

The addressed problem of matching a hierarchical structure is similar to the one addressed by the techniques based on the Spatial Pyramid Matching (SPM) introduced by Lazebnik et al [51]. This is an extension of the Bag of Features (BoF) model [80] already presented in Chapter 6. The SPM approach places a sequence of grids at different spatial resolutions $0, ..., L$, such that the grid at level l has 2^l rectangular cells along each of the two spatial dimensions. The grids of two images are matched by counting the amount of matches at every level of resolution l. Two points are said to match if they fall in the same cell of the grid. The final match distance is computed as a sum of the amount of matches that occur at every level, but weighting each of them depending on the level. Matches in smaller cells have larger weights associated as their matches are considered more significant. The SPM is in fact a two-dimensional extension of the Pyramid Match Kernel proposed by Grauman and Darrell [36]. The BoF histograms computed for each grid can also be used to define a vector representation of the image by concatenating them, each histogram after the other. The obtained feature vectors can be used for image classification, for example using SVMs [51] [108] or random forests [8]. Our work also work with a hierarchical matching based on

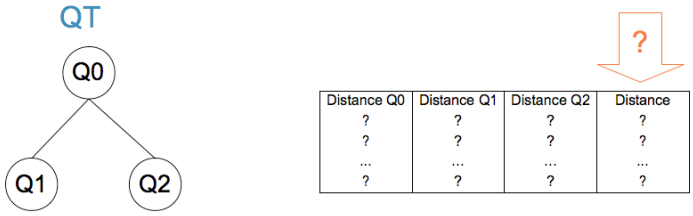


Figure 8.2: The distance problem for basic query OT.

the visual features at each level, but the hierarchy considered in this thesis is adapted to the content. Partition Trees define variable amounts of decomposition levels and determine the area of support of each level according to an initial partition of the image. This region-based approach allows using the highly informative shape descriptors. However, the diversity of topologies of PTs increases the complexity of the matching algorithm.

8.2 Matching Distance

The assessment of a matching between a QT and a set of the image parts represented in the target PT requires a distance to evaluate the visual similarity of the assignment. The metric proposed in this section will be introduced by considering a simple QT of three nodes Q_0, Q_1, Q_2, where Q_0 is the union of Q_1 and Q_2. Throughout this chapter, the numbering between parent and children in the QT will always assign lower IDs to those regions which are larger. So, for example, in Figure 8.2 the region associated to Q_1 is larger than its sibling Q_2 and, of course, their union Q_0 is larger than each of its two children.

It seems reasonable to quantify each of the individual matchings with the same criterion applied in the *atomic* case, that is, the region or codebook-visual distances presented in Part II. The main question arises when considering all these distances together with the topology. This work proposes a solution that considers both the relative size of the query parts as well as their structure in the QT. The two properties are expressed through a coefficient α_i, which combines the relative area between the part Q_i and its parent, as well its decomposition in further parts. Equation 8.1 presents the proposed formulation to compute α_i, where the notation has been simplified by referring to part Q_i with its index i. The expression computes the area ratio between the part Q_i and its parent, P_i, and divides the resulting quantity with the amount of children C^i, if any.

$$\alpha_i = \frac{1}{max(1, |C^i|)} \frac{x_{area}(i)}{x_{area}(P^i)} \tag{8.1}$$

The α coefficients associated to every query part allow the definition another set of ω_i coefficients. These are the responsible to scale the distances which are individually obtained by comparing a part in the query with a part in the target image. The complete matching is

quantified with a simple linear combination of the match distances d_i scaled by the associated ω_i:

$$d = \sum_{i=0}^{|Q|} \omega_i d_i \tag{8.2}$$

The relationship between the ω_i coefficients of the linear form and the computed α_i is presented in Equation 8.3. The expression states that ω_i corresponds to the product between the α_i associated to the part Q_i and all the rest of α_j coefficients associated to its ancestors which, in turn, correspond to the ω factor associated to the parent node, P^i. This way, the obtained weigh ω_i corresponds to the weight of its ancestors multiplied by the relative size of the part Q_i with respect to its parent P^i.

$$\omega_i = \alpha_i \prod_{j \in A^i} \alpha_j = \alpha_i \omega_{P^i} \tag{8.3}$$

This definition of the ω_i satisfies that their sum throughout a QT will always equal 1.0, as expressed in Equation 8.4. This property allows obtaining similarity metrics which are independent of the amount of nodes. Moreover, it also keeps the $[0, 1]$ range of values required to the similarity metrics.

$$\sum_{i=0}^{|Q|} \omega_i = 1 \tag{8.4}$$

Example: One level decomposition

The generic expressions introduced in Equations 8.2 and 8.3 are particularized in a simple example to facilitate their comprehension. If taking into account the one-level decomposition example represented in Figure 8.2, the proposed solution will assign half of the final weight to the matching of Q_0, whose relative importance in terms of area is of 50%. The other half depends on Q_1 and Q_2, on a proportion equal to their relative area.

$$\begin{aligned}
\alpha_0 &= \frac{1}{2}\frac{1}{1} = \frac{1}{2} \quad &\Rightarrow \quad \omega_0 &= \alpha_0 = \frac{1}{2} \\
\alpha_1 &= \frac{1}{1}\frac{x_{area}(Q_1)}{x_{area}(Q_0)} = \frac{x_{area}(Q_1)}{x_{area}(Q_0)} \quad &\Rightarrow \quad \omega_1 &= \alpha_1\alpha_0 = \alpha_1\omega_0 = \frac{x_{area}(Q_1)}{x_{area}(Q_0)}\frac{1}{2} \\
\alpha_2 &= \frac{1}{1}\frac{x_{area}(Q_2)}{x_{area}(Q_0)} = \frac{x_{area}(Q_2)}{x_{area}(Q_0)} \quad &\Rightarrow \quad \omega_2 &= \alpha_2\alpha_0 = \alpha_2\omega_0 = \frac{x_{area}(Q_2)}{x_{area}(Q_0)}\frac{1}{2}
\end{aligned}$$

The obtained coefficients can now be applied to the expression defined in Equation 8.2 for the considered QT. The formula linearly combines the three region distances d_i with weights ω_i.

$$
\begin{aligned}
d &= \omega_0 d_0 + \omega_1 d_1 + \omega_2 d_2 = \\
&= \alpha_0 (d_0 + \alpha_1 d_1 + \alpha_2 d_2) = \\
&= \alpha_0 d_0 + \alpha_0 \alpha_1 d_1 + \alpha_0 \alpha_2 d_2
\end{aligned}
$$

If considering the simple case where the two siblings Q_1 and Q_2 are equally large, the final expression for the computation of the distance corresponds to:

$$
d = \frac{1}{2}\left(d_0 + \frac{1}{2}d_1 + \frac{1}{2}d_2\right) = \frac{1}{2}d_0 + \frac{1}{4}d_1 + \frac{1}{4}d_2
$$

Example: Two levels decomposition

The proposed expression also applies to QTs decomposed into multiple levels. This example consider the case where the Q_2 part is further decomposed into Q_3 and Q_4, as in the example contained in Figure 8.1. Now the α_i and ω_i coefficients are computed as follows:

$$
\begin{aligned}
\alpha_0 &= \frac{1}{2}\frac{1}{1} = \frac{1}{2} && \Rightarrow && \omega_0 = \alpha_0 = \frac{1}{2} \\
\alpha_1 &= \frac{1}{2}\frac{x_{area}(Q_1)}{x_{area}(Q_0)} && \Rightarrow && \omega_1 = \alpha_1\alpha_0 = \alpha_1\omega_0 = \frac{1}{2}\frac{x_{area}(Q_1)}{x_{area}(Q_0)}\frac{1}{2} \\
\alpha_2 &= \frac{1}{2}\frac{x_{area}(Q_2)}{x_{area}(Q_0)} && \Rightarrow && \omega_2 = \alpha_2\alpha_0 = \alpha_2\omega_0 = \frac{1}{2}\frac{x_{area}(Q_2)}{x_{area}(Q_0)}\frac{1}{2} \\
\alpha_3 &= \frac{1}{1}\frac{x_{area}(Q_3)}{x_{area}(Q_2)} && \Rightarrow && \omega_3 = \alpha_3\alpha_2\alpha_0 = \alpha_3\omega_2 = \frac{x_{area}(Q_3)}{x_{area}(Q_2)}\frac{1}{2}\frac{x_{area}(Q_2)}{x_{area}(Q_0)}\frac{1}{2} \\
\alpha_4 &= \frac{1}{1}\frac{x_{area}(Q_4)}{x_{area}(Q_2)} && \Rightarrow && \omega_4 = \alpha_4\alpha_2\alpha_0 = \alpha_4\omega_2 = \frac{x_{area}(Q_4)}{x_{area}(Q_2)}\frac{1}{2}\frac{x_{area}(Q_2)}{x_{area}(Q_0)}\frac{1}{2}
\end{aligned}
$$

The expansion of the total distance d as a linear combination validates the development in Equation 8.2:

$$
\begin{aligned}
d &= \omega_0 d_0 + \omega_1 d_1 + \omega_2 d_2 + \omega_3 d_3 + \omega_4 d_4 = \\
&= \alpha_0 d_0 + \alpha_0\alpha_1 d_1 + \alpha_0\alpha_2 d_2 + \alpha_0\alpha_2\alpha_3 d_3 + \alpha_0\alpha_2\alpha_4 d_4 = \\
&= \alpha_0 (d_0 + \alpha_1 d_1 + \alpha_2 (d_2 + \alpha_3 d_3 + \alpha_4 d_4))
\end{aligned}
$$

As previously considered in the case of one level of decomposition, the expression can be evaluated if every split defines children of equal size:

$$
d = \frac{1}{2}\left(d_0 + \frac{1}{2}d_1 + \frac{1}{2}\left(d_2 + \frac{1}{2}d_3 + \frac{1}{2}d_4\right)\right) = \frac{1}{2}d_0 + \frac{1}{4}d_1 + \frac{1}{8}d_2 + \frac{1}{16}d_3 + \frac{1}{16}d_4
$$

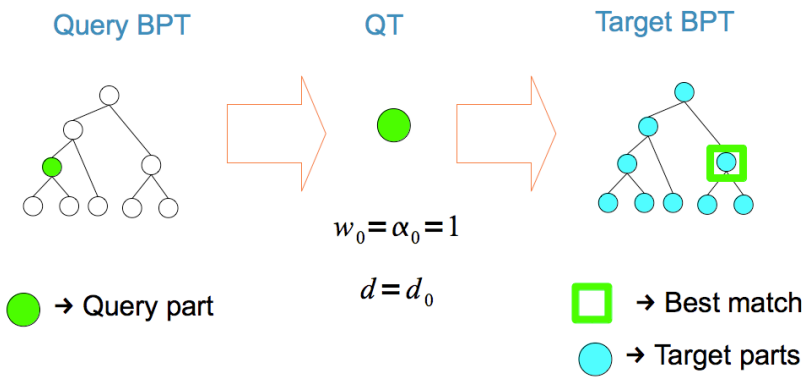


Figure 8.3: Atomic query.

8.3 QT-BPT Correspondence

This section proposes an algorithm to efficiently consider the possible matches between the QT and a target BPT. The technique is presented in three cases of increasing complexity:

- Atomic query: The basic case of a QT with a single node is addressed to show its equivalence with a classical region-based retrieval problem.
- Top-down QT Matching: The target BPT is explored to find the best correspondence of its nodes with the hierarchical decomposition defined in the QT.
- Bottom-Up QT Matching: The bottom-up strategy does not require the topology in the QT to be respected in the target BPT at the expense of a more demanding computational effort.

8.3.1 Atomic Query

The simplest case that can be addressed is a QT composed by a single node, the basic configuration that has been adopted during the experiments in Part II. In this situation, there exist no sub-parts for the query, so the matching problem only requires comparing the QT node with every single node in the target BPT. The order in which the BPT nodes are assessed is irrelevant, though its implementation is more simple if a top-down approach is taken. Figure 8.3 shows an example where one node in the query BPT is selected to define a single node QT. The region associated to the node in green is compared to every node in the target BPT, in cyan, and the match with the smallest distance is considered as the best match. In this configuration, the hierarchical decomposition of the non-leaves nodes in the query BPT is ignored.

The classical query by example problem at the global scale is also contemplated by the proposed framework when considering that the query BPT is composed by a single node

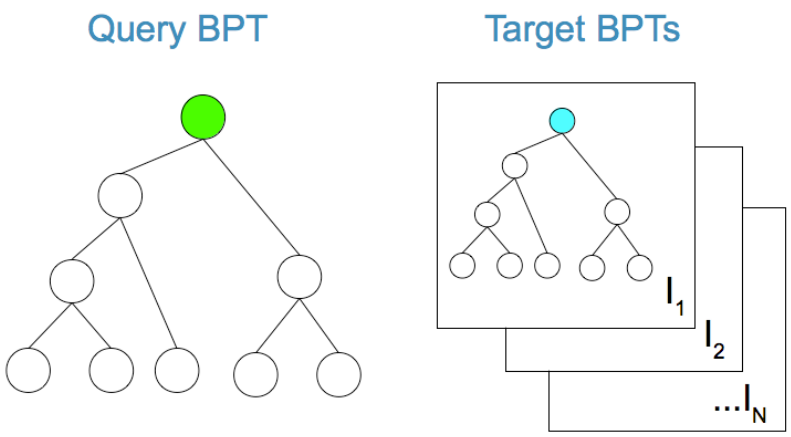


Figure 8.4: Global scale query.

representing the complete image. The same reasoning can be applied to the target BPTs, considering only matchings to their root node. Figure 8.4 displays this situation.

8.3.2 Top-down QT Matching

The top-down variant of the matching algorithm searches for the best correspondence between the QT in a subtree of the target BPT. This matching strategy assumes that the hierarchy of parts represented in the QT is kept in the target BPT. However, the proposed algorithm does not require the QT to be exactly replicated as a subtree in the target BPT, only the relative relationships of *inclusion* must be satisfied. The presented solution is resilient to both the insertion of intermediate nodes in the target BPT which are not in the QT, as well as the collapsing of different QT nodes into a single node in the target BPT.

The adopted top-down approach also allows an efficient matching strategy of the QT nodes by jointly exploiting the hierarchies of both QT and BPT. Instead of finding separate candidates for each QT node and later considering combinations among them, the matching of a QT node reduces the search space for its descendants.

A additional strategy for efficiency is proposed based on the distance metric introduced in Section 8.2. The proposed expression allows discarding many combinations even with a partial matching, avoiding this way having to deal with an explosive amount of combinations.

Matching algorithm with Hierarchical Reduction of the Search Space

The correspondence algorithm starts by matching the QT root, Q_0, with a BPT node. Then, the largest of Q_0's children, Q_1, is compared with every node in the subtree defined by the Q_0 anchor match. For every assignment to Q_1, the next stage will consider Q_1's children and, if any, the same top-down scheme will be applied. Later, the same recursive algorithm

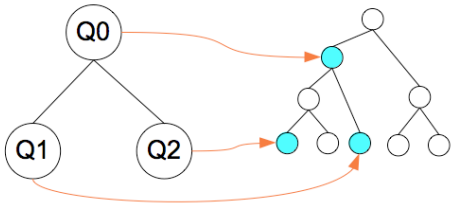


Figure 8.5: Top-down match.

will try to find matches for Q_1 sibling, if any. If this is the case, the matches for Q_2 must define a connected region with the considered Q_1 match. Figure 8.5 shows an example where a QT of three nodes has been matched to three nodes of a target BPT. Node Q_0 was matched first, then Q_1 and finally Q_2. The rational behind this algorithm is discussed in the following paragraphs.

The application of the matching algorithm defines a top-down exploration of the QT that processes the largest children first. The search space of BPT nodes considered for matching every QT node does not need to be the full BPT, but a portion of it. The previous matches with the parent and sibling (if any) of the QT node can efficiently restrict the search.

Given a non-root QT node to match, the first condition that a candidate BPT node must accomplish is to respect the *inclusion* relationship towards the matching of the QT node's parent. This requirement reduces the search space to the BPT subtree defined by the parent's match. This condition automatically guarantees the replication of the QT hierarchy on the matched BPT nodes.

In addition to the *inclusion* relation, the algorithm also forces a second *adjacency* requirement when trying to match the second of a pair of siblings. This requirement applies because this thesis only considers connected objects and, given the binary nature of the QT, the two parts will always be adjacent. This second restriction reduces even more the search space and, combined with the subtree limitation, offers a fast and efficient algorithm when compared with a blind and independent matching for every QT node. On the other hand, the assessment of the connectivity requires the computation of a *Region Adjacency Graph (RAG)* on the leaves of the BPT, an information that can be easily bottom-up propagated throughout the BPT. The computation of the RAG is light and worthy if compared with the amount of combinations that are discarded thanks to the *adjacency* information it provides.

Resiliency to collapses in the topology

Notice that, in the example of Figure 8.5, the matched BPT nodes do not present the same topology as the QT in terms of parent-children, but they do respect the same relationships in terms of ancestor-descendants. These common misalignments are a result of a subsegmentation on the query BPT or oversegmentation on the target BPT. Their existence justifies

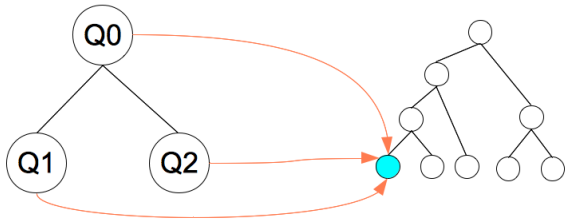


Figure 8.6: QT nodes collapse in a single BPT leaf.

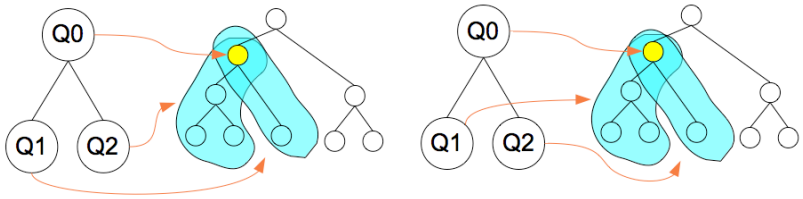


Figure 8.7: Permutations through sub-BPTs.

that the matching of a QT child is not only assessed on the children of its parent's match, but also on all its descendants. The presented solution is robust against these cases, because despite assuming that the QT hierarchy is present in the target BPT, it does not expect to necessarily satisfy the same parent-children topology.

On the other hand, it is also possible that different levels of decomposition in the QT would appear collapsed in the target BPT. For this reason, several QT nodes of the same branch can be matched to the same BPT node. In any case though, this collapse cannot break the hierarchy defined in the QT. That is, given a QT-BPT match, all QT node's descendants must be matched to the same BPT node or one of its descendants, but never to an ancestor on the BPT. The collapse situation is especially usual when reaching the leaves of the target BPT. In these situations, all descendant QT nodes are matched to the same BPT leaf, as exemplified in Figure 8.6. It is necessary to find a match to every node in the QT, even if dissimilar, in order to be able to compute the total matching distance, defined in Equation 8.2.

Every time a QT node is matched to a BPT node, the two possible permutations for matching the QT node's children must be assessed. The example in Figure 8.7 shows the two possible sub-trees that can be matched to Q_1 and Q_2 after having assigned Q_0 to a node in a BPT.

Dynamic Threshold Update

The goal of the intra-image search is to find the best possible QT-BPT match, and only the best one. The rest of sub-optimal combinations are useless when solving the instance search

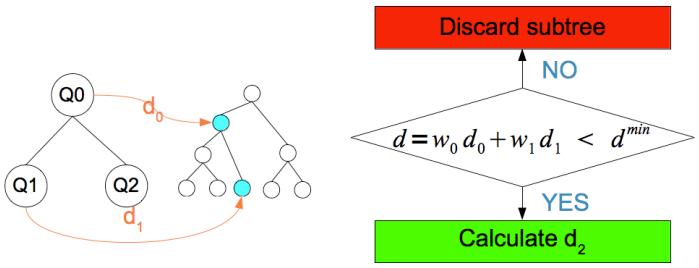


Figure 8.8: Efficiency in matching.

problem through image retrieval. This particularity of the requirements can be exploited in conjunction with the matching distance defined in Section 8.2. The computation of the distance defined in Equation 8.2 does not require a complete match to be partially evaluated. That is, every time a new QT node is matched, a new term can be added to the sum.

Consider that a first complete QT-BPT match has obtained a matching distance of d^{min}. In this situation, any combination that generates a distance d larger than d^{min} will be suboptimal with respect to the best solution obtained so far. Therefore, such combination can be discarded. As the accumulated distance can be sequentially computed after every matched node, the refusal of a combination can be decided even when the QT-BPT match is partial. For example, in Figure 8.8, there is no need to search for a match for Q_2 if the correspondences assigned to Q_0 and Q_1 already produce a total distance d higher than the minimum distance d^{min} found so far among all previous combinations.

Whenever a new full match generates a distance d smaller than the current d^{min}, this threshold is updated. This way, early good matches will rapidly set a low d^{min}, which will in turn reduce the amount of combinations to consider. Notice that the order of the matches promotes computing first the distances associated to those parts with higher ω_i, because it prioritizes the higher nodes in the QT first and, in the next level, the larger of the two children. This scheme facilitates that the terms whose contribution is more important to the distance function will be added first. This way, the chances of an early refusal of suboptimal combinations will increase.

This dynamic threshold update allows discarding a large amount of combinations, while ensuring that the optimal match is not among the discarded ones. This strategy is the main contribution of the top-down approach and it is tightly related to the definition of the similarity distance presented in Section 8.2.

8.3.3 Bottom-Up QT Matching

The top-down matching is an efficient technique as it uses the topologies of the QT and the target BPTs to limit the amount of combinations that are to be considered. However, it is based on a very strong assumption: the region hierarchy represented by the QT must

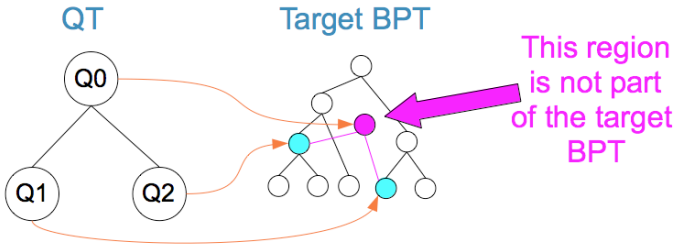


Figure 8.9: Bottom-Up matching.

be respected in the target BPT. In other words, not only the query parts must be present in the target BPT, but also be part of the same subtree. If this assumption is satisfied, the top-down matching is optimal as it considers all possibilities to find the best match. Unfortunately, the semantic gap between the perceptual criteria used for building the BPT and the actual semantic objects contained in the image makes this assumption untrue in many cases, as previously discussed in Section 2.4.

These assumptions are relaxed in this complementary approach that performs a bottom-up matching of the QT nodes. In this second case the only assumption is that the QT leaves are present in the BPT. It is no longer necessary that these parts appear under the same subtree, so it is robust to object splits in the BPT. This property is achieved by considering parts of the target image that are not present in its BPT. For example, Figure 8.9 shows a situation where Q_0 is matched to a region that does not belong to the target BPT, although its subparts do.

Matching algorithm

Opposite to the top-down case, the bottom-up algorithm starts looking for matches for the QT leaves. The upper QT nodes are assessed on adjacent pairs from their children matches, as long as their combination can improve the best distance found so far. This algorithm is designed to be executed after a top-down matching, which is much faster and will determine an initial threshold d^{min}.

Given the basic structure of a one-level QT shown in Figure 8.9, the algorithm starts by identifying the largest leaf and launches an atomic query with it, Q_1 in this simple example. As a result, a ranked list of all those regions whose distance to the Q1 part is below d^{min} is built. Similarly to Q_1, an atomic query through the target BPT is launched with Q_2. However, the maximum distance d_2 that can be accepted not only depends on the best global distance found so far, d^{min}, but also on the best match found for its sibling, d_1^{min}. Every potential match to Q_2 must satisfy that its associated distance d_2 combined and weighted with its sibling's best distance d_1^{min} will not exceed the global threshold d^{min}. This situation is illustrated in Figure 8.10.

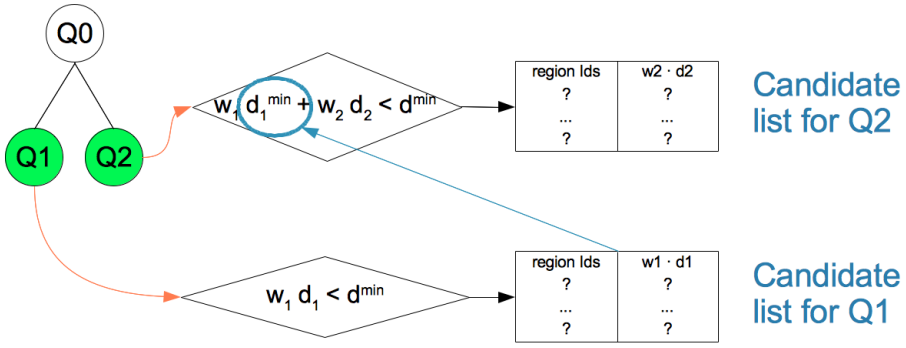


Figure 8.10: Bottom-Up matching of QT children.

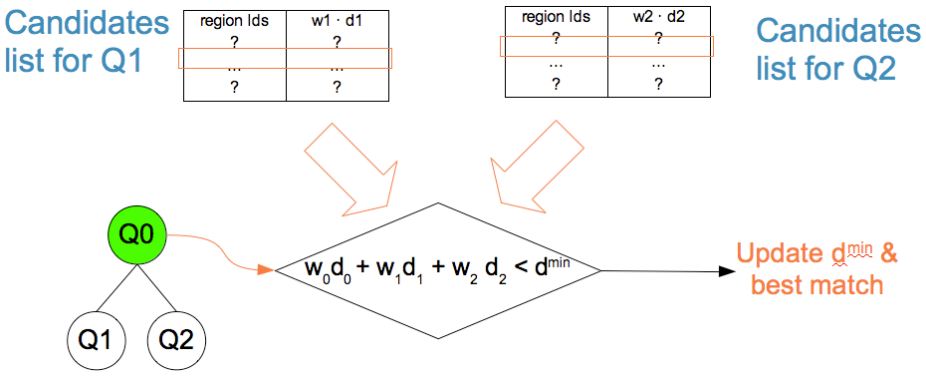


Figure 8.11: Bottom-Up matching of a QT root.

The candidate BPT nodes to be matched to Q_0 must come from the union of two candidates from the Q_1 and Q_2 lists, one from each list. Figure 8.11 shows a scheme where the two ranked lists for Q_1 and Q_2 are combined to generate candidates Q_0. Only those combinations whose distance is below the threshold d^{min} will be added to the ranked list associated to Q_0. Not all pairs must be considered: only those combinations which are not self-included, adjacent and that do not belong to the same subtree, as these were already considered in the top-down case. The unions of these pairs of candidates do not belong to the target BPT so, at this point, the potential object splits may be merged and considered as object parts. Finally, a ranked list of candidate matches to Q_0 is built, a ranked list that will only include those merges capable of improving the minimum matching distance d^{min} found so far.

The collapse case is also handled in the bottom-up scenario when the same region is a candidate for multiple QT nodes. Figure 8.12 shows two examples of concentration. In the one-level decomposition on the left, the matches of Q_1 and Q_2 on the same BPT node will inevitably drive to a match of Q_0 on the same node. In the two-level decomposition example on the right, the two sets of QT leaves point to the same pair of BPT nodes. When these two pairs merge, they define the same region, in pink. When considering the matches for the

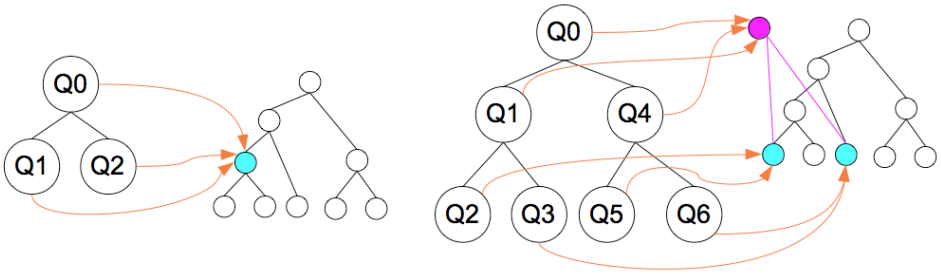


Figure 8.12: Collapse cases in Bottom-Up matching.

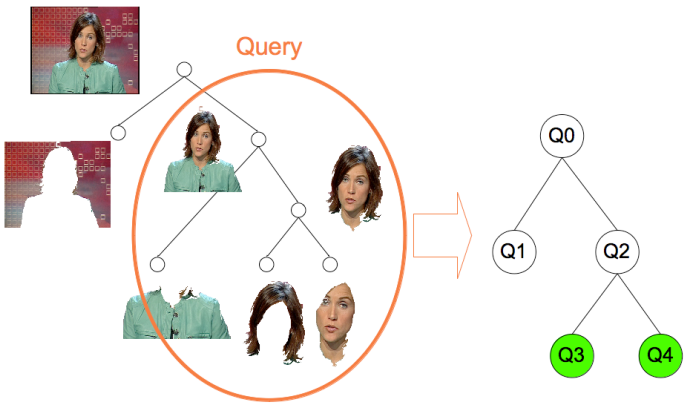


Figure 8.13: Multiple-level matching.

superior QT nodes, they also inevitably collapse in the same candidate region.

The one-level decomposition presented can be easily expanded to more complex topologies. In the general case of multiple levels in the QT, the candidate list linked to the parent is treated the same way as the children. In the example of Figure 8.13, this situation refers to using the candidates for matching Q_2 together with those assigned to Q_1 to generate new candidates to match Q_0.

The main problem of the bottom-up approach is its potential to generate a large amount of hypothesis that could make the problem untreatable in terms of computation. The exponential growth of candidates while climbing the QT could compromise the usability of the system. Considering only those cases that can improve d^{min} is already a technique to avoid these situations. However, and especially in those QTs with many decomposition levels, it may not be enough because the computed d_i are weighted with the correspondent ω_i, which could be a very small figure. A possible solution to this situation is to limit the amount of considered candidates to every QT node, by keeping only the best K matches. The impact of this parameter K will be studied in Section 8.4.2.

8.4 Evaluation of QT-BPT Matching

The presented solutions for QT-BPT matching have been evaluated on the *anchorwoman* dataset, which contains 11 instances of a frontal view of an anchor woman in a collection of 243 keyframes extracted from a news video asset. The experiments were designed taking into account the best configurations found in the previous Part II.

The ETHZ dataset used in Part II has been replaced in this Part III because the *anchorwoman* object presents distinguishable parts that repeatedly appear in the annotated instances. On the contrary, the objects in the ETHZ present a high level of visual diversity that make them less appropriate to evaluate the part-based analysis presented in this part. However, the different instances of the *anchorwoman* contained in the dataset do represent the diversity in the creation of the BPT. The same object appears on different topologies, many of them split in multiple sub-trees. These type of situations are precisely the ones addressed by the proposed solutions, so the dataset was considered appropriate to study the impact of the different parameters that govern the matching algorithm. However, Section 8.4.3 has been added with the same experiments run on the *anchorwoman* dataset when applied on the ETHZ dataset. The results obtained show the non-suitability of the QT-BPT matching algorithm for non-composite objects.

This study of the QT-BPT matching has focused on the two parameters that regulate the amount of combinations to consider: the maximum amount of query parts and the maximum amount of candidates for every QT node during the bottom-up matching. These two type tests have also brought additional information about the performance of the object decomposition of the objects in QTs as well as further experiment with the HBoR features presented in Section 6.1.

8.4.1 Query Parts

The first experiment has studied the impact of the amount of query parts in the overall performance. The five trials of the cross-validation scheme have been run on a configuration with no codebooks and with codebooks of different sizes, so that every graph corresponds to a different amount of codewords. Figures 8.14 8.15, 8.16, 8.17 and 8.18 include the results for no codebooks and codebooks of size 2, 4, 8 and 12, respectively. The horizontal axis represents the amount of query parts while the vertical one corresponds to the retrieval, detection, segmentation and search time measures. The codebooks have been generated following the *generic* option, that is, including as many regions from the foreground as from the background. Different codebooks have been considered for every feature, as this is the best option according to the results in Part II. Moreover, the codebook experiments have evaluated the gain obtained with the bottom-up matching when applied after the top-down solution, a configuration which is referred as *full* matching. In this case, the bottom-up

matching considered a maximum of 20 candidates per QT node. This maximum value of 20 candidates was chosen as an upper bound according to the experiments on the amount of candidates per part reported in Section 8.4.2. Finally, the impact of the top-down matching with the region descriptors or with the HBoR is also assessed, a comparison that complements the results obtained in the previous Part II.

Remark 1. *Splitting the query in the appropriate amount of parts improves accuracy*

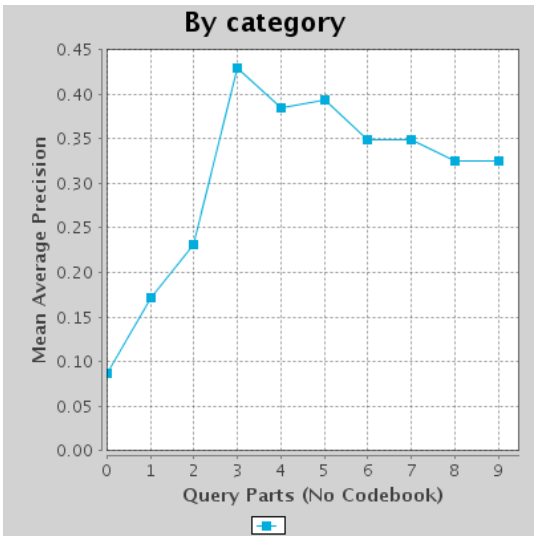
The first observation that can be drawn from the graphs is that splitting the query into parts is beneficial because, in all configurations, the atomic results (1 query part) are improved. This gain though is not unbounded as the results stabilize around the three to five query parts. These figures were expected because the *anchorwoman* object presents three distinctive visual parts: *hair*, *face* and *body*. These three parts appear split in many of the annotated instances, though in some other cases the *hair* and *face* parts appear merged in a *head* part. These basic parts correspond to the leaves in the QT which, in turn, define a total of three to five QT parts. Results indicate that splitting the query in parts is a good strategy. However, using too many parts reduces the performance, both in quality and, especially, in search time. The exponential growth in search time is due to the increasing amount of combinations that must be considered as more query parts are added to the search.

Remark 2. *The bqt-full case for region-based features is not considered due to its high computation requirements*

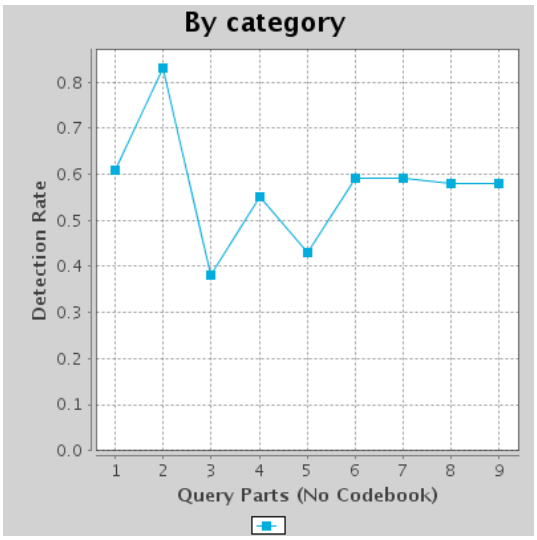
Region-based features are considered both with their native distances, in Figure 8.14, or as quantized visual words, in the plots labelled as *region bqt-topdown* of Figures 8.15 to 8.18. Both configurations have only considered the top-down matching case because this solution does not require the extraction of new visual features from the candidate merges, as it would require an hypothetical *region bqt-full* mode during the bottom-up stage. The extraction of region-based descriptors is a very consuming task because they must be computed from scratch. On the other hand, the fast bottom-up expansion used for HBoR features makes them convenient for assessing merges during the bottom-up approach. For this reason, the comparison between to top down (*bqt-topdown*) and the full solution (*bqt-full*) is only assessed on the HBoR features (*bor*).

Remark 3. *The quantization of region-based descriptors decreases retrieval performance but improves search time*

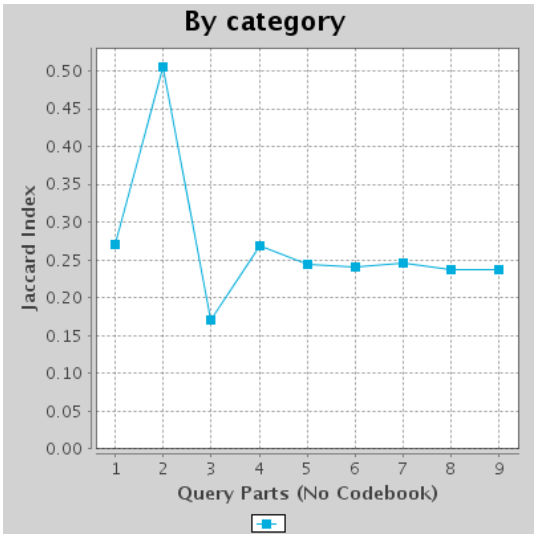
The graph in Figure 8.19 shows the loss in the image retrieval and gains in search times when comparing different codebook sizes with the results obtained by using only region descriptors in the best configuration, the one that decomposes the query in 3 parts. The first difference between using or not using codebooks is the search time, much faster with the simple cosine distance of the codebooks than with the specific distances. On the other hand,



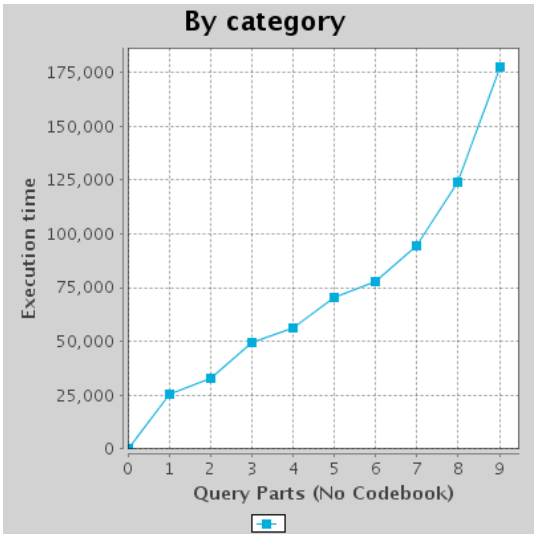
(a) Image Retrieval



(b) Object Detection

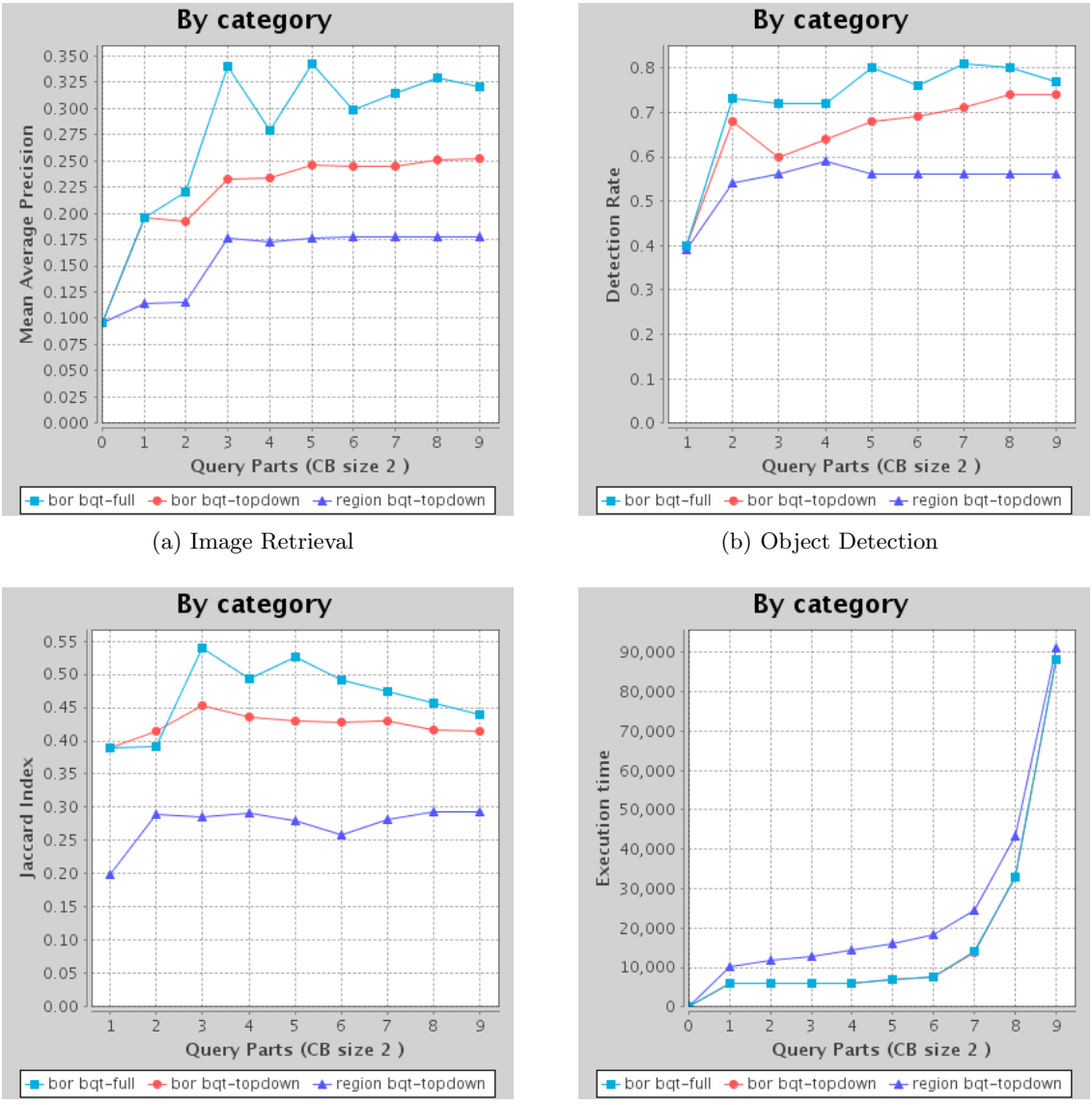


(c) Object Segmentation



(d) Search time

Figure 8.14: Performance dependent on query parts for region-based distances.



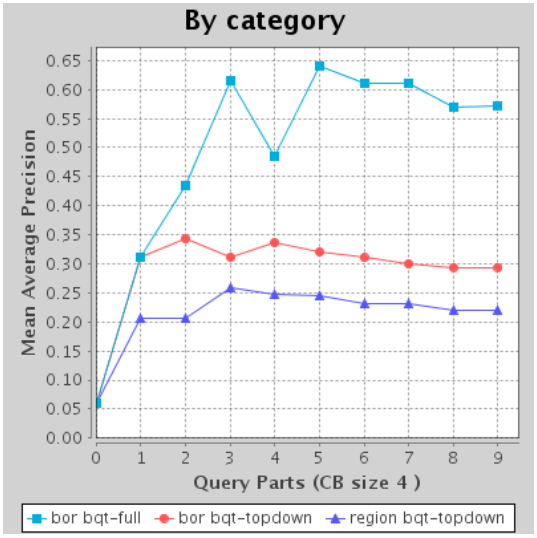
(a) Image Retrieval

(b) Object Detection

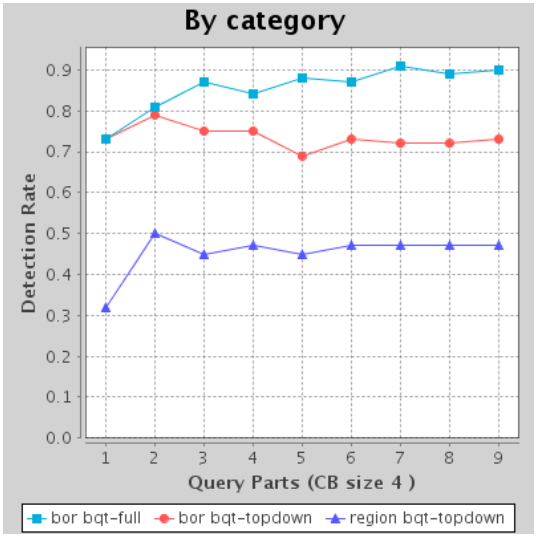
(c) Object Segmentation

(d) Search time

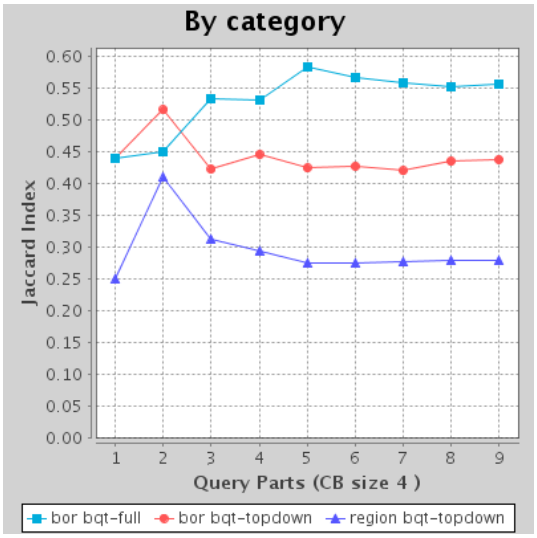
Figure 8.15: Performance dependent on query parts for a codebooks of size 2.



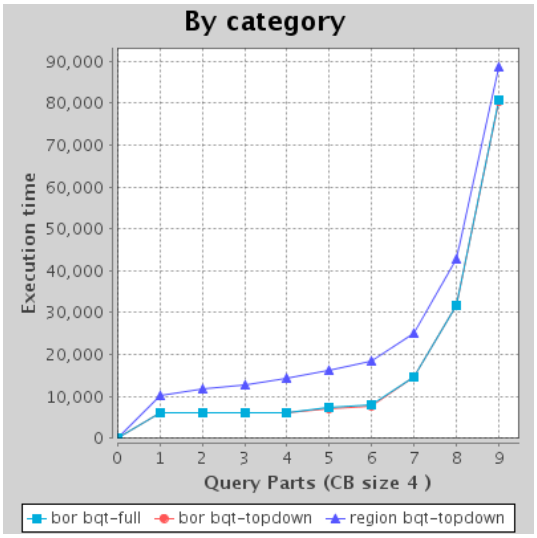
(a) Image Retrieval



(b) Object Detection

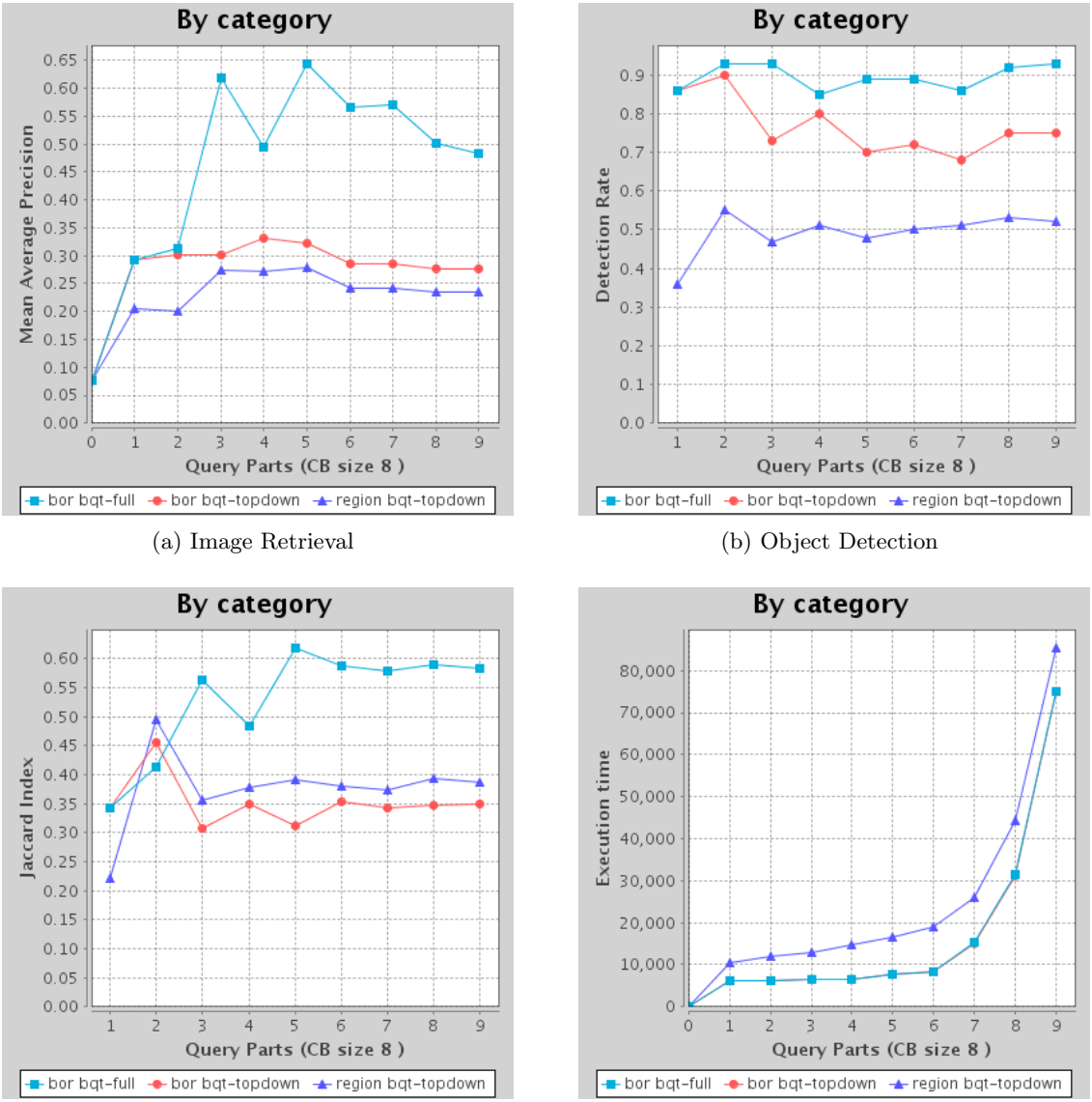


(c) Object Segmentation



(d) Search time

Figure 8.16: Performance dependent on query parts for a codebooks of size 4.



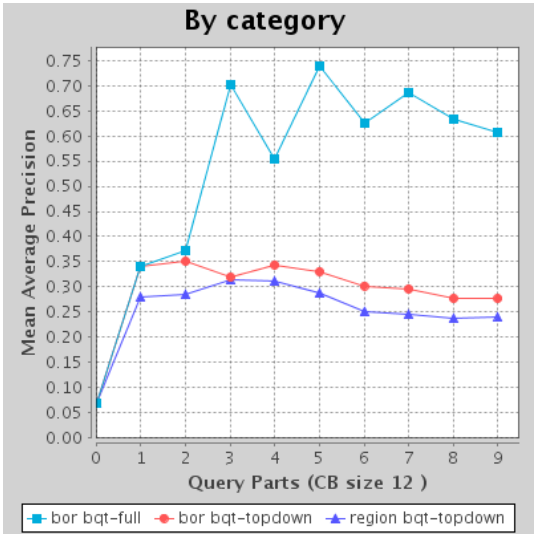
(a) Image Retrieval

(b) Object Detection

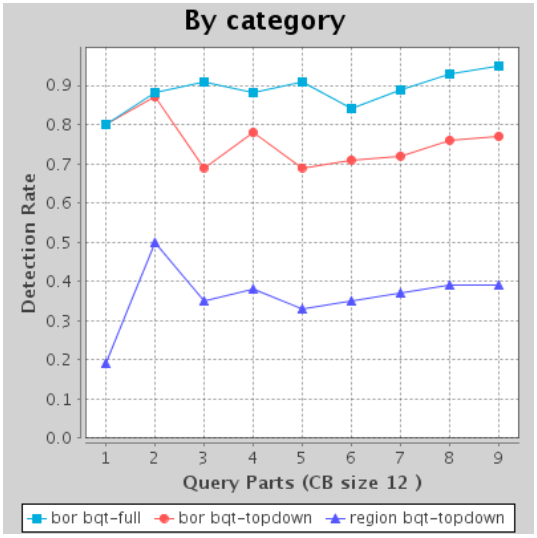
(c) Object Segmentation

(d) Search time

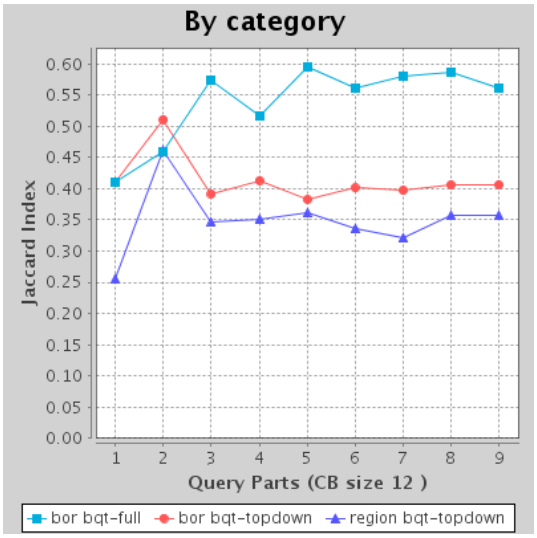
Figure 8.17: Performance dependent on query parts for a codebooks of size 8.



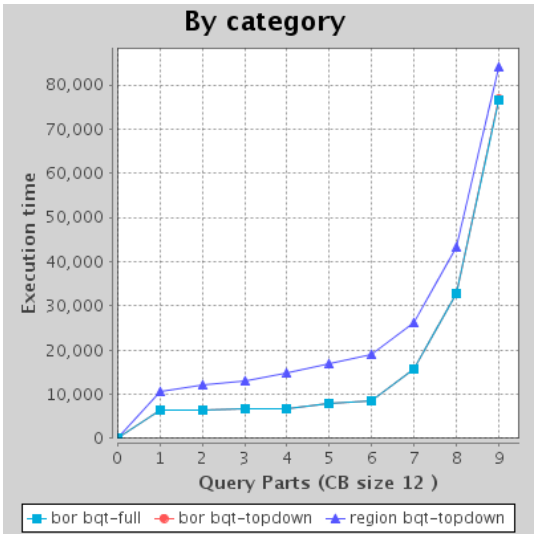
(a) Image Retrieval



(b) Object Detection



(c) Object Segmentation



(d) Search time

Figure 8.18: Performance dependent on query parts for a codebooks of size 12.

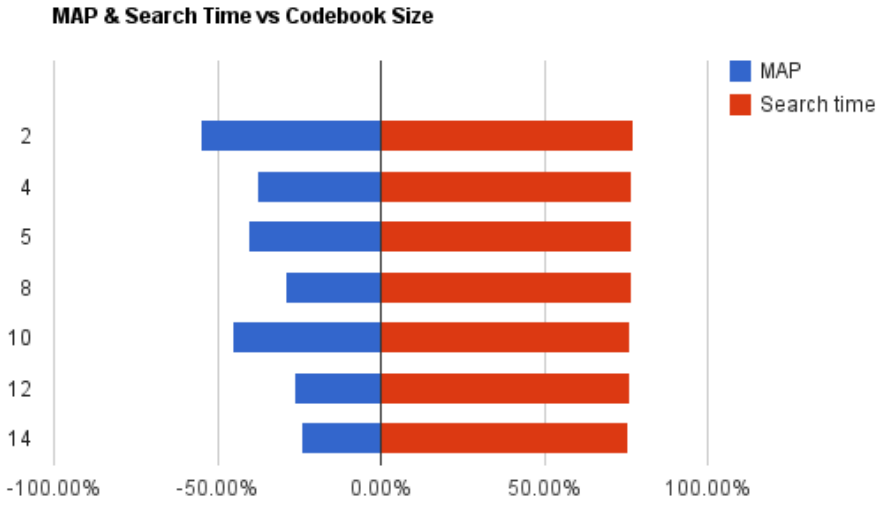


Figure 8.19: Gains and losses of region-based codebooks compared to the no codebook mode.

the results with codebooks are worse in terms of image retrieval. There exist a compromise then between these two factors if only region descriptors are considered.

Remark 4. *Bottom-up matching with HBoR is the best configuration in all aspects*

The graph contained in Figure 8.20 compares the image retrieval results for the best configuration of 3 query parts with different codebook sizes. The drawn lines correspond to the matching algorithm that only uses the *top-down* exploration with the one that also considers the *bottom-up*. Both cases are based on HBoR features. The plots clearly shows that applying the bottom-up matching strategy significantly increases the performance. Its robustness against object splits in the target BPT is clearly beneficial and can increase the retrieval MAP in more than a 100 %. Similar results are obtained for different amounts of query parts.

As expected, results improve in parallel with the size of the codebooks. The more codewords in them, the more precise are the feature vectors. For this reason, the best results are obtained with the largest codebook considered. Results also indicate that the impact in computation time of the codebook size is far away from the one on the amount of query parts.

In terms of computation time, the usage of HBoR features and the limitation of 20 candidates per region allow full and top-down times almost identical if compared to the region-based case. For this reason, the red top-down curve does not appear in the (d) graphs, as it is covered by the cyan one corresponding to the full solution.

Remark 5. *HBoR also outperform region-based features with the "anchorwoman" dataset*

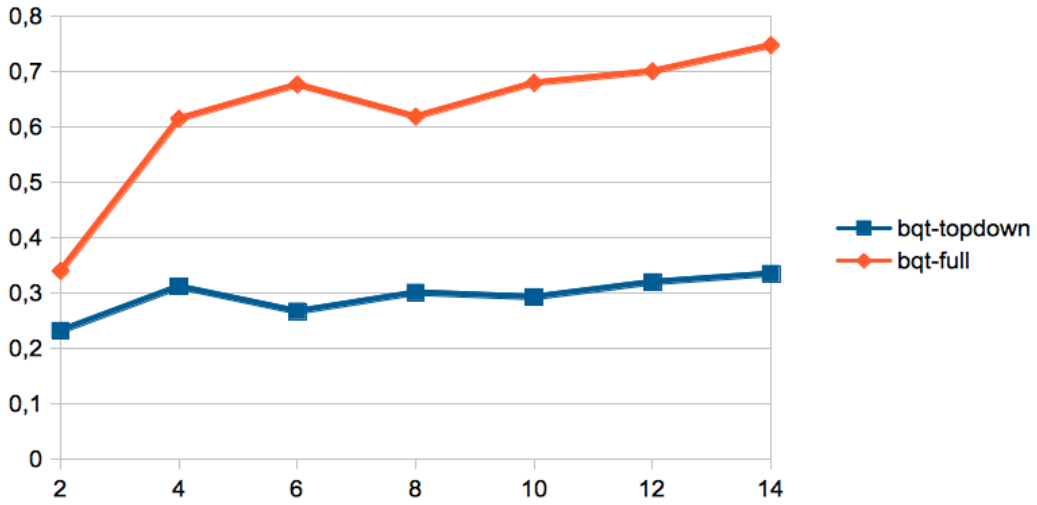


Figure 8.20: Top-down vs full QT-BPT matching with HBoR features.

If focusing on the top-down matching, region and HBoR features can be compared again, as previously studied in Part II. The HBoR features outperform the region-based ones. However, when codebooks are small, region-based features are slightly more reliable in terms of segmentation. This is because they are much more precise than the HBoR ones, which somehow make a raw approximation as they only consider the included parts, but not how they are structured. In terms of search time, the extraction of the region-based features from the query parts not included in the QT is the responsible of its difference with respect to the HBoR case.

8.4.2 Parts Candidates

The results obtained during the analysis of the object split in query parts have indicated that the bottom-up approach is a feasible strategy to overcome the object split problems in the target BPTs. This algorithm considers candidate regions that are not represented on the BPT. Despite the dynamic threshold update restriction, the amount of combinations that can be generated is explosive. The proposed method to control this proliferation of hypothesis is limiting the amount of candidates to every query part. This section studies the impact of this parameter in the *anchorwoman* dataset.

This experiment considers the full matching configuration, that is, the top-down and posterior bottom-up. The limitation on the amount of part candidates applies only in the bottom-up case, as the top-down is alred self-limited when considers regions in the sub-trees. Only the HBoR features have been considered because, as previously discussed, region-based features are not feasible in the bottom-up strategy. This time, the size of the codebook has been fixed to 14 and the amount of query parts to 5, because these values achieved best results

in the previous section. The results are shown in the four graphs of Figure 8.21, where the horizontal axis corresponds to the amount of candidates considered for every QT part. The three lines shown represent three different configurations for codebook creation: using only foreground parts (fused and non-fused) or also background parts.

Remark 6. *Most cases are already solved with only five candidates per part*

The graphs in Figure 8.21 indicate that most of the gain is obtained during the first five candidates, but that afterwards the improvement is minimum and could even slightly decrease the performance. This means that normally the best merges from a local perspective are already the best merges for the global match. It is not necessary to consider a large amount of candidates to get close to the best result. Notice that, due to the limiting nature of these ranked lists, the optimal result cannot be guaranteed in the bottom-up case. However, these empirical results suggest that the best solution may be found in most of the cases.

Remark 7. *Search time does not grow exponentially with the amount of part candidates*

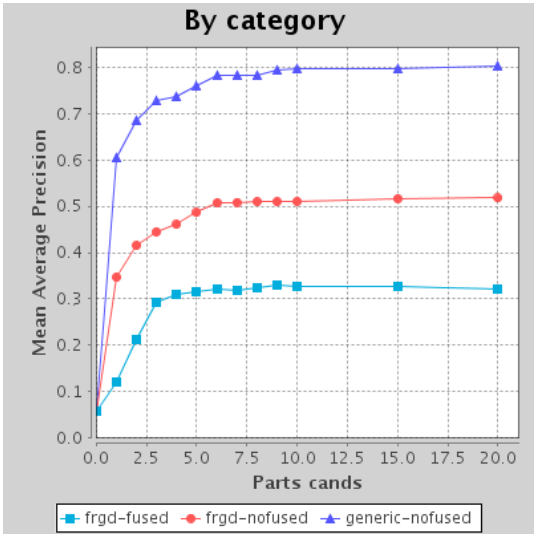
The search time curves offer an almost flat behaviour, although the more query part candidates, the more computational effort was expected. This flatness may be due to the minimum distance threshold set during the previous top-down matching. This value is already a limitation to avoid the explosive growth of combinations, especially in the considered five parts case. The ω_i weights associated to the QT leaves are large enough to discard combinations when multiplied with high d_i distances. The growing tendency would be more visible with more query parts, as the smaller ones would be associated to ω_i too small to allow a filtering by threshold. In these cases, the limitation by amount of parts would be more relevant.

8.4.3 Non-composite objects

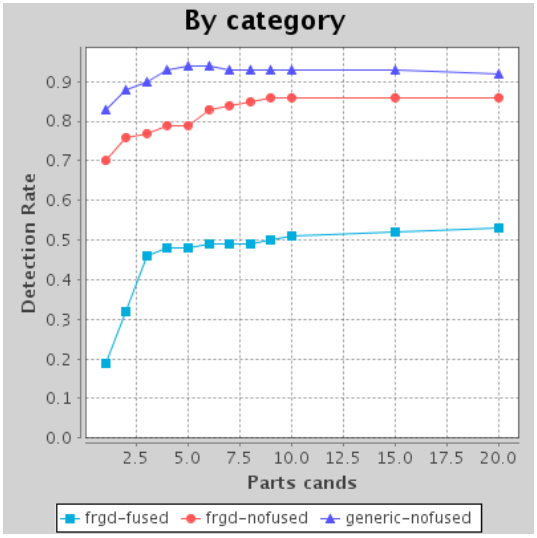
The experiments reported on amount of query parts have been repeated by considering the ETHZ dataset used in Part II. Only the best case of 120 codewords and 5 candidates per part have been considered as they already offer a valuable insight of the impact of QT-BPT matching techniques for this dataset.

Remark 8. *Little impact of query parts for non-composite objects*

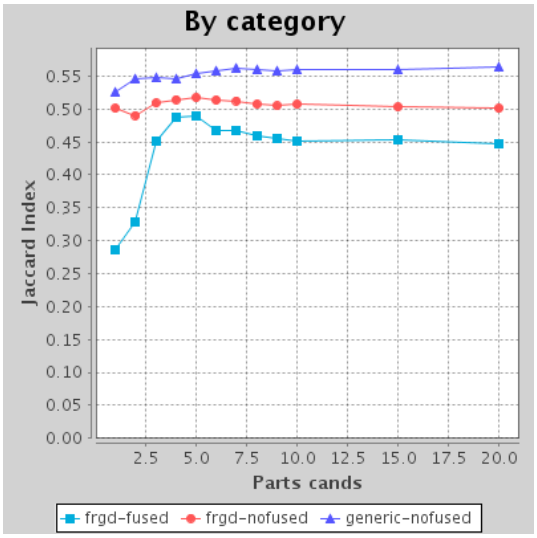
Figure 8.22 shows the obtained plots, which basically draw a flat line that indicates independence from the amount of query parts. These results state that there is no gain in this dataset when considering objects as a combination of parts. This is because, even if assuming that the BPT provides a correct segmentation in terms of object contours, it is very difficult that the decomposition in parts will be repeated between instances. The objects included in the ETHZ dataset were basically chosen for sharing a distinctive *shape*, but not to present any discriminative pattern in terms of parts.



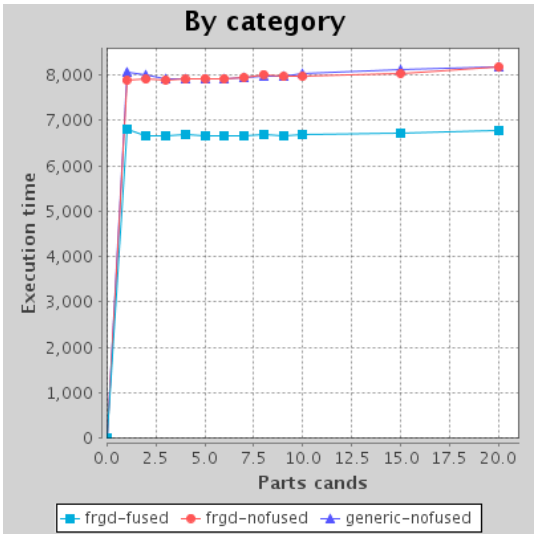
(a) Image Retrieval



(b) Object Detection



(c) Object Segmentation



(d) Search time

Figure 8.21: Performance dependent on the amount of candidates for every QT node.

Remark 9. *Splitting atomic objects unnecessary introduces diversity of parts*

The only exception to the neutral behaviour is in the *detection* plot b), where the single-part case outperforms the cases where the query is split in parts. This behaviour probably corresponds to the single-part objects (eg. white Apple logo), whose decomposition is negative because a non-composite object is forced to be treated as composite. The enforcement of object parts will inevitable decrease performance, because such decomposition will be significantly diverse between different instances and will introduce a pattern which is not repeated. This case indicates that the amount of parts should be object dependent to avoid introducing artificial parts.

This independent behaviour from the amount of query parts supports the decision adopted in this Part III to replace the challenging ETHZ dataset for the simpler but more appropriate *anchorwoman* dataset.

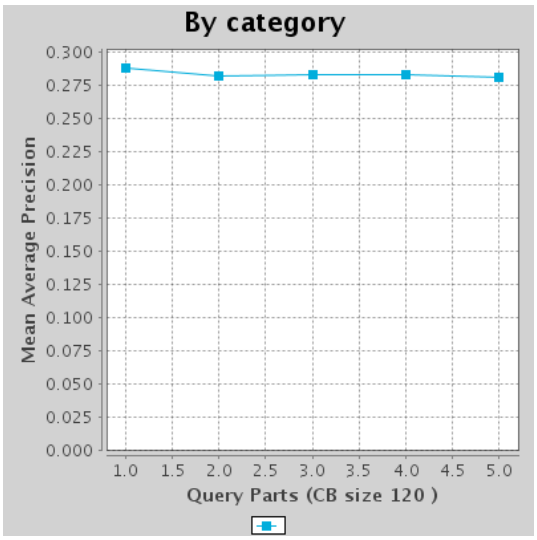
8.5 Summary

This chapter has addressed the query by example problem, where an instance of the object defines a query for image retrieval. The adopted solution represents the query object as a hierarchy, the *Query Tree (QT)*, so that the search algorithm corresponds to solving a correspondence problem between the QT and each BPT in the target dataset.

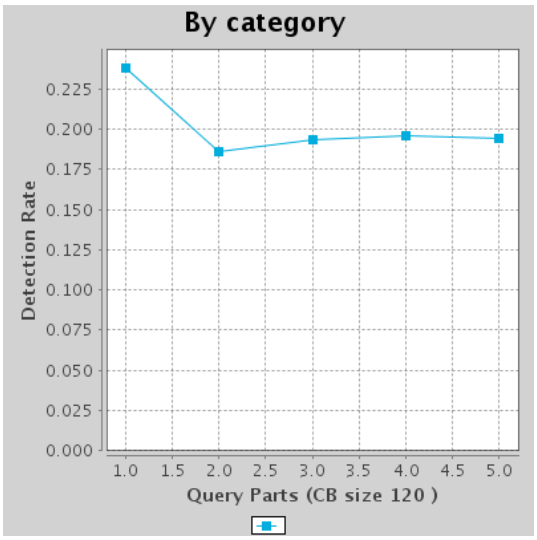
The quality of each considered matching is assessed by a newly proposed metric that assigns weights to each part in the QT according to their relative sizes. This distance is computed after every matching of a QT node to BPT node. This sequential increase allows an early reject of those correspondences that cannot improve the best match found so far.

The proposed algorithm has two distinctive stages. The first one follows a top-down approach in such a way that every QT node match reduces the search space for its descendants among the target sub-BPT. The second stage is a bottom-up strategy that handles the object splits in the target BPT at the expense of a most computational effort, as it will consider regions which were not defined during the construction of the target BPT. This situation can be handled by using light-computational features, such as the HBoR introduced in Chapter 6, as well as a dynamic update of the similarity distance that sets a threshold for every new matching.

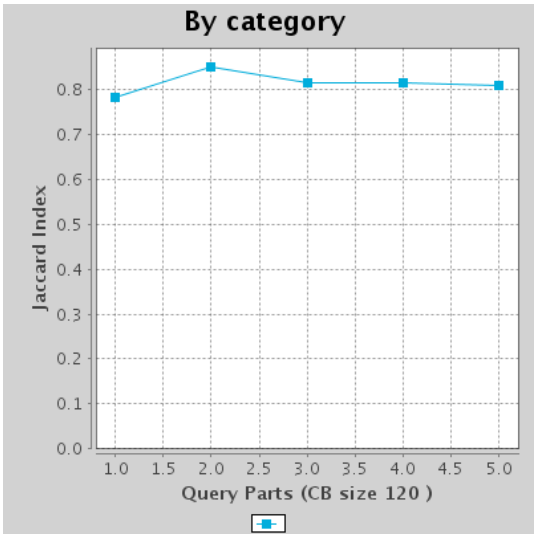
The proposed technique has been tested on a dataset that contains several instances of a composite and single-view object, an *anchorwoman*. The experiments showed that decomposing the query object in parts improves the retrieval performance at the expense of an exponential growth in search time. A second set of tests showed that it is not necessary to consider a large amount of candidates to match every query part in the bottom-up approach. In the considered dataset, working a maximum of five candidates achieved similar results to considering up to twenty.



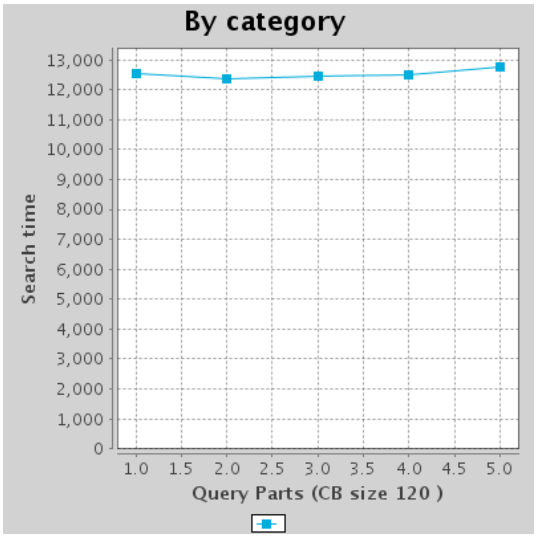
(a) Image Retrieval



(b) Object Detection



(c) Object Segmentation



(d) Search time

Figure 8.22: Performance for non-composite object classes.

Chapter 9

Part-based Object Model

The solutions considered so far in this thesis have addressed the problem of, given a single query object, finding the best matches among the images in the target dataset. This chapter addresses a new task where the query is not a single instance of an object but multiple ones. This situation is normally referred as *query by concept*, as the collection of query instances aims at modelling the object as a semantic class instead of an individual.

A first solution for this scenario would be formulating an individual query for every instance and applying a late fusion on the obtained ranked lists. This approach presents two drawbacks though. The first one is that, in terms of computation, performing as many individual searches as query instances linearly increases the response time with the amount of exemplars. Secondly, treating each query instance separately prevents from identifying the common patterns among them, which is precisely what may best define the object class.

The approach followed in this chapter is based on analysing the multiple query instances before the search to try to extract their common patterns. The result of this study will be a model of the object, which is to be compared with each image in the target dataset. This model assumes the same role that the query instance played in the previous Chapter 8. This problem is also referred in the literature as *Multiple-Instance Learning* [106] [107], defined as a variation of supervised learning where the task is to learn a concept given positive and negative bags of instances.

The solution proposed is also based on Partition Trees (PTs). However, as previously exposed in Section 2.4, assuming that the target PTs will contain good segmentations of an object among its nodes is a very restrictive requirement. Similarly to the matching case presented in Chapter 8, the object will not be considered as a whole but as a composition of parts.

The basic assumption in this chapter is that the common pattern among the query instances can be found among their visual parts. The adopted solution processes objects as sets of parts and tries to identify which parts appear repeatedly in the provided query exemplars.

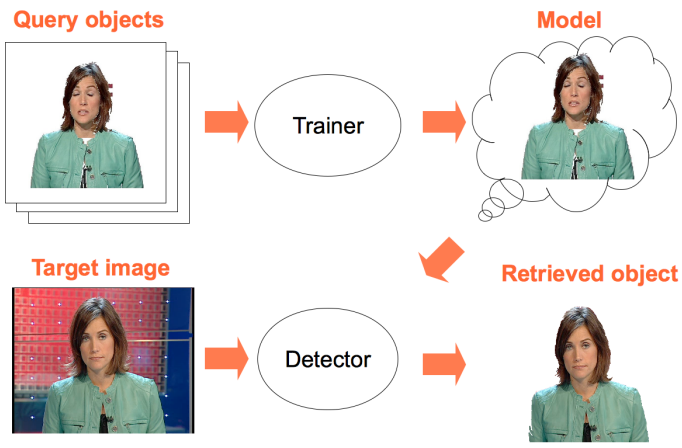


Figure 9.1: Model Training and Detection stages.

This repetition indicates the presence of a pattern and, when detected, it is added to the model.

It must be noticed that, although this chapter studies the presented technique applied to instance search, the algorithm basically provides a probability score that is used to generate the ranked list. This same framework can be used to solve an object detection and categorization problem, as argued in [61]. The adaptation only requires a decision threshold to be compared with the confidence score provided by the classifier. For example, in the case of the object class *anchorwoman* depicted in Figure 9.1, a set of annotated objects are manually segmented to train a model. This model is used to detect new instances of the *anchorwoman* object class in the target images.

The chapter is organised as follows: Section 9.1 discusses previous works based on image parts in the field of object detection and categorization. Our contributions begin in Section 9.2, that presents how user annotations are treated to determine the parts of the objects and how these annotations are processed to train a model for each part. Section 9.3 explains how the model of the complete object is built through a late fusion of the detected parts. Section 9.4 presents a post-filtering stage to give semantic coherence to the detections within an image. Finally Section 9.5 describes the evaluation of the presented techniques, where the model-based results are compared with the single-query strategy introduced in Chapter 8.

9.1 Related work

Jaimes and Chang (2001)

The detection of objects from its parts has received the attention of several works in the last decade. The most similar one to ours is the *Visual Apprentice* by Jaimes and Chang [41].

They claimed that successfully content-based retrieval systems should be based on generic techniques that would allow users to construct their own models. Their object model was built according to a *definition hierarchy*, a structure that represents objects at four scales and that was already introduced in Chapter 4. In their system, users were asked to manually define a hierarchy for each new concept and annotate a training dataset by individually assigning image parts to every node in the *definition hierarchy*. This approach presents the drawback that the user must explicitly define the decomposition of the object in parts, a direction that was also explored during this thesis [32]. However, this option was abandoned due to the high level of user interaction it required. Instead, a more simple object model was proposed with only two levels, *object-part* and *object*, where the user only needs to annotate the complete object. The composition of the *object-parts* level is automatically learned during training.

The Visual Apprentice worked with video objects whose training samples were generated by interactively segmenting and labelling a video frame. The manually generated labels were expanded in time by applying a region tracker, which allowed the characterization of objects also in terms of motion. Those regions that were not labelled were used as negative observations when training the classifiers of the object parts. This strategy is known as the closed-world assumption [10] and is also adopted in this thesis. Every node in the annotated hierarchies was processed to extract their features. Generic visual descriptors (color, shape, texture, area, location and motion) were extracted from each image part, but a different type of feature was computed for those elements in the hierarchy with a structure underneath. In these cases, the feature vector was built from an Attributed Relational Graph (ARG) [91], a type of graph whose nodes represent elements and arcs represent the spatial relationships between elements (above/below, right of / left of, near/far, adjacent, inside/contains). The classification of new images/videos followed the bottom-up order of the hierarchy defined by the user, which started by detecting instances for the lowest level of the hierarchy on their own. Afterwards, the groups of adjacent sub-parts were considered to define an instance of the upper part. The process was iteratively repeated until the complete object is created. Our work shares the bottom-up construction of objects by mapping the sub-parts into a feature vector. However, the requirements in terms of spatial relationships are simpler, as we only consider the *adjacency* between the detected parts.

Ravinovich, Vedaldi, Galleguillos, Wiewiora and Belongie (2007)

Ravinovich et al [69] use a region-based representation of the image to detect objects, putting special care in the semantic context of the detections. Their solution is based on a training set of images annotated at the global scale, allowing an image to contain several labels describing the objects it contains. Every region in the test image is analysed to estimate the probability of belonging to an image of each class, that is, there are as many probability values associated to every region as object classes. The probability for the best labelling is estimated by

comparing the similarity to the best match with the one obtained with the second best labelling. The higher the difference between the two distances, the higher the probability of a correct labelling. The probabilities for the rest of classes are considered equal and adjusted so that the sum of probabilities adds to one. This uniformity seems a very restrictive solution because it is treating the same way all non-top categories. For example, if a region is very similar to the training regions from different categories, only the top option will be treated as preferable. Our proposal does not apply this hard decision at this point and a region can be a part candidate to several object classes. In fact, our solution allows a region to belong to several candidate objects. It is not until a late contextual filtering on the detected objects that a final decision about the categorization of a region is taken.

Ravinovich's solution works with regions coming from different segmentations of the same image, as previously presented in Section 2.1. This scenario inevitably generates regions that overlap. The proposal of the authors to manage the situation is splitted in two parts. Firstly only those regions with the same label as the complete image are considered. All overlaps are resolved and only the k segments with a highest probability are considered in future steps. Secondly, those segments with labels different from the label associated to the whole image are also filtered to remove any overlap and again only the k remaining regions are further analysed. The authors do not clarify how the overlaps are solved and what is the reason for only considering the k most confident labels. The overlapping could be solved by prioritizing the region with the highest confidence and, probably reducing the among of considered regions to k is a solution to reduce computation time. However, applying hard decisions relying on the predicted label for the complete image or relying on the k best estimations may discard hypotheses with high confidence. In our solution the hypotheses are discarded based on a minimum probability threshold, that avoids losing highly confident labels and, on the other hand, can remove low confidence hypotheses that may be considered if applying the criterion of the k top elements.

Moreover, the probabilities obtained for every region are weighted according to the amount of interest points they contain. The authors justify this solution as a measure of reliability of the predicted label. However, this operation is also biasing the object categorization in favour of those objects that present textures with high frequencies. Our work also prioritizes regions when selecting anchor points to construct the objects. However, this selection is based only on the estimated probabilities of the labelling, a criterion that will evenly treat objects with high and low spatial frequencies. In the previous Section 8.2 addressing the matching problem, the parts of the query object were also prioritized based on the reliability of their matching, but the criterion was based on their sizes, as expressed in Equation 8.1.

Finally, the labels obtained for every region are filtered by introducing contextual information using a *Conditional Random Field (CRF)*. By doing this, the authors completely separate the initial categorization of segments with the addition of the contextual informa-

tion. This allows them to work with simpler CRFs, a type of graphical model that requires an important computation effort to be learned. The context is introduced in terms of a co-occurrence matrix, whose elements count the amount of images on the training set where two labels co-occur. So, in this solution, no spatial information between the objects is considered as no local annotations are available, the context only refers to the co-occurence of labels within the image. This global approach prevents from identifying if too many instances of the same object class are detected within the same image, as multiple detections would be anyway annotated with a single label for the whole image.

Gu, Lim, Arbelaez and Malik (2009)

Gu et al [37] face the object detection and segmentation problem also with a Partition Tree (PT), which they name *region tree*. However, the hierarchical structure of the tree is discarded after segmentation, and the obtained set of regions is further analysed structureless following the *bag of regions* paradigm. In our work, as presented in Section 6.1, the PT is not only used to create the bag of regions, but also to define the Hierarchical Bag of Regions (HBoR) features, as in in the Spatial Pyramid solution [51]. Each region of the bag is described by subdividing its bounding box into a 4x4 grid of cells. Every cell is analysed to extract four types of descriptors based on contours, edges, colours and textures. The description of the regions is generated by computing four histograms with the cell features, one for every type of visual descriptor.

In their work they use a previous technique from Frome et al [28] that estimates the significance of each region in the training set with respect to its associated object class. A class-dependent weight is learned for each region by minimizing a region-based distance between images from the same class but trying to maximize this same distance for images belonging to different classes. These weights allow the definition of a distance for each training image which, in turn, is used to estimate a probability value. These distances consider the complete image, so it is not required for users to generate local annotations. On the other hand, the algorithm may assign high weights to regions belonging to the background and, so, basing the object detections on the context. This approach may benefit from the context for the detection of objects, but at the cost of providing a solution which is not so robust to context changes.

The detection of objects begins with a *voting* process which, given a test image and an object category, generates hypothesis of bounding boxes that may include an instance of the object. These hypotheses are generated by comparing every region of every image in the training set of the object class with every region in the test image. Every possible match is quantified with a *voting* score that combines the weight learned from the region in the training image and the perceptual similarity with the considered region in the test image. Only the bounding boxes of those matches with a certain minimum *voting* score are considered in

future stages of the algorithm. Comparing every region in the training dataset with every region in the test image is a costly process that compromises scalability, as the response time of the system will linearly increase with the amount of training samples. For this reason, our solution is based on a single model from the training dataset that synthesizes the object patterns. Moreover, our proposal exploits the hierarchical structure of the PT to avoid the exploration of every possible region in the target image.

The bounding boxes of the matched regions are compared to estimate the translation and scaling transformations from the region in the training object to the region in the test image. The transformation parameters are applied on the bounding box of the complete object to obtain a candidate bounding box for the object in the test image. Afterwards, all bounding boxes in a same test image are clustered to fuse the different hypotheses suggested by the individual part matchings. This way, the location of the matching regions in the test image provides an anchor point for a bounding box to contain the complete object. Notice that, at this stage, the initial segmentation is discarded in favour of a bounding box. This is another difference from our solution, that always works with the regions defined in the initial Partition Tree. This way, during analysis, the computing effort required for extracting new visual features is minimal, as previously exposed in Section 6.1.3.

The next stage is the *verification*, where every candidate bounding box is compared again with every training image. The probability score is computed for every comparison and all values are averaged into a single *verification* score. The reason for this averaging is not clear in [37], given that the ETHZ dataset considered by the authors (described in Section 1.3.2) contains instances with a high visual variability inside the same object class. So, even if the candidate bounding box is exactly identical to one of the objects in the training set, if it is not similar to many of the rest the obtained verification score will be low. The final *detection* score is obtained by fusing the *voting* and *verification* score with product.

Finally, a third phase is responsible of the *segmentation* of the detected object. This stage recovers the Partition Tree (region tree) to propagate foreground and background labels through the hierarchy of regions [4]. An estimation of the object mask is generated for every matching between the training image and a region in the test image. These masks are weighted according to the detection score associated and combined all together on the test image. The result is a *confidence map* that shows the predicted locations of the object instances. This map allows the plot of accurate precision-recall curves depending on the minimum score required to classify a pixel as foreground (object) or background.

Vieux, Benois-Pineau, Domenger and Braquelaire (2011)

Vieux et al [94] also solve the object detection problem through a bottom-up approach that initially labels regions to later build composite objects. They work with local annotations at the pixel level that are mapped onto the regions generated by four different segmentation

algorithms. The training dataset only considers those regions with at least 80% of their area overlapping the same label in the ground truth segmentation.

The obtained annotation is used to train a *Support Vevctor Machine (SVM)* classifier [92] for each pair of considered classes to provide a discrete output. This result is later mapped into a posterior class probability through a sigmoid function whose parameters are learned for each pair of classes [68]. The resulting set of probabilities, one for each possible pair of object classes, are combined according to [103] to obtain the set of multi-class probabilities associated to every region. This multi-class approach at this stage presents similar limitations to the ones associated to the probabilistic feature vectors discussed in Section 6.1.1. If two object classes are visually similar, their associated probability weights will always be distributed and, by doing so, the labelling score decreases. Nevertheless, an independent probability estimation for each class based only on foreground and background labels, as proposed in our work, could avoid these effects.

After the multi-class labelling of every region, the next step introduces the contextual information through the *Relaxation Labelling* proposed by Rosenfeld et al [70]. This is an iterative algorithm based on two parameters. Firstly, the neighborhood between segments, which is quantified with the length of the shared boundaries. Secondly, the learned co-occurrence of adjacent segment labels.

Notice that the neighborhood of a region is only defined within a segmentation, while this technique considers multiple segmentations generated by different algorithms. The authors proposed to fuse the labelling by creating a new high resolution partition of the image that will incorporate all boundaries generated in the considered segmentations. The multi-class probabilities obtained from each segmentation are combined with a max, average or product fusion to decide the final label.

In their conclusions, the authors precisely point at multi-scale image partitions as future steps in their work, like the PT adopted in this thesis.

9.2 Extraction of Parts

Our solution for the detection of objects follows a bottom-up approach that begins with the exploration of the PT to try to discern which nodes might represent a part of the object. This section focuses on this initial labelling of PT nodes, while Section 9.3 studies how the detected parts can be combined to build the complete object.

Before trying to model the different parts of an object class, it is necessary to determine in which parts can an object be split. For this reason, this section is divided in two: the definition of parts is addressed in Section 9.2.1, while the modelling and detection of each of them is discussed in Section 9.2.2.

Figure 9.2: Local Annotations of the object of interest.

9.2.1 Parts Definition

The variability in the amount of parts between different objects can be great. While some object classes are very homogeneous in terms of visual features and can always be represented with a single and invariant node in the PT, others may always require multiple PT nodes which could at the same time greatly differ between instances of the same object class. The model we propose handles this variability by learning the amount of different parts into which an instance of a certain object class can be decomposed.

This strategy naturally handles multiview variability, as each view of an object can be modelled with its own set of parts. Afterwards, during detection, the full representation of the object may require only a subset of all modelled parts, just the necessary to fully represent a view. The modelled parts do not need to be specific for a particular view, they can be shared by models of different views.

The presented technique requires a training set of objects annotated as one or several nodes of the PT. The framework is both valid for local and global scales, corresponding the later case to an annotation linked to the root of the PT. Figure 9.2 provides an example of local annotation of the semantic class *anchorwoman*. Whether in a local or global case, one must decide which nodes of the PT are to be considered as parts of the object. Given the hierarchical nature of the PT, several scales are available, each of them associated to a different decomposition of the object. The proposed solution considers that, if the training set is representative enough, it is reasonable to assume that the object splits in the training set will similarly occur on the test data. For this reason, our solution considers the roots of the sub-BPTs that represent the object as the appropriate scale for the decomposition of objects into parts. This solution is consistent with the *Object Trees (OTs)* introduced in Chapter 4.

Once decided the spatial scale of the parts, the next question that arises is how many parts are necessary to model an object. The answer would be very simple if all OTs from the training set presented the same amount and type of sub-BPT roots but, in general, this is not the case. The same object from the same view may appear split in a different amount of sub-

Figure 9.3: Clustering of the object parts.

BPTs from different images due to the variability inherent to the segmentation. Moreover, the same object may be represented by different parts depending on the point of view (eg. front - back), or the same object class may itself be represented by dissimilar visual instances. To sum up, it cannot be assumed that all the instances of an object class will be composed by neither the same amount nor type of sub-BPTs roots.

Determining the amount of parts is basically an unsupervised clustering problem. The goal is to group those parts which are visually similar, an analogous problem to the one addressed during codebook creation. The difference now is that, instead of using a supervised algorithm where the amount of clusters (codebook size) is predefined, the amount of object parts is unknown and needs to be estimated during the clustering process. If results are satisfactory, a cluster for every object part will be created, as shown in Figure 9.3. This Figure represents an ideal output of such clustering process, where three types of object parts have been successfully identified.

Quality Threshold Clustering

In our work, the unsupervised clustering problem has been solved with the the Quality Threshold algorithm proposed by Heyer [39]. This technique imposes a minimum *quality threshold* between their members. In our context, the quality measure corresponds to the visual similarity between all the image parts in a cluster.

This algorithm requires two configuration parameters. Firstly, the quality threshold T that defines the maximum distance between the center of a cluster and any of its members. The second parameter is the minimum amount of necessary items N_{min} to form a cluster. This value allows discarding those image parts that only appear in few training samples and, for this reason, they cannot be considered representative enough of the object class.

The Quality Threshold algorithm assesses the mutual distance between all observations and identifies which of them defines the center of a circular area that includes the highest

amount of other observations. If this value satisfies N_{min}, then the cluster is created and all included observations are removed from the data set. The process is iteratively repeated until no cluster can be found so that it accomplishes the two T and N_{min} requirements.

The Quality Threshold algorithm has the advantage that only those clusters with a minimum amount of predefined members will be considered, providing a minimum quality that guarantees representativeness. On the other hand, all possible combinations must be considered, so all mutual distances between observations must be calculated. This approach is very demanding in terms of computation so, in the case of a large amount of samples, it is advisable to apply a pre-filtering to reduce the computation effort. [39]

9.2.2 Local Hierarchical Analysis

The detection of parts in a hierarchical structure such as the Partition Tree can be efficiently performed by exploiting the HBoR features presented in Section 6.1. The proposed algorithm follows a top-down exploration of a test Partition Tree and, at each node, evaluates two different conditions: whether the subtree defined by the node contains the modelled part and whether the region represented by the PT node is itself an instance of the part.

The decision regarding the inclusion or detection of a part is based on a binary Support Vector Machine classifier implemented in the libSVM library [16]. This tool provides a prediction and confidence score for a test observation according to a model learnt from a set of labelled observations in a training set. As presented in Section 7.2, the PT nodes may be represented by a single feature vector resulting from a previous fusion of the visual descriptors, but it may also be related to multiple feature vectors, one for each considered visual descriptor. While in the first case only one classifier will be learnt, the second situation requires some sort of fusion strategy. If this is the case, the *late fusion* solution has been adopted, which basically trains a classifier for every visual descriptor and uses the resulting predictions to build a new fused feature vector. This feature vector is the input to an additional classifier responsible to generate a single and final prediction value. This strategy has been reported in [84] to perform slightly better than *early fusion*, which corresponds to creating a fused vector of features by concatenating the vectors for every visual descriptor. The parameters that tune the SVM classifiers have been estimated with a grid search, as suggested in [40].

Efficient Top-down Exploration through the PT

The search for object parts among the PT nodes follows a top down exploration of the tree that starts from the root. Nevertheless, not all nodes need to be individually analysed. We propose an efficient exploration by using classifiers capable of discerning whether a subtree includes a part of the object. These classifiers are based on *recursive* features, which were presented in 5.2. This type of visual descriptors naturally contain information about the

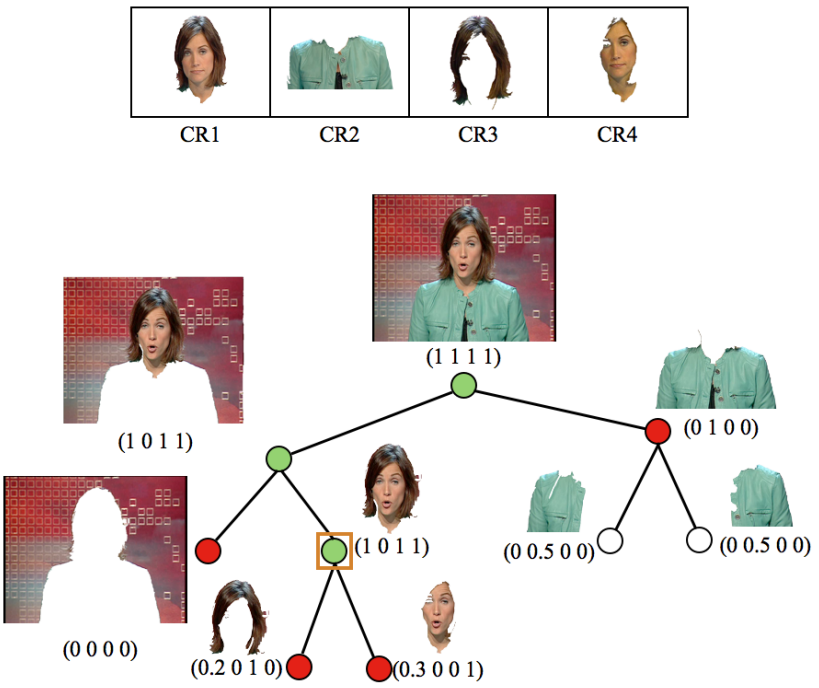


Figure 9.4: Training labels and HBoR feature vectors for the inclusion classifier.

regions under a PT node. For example, the HBoR features introduced in 6.1 satisfy this property and are adopted in the experiments presented in this chapter.

The efficient top-down exploration of the PT requires a dedicated *inclusion* classifier for each type of object part that has been identified, so a different training dataset must be built for each of them. The positive labels correspond to all PT nodes that include one or more instances of the modelled part among its children. On the other hand, negative observations may have two different origins: (a) the children of the nodes that represent a part, which correspond to splits of this part, and (b) the siblings of the nodes representing a part, as long as they do not include any other instance of the same type of part. This scheme generates a set of training samples that contains observations from near the frontier between the *included/non-included* classes. Figure 9.4 shows and example of positive and negative labelling in the case that the part *head* is modelled. Positive observations are colored in green and the negative ones in red. The HBoR feature vectors have been added to exemplify the application of these feature vectors. In this case, the classifier should be able to learn that the inclusion of the *head* part is related to the first component of the feature vector.

The efficiency during the top-down exploration of the test PT is achieved by applying the *inclusion* classifier at every node. The prediction of this classifier determines if the subtree must be further explored. Figure 9.5 represents the application of the trained classifier in the same PT. In this case, the nodes in white represent the regions that will not even be assessed

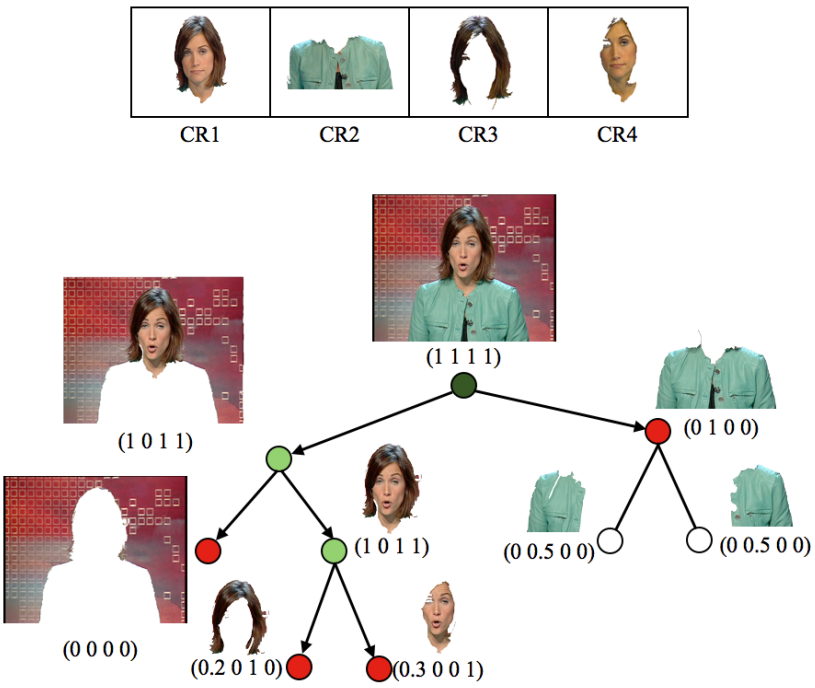


Figure 9.5: Top-down exploration with the HBoR-based inclusion classifier.

to represent the *head* part as they will have been previously discarded by a negative prediction (in red) on one of their ancestors. The brightness of the green represents the confidence of a subtree to contain a part, so the green nodes draw a path from the root to the actual *head*.

The decision about which subtrees to explore will in the end depend on a minimum threshold evaluated on the probability scores generated by the classifier. This threshold balances the recall of the results and the computational time.

Part Detection

During the top-down exploration of the PT, every visited node is also considered to be itself an instance of the object part. A different *detection* classifier is trained to perform this task. Like in the *inclusion* case, this classifier will provide a confidence score that must be compared with a threshold to decide about the presence of the part. This threshold value defines a compromise between the precision and recall of the results.

Similarly to the case of the inclusion classifier, it is necessary to determine which nodes from the annotated training set are to be used as observations for the binary classifier. For the positive case, the actual regions representing the part are the ones selected. In the negative case, given the variability of the non-part regions, the observations have been obtained from a random sampling among the rest of PT nodes. This random sampling contains as many observations as the positive training set to create a balanced amount of observations for both

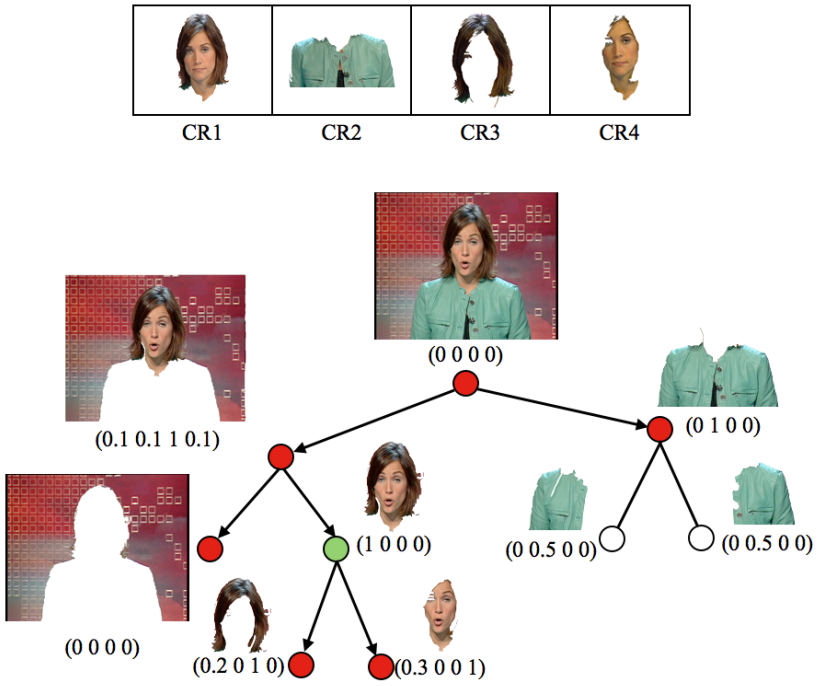


Figure 9.6: Parts detection used on region-based possibilistic features.

classes.

The *detection* classifier can be based on a different set of features from the ones used by the *inclusion* classifier. For example, while the contour of a region is very informative for recognizing a part, it provides little information about the contours of its sub-parts. In our work, the HBoR features used by the *inclusion* classifier are substituted in the *detection* classifier for the possibilistic vectors introduced in Section 6.1.1; that is, without the max pooling operation through the PT. This way, it is easier to discern between the actual nodes that represent a part from its ancestors. This situation is exemplified by the HBoR features depicted in Figure 9.5. They describe with the same vector the region representing the *head* as well as its parent, so they cannot be used to decide which of the two nodes actually corresponds to the *head* part. On the other hand, Figure 9.6 shows the same PT with region-based possibilistic features. In this case the feature vectors clearly allow the discrimination between the region that represents the *head* and the rest of nodes in the exploration path.

9.3 Late Fusion of Parts

Once the parts of the object have been determined and a classifier for each of them trained, the next step is learning which combinations of these parts define an object. The proposed technique is based on training another classifier that will learn the configuration of parts that

completely represent an object. This additional classifier plays the same role as the fusion classifier in the *late fusion* scheme described in Section 9.2.2. Now, instead of combining the probability scores obtained for each feature, it fuses the probability scores obtained for each part in the combination. This solution is complemented by requiring all composing parts to define a connected object.

This section describes the two basic stages of a supervised learning problem, a first one focused on the training stage and a second one describing the detection of the complete object.

9.3.1 Training

Every instance of the object contained in the training set is defined by one or multiple parts. During the parts definition stage presented in Section 9.2.1, each of these parts is assigned a unique label that identifies its type. With this information, it is possible to represent a combination of detected parts in a feature vector of fixed size, a requirement to train an SVM classifier.

The size of the feature vector is initially unknown, as different instances of the same object might present different amounts of parts due to variable topologies, splits/merges in the train Partition Trees or different views. In order to adapt the multiple configurations to a feature vector of finite length, the algorithm firstly counts the maximum amount of occurrences of each type of part in the training set. By obtaining this value, the parts of the model can be learned by assigning one position in the feature vector to each type of object part and occurrence index. For example, the object *face* might be decomposed in three types of visual parts: *skin*, *mouth* and *eye*. The three parts in the example have a semantic interpretation which is not necessary in general, but it has been adopted to facilitate the clarity of the text. An ideal segmentation of a face would create an image part of *skin* and *mouth*, but two parts for the *eye* class. Given the double occurrence of the *eye* class, the model will be composed of four parts, even if only three different types of parts were detected in the object.

Once the size is determined, each component in the feature vector must be assigned to a type of part and occurrence index. The order in which the types of parts are assigned within the feature vector is irrelevant, but a certain sorting criterion must be applied for coherence between training and testing. In our implementation, the blocks of part types are sorted according to the amount of samples in the clusters that defined the parts. So, in the *face* example, if all the training instances contained the four parts, the first two components of the vector would be assigned to the two occurrences of the *eye* class, as its defining cluster would contain twice the amount of elements of the *skin* and *mouth* clusters.

Within a block describing a type of part with multiple occurrences, the probability scores are added in decreasing order. Following the *face* example, this means that if the training set contains a *face* with two *eyes* which are recognized by the *eye* detector with a confidence score of 0.9 and 0.8 respectively, the two initial components of the training vector for the full

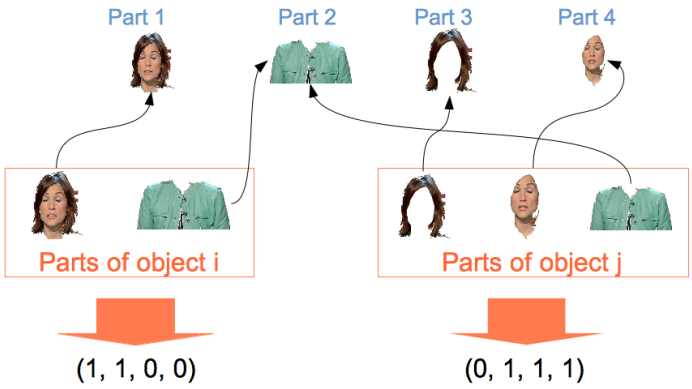


Figure 9.7: Positive training samples represent valid combinations of parts.

object will contain these two values, with this same decreasing sorting. The feature vector will be completed with the detection scores for the considered *skin* and *mouth* parts.

The example shown in Figure 9.7 presents a case where two instances are used to define the *anchorwoman* object with two different combinations of four different types of parts. Object instance i is composed of two image parts, one labelled as *Part 1* and another one labelled as *Part 2*. Object instance j is represented by three image parts, labelled as *Part 2*, *Part 3* and *Part 4*. Each of the two instances is coded with a feature vector whose components correspond to the detection score of the parts. Given the toy nature of the example in this case the detection score for each part is whether maximum (1) or minimum (0).

The training of the binary classifier that recognizes the full object also requires negative samples to model the invalid combinations of parts. These samples are synthetically generated by introducing a distortion on the feature vectors that represent the positive observations. They are built by swapping the non-zero values of these vectors. If no other combination in the positive training set presents the same configuration of zeros, then the distorted sample is added to the test set with a negative label. Figure 9.8 represents an example of this generation by using the two positive samples depicted in Figure 9.7. The generated negative samples correspond to combinations of parts that do not represent the full object. The underlined zero indicates the value that has been toggled at every stage.

9.3.2 Test

The detection of the object is a step posterior to the detection of its parts. That is, the individual extraction of parts is performed in a first stage. Then, combinations with the detected parts are assessed to define a complete object.

The detection of the object is based on the pattern described by its parts-based model. The composition algorithm sequentially considers every part in the model and builds combinations of detected object parts according to this sequence. The algorithm also considers the

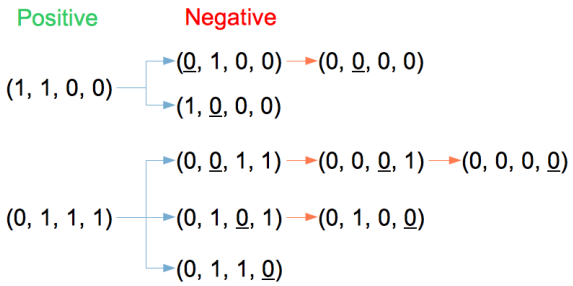


Figure 9.8: Negative training samples generated from two positive ones.

possibility of not matching any detected part to one of the model parts.

Not all combinations of detected parts are assessed, because the combined parts must be adjacent and non-overlapping. These two requirements significantly reduce the amount of possibilities which are to be assessed, at the expense of adding the information about the adjacency of regions to the inclusion relationships represented in the PT. The adjacency information is coded in a Region Adjacency Graph (RAG) of the PT leaves. Maintaining these data for the initial partition is enough as the adjacency information of any non-leave node in the PT can be easily extracted from the RAG of its underlying leaves.

The assignment of detected parts to components of the model allows building a test feature vector that contains the detection scores of every matched part, as done during the training stage. The feature vector is fed into the binary classifier, which produces a confidence score for the combination of parts to represent the complete object.

Analogously to the individual detection of parts, a threshold value is required as a parameter to decide which combinations are to be considered valid object detections.

9.4 Contextual Filtering

The previous Section 9.3 has presented how the analysis of the object parts can be used to learn detectors capable of combining these parts. The analysis so far is totally based on the visual features extracted on the PT nodes and the adjacency of the regions, but still lacks considering the detections in the context of an image framework that must present certain semantic consistency as a whole. In order to avoid over-detection and non acceptable results from a semantic point of view, some basic constraints have been introduced. These will reduce the amount of detections according to certain predefined or learned rules.

9.4.1 Parts overlapping

During the composition of object parts presented in Section 9.3.2, the detection of the object is based on valid combinations of adjacent parts. However, this approach does not take into

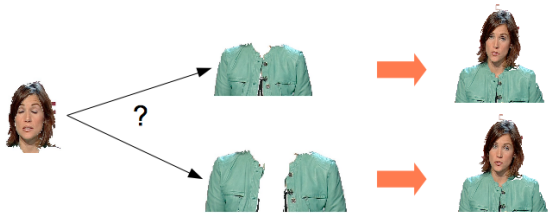


Figure 9.9: One image part cannot belong to multiple objects of the same class.

account that, in general, one object part can only belong to one object. It is possible that when a combination of parts completely defines an object, removing one or a few of its parts will still generate a confidence score over the detection threshold. If not handled appropriately, these situations may generate multiple detections of objects that would share some of their parts. In order to avoid this problem, when a part is shared among different objects, only the combination with the highest detection score is considered. The rest of cases are discarded. Notice that this rule applies also in the case that the image part is assigned to different classes of objects. However, this constrain should be adapted to ever application domain, because often images represent objects or concepts that include other objects and concepts.

9.4.2 Maximum Occurrence

A final filtering stage is performed in terms of occurrence. Given the local nature of the annotation used for training, it is also possible to learn the maximum amount of occurrences of an object class in an image. The implemented detectors include a final filtering layer that discards all those occurrences over the limit. The decision about what instances are to be discarded is based on the detection score, keeping the detections with the highest values.

9.5 Evaluation of the Model-based Object Retrieval

The model-based techniques for the detection and segmentation of local objects have been tested on the same *anchorwoman* dataset considered in the previous section 8.4. In this Chapter, though, instead of running a different search with every training instance and averaging the results, the training set has been used to build a part-based model of the object. The only considered features are the HBoR, as they have provided the best results in all previous experiments, reported in Chapters 7 and 8.

There are three parameters that affect the results. The first one is the size of the codebook, a variation that has already been studied in Sections 7.2.2 and 8.4. The diversity in codebook size is presented by plotting a different graph for each codebook size. Figures 9.10, 9.11, 9.12 and 9.13 contain the MAP, detection rate, Jaccard Index and search time for the same four codebook sizes reported in Section 8.4 (2, 4, 8 and 12 coderegions).

	2 codewords	4 codewords	8 codewords	12 codewords
CBminNum 2	58.31 %	10.45%	44.19 %	26.25%
CBMinNum 3	64.43%	27.46%	40.16 %	20.30%
CBMinNum 4	27.70%	26.68%	10.08 %	-6.22%

Table 9.1: Gain in MAP of a parts-based model compared to the best configuration for matching

The parameters represented in the graphs control the Quality Threshold algorithm responsible of the definition of the object parts. These two values corresponds to the radius of a cluster and the minimum amount of regions necessary in a cluster to define an part. The radius of the cluster is plotted in the X-axis of the resulting graphs, while the minimum amount of necessary elements (CBminNum) is represented by drawing different lines.

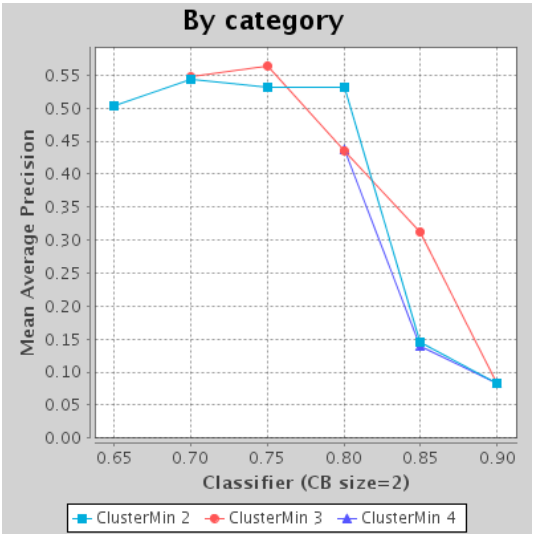
Remark 1. *Some combinations of Quality Threshold parameters cannot be satisfied with the considered training data*

The amount of points plotted in the graphs for each configuration differs depending on the *CBminNum* parameter. The missing points correspond to combinations of unsupervised clustering parameters that could not be satisfied with the provided training data. For example, in Figure 9.10 (a), the clustering algorithm cannot create any cluster with 3 parts with a radius of 0.65, but can do it if the radius is 0.7 or higher. If 4 parts are required to define the cluster, then the radius must be increased to 0.8. These problems are normal because the training dataset contains only 5-6 instances of the object, so it is reasonable that the requirement of finding 3-4 regions in the same cluster may not be satisfied for the smallest radius.

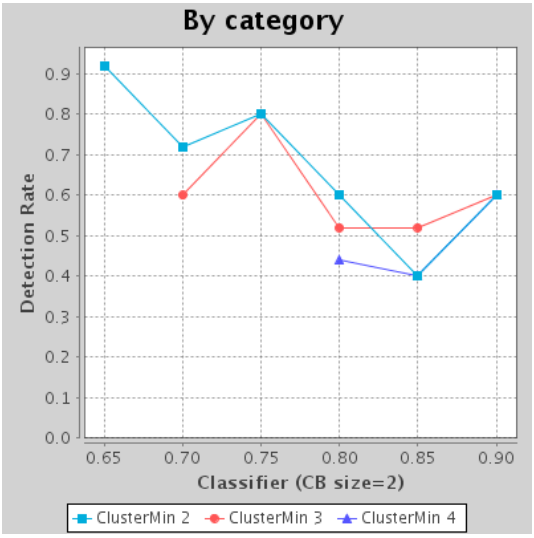
Remark 2. *Best performance in retrieval with the appropriate parameters for unsupervised clustering*

The values with the part-based model have been compared with the previously achieved in Section 8.4. Those were generated by averaging the results obtained by using each instance in the training set as a separate query. Table 9.1 shows the gains or losses obtained by comparing the best configuration with the part-based model against the best configuration for object matching, which corresponds to the top-down and bottom-up exploration using HBoR features and a maximum of 5 parts per query.

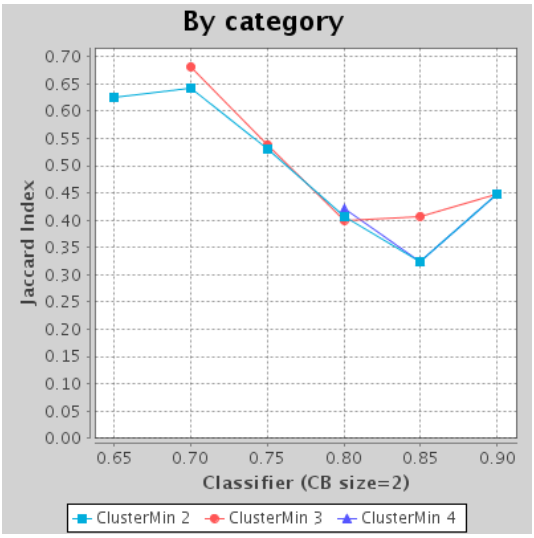
The values in the table indicate that some configurations for the Quality Threshold clustering significantly succeed in improving the results for all considered codebook sizes, especially by requiring only 2 or 3 regions to form a cluster. These configurations correspond to the radius that provides the best results, which corresponds to the 0.7 or 0.75 values. These maxima suggest that this range of parameters are appropriate to successfully identify the types of object parts with the Quality Threshold algorithm.



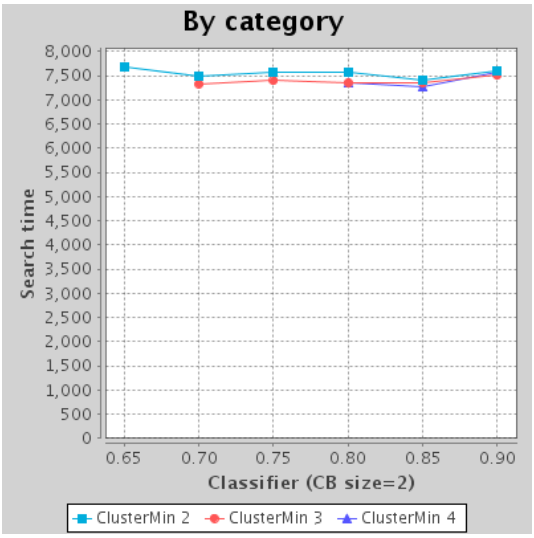
(a) Image Retrieval



(b) Object Detection

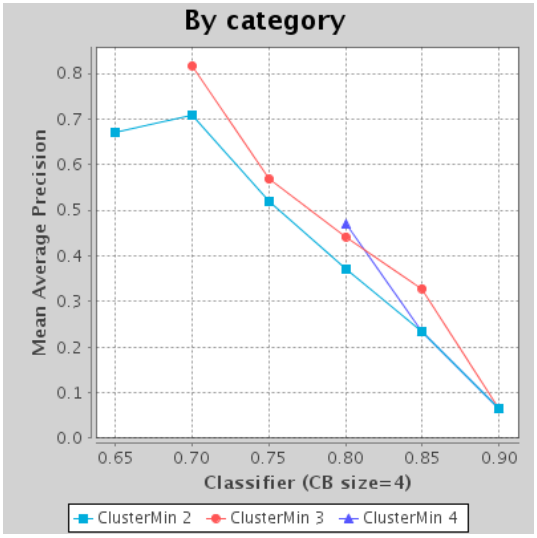


(c) Object Segmentation

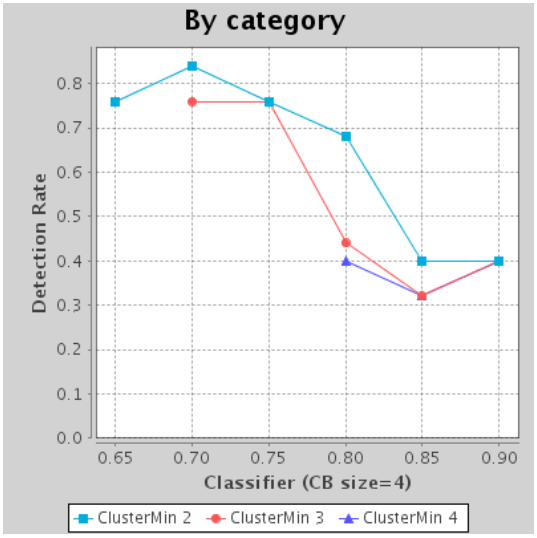


(d) Search time

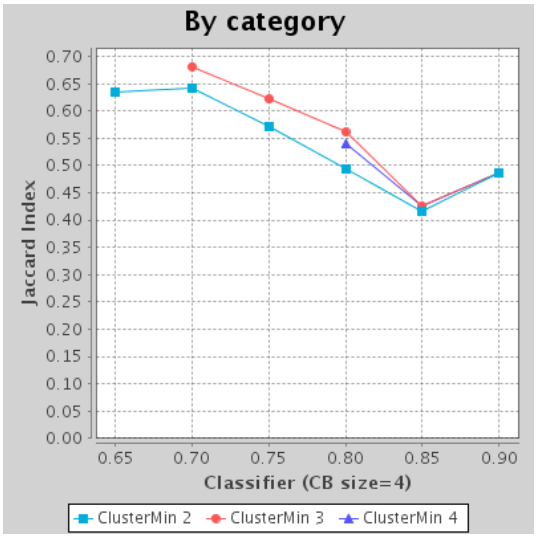
Figure 9.10: Performance dependent on clustering parameters for a codebook of size 2.



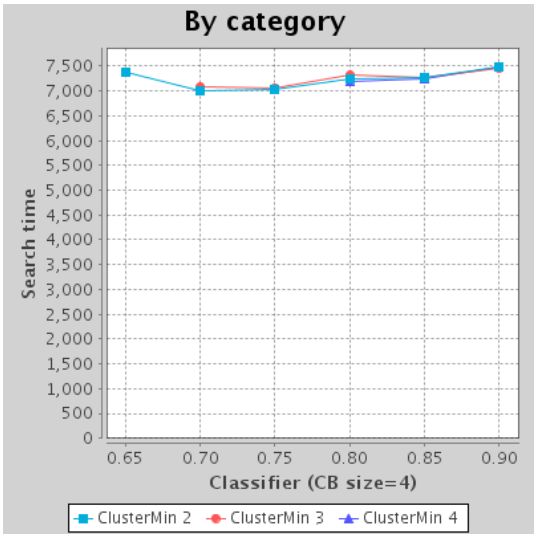
(a) Image Retrieval



(b) Object Detection

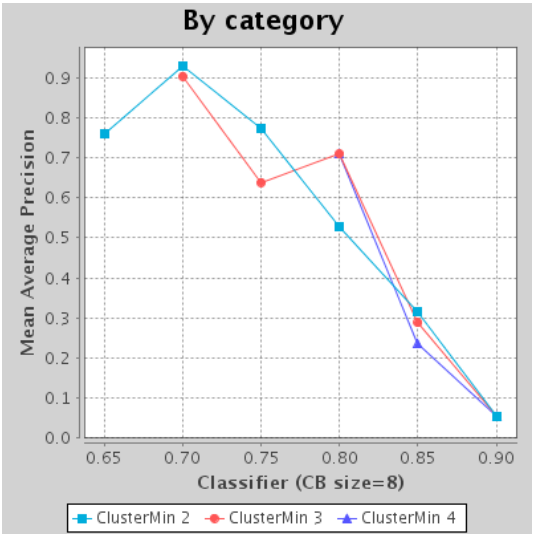


(c) Object Segmentation

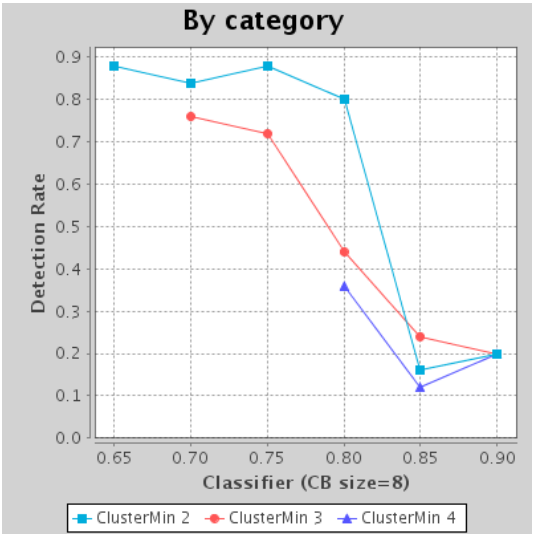


(d) Search time

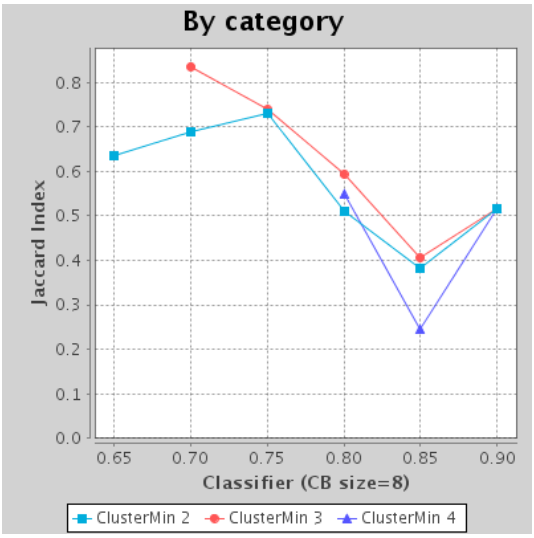
Figure 9.11: Performance dependent on clustering parameters for a codebook of size 4.



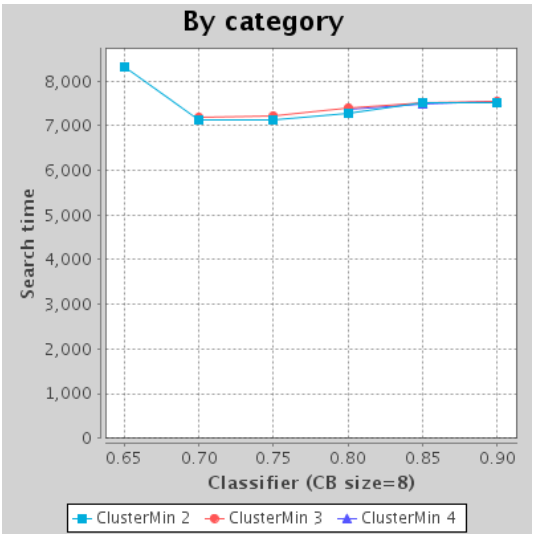
(a) Image Retrieval



(b) Object Detection

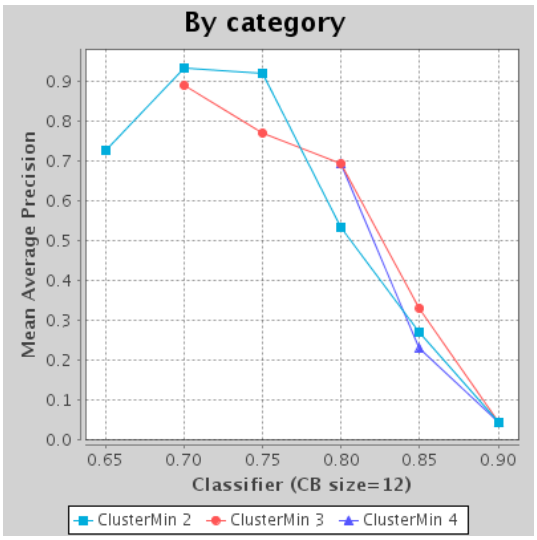


(c) Object Segmentation

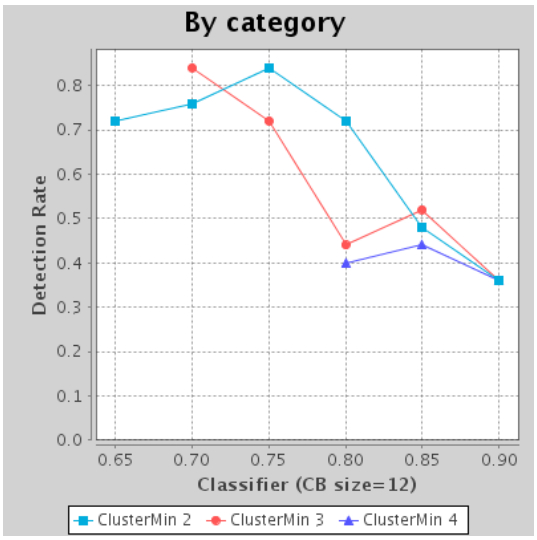


(d) Search time

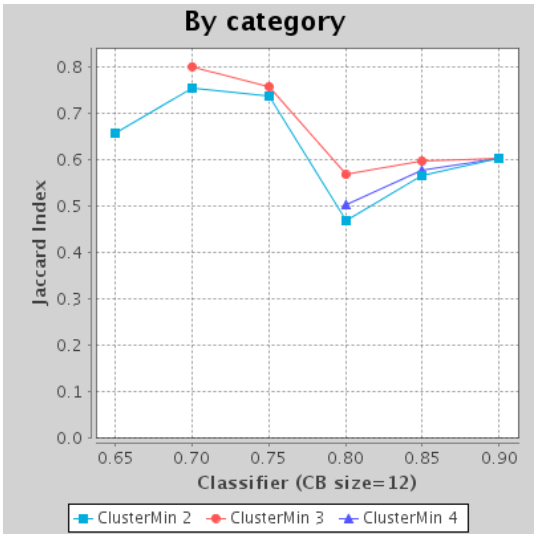
Figure 9.12: Performance dependent on clustering parameters for a codebook of size 8.



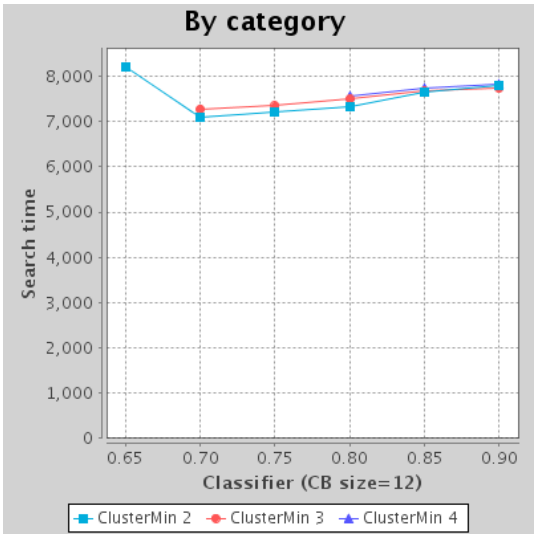
(a) Image Retrieval



(b) Object Detection



(c) Object Segmentation



(d) Search time

Figure 9.13: Performance dependent on clustering parameters for a codebook of size 12.

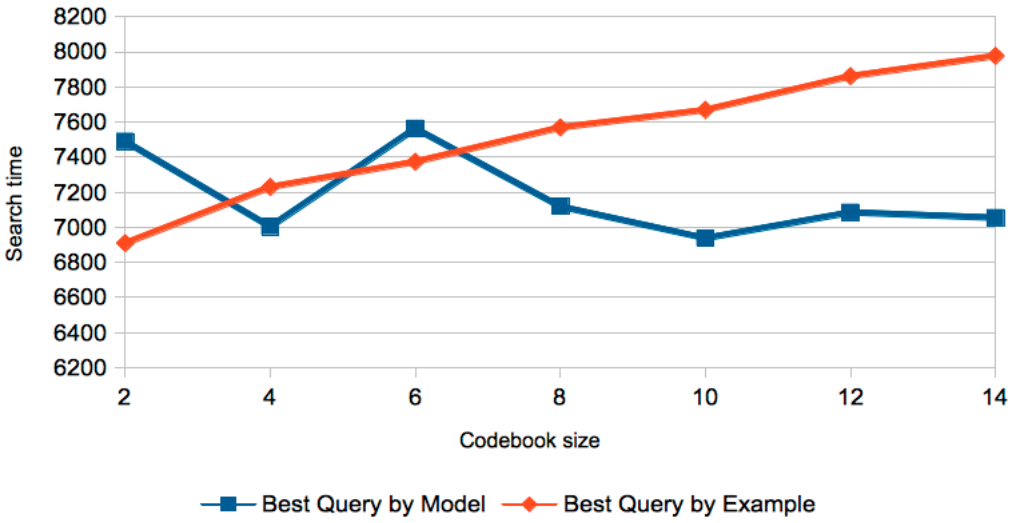


Figure 9.14: Search time vs codebook size for matching or model-based retrieval.

However, some other Quality Threshold configurations decrease the performance, especially for high radius values. This loss is because the clusters that define the object parts are too large and include regions that are too dissimilar among themselves, as much as the dissimilarity between the regions considered as *non-parts*. As a result, the binary classifiers responsible to discern between *part* and *non-part* are trained with two sets of samples with an equal dispersion. In this scenario, the classifier fails into discerning between the two classes.

Remark 3. *Search time does not increase with the considered codebook sizes*

Figure 9.14 compares the search time between the best configuration for matching (5 parts, BQT full and HBoR) and the best configuration for part-based model (CBminNum2 and radius of 0.7). The values in the X-axis correspond to the size of the codebooks. Results suggest that the search time does not necessarily grow with codebook size for the model-based search, while it increases linearly when considering the matching solution. The result for the model-based solution may seem surprising because, the larger the codebook, the longer the feature vectors, so a higher computation effort should be expected. Nevertheless, as larger codebooks also offer more precise feature vectors, which will result on a better performance of both the *inclusion* and *detection* classifiers: the *inclusion* classifier can be more selective and discard more sub-trees during the analysis, while the *detection* classifier will reduce the amount of false positives during the early detection of parts, which will reduce the amount of combinations to consider in the later fusion. These gains compensate the increase of computation effort introduced by using longer feature vectors.

The stable behaviour in the search time of the model-based solution presents a penalty during the training period, as the creation of the models also consumes computational re-

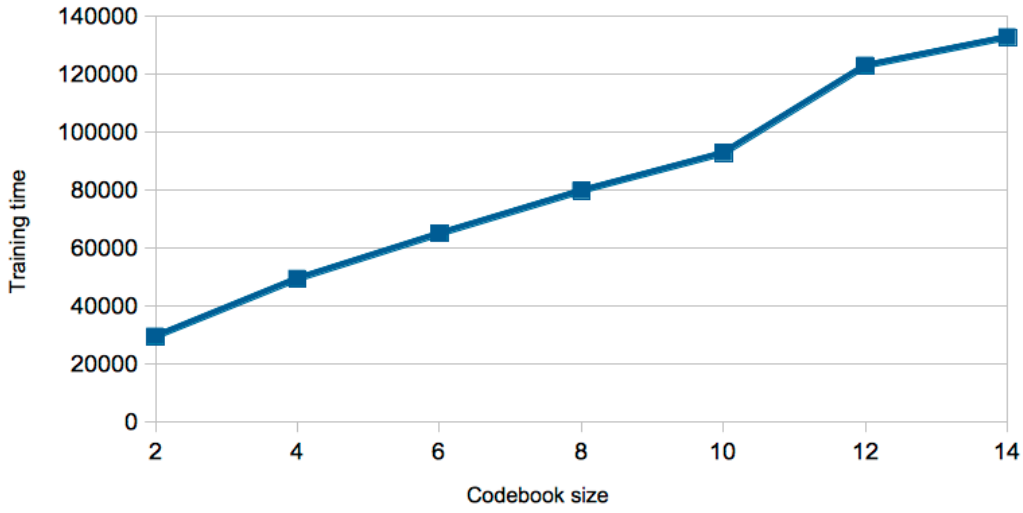


Figure 9.15: Search time vs codebook size for matching or model-based retrieval.

sources. This process, though, is typically performed before query time, so it is not critical for this retrieval application. The impact of the codebook size with the training time is plotted in Figure 9.15. In this case the required time increases linearly with the codebook size, as expected.

9.6 Summary

This chapter has described a model of objects suitable for the analysis of Partition Trees (PTs). This modelling is abstracted from a collection of exemplars represented with Object Trees (OTs). The instances within the same object class can present different degrees of diversity of different nature: appearance, view or segmentation. The proposed solution addresses these problems with a part-based model.

The parts that define the object are learned by processing a training set of OTs with an unsupervised clustering algorithm, the Quality Threshold (QT). This algorithm automatically determines the amount of parts for the model and naturally handles the diversity within the object class. The samples assigned to each cluster are used to train two types of binary classifiers, a first one to determine if a sub-PT includes a part, and a second one to assess if a PT node represents a part itself. This duality of classifiers allows an efficient top-down exploration of PT to detect those nodes which are candidates to contain parts of the object.

Those candidate parts which are adjacent are combined at a later stage, being each combination represented by a feature vector that is used to train another classifier. This classifier performs a late fusion of the parts and generates a score to assess the probability of the combination to represent the complete object. All detected objects are further filtered to

guarantee a coherent result within every image.

The same *Anchorwoman* dataset used in Chapter 8 was adopted to compare the differences between the *Query by Example* and *Query by Concept (model-based)* strategies. Results indicate that the model-based solution can successfully deal with the diversity of PTs present in the *Anchorwoman* dataset as long as the unsupervised clustering can successfully determine the parts of the object. As expected, the performance of the model-based solution is better than averaging the individual results from the *Query by Example* solution, both in retrieval accuracy and search time.

Part IV

Conclusions

Contributions

This thesis had as a starting point the previous work in the multi-scale and region-based representation of images with Binary Partition Trees (BPTs) by Salembier and Garrido [76]. In their work they already highlighted the potentials of BPTs in the field of image analysis, potentials that have been exploited and further studied in this thesis according to the state of the art on computer vision. It has not been the goal of this thesis to improve the quality of the hierarchical image partitions, but to exploit their opportunities and try to overcome some of their limitations. The reader is referred to the work by Pont-Tuset and Marques on the different strategies for BPT creation and quality assessment [96].

This thesis has re-afirmed that BPTs are valid tools for the semantic analysis of images at the local scale, as they can successfully estimate the location and contours of the semantic objects, or their parts. However, the semantic gap between the regions defined in the BPT and the object represented in the images poses several challenges. The main one is that composite objects may appear split in different subtrees, instead of being represented by a single node. This observation has driven us to a part-based analysis of both the images, with the BPT, and the objects, with the proposed *Object Tree*.

A first application of the BPT has been precisely the interactive segmentation to assist users into manually defining *Object Trees*. User generated labels and actions on the hierarchical structure of the BPT allow the development of intuitive mechanisms for interactive segmentation. This thesis has proposed and implemented strategies to define a query object in the instance search problem, or generate several local instances to train an object detector.

The adoption of a region-based approach inevitably poses a challenge when describing and analysing their visual properties. The visual properties of regions can be quantized from different points of view, typically with color, shape and texture. This diversity arises several questions about how to combine them. The adopted approach is based on a previous work by Scheirer [78], where the similarity distances of each considered visual descriptor are fitted to a Weibull distribution. This solution, based on the Extreme Value Theorem (EVT), provides a solid ground whose benefits have been confirmed by the reported experiments.

The quantized nature of point-based features such as SIFT or SURF has inspired the design of the *Hierarchical Bag of Regions (HBoR)*, a new type of region-based features referred to codebooks. The first particularity of the HBoRs is that, instead of a hard mapping of the regions into a single codeword, a possibilistic approach has been adopted, which keeps the similarity between the represented region and every codeword. Moreover, the hierarchical structure of the BPT has been exploited to perform a bottom-up max pooling of the quantized features, which allows the description of every BPT node according to its sub-parts. These two techniques allow the definition of a *Parts Space*, a new feature space where every BPT node is located according to the similarity of its sub-parts to the codewords. This strategy

allows that, if the codewords are representative enough, the BPTs nodes that contain all the parts of an object will occupy a distinctive area in the feature space. The experimental results have shown that these HBoR features outperform the simpler region-based ones when tested with the challenging ETHZ dataset.

The hierarchical structure of regions has also been exploited in the correspondence problem posed by matching a *Query Tree (QT)* with a sub-tree in a BPT. This problem arises when trying to find an instance of an object in a target image. If assuming that the hierarchy of regions defined in the QT will also be kept in the target BPT, the problem is solved with a fast top-down approach that iteratively matches nodes of the QT on the target BPT and searches among their subtrees the rest of QT nodes below. However, this assumption is not satisfied in those cases where the object is split among different subtrees. In these situations, a bottom-up strategy has been proposed that is much more costly as it considers regions that are not defined in the target BPT. This solution can only be applied with features that can be very rapidly computed, such as the HBoR. Both the top-down and bottom-up algorithms have been designed with special care to avoid an explosive amount of combinations.

Finally, this thesis has also addressed the problem of how to create a part-based model of a hierarchical object. The main problem in this case is the great diversity between the different visual instances of the same object class. Firstly, different instances may present completely different appearances despite belonging to the same class. Secondly, a single instance can generate different representations depending on the point of view of the image acquisition. Thirdly, the segmentation algorithm can build a variety of BPTs with different topologies and initial partitions. These three problems have been addressed by assuming that all these variabilities must be represented in the training dataset, which must be rich enough to cope with all of them. Then, following the part-based approach of the thesis, the model is built by firstly identifying that composing parts of the model and later training a detector for each part. The detection of individual parts is performed with an efficient top-down exploration of the target BPT, that is based in another inclusion classifier capable of discarding subtrees by just analysing their root. Finally, an additional classifier is trained to perform a late fusion of the parts, inspired by the late fusion approach broadly applied for feature fusion. The experimental results have shown that it is important to correctly determine the composing parts of the model, whose amount is to be dependent on every object class. The result is a very versatile model that could also be used for multiview detections at both local and global scales, as the parts associated to each different views would be naturally handled by the late fusion of parts.

Future work

The presented work has tried to cover a broad range of aspects related to the exploitation of BPTs for object analysis. One of the main conclusions of this thesis is that the potential of this structure can be further exploited, so this work must be considered still open and under progress.

A first opportunity for future work is in the field of Human-Computer Interaction. This thesis has implemented and exploited different solutions to exploit the structure, but has not focused on their comparison and evaluation. The next step should focus on a research with users, for example, according to the guidelines proposed in [58]. Moreover, interaction possibilities should not limit to mouse devices but be extended to newer interfaces, such as touch screens, cameras capturing gestures or brain-computer interfaces.

The codebooks used in this thesis have adopted the basic approach of a K-means supervised clustering for a predefined size. This solution is somehow contradictory with the unsupervised clustering applied for the definition of object parts because the problem is very similar. Creating codebooks whose size would adapt to the dataset would probably save some codewords without any loss in performance. Additionally, the current solution considers equally all regions for the clustering, no matter their frequency of appearance in the dataset. Another possible improvement would be the removal of highly repetitive and uninformative codewords, the so called *stop words*, as already suggested in [80]. The same way, the remaining codewords could be weighted according to their entropy, as the TF-IDF descriptors for text which were already adapted for point-based features in [42].

In terms of image representation, the adopted solution has always considered a single PT for every image. This type of structure is very restrictive because during its creation is limited to only one the number of merges for each region. A bad decision at this stage is the responsible of the object splits. The whole system could be much more reliable by relaxing this condition and considering multiple hierarchies, as suggested in [20]. The merging decision may not only rely on the best neighbour but consider all those adjacent regions with a similarity under a certain threshold. This would break the tree structure but would still keep a hierarchy which is the only requirement for the HBoR features. Moreover, this solution would discard many of the regions at the highest part of the PTs, which normally merge different semantic entities.

At the feature level, it is exciting to imagine the potential of combining region- and point-based descriptors. This thesis has ported some of the most popular techniques from the point-based features to the region-based world, but it is still a fully region-based approach. It seems reasonable to expect that the accurate and hierarchical segmentations provided by the PTs and the highly informative point-based features could co-operate to obtain more precise analysis.

The use of the SVM discriminative classifiers could also be explored by considering the perceptual distances proposed by the MPEG-7 standard. The presented experiments were always based on the Radial Basis Function (RBF) as a kernel because the feature vectors were already quantized with the aid of the codebooks. However, this stage may be skipped by directly representing regions with their visual descriptors and using the specific-distances as kernels for SVM. It would also be interesting to evaluate the performance of this non-quantized configuration.

At the parts level, it is the will of the author to study and analyse the existing solutions of probabilistic graphical models to better estimate the right configurations of parts. The consideration of richer relationships (both spatial and appearance-based) should allow more precise models and, probably, even more efficient methods to assess the combinations between parts.

The presented object retrieval experiments have succeeded in indicating what design strategies worked best and, this way, directing the research at every stage. Part II has used the challenging ETHZ dataset to select the HBoR features with separate codebooks and Weibull-based fusion as the best configuration in terms of features, Part III has focused on a simple case of single-view composite object to determine the most promising solution among all the presented ones. However, the best configuration should still be compared with other techniques to evaluate their impact in the state of the art. This thesis does not provide comparative results on public evaluation benchmarks, but it is the author's will to fulfil this stage in the near future and publish the results for the scientific community to evaluate.

Although this thesis has limited its scope to visual queries, the multi-instance visual search problem can also be exploited in the context of text-based queries. Once a model of a certain concept is built from a set of instances, assigning a textual label to the model allows its reuse on future text-based queries. In these frameworks, it has been proposed [111] the pre-processing the target database with the models of a closed set of concepts to generate automatic labels weighted with the obtained probability scores. This architecture allows the creation of new concept detectors on top of the closed set of models, using their probability scores as components of a new feature vector [45]. This approach has also motivated the public release of detectors entitled to be used as a basic component of more complex detectors [86] [43].

Finally, the most exciting research line from the author's perspective is exploiting the annotations at the global scale to extract local objects [15]. The solution to these problems should detect the similar image parts in the positively annotated images which do not appear in the negatively annotated images. The advantage of this approach is that the annotation requires much less user interaction and, in many cases, these annotations can be easily obtained from online resources . Previous works [107] [69] [37] already exists using several images representations, so it would be interesting evaluation the proposed hierarchical techniques on

similar benchmarks.

Final words

This thesis basically confirms the opportunities of the BPT in the field of visual retrieval. The work has been performed during the early 2000's, a moment in history extremely influenced by the popularity of multimedia capturing devices and interconnectivity between people thanks to the advances in the information technologies. At the beginning of this work digital cameras and Internet access were only available by a minority, while in 2012 the popularity of mobile devices equipped with high-definition cameras and Internet connectivity is almost universal in the developed world, especially among youngsters.

This growing trend defines a connected society of cameras and other types of sensors that offers an enormous amount of opportunities, but also poses thousands of new questions. I expect in the forthcoming years a great debate on privacy issues because soon the maturity of the automatic analysis technologies will technically allow anybody to know almost everything of what is happening and what has happened in the world. Some may argue that this situation is undesirable and that the research of these topics should be cancelled. History has shown that this is inevitable and that advances will happen. However, the usage of these technologies will indeed require an extensive research. Not from a technical perspective, but from a humanistic one. I fully believe that the inter-connectivity between people and free circulation of information are the great antidotes to eradicate many of the current human dramas, such as wars, hate or poverty. In my opinion, many of them are based on the deliberate hiding and distortion of information, commonly known as *lies*. On the other hand, I also believe that a strong social network is one of the key ingredients for *happinness* and that technology can assist in better human connectivity. Automatic analysis of multimedia data is one more piece for a better communication between persons, but its success is inevitably linked on the usage humans will make of it.

I have both enjoyed and hated this thesis for the personal price it supposes a research at this level. At all time, though, it has been my will to contribute with this work to a little bit better world with the skills I consider myself more talented. This part of my contribution to society reaches to an end here, I wish I succeeded in making it enriching and valuable for the reader. Thank you.

Part V

Annexes

Chapter 10

System Architecture

10.1 Introduction

The growing amount of audiovisual digital data is following the path of a first type of information whose generation and distribution has dramatically increased in the last decades: text. Thanks to the universalization of Internet access, the amount of text information available has exponentially grown and has become the prime knowledge repository nowadays. All these data are in most cases retrieved thanks to text-based search engines which are capable of fast and precise indexing and retrieval. These search engines were initially accessible through web browsers but have increased their range of interfaces thanks to flexible APIs that allow their integration with any other application. From this persepective, search engines can be understood as one of the first types of web services, an online tool that processes distributed data and provides information to all clients that request it. The externalization of tasks from the local machines is a growing trend in software engineering and is normally referred as cloud computing. Software, data and hardware are moved away from the user computer, which becomes a terminal to a distributed online system.

Visual data are following the path openened by text documents and are also being indexed in the cloud [47]. Online image and video repositories offer flexible APIs to third-part applications that access and process them as if they were stored locally. Among these applications, content-based indexing systems are trying to become as successful and useful as their text-based cousins, bringing their workflow and architectures to the cloud. This paper presents UPSeek, a cloud computing solution that contemplates the whole life cycle of a Content-Based Image Retrieval (CBIR) system. From the content ingest to the results exploration, including the training of models or the queries by example. The presented system uses regions as the basic analysis unit, which are defined on a multi-resolution partition tree. The system has been developed according to the requirements of two industrial partners from the broadcast television. Their growing archives, already on explotation, require software solutions that can

be easily integrated with their established workflow. In this scenario, the access to the image processing tools as web services is a flexible and modular solution that allows the integration of new indexing tools without any traumatic change on their databases.

10.2 Interfaces

The UPSeek system is a type of cloud computing application that follows the *Software as a Service (SaaS)* paradigm. In this type of software deployment, the provider allows the customer to run the software without owning it, neither the binaries nor the source code. The service is the object of the contract and, when available online, it is referred as a web service. In this case, the service provides tools for managing an image database with content-based indexing and retrieval functionalities.

UPSeek consist on a set of five different pieces of software, two of them running on a server where the content is stored and the other three running in the client machines. The communication between them is established through state of the art protocols (SOAP and REST), broadly used on the web. All elements in the architecture have been implemented in Java, but the underlying image processing algorithms were developed in C/C++. The whole system is based on the MPEG-7 standard [11] to guarantee its inter-operatibility with third-part tools.

10.2.1 Applications Server

The core of the system is an applications server that is responsible for the execution of the image processing algorithms. This server handles the incoming requests and input parameters, run the analysis engines and, when their execution is over, returns the results. At the moment, this application server includes a segmentation tool, a features extractor, a query by example search engine, a text detector and a trainer-detector pair based on region features. In most cases, these image processing algorithms have not been developed considering a web architecture, so they lack connectivity outside the server where they are to be run. For this reason, the applications server implemented in Java can be understood as a wrapper that provides these connectivity features

The developed software has followed a modular architecture for its flexibility. Two projects have been built, one for the preanalysis process and another for the analysis one. The preanalysis software generates and MPEG-7/XML description of the image. The analysis software takes these input data and accesses the encyclopedia of models to find any instances. All XML documents are manipulated through the ANSI-C XML C Parser and toolkit of Gnome [libxml2]. A more detailed description is included in Annex B.

The proposed architecture is coherent with the implemented according to the proposed described for the detection algorithm. It is important to highlight that the most costly

operations, concentrated in the preanalysis, are only performed when data are added to the database. This differentation allows MMDBMS to work in query and update mode [Avrithis03]. In query mode, the system serves its users by using the semantic indexes and the stored perceptual image descriptions. In update mode, the system generates the perceptual image descriptions and generates indexes for future use in query mode.

Figure 10.1 and Figure 10.2 show the software architecture in implemented for the perceptual and semantic analysis processes.

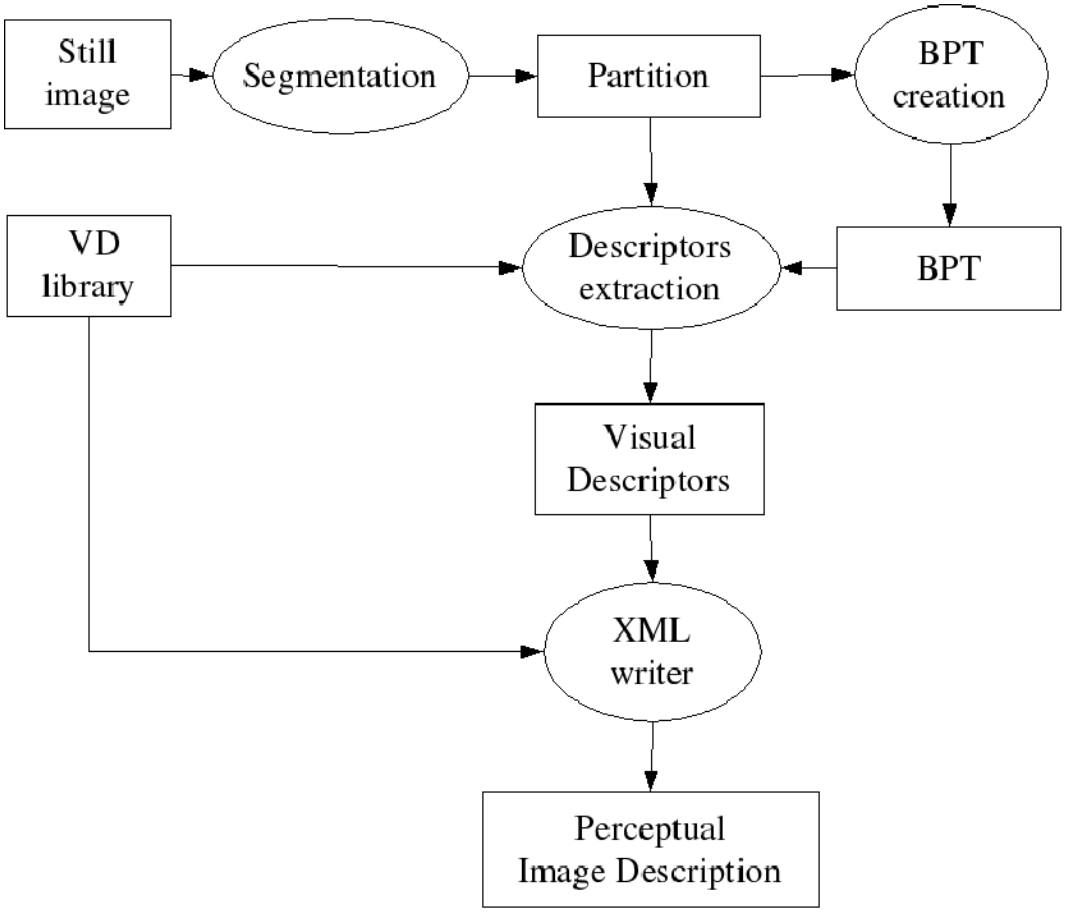


Figure 10.1: Software architecture for the perceptual analysis.

10.2.2 Repository: Fedora Commons

Images are stored in a Fedora Commons repository [50], an open-source web service that manages the storage and online access to the content. The Fedora Commons working unit is an *object* and each object can have a specific type. In the UPSeek case, two types of objects are defined, the *image* type representing a single image and a *video* type. Each object is identified by its PID (Persistent ID), no matter the modality of the data it represents.

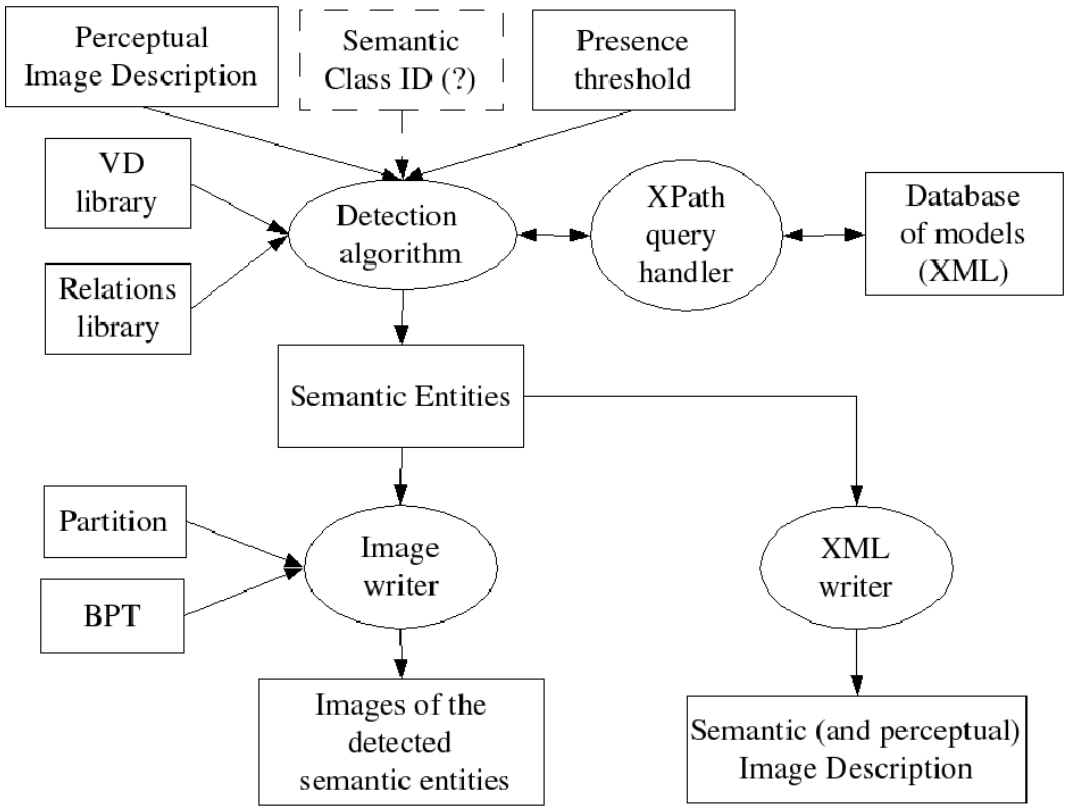


Figure 10.2: Software architecture for the semantic analysis.

Relationships between different obects in the repository is established through a generic Resource Description Framework (RDF), an XML-based language which is broadly used by the semantic community. For example, when UPSeek manages a set of keyframes extracted from a video asset, each of them is ingested in the repository as a single object, but a video object is also created to link all of them as a belonging to a common source.

Every object can have several representations, some of them dynamically produced in the server instead of ingested from the exterior. Each of these representation is called a *datastream*. Image processing algorithms similar to the ones handled by the applications server are used to generate datastreams. In this case, the JMS messaging implemented in Fedora Commons is the responsible to execute the image processing software. This method allows the scheduling of a sequence of operations without blocking the activity of the client that is sending the set of images being ingested. The chosen strategy is to generate perceptual information (eg. thumbnail, segmentation, features...) at ingest time so that, when a more semantic analysis is requested by the user, these data is already available to reduce the response time.

The datastreams defined for the image type objects for this Fedora Commons installation are the following:

- *DC:* A Dublin Core document containing textual metadata about the image.
- *RELS-EXT:* Semantic description of the object in RDF/XML. If the image is part of a larger set, it provides the PID of the associated video object.
- *image:* The original image initially ingested. All analysis have in this datastream their origin.
- *thumbnail:* A light version of the image generated for visualization purposes. When clients want to visualize a large amount of images, it is not advisable to send their full resolution version due to the high amount of data that it is to be transmitted and managed by the client. In order to save bandwidth and memory resources, thumbnails are advisable.
- *partition:* The region-based representation requires an initial segmentation of each image.
- *image-partition* An image showing the regions in the initial partition by painting them with their mean color. Used only for debugging purposes.
- *bpt:* A Binary Partition Tree (BPT) [76], a hierarchical region-based structure built by iteratively merging the two most similar neighbouring regions from the initial partition. Coded in MPEG-7/XML.
- *feature-image:* Global features computed on the complete image. The list of features corresponds to MPEG-7 visual descriptors defined on the whole image.
- *feature-region:* Local features computed for each region described at the BPT. The list of features includes most MPEG-7 visual descriptors extendend with other features.
- *annotation:* Manual and/or automatic annotation of the image describing the represented semantic classes and providing a confidence value for each instance. The semantic representation refers to an external ontology where semantic classes are defined.

Fedora Commons already includes a web-based user interface that can be used for searching and checking the datastreams and metadata associated to each object. Two search engines are available. Firstly, a textual one based on the Dublin Core metadata and, secondly, a semantic one to query the information contained in the RDF descriptions. When an objected is found, the available datastreams are shown and metadata fields can be checked on the web browser.

Finally, Fedora Commons implements a security layer based on username and password. This features provides privacy in the stored data, a requirement in certain domains, such as propietary-protected content or security aplications. If active, all clients must provide the authentication data in every transaction.

10.2.3 Ingester: UPLoad

At the beginning of their life cycle, images are considered to be available at the client side after an acquisition or data transfer. The first step in any operation provided by UPSeek is ingesting the images to the Fedora Commons repository. This step requires a human-computer interaction so a graphical and user friendly interface is introduced, as shown in Figure 10.3. The goal of *UPLoad* is assisting the user into browsing the content in the local file system and selection what images or image directories are to be sent to the server.

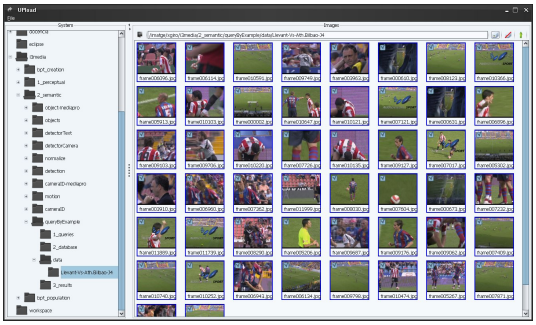


Figure 10.3: UPLoad user interface.

Users can explore their filesystem through a tree-view of the directory structure. When a directory is chosen, thumbnails of the contained images appear in the main panel so that the user can check the contents of the directory and select which of them are to be uploaded. If the selected directory contains a very large amount of images, the generation of automatic thumbnails is disabled to avoid memory overload, but the user can still generate individual thumbnails by clicking on them.

10.2.4 Annotator: GAT

A common feature in most CBIR systems is the possibility of training the system models through the manual annotation of a part of its content. In the present system, most of the analysis is both performed at the image or region level, so its manual annotation tool must also be able to work at both scales. This is the goal of GAT (Graphical Annotation Tool) [32], a client application that can annotate sets of images at the global level as well as sets of regions to generate local-scale annotations. With this interface the user can create or load a semantic ontology coded in MPEG-7/XML or OWL and define instances of semantic classes to images or regions. When working at the global scale, the user selects a directory in the file system or a collection of images on the remote repository and visualize their thumbnails to select those ones that contain the concept being annotated. If a local annotation is necessary, a double click on the thumbnail zooms in the complete image and the user is prompted to select regions by navigating through the BPT. If navigating through the file system, the BPT may not be

locally available nor the segmentation tool that generates it. In this case, the image is sent to applications server so that it generates the BPT and returns it to GAT. Figure 10.4 shows the selection of a region in the BPT to generate an instance of the semantic class "face". This tool is described in detail in Chapter 11.

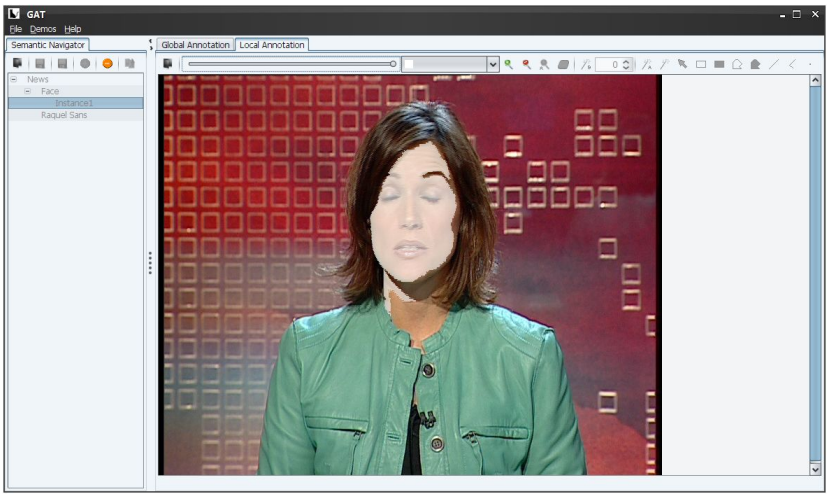


Figure 10.4: GAT user interface.

10.2.5 Searcher: GOS

A classic application of CBIR systems is the retrieval of images that are similar to a query image or that contain a region similar to a query region. The definition of such queries requires a user interface capable of selecting the query image or region. The query may also contain additional parameters such as which visual descriptors are to be considered, the dataset that defines the search space or the similarity measures to be used. The Graphical Object Searcher (GOS) is the client tool proposed in this system that, analogously to GAT, allows the formulation of global or local queries.

The user interface collects the query parameters, sends them to the applications server and, after the search is performed, displays the thumbnails of the results. The query image can be read from the local file system or retrieved from the repository. In the first case, the image is firstly ingested so that the BPT and the visual features are computed on the server. The graphic interface is shown in Figure 10.5, where the query parameters are specified on the left par of the screen and results are explored in the large panel located on its right.

10.2.6 Searchers: Digimatge v.1 and v.2

The visual search system has been combined with a textual search engine provided by CCMA, the Catalan public broadcaster. The starting point for the system integration was Digition,

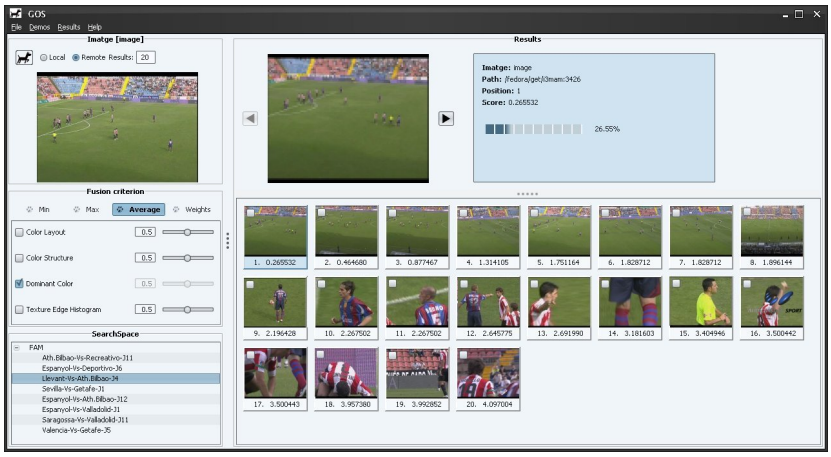


Figure 10.5: GOS user interface.

the Multimedia Asset Manager (MAM) previously developed by the CCMA. This search system was based only on matching query keywords and the textual metadata manually generated by the documentalists. *Digimatge* is a new search system that adds two new services to the ones previously offered by Digition: an external tag suggestion for text-based queries and a ranking of retrieved keyframes based on image similarity.

Multimodal search

Semantic information is normally expressed under the form of text descriptors and it is the most used modality in search engines. Users can directly formulate their queries by entering keywords or full sentences to the system, and the generated query descriptors are compared to the text or semantic descriptors previously associated to the asset. The text-to-text video search tends to provide good results as humans can be very precise when using text to express semantics. Nevertheless, this approach requires associating textual metadata for the multimedia content. A first option is the manual annotation of the content, a very consuming task in terms of human interaction. As an alternative, descriptors can also be automatically parsed from the image filenames or contextual text, or be generated through signal processing techniques like OCR-based solutions [55] or semantic classifiers [42].

A second family of retrieval systems are based only on visual descriptors, as the one presented in this thesis. The automatic nature of the process allows batch processing of large amounts of content to generate perceptual descriptions that do not require any human interaction. This solution, though, presents two drawbacks when compared to text annotation. Firstly, the correlation between perceptual descriptors and semantic concepts is not as close as in the textual case and, secondly, the final user needs to become familiar with a new interface to formulate queries in terms of perceptual descriptors instead of text. In these case, queries are no longer formulated through text but whether by directly providing a quantified value

of the descriptor or by giving the system some sort of visual content from which the query descriptors can be extracted, such as an example or a sketch [27] [100] [49] .

Although the visual-based similarity provides reasonable good results from a semantic point of view, many systems that exploit these technologies also rely on textual descriptors, providing hybrid search solutions that combine both. Fast indexing and retrieval algorithms are more mature in the text mode than in the visual, so many designs use text queries to retrieve a first set of results which are later refined by a second search based on visual descriptors. This third family of retrieval systems are the natural evolution of the two previous, introducing multimodality to the search experience. Figure 10.6 shows an example of a combined text and visual query, in which the semantic class of the object *dog* is expressed through text, but the color of its hair is represented by a choice in a palette. Commercial examples like Google Similar Images, Pixolu or Xcavator have applied this strategy.

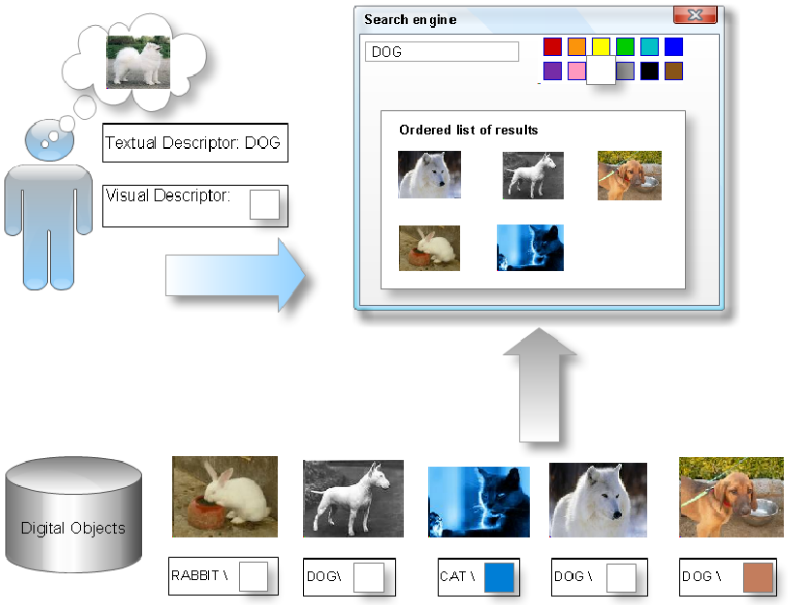


Figure 10.6: Visual and textural query.

Integration with a Tag Suggester

The search process begins with the input of a textual query in a box located at the upper left side of the graphic interface. While typing the query, an autocompletion combobox appears suggesting terms from a thesaurus maintained by the documentalists. This tool speeds up the typing process as well as decreases the probability of spelling errors. The words in the thesaurus are preferably used by documentalists, but during asset metadata may contain many more words are used. For this reason, the user can also freely enter keywords that are not included in the thesaurus.

Once the textual query is introduced, the search is executed and results are displayed in the graphic interface shown in Figure 10.7. The upper part of the screen shows a table with the numeric ID of the retrieved video assets as well as their title. The lower half contains the ten first keyframes of each retrieved video object. A double-click on any row in the results table repaints both upper and lower parts of the interface with more data about the selected video object: the results table is replaced by a all textual metadata associated to the asset, while in the lower part a new tab is created to show all keyframes from the selected asset.

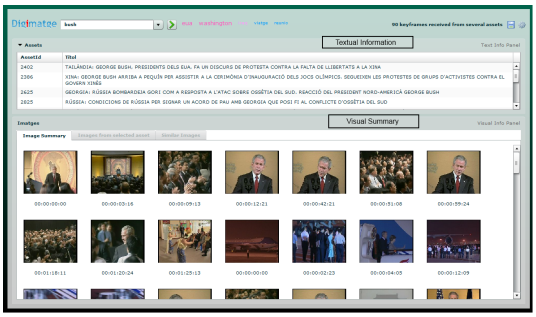


Figure 10.7: Digimatge v.1.

A tag suggestion service [29] is also executed at query time to retrieve similar terms to the entered keyword. This service is based on a statistic analysis of words in the metadata database and shows its results in a word cloud. By doing so, the system is proposing new query terms aimed at assisting the user when its search idea is vague or fuzzy.

In addition, a new visual search service is available by double clicking on any thumbnail. When activated, an image similarity search is run between the selected keyframe and the rest of the keyframes of the retrieved assets. The process can be understood as a reranking of the keyframes extracted from the asset, which in many cases contains several clips from different semantic nature, as it is the case of those assets that represent a TV news program. The results are displayed in a new tab, where any click on a keyframe will show the asset summary on the upper part of the interface.

10.3 Web services

The elements of the system presented in Section 10.2 interact to build a complete CBIR system. The user actions performed at the client side trigger image analysis processes run on the server. In the present section, each of the six web services are analyzed in detail to describe the complete workflow. They are the following: the ingest of images, the access to visual features, the text detector, the query by text and the manual annotation and train of classifiers.

10.3.1 Image ingest

The image ingest web service is provided by the Fedora Commons repository and two image processing tools accessed via JMS messaging. The client application, UPLoad, calls it after the selection of an image or a directory on the client file system. If an image is selected by clicking on its thumbnails, a digital object of the *image* type is created. If, on the other hand, a directory is selected, not only an *image* object is created for each of the images contained, but also a *video* object is built. UPLoad indicates Fedora Commons the relationship between the images and the video object, establishing the semantic relation between them that is coded in the RDF semantic descriptions of both objects.

Due to the region-based nature of the UPSeek system, all images in the repository are segmented and their features extracted at ingest time. This processing involves the execution of two different pieces of software via JMS messaging. The first algorithm creates a thumbnail, an initial partition and a BPT of the image. The second one extracts the visual descriptors at the global (image) and local (region) scale using, in the later case, the BPT and partition previosuly generated. Figure 10.8 shows the datastreams generated after the ingest of the image. All datastreams are added as representations of the digital object initially created by UPLoad.

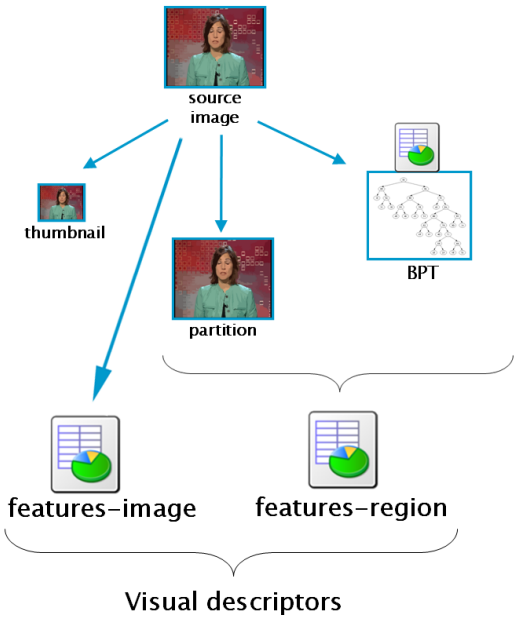


Figure 10.8: Datastreams generated at ingest time.

10.3.2 Visual descriptors

The access to the visual descriptors generated at ingest time defines the first family of web services offered by UPSeek. The computed features are already valuable for some applications as they provide low-level representations of the images valid for visualization interfaces. For example, the MPEG-7 dominant color descriptor is used by an industry partner in the broadcasting industry to implement a fast browsing of their archive based on color. In this case, the partner metadata fields are stored and indexed in the DC document in order to translate their queries to Fedora Commons PID. The response of the visual descriptor web services is an MPEG-7/XML document that has been created by reading the requested descriptor from the *feature-image* datastream.

10.3.3 Query by Example

The query by example web service must process the queries formulated at GOS to retrieve rank list of images from the Fedora Commons repository. This list is built according to the similarity the images in the search space when compared to the query. The web service considers that the query image is already ingested in the repository and, if it was not because it was read from the client file system, GOS can ingest it on the server so that its regions and/or visual features are computed.

The web service allows the user to limit the search space to a subset of video objects. In this case, the query will determine what video objects are to be assesed and, through the RDF relations, the images to be considered are located. Other filtering criteria could be applied based on a previous query on the Dublin Core metadata indexed by the system. This strategy can significantly speed up the retrieval time but requires a previous knowledge of where to look for. A possible method for a search space filtering is a query by text, a practise applied by *Digimatge* [33], a third-party interface that runs this web service on a subset of the video objects.

The web service generates an MPEG-7/XML document describing a collection of PIDs and their associated similarity score. With this information, the GOS client can retrieve the thumbnails for visualization and, if necessary, access to the metadata associated to each image object.

10.3.4 Query by Text

The query by text engine is provided by the Fedora Commons web service. It integrates an SQL database where Dublin Core metadata is indexed for a fast search. Queries can be performed on any Dublin Core field by a client application with an SQL API or by a user through a web interface, shown in Figure 10.9.

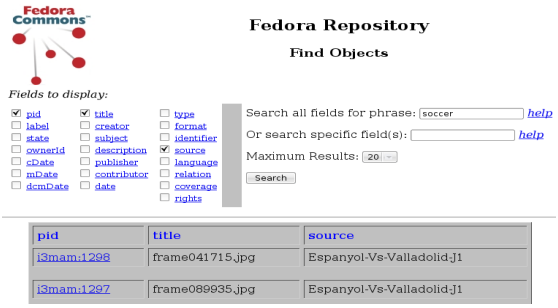


Figure 10.9: Fedora Commons interface for query by text.

10.3.5 Annotation

UPSeek considers two methods for the semantic annotation of content: manual and automatic. Manual annotation is supervised by the user while automatic annotation is the result of appliying pattern recognition techniques on the repository content. GAT is the client tool designed to manage all tasks related to content annotation, whether at the image and/or region scale. Local annotation requires the BPT datastream, while global ones need the thumbnails, both of them generated during the image ingest.

The manual annotation provides data for the query by text web service. Firstly, because the keywords associated to the annotated semantic class are added to the Dublin Core datastream. Secondly, because the selected images and/or regions are added as representation of the selected semantic class to the *annotation* datastream. The manual annotations are considered as training samples of classifiers whose goal is to generate new automatic annotations on unlabelled data.

The training of the classifiers is currently launched by the user when enough training samples are available. A confidence value is associated to every annotation, giving to the manual ones the highest score. This way a review process can easily validate the automatic annotations as the resulting detections are sorted by confidence. The validation process also offers the chance to define a minimum threshold for the automatic annotations are automatically considered valid and added to the Dublin Core metadata. The reader is referred to [13] for more details on the GUI that exploits this web service.

10.3.6 Text detector

The detection of semantic entities is at the same time the most powerful and challenging of the services offered by a CBIR. Among the broad range of interests in the automatic recognition field, one of the most studied cases is the text recognition [110]. These types of algorithms provide a direct textual annotation of the content as the detected words can be directly indexed and queried by traditional text-based engines. UPSeek offers a text detector build in two separate pieces of software. The first one uses the BPT and local features to find

boxes containig text using a pre-trained region-classifier [53]. Each box is binarized and the resulting segment processed with the tesseract OCR engine[83].

10.4 Conclusions

This paper has provided an overview of a content-based image retrieval system following a server-client architecture on the cloud. The most demanding tasks in terms of storage and processing are executed on the server, while those processes that require an intensive user interaction are run on the client-side. The connectivity between all parts is achieved through web protocols valid for content and metadata transport as well as for application signalling. The proposed design offers state of the art image analysis algorithms under the form of web services, which can be accessed from the proposed user applications but also by third part engines thanks to an API based on REST protocol and MPEG-7/XML data format. This type of architecture follows the growing trend in the software engineering world of offering solutions on the cloud, replacing costly and complicated local installations for simple calls to remote services via well designed APIs.

These services are being tested with industry partners that use them to check the development of the research results in a simple and convenient way for both parts. Although establishing these communications channels requires an important and initial effort from both sides, this design introduces an powerful flexibility to the scheme making it modular an extendible. The author considers that a similar solution could also applied to evaluation campaigns of CBIR systems, that would put the base of an eco-system of signal processing services available online. Following this philosophy, the client parts in UPSeek (UPLoad, GAT and GOS) are published [1] under an open source license for evaluation and future development from the research community.

Future steps in the architecture of the system will address the efficient indexing of the huge amount of region-based features generated, whose access currently limitates the performance of the system in terms of speed. On the client side, a better integration of the presented tools with the objects and datastreams in Fedora Commons will also be explored, so provide a similar feel to the user whether exploring the local filesystem or the remote repository.

[1] http://upseek.upc.edu

Chapter 11

GAT: Graphical Annotation Tool

11.1 Motivation

The annotation of visual content has become a topic of interest in scientific community due to the recent advances in pattern recognition applied on computer vision systems. Most of automatic detection solutions rely on a training phase during which classifiers are trained with a corpus of manual annotated data. While traditionally visual content has been annotated at a large scale (whole image or video files), more recent region-based indexing techniques require a finer resolution to train those detectors that are able to localize objects in a specific part of the image or video. To do so, new initiatives have been developed, such as the project LabelMe [73] that provides a web interface to annotate polygonal regions by the definition of their vertices. A previous analysis of the visual data can help such a region-based annotation process, as the tablet PC techniques proposed in the Videotater project [21], where users refine and tag video segments that have been automatically delimited. In the case of still images, a hierarchical organization of partitions generated from the image can facilitate a multiresolution navigation in the image, as presented in [25].

The annotation process not only requires selecting visual data but also associating it to a semantic class. If this class has a semantic meaning, as in most computer vision tasks, these semantics must be defined in an additional data structure. Ontologies are the most common solutions adopted by the scientific community as they define classes in a formal and structured manner. Successful computer vision techniques not only base their results on the signal processing algorithms but also on semantic reasoning processed at a higher level. The use of ontologies introduces context in the analysis task and provide a natural methodology to fuse image analysis with other modalities such as text and audio. For these reasons, annotation tools not only need to offer a workspace to select image and regions but must also provide mechanisms to handle ontologies [67].

This chapter presents GAT, a Graphical Annotation Tool that integrates in the same

interface a framework to annotate images at global and local scales as well as to import, create and edit ontologies. The resulting tool is implemented in Java and uses MPEG-7 XML compliant files written by Java binding classes automatically generated by Apache XMLBeans and the MPEG-7 schemas. This software is addressed to scientific academic audience that can find in GAT a solution to generate MPEG-7/XML standard annotations to be later used to test their own algorithms.

The rest of the chapter is structured as follows. Section 11.2 presents the different options to select areas of support at the global and local scales. Section 11.3 describes how semantic data is displayed while Section 11.4 explains the intended architecture and used data formats.

11.2 Visual Annotation

The annotation of images can be performed at two basic visual scales: global or local. In the global case the area of support is the full image, while local annotations mark a subset of the image pixels that depict a semantic object. GAT provides different tools to assist users for a quick interaction with images at both scales.

All presented strategies share a basic workflow for annotation. Firstly, the user selects the segments of images that are to be annotated and, as a response, the interface clearly highlights the selection. At this point, the user can decide to modify the selection or validate it with a right-click on the mouse. After validation, the new instance(s) are created in the annotation and clearly marked on the interface. This way, the right-click becomes the common action for validation in all annotation variants.

11.2.1 Global scale

Annotations at global scale normally consider collections of images. GAT provides a dedicated *Collection* tab that explores the content of a folder in the file system and shows the thumbnails of the included images. In most cases, viewing the thumbnails is enough for users to decide about the image semantics but, if necessary, a double click on any of them displays the full image on a new *Image* tab.

A broad range of machine learning techniques require that annotations consider not only what observations correspond to a semantic class but also which of them do not correspond to the class. A classic example are binary classifiers, that use two types of labels: positive and negative. In some situations a third type of label, the neutral one, is also used. This label just states the existence of the observation but that does not associate it to any of two basic labels. These observations are usually discarded for training or experimentation [98] as its inclusion may harm the overall performance. These three types of labels are supported in GAT only in the case of global annotations, as local annotations usually imply a positive label for the selected segment and a negative label for the rest of the image.

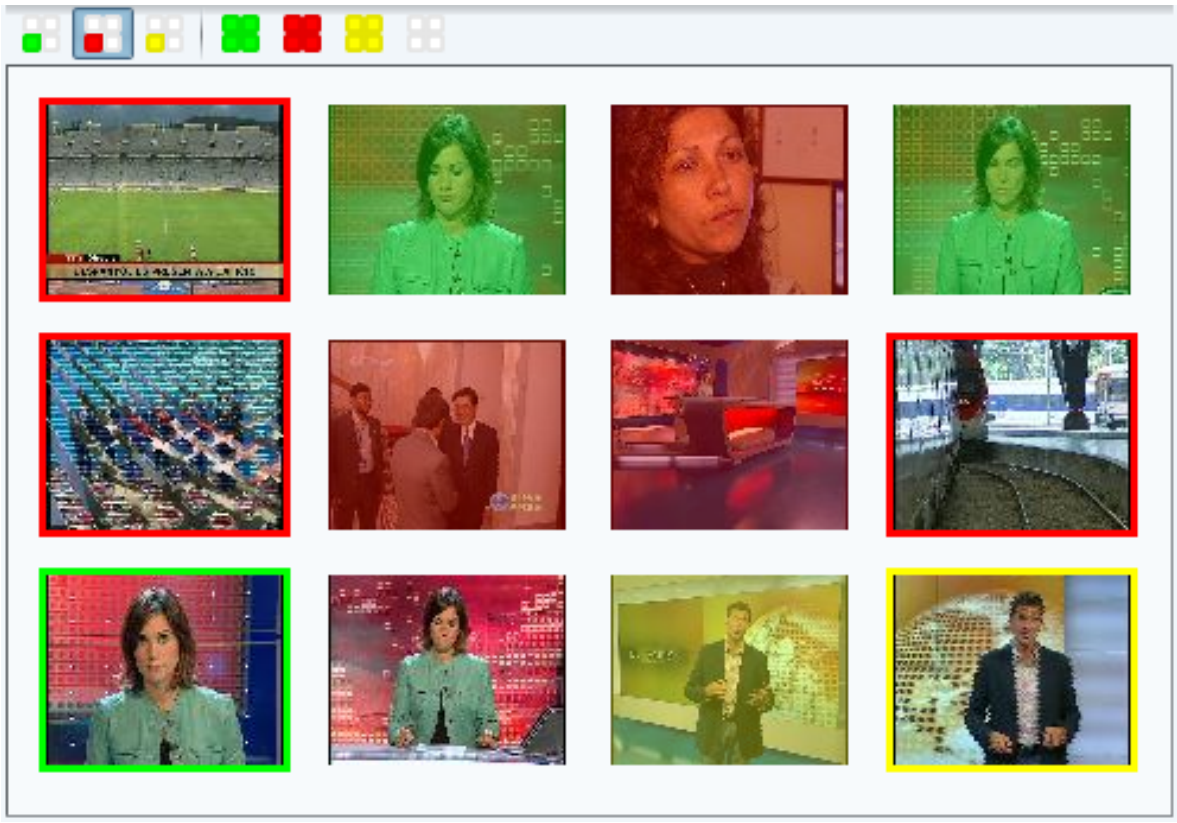


Figure 11.1: Selected (with frame) vs annotated (alpha mask) images. The color indicates the type of label: positive (green), neutral (yellow) and negative (red)

The assignment of global labels starts by clicking on one of the six icons located on the *Collection* tab's toolbar. Their color intuitively indicates what label are they related to: green (positive), red (negative) or yellow (neutral). These icons provide two different types of selection tools: individual or all. The first group activates the associated label so that every new click will associate the label to the image. The second group sets the selected labels to all currently unannotated images. For example, this functionality becomes very practical in the case when only a few of the displayed images belong to the class. In this situation, an initial red labeling to all thumbnails can be later be corrected by adding the remaining green instances.

Figure 11.1 shows how selected and annotated thumbnails are discriminated. When a thumbnail is selected, a frame of the associated label's color is painted around it, but when this selection becomes a validated annotation, a semi-transparent mask of the label's color is painted over the thumbnail.

11.2.2 Local scale

As previously explained, a double-click on a thumbnail of the Collection tab will activate to the *Image* tab where the selected image is shown at full view. Apart from providing a more detailed view of the image, this tab allows its local annotation. All local annotations are assigned to the positive label so, in this mode the color code used for global annotations does not apply. The color of the markers used for local selection can be defined by the user to avoid visual confusion between the instance selection and the background.

Local-scale solutions can be divided in two groups depending on the sought precision. A first family of techniques provides *rough* descriptions of the objects [73], giving approximate information about their location and shape, normally, using geometric figures. A second option for local annotations is the precise *segmentation* of those pixels that represent the object, by defining the exact area of support associated to the object [74]. GAT provides tools for both options, with special emphasis on interactive segmentation strategies for the second case.

Rough annotation

GAT allows drawing geometric markers over the image to indicate the local presence of a semantic instance. The catalogue includes points, lines and rectangles, which can be combined on a single annotation. Figure 11.2 shows an example of a rectangle-based rough annotation of two soccer players.

Interactive segmentation

The interactive segmentation modules allows a precise extraction of the object from the background. The proposed techniques are described in the core of the thesis, in Chapter 3.

11.3 Semantic data

The presented tools for the selection of visual data are complemented with a *Semantic* Panel located on the right-side of the interface. When working on the Collection tab, this panel shows the annotated instances in the current directory, while in the Image tab case it displays the local instances found in the image. The whole interface is shown in Figure 3.2, where the Semantic Panel highlights one instance of the semantic class “Anchor”. The classes defined in the active ontology are listed there and the annotated instances appear as child nodes in a tree-based structure. The selection of instance nodes in this tree or thumbnails on the visual panels is synchronized to simplify navigation. The upper part of the panel includes a search form to find entries on those ontologies with a high amount of classes.

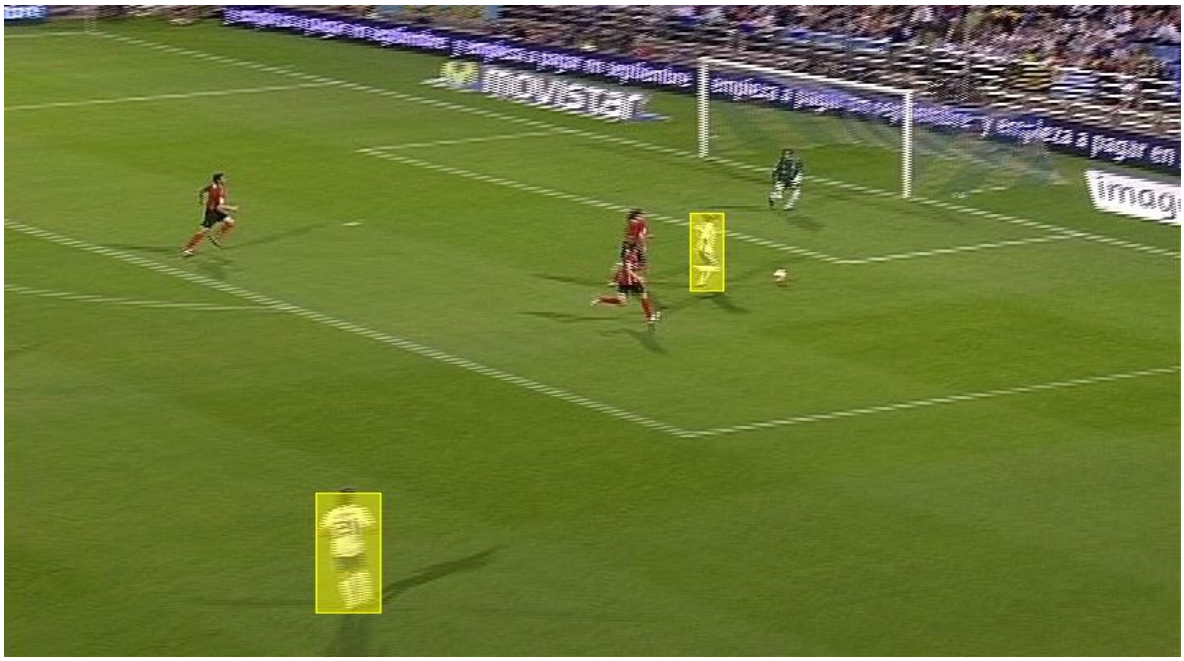


Figure 11.2: Rough annotation of soccer players.

In addition to the Collection and Image Tabs, GAT offers a third tab called *Annotation*. This tab displays thumbnails of all the annotated instances from the selected class in the Semantic Panel, independently from their scale scale, local path or label. If the root node of the Semantic Panel is clicked, the Annotation tab summarizes the complete annotation session and represents the information that GAT saves in the annotation files presented in Section 11.4.

Any new annotation must start by importing or creating an ontology. GAT also includes a simple ontology editor that lets users add and delete semantic classes as well as edit their textual labels.

11.4 Architecture and data formats

Regarding data formats, GAT is based on MPEG-7/XML to code the ontologies, annotations and BPTs. Examples of all types of documents are provided with the software package. In addition, GAT can also read OWL ontologies, a very popular format among the semantic community. The most common image coding formats are also supported by GAT, as it uses the native Java classes. In addition, it also supports a developed PRL data format to code image partitions of 32 bits per pixel. Nevertheless, partitions can be coded in any other format supported by Java (PNG, BMP,...).

GAT is designed to both read precomputed BPTs or use an external tool to use them

whenever need. In the second case, this additional tool can be a binary file in the local machine or a web service accessed through the Internet.

11.4.1 Input and Output Data

The presented tool uses two main sources of information: visual data associated to the still image, and a semantic ontology from where objects or their parts are selected. The annotation tool allows the user to generate an output file describing which regions of the input image represent instances of semantic classes defined in the ontology. Moreover, the same interface includes tools for the creation and edition of the semantic ontology.

Visual Input

Three different types of data related to the visual content are necessary for the annotation tool to work. Firstly, the still image itself; that is, the actual pixels in any standard format (JPG, PNG...). Secondly, an initial partition of the still image previously generated through a segmentation process. Such data is represented by another image whose pixel values correspond to region labels. Finally, the annotation tool requires an additional set of regions defined as combinations of the regions in the initial partition and structured as a Partition Tree (PT). The PT leaves correspond to the regions in the initial partition, while the root represents the whole image. The tree structure guarantees that there exists a single path between any leaf and the root. The access to the three sources of data is achieved through a single MPEG-7 XML file that contains the parent-child relations among the nodes in the PT as well as references to the files with the input image and initial partition.

Semantic Input

The annotated concepts are uniquely identified and structured in a library of semantic classes. There exist two types of classes: atomic and composite. While the visual representation of *atomic* classes cannot be further splitted from a semantic point of view, *composite* classes are defined as a combination of parts visually distinguishable and semantically meaningful, that is, as a combination of other classes. The decomposition process can be iterated to define semantic hierarchies under the form of ontologies. The current tool version reads the semantic data from an MPEG-7 XML compliant file and allows its edition through a graphical user interface.

Annotation Output

The output data is a description of which regions or combinations of regions from the initial partition represent instances of those classes included in the semantic library. In the case of

atomic classes, each annotated instance includes the list of associated regions from the initial partition. On the other hand, annotated instances of composite classes refer to the annotated instances of their parts. Such information is written in an MPEG-7 XML compliant file which also includes references to the input image and initial partition files.

11.4.2 User Interface

The graphical user interface shown in Figure 11.3 is divided in three parts: on the left, the ontology panel for the selection and management of classes and, below it, the instances panel for the review of annotated data. Occupying most of the window, the image panel for the selection and visual display of regions.

Ontology Panel

The ontology panel is located shows the semantic classes defined in the library. When the user focus the interest in one of them, the image panel highlights the regions associated to those instances that have been previously annotated, if any. The ontology panel includes an edition tool that allows the definition of new classes as well as the edition of existing ones. Users can add and delete classes, but also add or remove parts from composite classes. The addition of parts is always based on those classes already defined, so when a brand new part is to be added to a composite class, it must be firstly defined as an atomic class and later added to the composite class. The editor allows multiple instances of the same part in the definition of a composite class, but prevents the creation of semantic cycles, that is, an ancestor class cannot be a child of its descendants.

Image Panel

The image panel is used to select those regions containing instances of the class selected on the ontology panel. After reading from disk the input visual data described in Section 11.4.1, the application shows the image on screen and activates a region-based navigation system through the PT. Users can choose any pixel from the image as an anchor point by left-clicking on it, an action that toggles the state of the associated PT leaf between selected and unselected. The new state can be propagated to upper PT nodes by using the mouse wheel. The selected nodes are visualized by making their associated pixels transparent on an overlay mask of selectable colour and transparency.

Instances Window

The instances panel shows in a tree the annotated instances of the class selected in the ontology panel. By selecting any of them, their associated regions are shown in the image

panel.

11.5 Annotation Cycle

The process of annotation follows a cycle that can be repeated as many times as semantic instances in the image. Firstly, the user must select a class from the ontology panel by left-clicking on it or using the mouse wheel. In case of selecting an atomic class, the next step is to choose which regions instantiate it, while in the case of composite classes, a new annotation cycle is initiated for each of its parts.

The selection of regions starts with the first left-click on the annotation panel and it is achieved with the tools described in Subsection 11.4.2. As a result of the process, a set of regions from the initial partition is selected and must be validated with a right-click on the mouse. Afterwards, a new class can be selected on the ontology panel, or another instance of the same class annotated by selecting another set of regions. In the case of composite classes, the annotation can also be performed by selecting the object parts from the previously annotated instances selected on the instances panel.

11.5.1 Example

In our case, the Binary Partition Tree (BPT) [75] image representation was selected to generate the PT. In the BPT, intermediate nodes are iteratively defined by merging the two adjacent regions which minimize a certain fusion criterion (eg. based on colour). Figure11.3 shows a snapshot of the tool where the image contains four instances of the semantic class *Laptop*. The instances panel shows that two screens have already been annotated while a third is being selected on the image panel.

Figure 11.3: Snapshot of the user interface

Chapter 12

ETHZ Dataset processing

The ETHZ dataset [26] has been processed in order to generate a framework valid for the evaluation purposes reported in this thesis. The annotation of the partitions from the provided bounding boxes and masks has required an adaptation process described in this section. The process has been divided in two stages, a first one where pixel-wise masks have been built with the provided data, and a second one where masks are defined as connected components in a partition.

12.1 Pixel-wise ground truth masks

The ETHZ dataset provides for every image the contours of the annotated objects, but not all of them are closed. This characteristic generates a conflict in the considered region-based approach because, by definition, a region is defined by a closed contour. For this reason, the provided contours were reviewed to guarantee all of them are closed. Figure 12.2 shows an example of one of the manually closed contours.

The next problem was the variability in the file naming. Three object categories (applelogos, bottles and giraffes) use a suffix for the contour files, and the two other categories (mugs and swans) use a different extension. This diversity introduced some complexity when

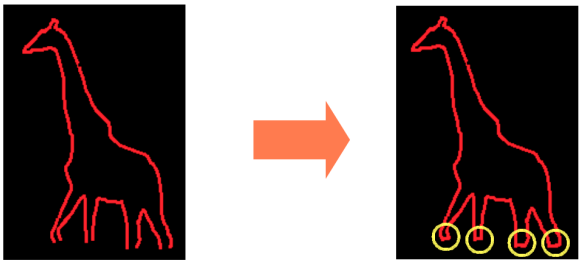


Figure 12.1: Manually closed contours.

Figure 12.2: Filename diversity on contour files.

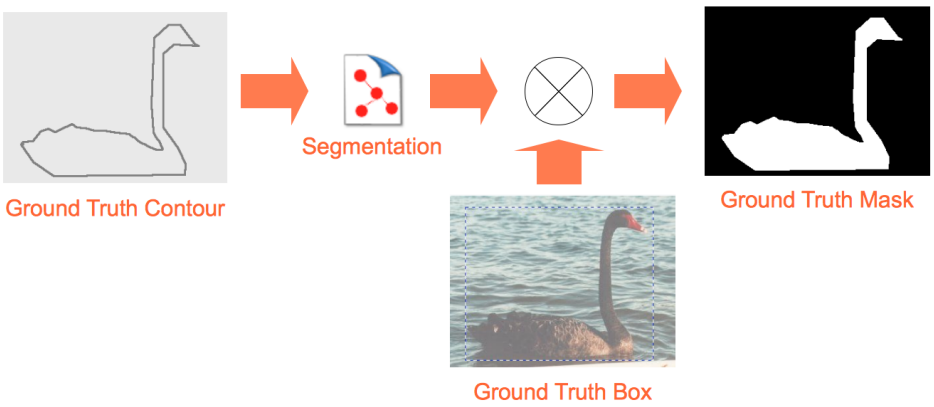


Figure 12.3: Generation of the pixel-wise mask.

writing the software to manage the dataset. Moreover, in the first set every instance of an object was in a separate mask file, while in the two last categories they were all in the same image, so they had to be manually split and renamed consistently with the first scheme.

Figure 12.3 presents the architecture used to process the images with the closed contours to generate the pixel-wise masks. Firstly the contour images were segmented to generate a simple partition. Afterwards, the provided boxes were used to mark the regions in the partition of the contour image, as presented in Section 3.1.2. As a result, a set of regions of the contour partition were selected to generate the pixel-wise mask of the ETHZ dataset.

This stage also generated some more problems, as the bounding boxes provided with the dataset do not correspond to the positions of the provided contours. Especially in the giraffes category, many of the rectangles did not include completely the contour, so the mapping algorithm would miss them. The solution was to modify the provided ground truth by manually determining the bounding boxes again.

12.2 Partition-based ground truth masks

Once the mask for the object was generated, this mask was used onto the resulting segmentation of the input image, as presented in Section . In this case a minimum value for the Jaccard index for a region to be considered was empirically set to 0.7. Figure 12.4 presents a case

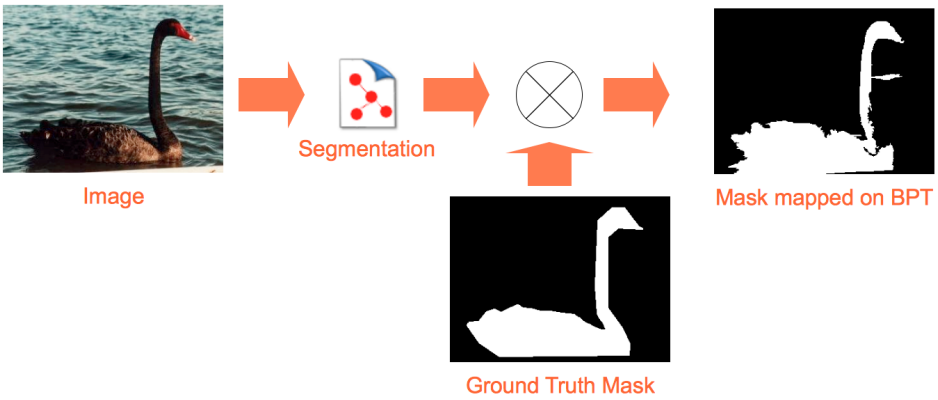


Figure 12.4: Generation of the partition-based mask.

	Original		Filtered	
Category	# images	# instances	# images	# instances
Applelogos	40	44	39	43
Bottles	48	55	48	54
Giraffes	87	91	72	78
Mugs	48	66	48	60
Swans	32	33	29	29
TOTAL	255	289	236	264

Table 12.1: ETHZ dataset

where the resulting mapping generates a mask that introduces portions of the background introduced by the image segmentation.

At the end, the mapping of the ground truth masks on the image segmentations defined a set of regions that could be compared with the ground truth to assess their quality. As this dataset was to be used in further retrieval and classification tasks, there was not interest in working with a dataset that does not include the objects that are supposed to be represented. For this reason, it was decided to work only with those object instances whose representation on the image partitions was equal or better than a minimum value of the Jaccard Index, which was also taken as 0.7. These instances are the ones used in any further experimentation with the ETHZ dataset. Table 12.1 presents an overview on the amount of images and objects which are defined before and after this mapping.

Bibliography

[1] *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Tomasz Adamek. *Using contour information and segmentation for object registration, modeling and retrieval*. PhD thesis, Dublin City University, 2006.

[3] Ethem Alpaydin. Soft vector quantization and the em algorithm. *Neural Netw.*, 11:467–477, April 1998.

[4] P. Arbelaez and L. Cohen. Constrained image segmentation from hierarchical boundaries. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, june 2008.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Ale Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin / Heidelberg, 2006.

[6] N. J. Belkin and P. Kantor. Combining the evidence of multiple query representations for information retrieval. In *Information Processing & Management*, 1995.

[7] James C. Bezdek and Nikhil R. Pal. Two soft relatives of learning vector quantization. *Neural Networks*, 8(5):729 – 743, 1995.

[8] A. Bosch, A. Zisserman, and X. Muoz. Image classification using random forests and ferns. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1 –8, oct. 2007.

[9] Y-Lan Boureau, Jean Ponce, and Yann Lecun. A theoretical analysis of feature pooling in visual recognition. In *Proc. 27th International Conference on Machine Learning (ICML), Haifa, Israel*, 2010.

[10] G. Briscoe and T. Caella. *A Compendium of Machine Learning*. Ablex Publishing, Norwood, NJ., 1996.

[11] P. Salembier B.S. Manjunath and Eds. T. Sikora, editors. *Introduction to MPEG-7: Multimedia Content Description Interface*. Wiley, Chichester, West Sussex, UK, 2002.

[12] V.V.Vasudevan A.Yamada B.S. Manjunath, J.R. Ohm. Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 2(6):703–715, June 2001.

[13] Elisabet Carcel, Manuel Martos, Xavier Giro-i Nieto, and Ferran Marques. Rich internet applicatiosn for semi-automatic annotation of semantic shots in keyframes. In *Proceedings of the MUSCLE International Workshop on Computational Intelligence for Multimedia Understanding*, ICMR '11, 2011.

[14] Chad Carson, Megan Thomas, Serge Belongie, Joseph Hellerstein, and Jitendra Malik. Blobworld: A system for region-based image indexing and retrieval. In Dionysius Huijsmans and Arnold Smeulders, editors, *Visual Information and Information Systems*, volume 1614 of *Lecture Notes in Computer Science*, pages 660–660. Springer Berlin / Heidelberg, 1999.

[15] Yuning Chai, V. Lempitsky, and A. Zisserman. Bicos: A bi-level co-segmentation method for image classification. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2579 –2586, nov. 2011.

[16] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[17] Shi-Kuo Chang, Qing-Yun Shi, and Cheng-Wen Yan. Iconic indexing by 2-d strings. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-9(3):413 – 428, may 1987.

[18] Shih-Fu Chang, John R. Smith, Mandis Beigi, and Ana Benitez. Visual information retrieval from large distributed online repositories. *Commun. ACM*, 40(12):63–71, 1997.

[19] Michael Christel and Alexander Hauptmann. The use and utility of high-level semantic features in video retrieval. In Wee-Kheng Leow, Michael Lew, Tat-Seng Chua, Wei-Ying Ma, Lekha Chaisorn, and Erwin Bakker, editors, *Image and Video Retrieval*, volume 3568 of *Lecture Notes in Computer Science*, pages 588–588. Springer Berlin / Heidelberg, 2005.

[20] L. Cohen, L. Vinet, P.T. Sander, and A. Gagalowicz. Hierarchical region based stereo matching. In *Computer Vision and Pattern Recognition, 1989. Proceedings CVPR '89., IEEE Computer Society Conference on*, pages 416 –421, jun 1989.

[21] D. Diakopoulous and I. Essa. Videotater: an approach for pen-based digital video segmentation and tagging. In *ACM Symp. on User Interface Software and Technology*, pages 221–224, Montreux, October, 2006.

[22] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2 edition, November 2001.

[23] Max J. Egenhofer. Query processing in spatial-query-by-sketch. *Journal of Visual Languages and Computing*, 8:403–424, 1997.

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

[25] et al F. Marques. Partition-based image representation as basis for user-assisted segmentation. In *Proc. of the Int. Conference on Image Processing (ICIP)*, volume 1, pages 312–315, Vancouver, October, 2000.

[26] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Object detection by contour segment networks. In *Proceeding of the European Conference on Computer Vision*, volume 3953 of *LNCS*, pages 14–28. Elsevier, June 2006.

[27] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Qian Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: the qbic system. *Computer*, 28(9):23 –32, sep 1995.

[28] Andrea Frome, Yoram Singer, and Jitendra Malik. Image Retrieval and Classification Using Local Distance Functions. In Bernhard Schölkopf, John Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems 19*, number 4, pages 417–424. The MIT Press, 2006.

[29] Nikhil Garg and Ingmar Weber. Personalized tag suggestion for flickr. In *WWW 2008 Conference Proceedings*, pages 1063–1064. ACM, 1997.

[30] Xavier Giró and Ferran Marqués. System architecture for indexing regions in keyframes. In *Poster and Demo Proc. 3rd International Conference on Semantic and digital Media Technologies*, Koblenz, Germany, December 2008.

[31] Xavier Giro-i Nieto, Monica Alfaro, and Ferran Marques. Diversity ranking for video retrieval from a broadcaster archive. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, ICMR '11, pages 56:1–56:8, New York, NY, USA, 2011. ACM.

[32] Xavier Giro-i Nieto, Neus Camps, and Ferran Marques. Gat, a graphical annotation tool for semantic regions. *Multimedia Tools and Applications*, 46(2):155–174, 2010.

[33] Xavier Giro-i Nieto, Ramon Salla, and Xavier Vives. Digimatge, a rich internet application for video retrieval from a multimedia asset management system. In *Proc. ACM SIGMM Intl. Conf. on Multimedia Information Retrival*, Philadelphia, PA, USA, March 2010.

[34] Xavier Giro-i Nieto, Carles Ventura, Jordi Pont-Tuset, Silvia Cortes, and Ferran Marques. System architecture of a web service for content-based image retrieval. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '10, pages 358–365, New York, NY, USA, 2010. ACM.

[35] D. Gokalp and S. Aksoy. Scene classification using bag-of-regions representations. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1 –8, june 2007.

[36] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Efficient learning with sets of features. *J. Mach. Learn. Res.*, 8:725–760, May 2007.

[37] Chunhui Gu, J.J. Lim, P. Arbelaez, and J. Malik. Recognition using regions. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1030 –1037, june 2009.

[38] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):pp. 100–108, 1979.

[39] Laurie J. Heyer, Semyon Kruglyak, and Shibu Yooseph. Exploring expression data: Identification and analysis of coexpressed genes. *Genome Research*, 9(11):1106–1115, 1999.

[40] W.H.-M. Hsu and Shih-Fu Chang. Generative, discriminative, and ensemble learning on multi-modal perceptual fusion toward news video story segmentation. In *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, volume 2, pages 1091 –1094 Vol.2, june 2004.

[41] Alejandro Jaimes and Shih-Fu Chang. Learning structured visual detectors from user input at multiple levels. *Invited Paper, International Journal of Image and Graphics (IJIG), Special Issue on Image and Video Databases*, 1(3):415–44, August 2001.

[42] Yu-Gang Jiang, Ngo Chong-Wah, and Jun Yang Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *CIVR 2007 Conference Proceedings*, pages 494–501. ACM, July 2007.

[43] Yu-Gang Jiang, Akira Yanagawa, Shih-Fu Chang, and Chong-Wah Ngo. CU-VIREO374: Fusing Columbia374 and VIREO374 for Large Scale Semantic Concept Detection. Technical report, Columbia University ADVENT #223-2008-1, August 2008.

[44] T. Kato, T. Kurita, N. Otsu, and K. Hirata. A sketch retrieval method for full color image database-query by visual example. In *Pattern Recognition, 1992. Vol.I. Conference A: Computer Vision and Applications, Proceedings., 11th IAPR International Conference on*, pages 530 –533, aug-3 sep 1992.

[45] Lyndon S. Kennedy and Shih-Fu Chang. A reranking approach for context-based concept fusion in video indexing and retrieval. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, CIVR '07, pages 333–340, New York, NY, USA, 2007. ACM.

[46] J.F. Kenney and E.S. Keeping. *Mathematics of Statistics, Pt. 1*, chapter 15, pages 252–285. Van Nostrand, 1962.

[47] H. Kosch. *Distributed Multimedia Database Technologies supported by MPEG-7 and MPEG-21.* CRC Press, 2004.

[48] Neeraj Kumar. *Describable Visual Attributes for Face Images.* PhD thesis, Columbia University, 2011.

[49] Rajendran Kumar and Shih-Fu Chang. Image retrieval with sketches and compositions. In *ICME 2000 Conference Proceedings*, pages 84–89. IEEE, August 2000.

[50] C. Lagoze, S. Payette, E. Shin, and C. Wilper. Fedora: an architecture for complex objects and their relationships. *International Journal on Digital Libraries*, 6(2):124–138, April 2006.

[51] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2169–2178, 2006.

[52] M. Leon, V. Vilaplana, A. Gasull, and F. Marques. Region-based caption text extraction. In *Image Analysis for Multimedia Interactive Services (WIAMIS), 2010 11th International Workshop on*, pages 1 –4, april 2010.

[53] Miriam Leon, Veronica Vilaplana, Antoni Gasull, and Ferran Marques. Caption text extraction for indexing purposes using a hierarchical region-based image model. In *Proc. ICIP-09, IEEE International Conference on Image Processing*, El Cairo, Egypt, November 2009.

[54] Jia Li, James Z. Wang, and Gio Wiederhold. Irm: integrated region matching for image retrieval. In *Proceedings of the eighth ACM international conference on Multimedia*, MULTIMEDIA '00, pages 147–156, New York, NY, USA, 2000. ACM.

[55] Rainer Lienhart. Automatic text recognition for video indexing. In *ACM Multimedia 1997 Conference Proceedings*, pages 11–20. ACM, 1997.

[56] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004. 10.1023/B:VISI.0000029664.99615.94.

[57] Mathias Lux. Caliph & emir: Mpeg-7 photo annotation and retrieval. In *MM '09: Proceedings of the seventeen ACM international conference on Multimedia*, pages 925–926, New York, NY, USA, 2009. ACM.

[58] Kevin McGuinness and Noel E. OConnor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 43(2):434 – 444, 2010.

[59] Javier Molina, Evaggelos Spyrou, Natasa Sofou, and Jos Martnez. On the selection of mpeg-7 visual descriptors and their level of detail for nature disaster video sequences classification. In Bianca Falcidieno, Michela Spagnuolo, Yannis Avrithis, Ioannis Kompatsiaris, and Paul Buitelaar, editors, *Semantic Multimedia*, volume 4816 of *Lecture Notes in Computer Science*, pages 70–73. Springer Berlin / Heidelberg, 2007.

[60] P. Mylonas, E. Spyrou, Y. Avrithis, and S. Kollias. Using visual context and region semantics for high-level concept detection. *Multimedia, IEEE Transactions on*, 11(2):229 –243, feb. 2009.

[61] Apostol (Paul) Natsev, Milind R. Naphade, and Jelena TešiĆ. Learning the semantics of multimedia queries and concepts from a small number of examples. In *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA '05, pages 598–607, New York, NY, USA, 2005. ACM.

[62] Juan Carlos Niebles and null Li Fei-Fei. A hierarchical model of shape and appearance for human action classification. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[63] Alexandre Noma, Ana B.V. Graciano, Roberto M. Cesar Jr, Luis A. Consularo, and Isabelle Bloch. Interactive image segmentation by matching attributed relational graphs. *Pattern Recognition*, 45(3):1159 – 1179, 2012.

[64] Eric Nowak, Frdric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In Ale Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision ECCV 2006*, volume 3954 of *Lecture Notes in Computer Science*, pages 490–503. Springer Berlin / Heidelberg, 2006.

[65] T. Ojala, M. Aittola, and E. Matinmikko. Empirical evaluation of mpeg-7 xm color descriptors in content-based retrieval of semantic image categories. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 2, pages 1021 – 1024 vol.2, 2002.

[66] Paul Over, George Awad, Brian Antonishek, Martial Michel, Wessel Kraaij, Alan F. Smeaton, and Georges Quenot. Trecvid 2010 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proceedings of TRECVID 2010*. NIST, USA, 2010.

[67] Kosmas Petridis, Dionysios Anastasopoulos, Carsten Saathoff, Yiannis Kompatsiaris, and Steffen Staab. Montomat-annotizer: Image annotation, linking ontologies and multimedia low-level features. In *KES 2006 - 10th Intl. Conf. on Knowledge Based, Intelligent Information and Engineering Systems*, 2006.

[68] John C Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.

[69] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1 –8, oct. 2007.

[70] Azriel Rosenfeld, Robert A. Hummel, and Steven W. Zucker. Scene labeling by relaxation operations. *Systems, Man and Cybernetics, IEEE Transactions on*, 6(6):420 –433, june 1976.

[71] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, August 2004.

[72] Yong Rui, T.S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: a power tool for interactive content-based image retrieval. *Circuits and Systems for Video Technology, IEEE Transactions on*, 8(5):644 –655, sep 1998.

[73] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Labelme: A database and web-based tool for image annotation. *Int. J. Comput. Vision*, 77(1-3):157–173, 2008.

[74] Carsten Saathoff, Simon Schenk, and Ansgar Scherp. Kat: The k-space annotation tool. In *Proccedings of the SAMT, 2008*.

[75] 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, apr 2000.

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

[77] P. Salembier and F. Marqués. Region-based representations of image and video: segmentation tools for multimedia services. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1147–1169, December 1999.

[78] Walter Scheirer, Anderson Rocha, Ross Micheals, and Terrance Boult. Robust fusion: Extreme value theory for recognition score normalization. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision ECCV 2010*, volume 6313 of *Lecture Notes in Computer Science*, pages 481–495. Springer Berlin / Heidelberg, 2010.

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

[80] Josef Sivic and Andrew Zisserman. Video Google: a text retrieval approach to object matching in videos. *Proceedings Ninth IEEE International Conference on Computer Vision*, (Iccv):1470–1477 vol.2, 2003.

[81] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(12):1349 –1380, dec 2000.

[82] John R. Smith and Shih-Fu Chang. Visualseek: a fully automated content-based image query system. In *Proceedings of the fourth ACM international conference on Multimedia*, MULTIMEDIA '96, pages 87–98, New York, NY, USA, 1996. ACM.

[83] R. Smith. An overview of the tesseract ocr engine. In *Proc. 9th Intl. Conf. on Document Analysis and Recognition*, volume 2, pages 629–633, Curitiba, Brazil, September 2007.

[84] Cees G. M. Snoek, Marcel Worring, and Arnold W. M. Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA '05, pages 399–402, New York, NY, USA, 2005. ACM.

[85] Cees G. M. Snoek, Marcel Worring, Jan van Gemert, Jan-Mark Geusebroek, Dennis Koelma, Giang P. Nguyen, Ork de Rooij, and Frank Seinstra. Mediamill: exploring news video archives based on learned semantics. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 225–226, New York, NY, USA, 2005. ACM.

[86] Cees G. M. Snoek, Marcel Worring, Jan C. van Gemert, Jan-Mark Geusebroek, and Arnold W. M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th annual ACM international conference on Multimedia*, MULTIMEDIA '06, pages 421–430, New York, NY, USA, 2006. ACM.

[87] R. Socher and Li Fei-Fei. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 966 –973, june 2010.

[88] Jeroen Steggink and Cees Snoek. Adding semantics to image-region annotations with the name-it-game. *Multimedia Systems*, 17:367–378, 2011. 10.1007/s00530-010-0220-y.

[89] Erik B. Sudderth, Antonio Torralba, William T. Freeman, and Alan S. Willsky. Learning hierarchical models of scenes, objects, and parts. *Computer Vision, IEEE International Conference on*, 2:1331–1338, 2005.

[90] David Tax and Robert Duin. Combining one-class classifiers. In Josef Kittler and Fabio Roli, editors, *Multiple Classifier Systems*, volume 2096 of *Lecture Notes in Computer Science*, pages 299–308. Springer Berlin / Heidelberg, 2001.

[91] Wen-Hsiang Tsai and King-Sun Fu. Error-correcting isomorphisms of attributed relational graphs for pattern analysis. *Systems, Man and Cybernetics, IEEE Transactions on*, 9(12):757 –768, dec. 1979.

[92] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[93] Carles Ventura. Image-based query by example using mpeg-7 visual descriptors. Master's thesis, Barcelona Telecom, 2010.

[94] Remi Vieux, Jenny Benois-Pineau, Jean-Philippe Domenger, and Achille Braquelaire. Segmentation-based multi-class semantic object detection. *Multimedia Tools and Applications*, pages 1–22. 10.1007/s11042-010-0611-2.

[95] V. Vilaplana, F. Marques, M. Leon, and A. Gasull. Object detection and segmentation on a hierarchical region-based image representation. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 3933 –3936, sept. 2010.

[96] V. Vilaplana, F. Marques, and P. Salembier. Binary partition trees for object detection. *Image Processing, IEEE Transactions on*, 17(11):2201 –2216, nov. 2008.

[97] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:511, 2001.

[98] Timo Volkmer, John R. Smith, and Apostol (Paul) Natsev. A web-based system for collaborative annotation of large image and video collections: an evaluation and user study. In *13th annual ACM Intl' Conference on Multimedia*, pages 892–901, New York, NY, USA, 2005. ACM.

[99] Dong Wang, Xirong Li, Jianmin Li, and Bo Zhang. The importance of query-concept-mapping for automatic video retrieval. In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pages 285–288, New York, NY, USA, 2007. ACM.

[100] James Z. Wang, Jia Li, and Gio Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(9):947–963, September 2001.

[101] Jue Wang, Pravin Bhat, R. Alex Colburn, Maneesh Agrawala, and Michael F. Cohen. Interactive video cutout. *ACM Trans. Graph.*, 24:585–594, July 2005.

[102] Tao Wang, Yong Rui, and Jia-Guang Sun. Constraint based region matching for image retrieval. *International Journal of Computer Vision*, 56:37–45, 2004. 10.1023/B:VISI.0000004831.53436.88.

[103] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability estimates for multi-class classification by pairwise coupling. *J. Mach. Learn. Res.*, 5:975–1005, December 2004.

[104] Rong Yan, Alexander Hauptmann, and Rong Jin. Multimedia search with pseudo-relevance feedback. In Erwin Bakker, Michael Lew, Thomas Huang, Nicu Sebe, and Xiang Zhou, editors, *Image and Video Retrieval*, volume 2728 of *Lecture Notes in Computer Science*, pages 649–654. Springer Berlin / Heidelberg, 2003.

[105] Keiji Yanai and Kobus Barnard. Region-based automatic web image selection. In *Proceedings of the international conference on Multimedia information retrieval*, MIR '10, pages 305–312, New York, NY, USA, 2010. ACM.

[106] C. Yang and T. Lozano-Perez. Image database retrieval with multiple-instance learning techniques. In *Data Engineering, 2000. Proceedings. 16th International Conference on*, pages 233 –243, 2000.

[107] Changbo Yang, Ming Dong, and Jing Hua. Region-based image annotation using asymmetrical support vector machine-based multiple-instance learning. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2057 – 2063, 2006.

[108] Jianchao Yang, Kai Yu, Yihong Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794 –1801, june 2009.

[109] Jun Yang, Yu-Gang Jiang, Alexander G. Hauptmann, and Chong-Wah Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, MIR '07, pages 197–206, New York, NY, USA, 2007. ACM.

[110] K. Zagoris, E. Kavallieratou, and N. Papamarkos. Developing document image retrieval system. In *IADIS Intl. Conf. on Computer Graphics and Visualization*, Amsterdam, The Netherlands, July 2008.

[111] Eric Zavesky. *A Guided, Low-Latency, and Relevance Propagation Framework for Interactive Multimedia Search.* PhD thesis, Columbia University, 2010.