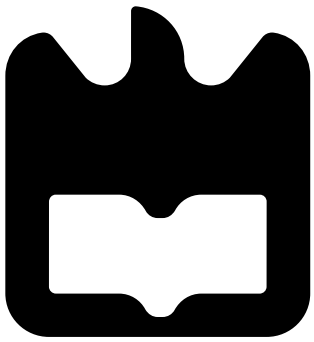
Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2011

Mário Luís
Pinto Antunes

Agente de Visão Semântica para Robótica

Semantic Vision Agent for Robotics



Mário Luís Pinto Antunes

Agente de Visão Semântica para Robótica

Semantic Vision Agent for Robotics

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenheria de Computadores e Telemática (M.I.E.C.T), realizada sob a orientação científica do Prof. Doutor Luís Seabra Lopes, Professor do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Armando José Formoso de Pinho

Professor Associado da Universidade de Aveiro (por delegação da Reitoria da Universidade de Aveiro)

vogais / examiners committee

Alexandre José Malheiro Bernardino

Professor Auxiliar do Instituto Superior Técnico da Universidade Técnica de Lisboa

Luís Seabra Lopes

Professor Auxiliar da Universidade de Aveiro (Orientador)

agradecimentos / acknowledgements

É com o maior gosto que aproveito esta oportunidade para agradecer a todos os que me ajudaram durante a realização deste trabalho. Começo por agradecer ao meu orientador Professor Luís Seabra Lopes, pelo apoio no trabalho e pelas sugestões para a correcção e realização desta dissertação.
Desejo também agradecer ao meu colega de laboratorio, Aneesh Chauhan, pela ajuda que prestou e disponibilidade demonstrada.
Finalmente, quero agradecer e dedicar este trabalho aos meus pais, Maria e António e aos amigos de longa data em especial a Sofia e a Daniela. Foram eles em conjunto que, embora entrando a minha vida em diferentes alturas, me deram a maior compreensão e ajuda que foram cruciais ao longo de todo o meu processo académico. Sem eles nada disto teria sido possível.

Resumo

Visão semântica é uma importante linha de investigação na área de visão por computador. A palavra-chave "semântica" implica a extracção de características não apenas visuais (cor, forma, textura), mas também qualquer tipo de informação de "alto-nível". Em particular, a visão semântica procura compreender ou interpretar imagens de cenas em termos dos objectos presentes e eventualmente das relações entre eles. Uma das principais áreas de aplicação actual é a robótica. Sendo o mundo que nos rodeia extremamente visual, a interacção entre um utilizador humano não especializado e um robô requer que o robô seja capaz de detectar, reconhecer e compreender qualquer tipo de referências visuais fornecidas no âmbito da comunicação entre o utilizador e o robô.

Para que tal seja possível, é necessária uma fase de aprendizagem, através da qual várias categorias de objectos são aprendidas pelo robô. Depois deste processo, o robô será capaz de reconhecer novas instâncias das categorias anteriormente aprendidas.

Foi desenvolvido um novo agente de visão semântica que recorre a serviços de pesquisa de imagens na Web para aprender um conjunto de categorias gerais a partir apenas dos seus respectivos nomes. O trabalho teve como ponto de partida o agente UA@SRVC, anteriormente desenvolvido na Universidade de Aveiro para participação no Semantic Robot Vision Challenge. O trabalho começou pelo desenvolvimento de uma nova técnica de segmentação de objectos baseada nas suas arestas e na diversidade de cor. De seguida, a técnica de pesquisa semântica e selecção de imagens de treino do agente UA@SRVC foi revista e reimplementada utilizando, entre outros componentes, o novo módulo de segmentação. Por fim foram desenvolvidos novos classificadores para o reconhecimento de objectos.

Apreendemos que, mesmo com pouca informação prévia sobre um objecto, é possível segmentá-lo correctamente utilizando para isso uma heurística simples que combina a diversidade da cor e a distância entre segmentos. Recorrendo a uma técnica de agrupamento conceptual, é possível criar um sistema de votos que permite efectuar uma boa selecção de instâncias para o treino de categorias. Conclui-se também que diferentes classificadores são mais eficientes quando a fase de aprendizagem é supervisionada ou automatizada.

Abstract

Semantic vision is an important line of research in computer vision. The keyword "semantic" means the extraction of features, not only visual (color, shape, texture), but also any "higher level" information. In particular, semantic vision seeks to understand or interpret images of scenes in terms of present objects and possible relations between them. One of the main areas of current application is robotics. As the world around us is extremely visual, interaction between a non specialized human user and a robot requires the robot to be able to detect, recognize and understand any kind of visual cues provided in the communication between user and robot.

To make this possible, a learning phase is needed, in which various categories of objects are learned by the robot. After this process, the robot will be able to recognize new instances of the categories previously learned.

We developed a new semantic vision agent that uses image search web services to learn a set of general categories based only on their respective names. The work had as starting point the agent UA@SRVC, previously developed at the University of Aveiro for participation in the Semantic Robot Vision Challenge.

This work began by developing a new technique for segmentation of objects based on their edges and diversity of color. Then, the technique of semantic search and selection of images from the agent UA@SRVC was revised and reimplemented using, among other components, the new object extracting module. Finally new classifiers were developed for the recognition of objects.

We learned that, even with little prior information about an object, it is possible to segment it correctly using a simple heuristic that combines colour disparity and distance between segments. Drawing on a conceptual clustering technique, we can create a voting system that allows a good selection of instances for training the categories. We also conclude that various classifiers are most effective when the learning phase is supervised or automated.

Contents

Contents **i**

List of Figures **v**

List of Tables **vii**

1 Introduction **1**

1.1 Motivation . . . 1

1.1.1 Semantic Robot Vision Challenge . . . 2

1.1.2 Robocup . . . 3

1.2 Research problems . . . 4

1.2.1 Object Recognition . . . 4

1.2.2 Contour-based segmentation . . . 5

1.2.3 Semantic image retrieval . . . 6

1.3 Objectives . . . 8

1.4 Overview of the agent . . . 8

1.5 Dissertation structure . . . 9

2 Related work **11**

2.1 Contour-Based Segmentation . . . 11

2.1.1 Canny edge detector . . . 11

2.1.2 Watershed . . . 13

2.1.3 Active contour . . . 15

2.1.4 Pb . . . 17

2.1.5 gPb . . . 18

2.2 Image Retrieval and Selection . . . 19

2.3 Generic object representation and recognition . . . 23

2.3.1 Shape Context . . . 23

2.3.2 Global Shape Context (GSC) . . . 24

2.3.3 Roy's Shape Representation (RSR) . . . 26
2.3.4 Object Recognition Using Junctions . . . 26

3 Object extraction and clutter removal 29
3.1 From edges to contours . . . 30
3.2 Contour aggregation metrics and criteria . . . 31
3.2.1 Bounding box criterion . . . 32
3.2.2 Distance . . . 33
3.2.3 Colour Disparity . . . 34
3.3 Contour aggregation algorithm . . . 35
3.4 Clutter Removal . . . 36
3.5 Application example . . . 38

4 Category learning 41
4.1 Internet-based image retrieval . . . 41
4.2 From images to object models . . . 42
4.3 Unsupervised object relevance evaluation . . . 43
4.3.1 Object clustering . . . 44
4.3.2 Object scoring . . . 46
4.4 Object selection and category model building . . . 46

5 Object recognition 49
5.1 Category membership measures . . . 49
5.1.1 Maximum inverse χ^2 distance . . . 49
5.1.2 Penalized membership . . . 51
5.2 Instance-based classification . . . 51
5.2.1 Nearest Neighbour . . . 52
5.2.2 k-Nearest-Neighbour (k-NN) . . . 52
5.2.3 Weighted k-NN . . . 53

6 Performance evaluation 55
6.1 Basic evaluations measures . . . 55
6.2 Contour-based object extraction . . . 56
6.3 Unsupervised object subset selection . . . 56
6.3.1 Original UA@SRVC2008 object extraction and selection modules . . . 57
6.3.2 Original extraction module combined with the new selection module . . . 58
6.3.3 Using the developed modules for object extraction and selection . . . 59

6.4 Classifiers 60
6.4.1 Cross validation with all available images 62
6.4.2 Cross-validation with increasing number of categories 63
6.5 Semantic vision agent 65
6.5.1 Maximum inverse χ^2 distance 65
6.5.2 Penalized membership 66

7 Conclusion **69**
7.1 Overview 69
7.2 Conclusions 70
7.3 Future work 71

A Performance tables **73**
A.1 Contour-based object extraction 75
A.2 Unsupervised subset selection 77
A.3 Classifiers 78

Bibliography **81**

List of Figures

1.1 Example of detection of objects in SRVC. 2
1.2 Example of an air ball detection and recognition. 3
1.3 CAMBADA@Home robot. 4
1.4 A comparison between generic and specific categories. 5
1.5 A comparison between edge detection and contour segmentation. 6
1.6 Google search for "scissors". 7
1.7 Semantic similarity $\neq$ visual similarity. 7
1.8 Architecture of the developed agent. 9

2.1 Example of application of the Canny edge detector. 13
2.2 Example of one dimension watershed. 14
2.3 Watershed practical example. 15
2.4 Active contour or snake simple example. 16
2.5 Active contour or snake practical example. 17
2.6 *Pb* for one point, scale and direction. 17
2.7 Architecture of the gPb detector. 19
2.8 Various applications of the gPb detector. 19
2.9 Shape Context and matching. 24
2.10 Shape Context as a global descriptor. 25
2.11 Roy's Shape Representation. 26
2.12 First junction feature. 27
2.13 Second junction feature. 27

3.1 Proposed algorithm for border tracing. 31
3.2 Bounding box criterion. 32
3.3 Distance criterion. 34
3.4 An complete example of the proposed algorithm. 38
3.5 Example of poor contour extraction because of pixel neighbourhood. 39

3.6 Example of inappropriate aggregation of contours. 39

4.1 Architecture of the Internet search module 42
4.2 Example of unsupervised subset selection. 47
4.3 Example of bottle diversity. 48

5.1 Rotation of the GSC histogram. 50

6.1 Ratio of good objects before and after the unsupervised subset selection. 57
6.2 Percentages of good training objects after the subset selection. 58
6.3 Ratio of good objects before and after the unsupervised subset selection. 58
6.4 Percentages of good training objects after the subset selection. 59
6.5 Ratio of good objects before and after the unsupervised subset selection. 60
6.6 Percentages of good training objects after the subset selection. 60
6.7 Performance of the classifiers for 68 homogeneous categories. 64
6.8 Performance of the classifiers for 49 heterogeneous categories. 64
6.9 Performance of the agent, using maximum inverse χ^2 distance membership measure. 65
6.10 Learning efficient of the developed agent. 66
6.11 Performance of the agent, using penalized membership measure. 67
6.12 Learning efficiency of the developed agent. 67

List of Tables

3.1 Example of one aggregation matrix with 5 elements. 35
3.2 Example of one simplified aggregation matrix with 5 elements. 36

4.1 Example of a distance matrix for five objects 43

6.1 Performance analysis of the original and the proposed contour-based object extraction algorithms. 56
6.2 Generic categories from the three editions of SRVC. 57
6.3 The 68 homogeneous categories in the LANG image set. 61
6.4 The 49 heterogeneous categories in the LANG image set. 62
6.5 10-Fold cross validation of classifiers for 68 homogeneous categories. 62
6.6 10-Fold cross validation of classifiers for 49 heterogeneous categories. 63

A.1 Results from the original contour-based object extraction algorithm. 75
A.2 Results from the proposed contour-based object extraction algorithm. 76
A.3 Results from the original subset selection algorithm. 77
A.4 Results from the proposed subset selection algorithm. 77
A.5 Results from the proposed subset selection combined with the contour-based object extraction. 78
A.6 Results from three classifiers for the 68 categories set. 78
A.7 Results from three classifiers for the 49 categories set. 79
A.8 Results from three classifiers for the 49 categories set, with maximum inverse χ^2 distance membership measure. 79
A.9 Results from three classifiers for the 49 categories set, with penalized membership measure. 80

Chapter 1

Introduction

The purpose of this dissertation is the study, development and implementation of a semantic vision agent capable of autonomously learning generic categories from the Internet, using image retrieval web-services. After the training phase, the agent should be capable of robustly and correctly recognizing objects, extracted from a complex scene or provided by a human user.

1.1 Motivation

Humans primarily perceive the world through vision. This is visible for example in human communication, where visual cues are used to transmit ideas.

For a human user to interact effectively with a robot it is necessary that the robot understands the visual cues. Understanding a visual cue can be defined as the process of associating visual information to the words expressed by the user. For example given a category name, the robot should be able to learn how to robustly recognize instances of that category. The ability of building a category representation without human support opens new possibilities for robot vision.

Semantic vision enables efficient robot interaction with the visual world perceived by humans. While this problem was very explored for highly textured categories, for more generic categories it is still an open problem. For this reason the main focus of this dissertation is generic categories segmentation and recognition.

There are two robot competitions where the knowledge developed in this dissertation can be integrated. An overview and the contribution to the competition is describe further.

1.1.1 Semantic Robot Vision Challenge

SRVC[1] is a competition related to semantic vision agents. There are two possible modalities, robot league and software league. Each team, in each modality, receives a list of category names. Instances of these categories will be scattered in a complex environment, and the goal is to detect and classify them correctly. In the first phase, called the learning phase, each agent can connect to the Internet and retrieve images related to the names of the categories, so that they can build representations of the categories.

After this first step, a performance phase is executed, which depends on the modality. In the robot league, the robot can navigate inside the environment taking as many pictures as needed. Then, it tries to detect and recognize instances of the previously learned categories in the pictures.

In the case of the software league, a set of pictures of the environment are provided by the competition organizers. Figure 1.1 shows a detection of an object that was correctly recognized, as belonging to the category "Peperidge Farm Goldfish Baked Snack Crackers", by UA@SRVC agent on SRVC'09 competition.

Figure 1.1: Example of detection of objects in SRVC. In SRVC'09 the UA@SRVC agent correctly detected and recognized the category "Peperidge Farm Goldfish Baked Snack Crackers", as illustrated.

[1] http://www.semantic-robot-vision-challenge.org/

1.1.2 Robocup

As RoboCup Middle Size League [2] (MSL) competition advances, new challenges are proposed to the teams. One of them is the ability of the robots to play with an official FIFA soccer ball. At the beginning of the competition the colour of the ball is announced. Since the field is uniformly green, a simple colour based segmentation algorithm is enough to detect and recognize the ball.

However to detect an airborne ball a colour segmentation algorithm is not enough since the background is not uniform nor known. A new technique that relies on shape and colour is being developed for CAMBADA MSL team [3].

The descriptor presented in section 2.3.2 (Global Shape Context or GSC) was used as a complement to a colour segmentation algorithm for ball detection. A colour segmentation algorithm extracts candidates from the environment that are verified for a circularity measure extracted from the GSC descriptor. A candidate is considered a ball if there exists a high coherence between the colour segmentation information and the descriptor information. Figure 1.2 shows an example of ball detection and recognition.

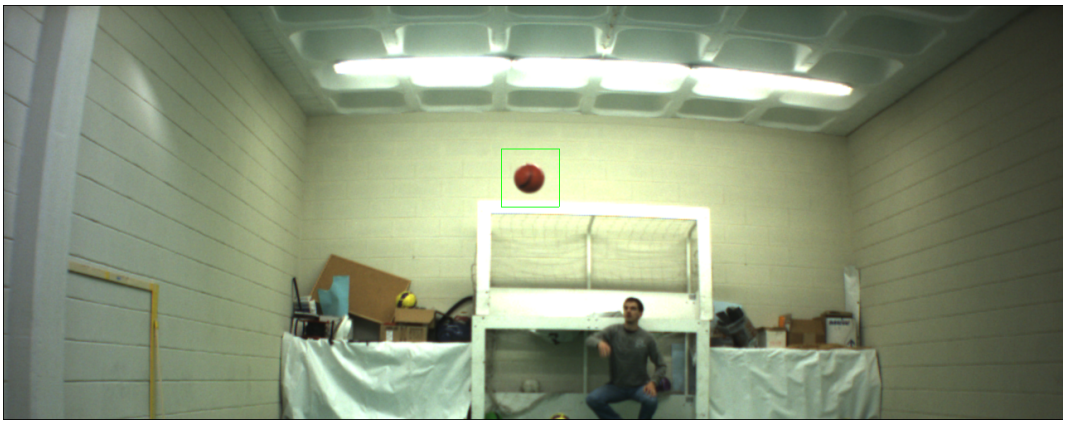


Figure 1.2: Example of an air ball detection and recognition.

For the first time this year, the team CAMBADA@Home [4] participated at Robocup@Home [5] competition. Various challenges proposed in this competition require object recognition and tracking.

For example the "Follow Me" challenge requires that the robot is capable of recognizing one specific person and then tracking that person while the person walks on the scene.

Other example is the "Shopping Mall" challenge that requires that the robot detects and

[2] http://wiki.msl.robocup-federation.org/
[3] http://www.ieeta.pt/atri/cambada/
[4] http://www.livinglab.pt/
[5] http://www.ieeta.pt/atri/cambada/athome/

recognize a set of instances from previous announced categories, this challenge is very similar to the SRVC competition.

Various modules from UA@SRVC and the knowledge acquired with this dissertation are being adopted to compete on these challenges. Figure 1.3 shows CAMBADA@Home robot.

Figure 1.3: CAMBADA@Home robot.

1.2 Research problems

In this section, the main areas of research are briefly introduced. First, an introduction about object recognition, especially for generic categories that are the focus of this dissertation. A generic category is a category that is better defined by the shape of it instances than other type of features.

Second, contour-based algorithms are introduced. These are important because they allow the extraction of shape features used in the learning and recognition phases.

Third, the concept and relevance of unsupervised learning is explained.

1.2.1 Object Recognition

Object detection is the task of, given an input image, being able to select a Region Of Interest (ROI) that is, a region that is likely to contain an object. The detection problem was not addressed in this dissertation. An approach to solve this problem based in color saliency was developed for SRVC'2008 [1].

The focus of this dissertation is on the recognition of instances of generic categories. A generic category is better defined by its shape than by local features. On the other hand, a specific category usually has a high texture saliency and can be more effectively recognized using SIFT[2] or SURF[3] features. For this reason, in this dissertation, a great effort was placed in the shape extraction module. Since the quality of the shape is directly related to

the quality of the recognition process.

Figure 1.4 shows instances of both types of categories.

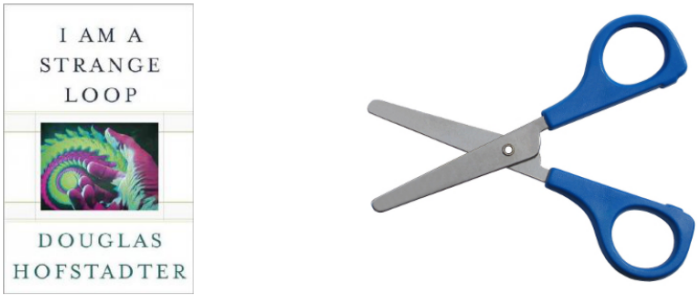


Figure 1.4: A comparison between generic and specific categories. The image on the left contains an instance of a specific category (book "I am a Strange Loop"). The image on the right contains an instance of a generic category (scissor).

Before any type of recognition can take place, it's necessary to acquire knowledge. This step is usually called the training phase and can be of two different types: supervised by a human user, when the human teaches the agent the categories of different instances; or unsupervised learning, where the agent autonomously creates categories with minimal previous information. This dissertation focuses on unsupervised learning.

1.2.2 Contour-based segmentation

The focus of this dissertation is learning and recognizing generic categories. As previously stated, instances of a generic category are better defined by their shape than by other types of features. For this reason, it's necessary algorithms for shape extraction, usually rely on contour-based segmentation.

It's important to distinguish edge detectors from contour-based segmentation. A popular definition in image processing, according to Milan Sonka *et. al.* [4, pp. 21], is that an edge is *"a local property of a pixel and its immediate neighbourhood"* while a border or contour of a region R [4, pp. 22] *"is the set of pixels within the region that have one or more neighbours outside R"*. Intuitively we can define a boundary as a set o pixels with a high edge property.

Edge detectors are a collection of very important local image pre-processing methods used to locate abrupt changes in pixel intensity. An edge detector creates an edge map based only on the pixel intensity and its immediate neighbourhood. It's blind to the coherence of the edge map, which lead to gaps, noisy contours and various other problems.

A contour-based segmentation uses more information than only pixel intensity to extract coherent contours. Usually these algorithms are specific for some type of images and need some type of human interaction in order to get good results. Some of the most relevant techniques

related to this problem will be presented in the following chapter.

Figure 1.5 illustrates the difference between: a edge detection and a contour-based segmentation. Image A is the original image, image B is obtained applying an edge detector. Image C is obtained applying contour-based segmentation. While in image C the plane is correctly segmented in image B the shape of the plane is noisy and there is some clutter.

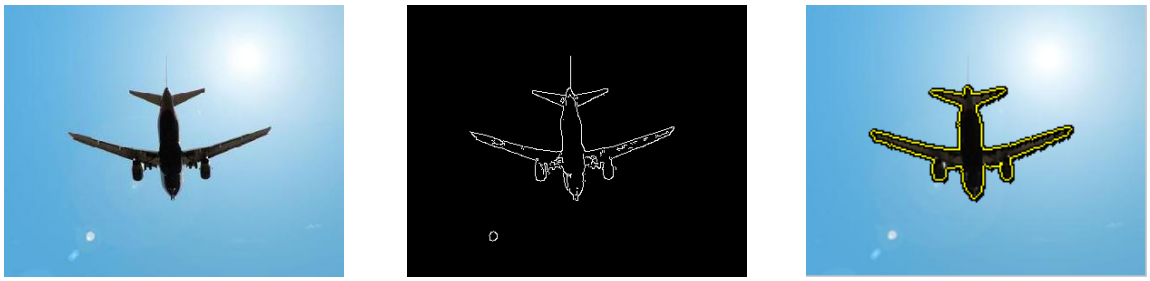


Figure 1.5: A comparison between edge detection and contour segmentation. Image A shows the original image. Image B shows the edge map returned by an edge detector. Image C shows a shape extraction using a contour-based segmentation algorithm.

In the work presented in this dissertation, the only information previously known is the category name. This information is rather insufficient to extract an correct boundary of a object. In the following chapters, a technique to tackle this problem, based in two heuristic rules, is presented.

1.2.3 Semantic image retrieval

As it was previously mentioned, the agent developed in this thesis should have the ability to classify objects with minimal human intervention. Given only the names of the categories, it should be able to construct a model that allows recognition of instances of those categories.

For this to happen the agent needs the ability to determine the meaning of the names of the categories and then retrieve images that are relevant to learning the category. The images can be downloaded though a common web service, for example Google Images[6]. The problem with this approach is that most web services are syntax-based and not semantics-based [5]. That, combined with a exponential growth in the number of images in Internet, makes it not possible to straightforwardly do a web search and use all obtained images to build a category model. The quantity of unrelated images retrieved in a simple search is so high that the model wouldn't be reliable, as illustrated in figure 1.6.

[6]http://www.google.com/imghp

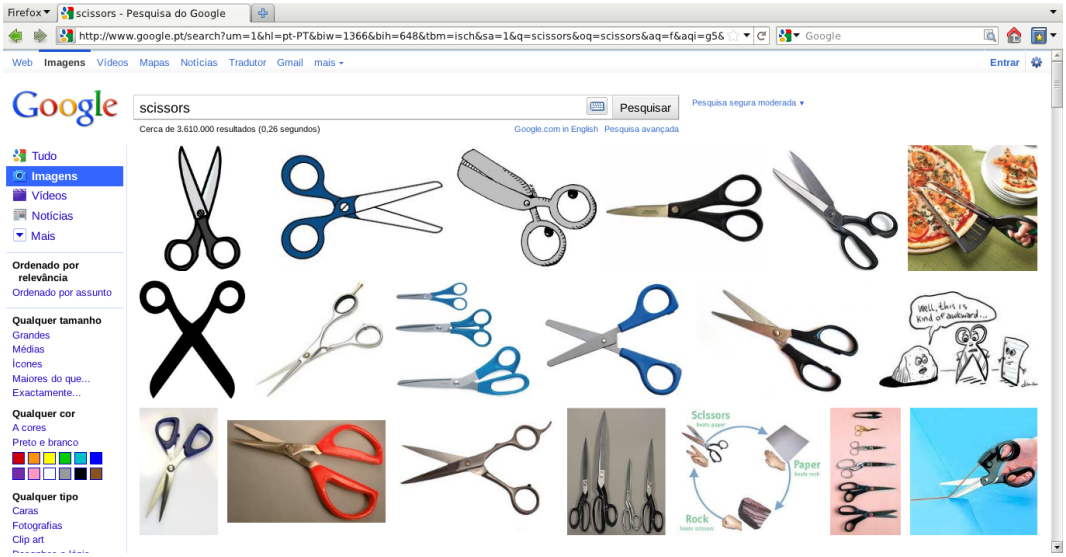


Figure 1.6: Google search for "scissors". Not all results are relevant.

To achieve this goal there are roughly two possible solutions. One of then is to use a semantic image search. However there are no publicly available semantic services. One of the reasons of the success of the Internet is its simplicity. Such complex service is difficult to maintain and the service provided by syntax-based search are sufficient for most uses.

The second solution is to create an abstraction layer on top of a syntax-based web service. This layer will be responsible for verifying the semantic value of the images and the relevance for the category. In this dissertation, unsupervised subset selection is used for selecting image segments that can be used as training images for category learning.

For generic categories, there is another problem in semantic image retrieval. Semantic similarity and visual similarity are different, as is explained by Mei *et. al.*[6]. In Figure 1.7 we can see that the peach in image A and the peach in image B aren't visually similar but they are semantically similar. In contrast the peach in image B is visually similar to the sun in image C, but they are not related semantically.

(a) peach (b) peach (c) sun

Figure 1.7: Semantic similarity $\neq$ visual similarity. Image A and B are semantically similar but not visual similar. Image B and C are visually similar but not semantically similar.

1.3 Objectives

The starting point of this dissertation was the Semantic Robot Vision Challenged (section 1.1.1) competition, that unfortunately is on hiatus now.

The main objectives of this dissertation are:

- Integrate the various modules of the agent UA@SRVC in only two process, one dedicated to the learning phase and another one dedicated to the performance phase.
- Review and improve the functioning of the contour-based segmentation module.
- Review and improve the functioning of the classification module.

The unsupervised subset selection algorithm developed for the UA@SRVC agent was also reviewed and improved, as this inevitably contributes to improving the recognition system.

Initially the work developed in this dissertation was intended to integrate the SRVC agent (UA@SRVC) developed in 2008-2009 [1, 7, 8, 9]. With this in mind the old agent was completely converted to C code All libraries were updated to the last stable versions and the code was cleaned for better performance. Unfortunately in the middle of the development of this dissertation the SRVC competition entered on hiatus. From this point the development of the agent UA@SRVC was suspended. Some of the developments were modified and used by CAMBADA team and LUL team.

1.4 Overview of the agent

As previously stated, the SRVC competition entered on hiatus. Because of this, the development of the UA@SRVC agent stopped. However, in the course of this dissertation, a simplified version of the UA@SRVC agent was build. This smaller agent was developed to test the new modules created and compare them to the modules used by UA@SRVC.

The agent receives a list of category names and using a public web service retrieves several images from the Internet. From each image, the objects in it are extracted. Using an unsupervised technique, that ranks and selects the most relevant objects, a category is created.

After the training phase, the agent is ready to perform. From given images, the present objects are extracted and recognized. Figure 1.8 shows a scheme of the developed agent.

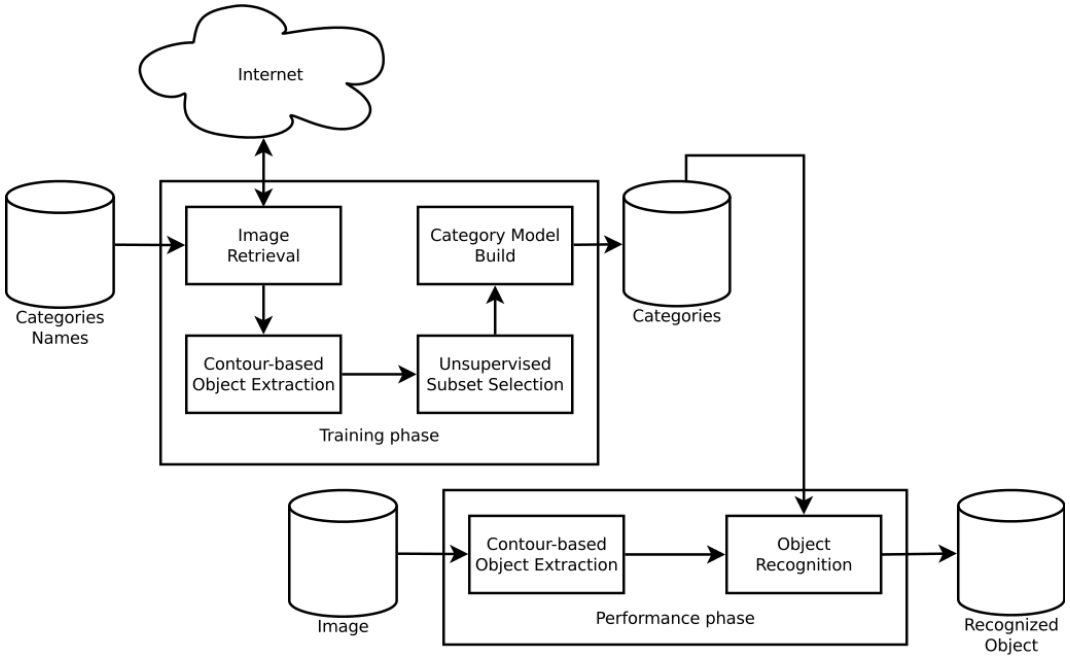


Figure 1.8: Architecture of the developed agent.

1.5 Dissertation structure

In this section we summarize the remaining contents of the dissertation. The dissertation is organized as follows:

Chapter 2 - Related work: In this chapter, some of the most relevant algorithms/techniques related to the work presented in this dissertation are surveyed. The chapter is divided in 3 sections. In the first section, the most relevant contour-based segmentation techniques are presented and described in detail. The following section presents algorithms for semantic image retrieval using syntax-based web services. The last section presents descriptors that explore shape features to recognize objects belonging generic categories.

Chapter 3 - Object extraction and clutter removal: Completely automated extraction of objects from an image is a very hard task. The most successful approaches either require active participation from a human user or some information regarding the context of the object in the image. In the absence of context, the segmentation process can only rely on information derivable from pixel intensity. This chapter describes a methodology to automatically derive the context of the object from pixel intensity values.

Chapter 4 - Category Learning: Recognizing an object consists on extracting a representation from the object and comparing it with representations of instances of known

categories. A category is composed by a set of Global Shape Context descriptors. The process of recognition of recognition normally relies on knowledge acquired on a training phase. I most works, the training phase is completely manual or supervised by a human user. In this chapter an unsupervised category learning phase, that requires only the category names, is presented.

Chapter 5 - Object Recognition: The agent, to be able to recognize new instances from the previously learned categories, needs a classification system. This chapter describes the classification system developed for the agent.

Chapter 6 - Performance Evaluation: This chapter presents an evaluation of the models developed in this dissertation. It will be divided in four sections. First, it is evaluated the object extraction and clutter removal algorithm (Chapter 3) comparing it with the object extraction previously used in the UA@SRVC agent. Second, it is evaluated the performance of the unsupervised object selection algorithm proposed in this dissertation (Chapter 4), comparing it with the algorithm proposed by Pereira *et. al.* [7, 9]. Third, the instance-based classifiers integrated in the agent (Chapter 5) are evaluated on two different tests. The overall performance is tested with a standard k-fold cross validation. Then a test similar to the one proposed by Pereira et al, with an increasing number of categories, is performed allowing direct comparison. Finally a last test is presented, where the new agent will learn categories from the Internet without any kind of human intervention and then tries to recognize a set of objects.

Chapter 7 - Conclusion: In this dissertation, a semantic vision agent was developed and evaluated. This final chapter summarizes the work produced and presents the major conclusions of this dissertation. Ends with the author opinion in which path to follow for improving the agent.

Chapter 2

Related work

In this chapter, some of the most relevant algorithms/techniques related to the work presented in this dissertation are surveyed. The chapter is divided in 3 sections. In the first section, the most relevant contour-based segmentation techniques are presented and described in detail. The following section presents algorithms for semantic image retrieval using syntax-based web services. The last section presents descriptors that explore shape features to recognize objects belonging generic categories.

2.1 Contour-Based Segmentation

Edge detectors are image operators that detect pixels with an abrupt change in intensity. These operators are a very important tool in image processing because the edges of an object can characterize its boundaries. Relying only in the pixel intensity and its neighbourhood, these operator works at a low level.

Other, more complex, contour-based segmentation techniques exist which take into account other type of information in order to produce more coherent object boundaries. In the context of this thesis, these techniques are useful to extract objects from images with multiple objects and also for clutter removal.

The remaining of this section presents several relevant contour-based algorithms.

2.1.1 Canny edge detector

One of the most widely used edge detectors available was proposed by John F. Canny in 1986 [10]. The author proposed three criteria to achieve theoretically optimal edge detection:

Detection criterion: important edges shouldn't be missed.

Localization criterion: the distance between the actual and the located edge should be minimal.

One response criterion: image noise shouldn't be considered edge, and one edge should produce only one response.

The first step to achieve these three criteria is to reduce the effects of noise in the image. This can be achieved with a filter based on the first derivative of a Gaussian curve. This step helps reducing false edges produced by noise in the image. The result of this step is a slightly blurred version of the original image.

An edge is a point where the intensity of the image changes abruptly. These points can be found determining the gradient of the image. The gradient is extracted using filters based on the first derivative of a 2D Gaussian curve in a direction $\mathbf{n}$. The localization of the edge is the local maximum of the image gradient.

The final edges are selected using a threshold with hysteresis. If any edge response is above a certain threshold then those pixels are accepted as edges. On the other hand, if some edge has a very weak response and it is not connected to a strong edge, then that edge is most likely noise in the image. But an edge with a weak response that is connected to an edge with a high response is more likely to be an actual edge than noise in the image.

The complete Canny edge detector algorithm:

1. Convolve an image f with a Gaussian filter of scale σ.
2. Find the location of the edges (non-maximal suppression).
3. Estimate local edge normal directions $\mathbf{n}$ for each pixel in the image.
4. Compute the magnitude of the edge.
5. Threshold edges in the image with hysteresis to eliminate spurious responses.
6. Repeat steps (1) to (5) for ascending values of σ.
7. Using feature synthesis, aggregate the final information about edges at multiple scales σ.

Figure 2.1 illustrates the main steps of the Canny algorithm. Image A is the original image. Applying a filter based on the first derivative of a Gaussian curve produces the slightly blurred image B (algorithm step 1). The gradient operation applied to the image B produces image C (algorithm step 2). The direction of the edges are represented in image D (algorithm step 3). Image E is the edge map without hysteresis. Here it is visible the excess of edges

(algorithm step 4). Finally image F is the final edge map obtained with a threshold operation (algorithm step 7).

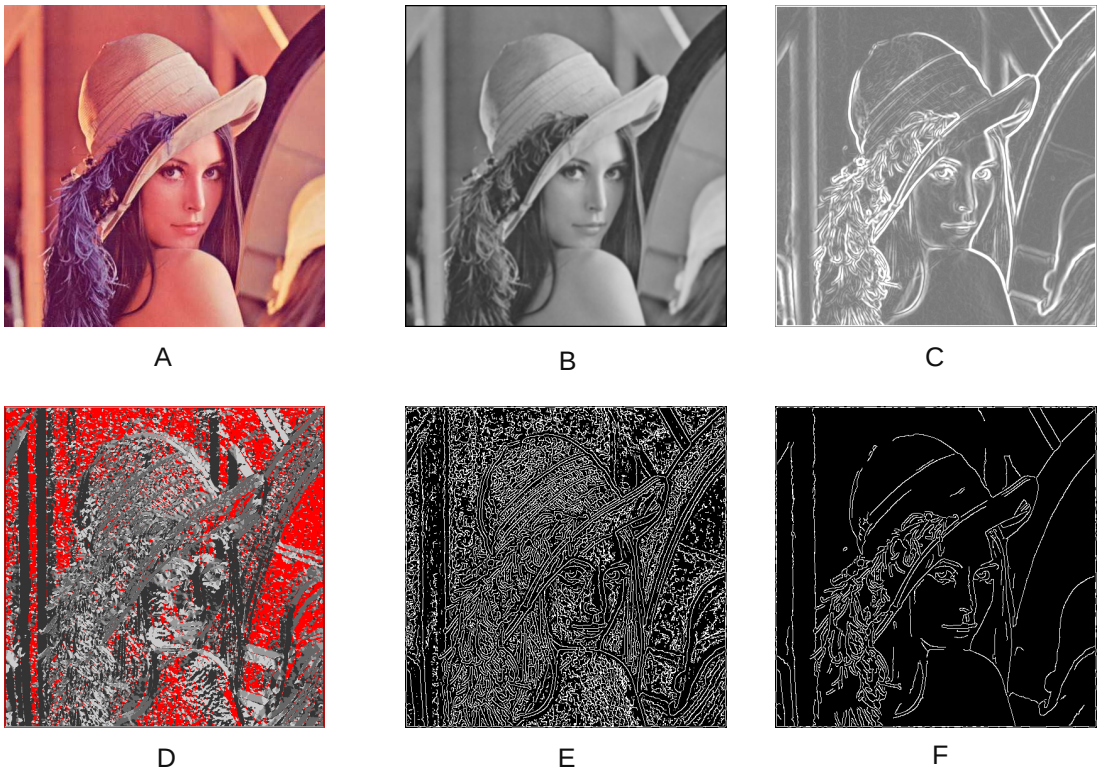


Figure 2.1: Example of application of the Canny edge detector. Image A shows the original image. Image B is obtained with a smoothing operation over image A. Images C and D represent the magnitude and direction of the gradient respectively. Image E represents the edge map without threshold operation. Image F represents the final edge map.

Since Canny edge detector relies only in pixel intensity and it's neighbourhood, it can't, by it self, extract good object boundaries. This method will produce an edge map with all edge points in the image. At the edge level there isn't sufficient information to produce a coherent contour. Nevertheless the edge map can be processed for coherent contour extraction.

2.1.2 Watershed

Watershed is a very well known segmentation technique in mathematical morphology. A gray-level image can be interpreted as a topographic map, where the gray level of a pixel is interpreted as its altitude in the relief.

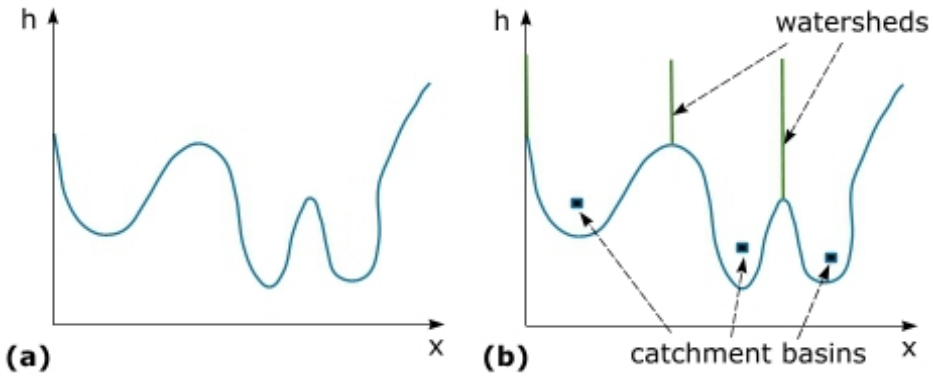


Figure 2.2: Example of one dimension watershed. Image A shows an one dimensional "image". In image B the squares represents water holes, and the vertical lines represents the watersheds.

Figure 2.2 shows a very simple example of a watershed in one dimension. In each catchment basin, there is a "hole" from where water can emerge. If the flows of water from two different catchment basins would merge, a dam is build all the way to the surface. This dam represents the watershed line. Applying this technique for a gray scale image, where the level of gray is interpreted has the altitude, the result is the contour of the object.

Meyer's flooding algorithm is one of the most common Watershed algorithms, introduced in the early 90's [11]:

1. A set of markers, pixels where the flooding shall start, are chosen. Each is given a different label.
2. The neighbouring pixels of each marked area are inserted into a priority queue with a priority level corresponding to the gray level of the pixel.
3. The pixel with the highest priority level is extracted from the priority queue. If the neighbours of the extracted pixel that have already been labelled all have the same label, then the pixel is labelled with their label. All non-marked neighbours that are not yet in the priority queue are put into the priority queue.
4. Redo step 3 until the priority queue is empty.

In the end, the pixels without label are watershed lines and become the final boundaries in the image.

The most important aspect in this method is choosing the correct catchment basins and mark them as a "hole". This process can be manually made by a human which, usually grants good results. However for a complex scene this task consumes much time. Such manual methods is also not suited for autonomous agents. There are methods to automatically select and mark good catchment basins, but usually previous knowledge of the context of the image is needed.

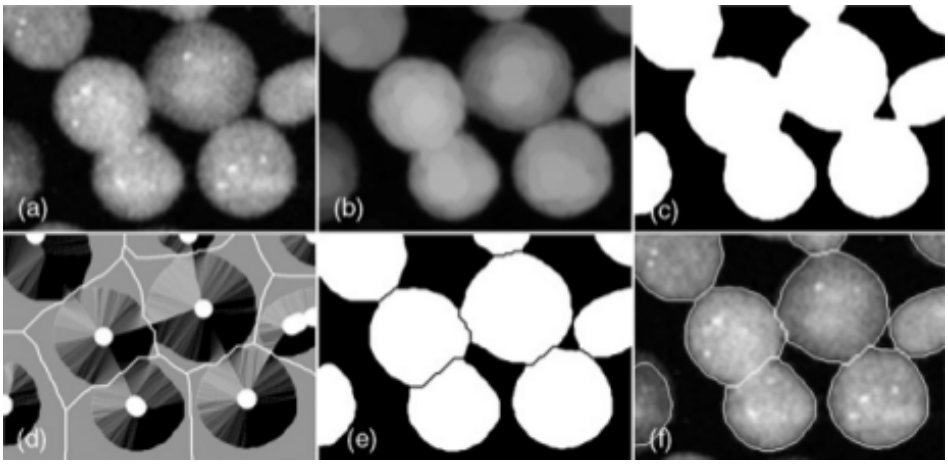


Figure 2.3: Watershed practical example. Image A shows the original image. Applying a smoothing operation on image A, image B is obtained. Image C is the result of a threshold operation applied in B. Computing the distance transform of the complement of image C gives good marker for water holes, these markers are represented in image D as white circles. Image E and F shows the watershed lines super impose over image C and image A respectively.

Figure 2.3 shows a practical example of a watershed segmentation of a group of cells. Based on the type of image, it's possible to developed a technique to automatically find water holes that produce good segmentation. In this example a distance transform operation is applied in image C and the result is used as markers for water holes.

As we can see by the previous example this method requires precise tuning to get good results. The distribution of water holes can be either by hand or, knowing the image type, exploring some property that allows to automatically find good markers. Also it isn't efficient enough to be executed in real time.

2.1.3 Active contour

Active contours, or snakes [12], have become one of the more important object delineation methods in image analysis, particularly in biomedical image analysis. The active contour can be perceived as an elastic band that is placed on the object and will shrink until it delineates the object.

A snake model is defined as an energy-minimizing spline. The snake's energy depends on its shape and location within the image.

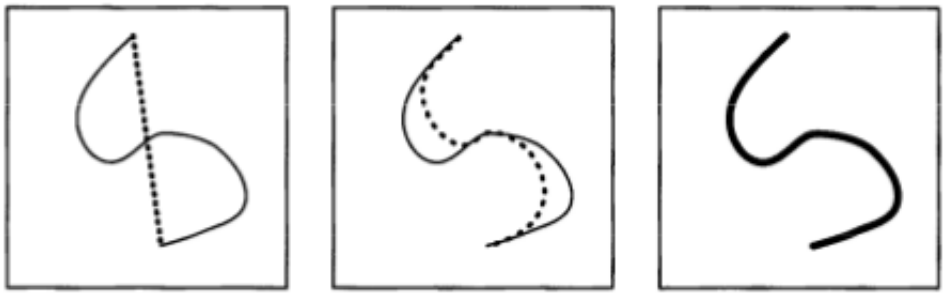


Figure 2.4: Active contour or snake simple example. Image A shows the contour as a continuous line and the snake as a dotted line. Image B shows the snake being attracted to the contour line. Image C shows that the snake completely adopted the shape of the contour.

Figure 2.4 shows a very simple example of a snake. With multiple iteration steps of snake energy minimization, the snake is pulled towards the true contour and finally adopts the contour shape.

The energy function to be minimized may be written as

$$E^*_{snake} = \int_0^1 E_{snake}(v(s)))ds = \int_0^1 (E_{int}(v(s)) + E_{image}(v(s)) + E_{con}(v(s)))ds, \tag{2.1}$$

where E_{int} represents the internal energy of the spline due to bending, E_{image} represents the image forces, E_{con} represents externals constrains and $v(s)$ represents the spline equation.

The internal spline energy can be defined as:

$$E_{int} = \alpha(s)\left|\frac{dv}{ds}\right|^2 + \beta(s)\left|\frac{d^2v}{ds^2}\right|^2, \tag{2.2}$$

where $\alpha(s)$ and $\beta(s)$ specify the elasticity and stiffness of the snake.

The image energy can be defined as:

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term}, \tag{2.3}$$

and it is composed by three terms, each one with its own weight that depends on the type of scene in the image. The first term dictates if the snake will be attracted to bright or dark lines. The second term controls the attraction of the snake towards contours with large image gradients, that is, to locations of strong edges. Finally the last term controls the attraction that line termination and corners have in the snake.

The last term of the equation 2.1 represents external constraints. These constraints can be imposed by either a user or some other higher-level process which forces the snake towards

or away from particular features. The final contour is the curve $v(s)$ that minimizes the value of the equation 2.1.

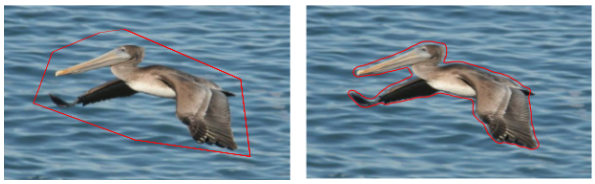


Figure 2.5: Active contour or snake practical example. The first image shows a rough snake outlined by a human user. The second image shows the snake, after multiple iteration, delineating the object.

The implementation of the algorithm uses multiple iterations to approach the final result. Because of this reason, and the need of manually outlining the first snake, this algorithm is not appropriated for object segmentation in autonomous agents.

2.1.4 Pb

Pb (Probability of Boundary) [13] is a boundary detector. Instead of using only pixel intensity, like regular edge detectors, it's constructed from brightness, color and texture cues at multiple scales. For each cue its employed the $Pb_{C,\sigma}(x, y, \theta)$ which estimates the probability of boundary for a given image channel C, scale σ, pixel (x, y) and orientation θ.

The probability of boundary is given by the difference in the image channel between two half of a disk of radius σ centred at (x, y) and oriented at angle θ. Figure 2.6 shows an example of one *Pb*.

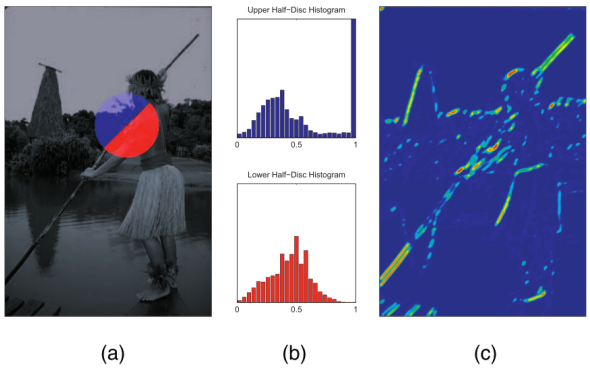


Figure 2.6: *Pb* for one point, scale and direction. Image A shows the computation of a *Pb* for one point, scale and direction. Image B shows the histogram with the value of the disk in image A. Image C shows the *Pb* result for one scale and direction. Contour in the direction of the disk, in image A, are highlighted.

The cues are computed over four channels: the CIELAB 1976 L channel, which measures

brightness, and A, B channels, which measure color, as well as a texture channel derived from texton labels [14]. The cues are also computed over three different scales $[\frac{\sigma}{2}, \sigma, 2\sigma]$ and eight orientations, in the interval $[0, \pi]$. The Pb detector is then constructed as a linear combination of the local cues

$$Pb(x, y, \theta) = \sum_{i=1}^{4} \sum_{j=1}^{3} \alpha_{i,j} Pb_{C_i,\sigma_j}(x, y, \theta), \tag{2.4}$$

where the weights $\alpha_{i,j}$ are learned by training in an image database.

2.1.5 gPb

According to Berkeley Segmentation Dataset[1], the gPb [15] is the most efficient contour detector until this day. It is also the most computationally expensive contour detector presented in this dissertation. For real-time application this method is not viable, at least for now.

The gPb detector consists in two main components:

mPb: a detector based on local image analysis at multiple scales (the Pb detector).

sPb: a detector based on the Normalized Cuts criterion.

The output of the **mPb** (previous Pb detector) is reduced to an affinity matrix W, whose $W_{i,j}$ define the similarity between pixel i and pixel j.

The **sPb** detector uses the affinity matrix W to solve the generalized eigenproblem, following the Normalized Cuts approach [16], to compute the oriented contour signal $sPb_{v_j}(x, y, \theta)$.

The **sPb** detector is defined as follows:

$$sPb(x, y, \theta) = \sum_{j=2}^{k+1} \frac{1}{\sqrt{\lambda_j}} sPb_{v_j}(x, y, \theta), \tag{2.5}$$

and the **gPb** is defined by

$$gPb(x, y, \theta) = \gamma \cdot sPb(x, y, \theta) + \sum_{i=1}^{4} \sum_{j=1}^{3} \beta_{i,j} Pb_{C_i,\sigma_j}(x, y, \theta), \tag{2.6}$$

where γ and $\beta_{i,j}$ are weights also learned by training. To get the final $gPb(x, y)$ signal, just maximize the previous function by θ, threshold to remove pixels with very low probability of being a contour pixel, skeletonize, and then renormalize. Figure 2.7 shows the full structure of the **gPb** detector. Figure 2.8 shows an example of the **gPb** detector in various images.

[1] http://www.eecs.berkeley.edu/

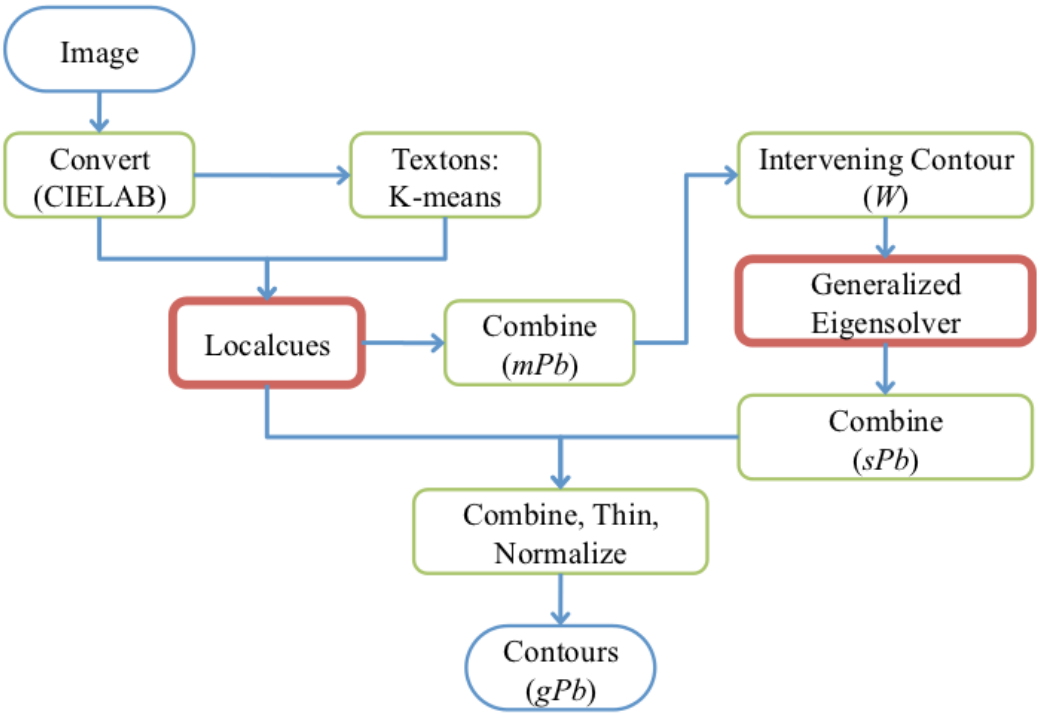


Figure 2.7: Architecture of the gPb detector.

Figure 2.8: Various applications of the gPb detector.

2.2 Image Retrieval and Selection

Before any attempt to recognize objects, it is necessary to learn object categories. Fergus *et. al.* [17] refer that the current paradigm consists of manually collecting a large training set of good examples of the desired object category and then performing a training phase over the set. But there is a plentiful supply of images available at the typing of a single word using Internet image search engines. The majority of the public web-services are syntax-based. This means that there is no meaning extracted from the query. The result is purely based in string matching from the resources label with the query. For this reason the quality of the images

retrieved from Internet will be poor, e. g., the number of objects per image, the position of the objects and scale of the objects is unknown, variable and uncontrolled.

Some researchers [18, 19, 20] developed a technique to learn categories from unlabelled and unsegmented clustered scenes. Each object is modelled as flexible constellations of rigid parts, where parts are very distinctive and stable image patches. These features will be represented with a probabilist representation.

The training stage is performed over a set of images. The source of the images can be the Internet. First the relevant object parts are detected using a clustering algorithm. Then the statistical shape model is learned using expectation maximization. Fergus *et. al.* propose a learning phase with limited supervision. In this approach, the user chooses 10 images from the initial set and the category is learned only from those.

It preforms very well with synthetic testing (up to 90% description of desired objects vs. background images). With images retrieved from Google Images, with an initial percentage of 44.3% of good images, it improves to 58.9% using the completely unsupervised technique and 65.9% with the limited supervision technique.

Fe-Fei *et. al.* [21] developed a method to learn object categories from only a few images [$1 \sim 5$]. It incorporates knowledge obtained from previously learnt models of unrelated categories. In this approach object categories are represented by probabilist models, while "prior" knowledge is represented as a probability density function.

It is proposed by the authors that "The appearance of the categories we know and, more importantly, the variability in their appearance gives us important information on what to expect in a new category.". An object is modelled as a flexible constellations of rigid parts [18, 19]. They use variational Bayesian methods lo learn new unrelated categories with few training examples. With only $1 \sim 5$ training images, the system produces models that, when used to recognize new instances of learned categories, discriminated correctly with an error rate of $8 \sim 22\%$.

Well known techniques used to discover relevant topics in large amounts of text, e. g., Latent Dirichlet Allocation (LDA) and probabilist Latent Semantic Analysis (pLSA), have been adapted for unsupervised category learning [22, 17].

Sivic *et. al.* [22] propose modelling images as vectors quantizing SIFT like features. These vectors are analogies of words and used as input for the previously mentioned techniques. After the process, the most relevant topics (in this case vectors of SIFT features) will be used to describe the category.

Fergus *et. al.* [17] propose that the problem of extracting coherent components from a large corpus of data in an unsupervised manner had many parallels with problems in the field of textual analysis. Taking that into account the authors proposed a extended version of

the pLSA model, developing a new model designated Translation and Scale Invariant pSLA (TSI-pSLA).

The new technique allows to incorporate spatial information in a translation and scale invariant manner, which is not possible with the regular technique. During the training phase, interest regions are detected and described using SIFT features. These vectors of features will be used as words, similar to the previous technique.

Even with a very noisy training set, the presented technique competes with existing methods that require hand-gathered collections of images.

Grauman *et. al.* [23] presented a method that allows automatic unsupervised category learning from unlabelled images.

Each image is decomposed into a set of local descriptors. Those sets are nodes in a graph. Here the weight of the edges corresponds to similarity of some subsets of the two sets of descriptors.

After efficiently computing the pairwise affinities between the input in this space, a spectral clustering technique is used to recover the primary grouping among the images.

Other investigators propose using meta-data to do an initial ranking of the training set of images. These meta-data consist in words nearby the image link in the respective web page.

Berg and Forsyth [24] present a method for identifying images containing categories of animals, relying only on four cues: text, colour, shape and texture.

The training phase consists into two steps. In the first step it is applied, a LDA method to the words, nearby the image link, to discover a set of latent topics for each category. The authors point out that words in images can be ambiguous, and since there is no automatic method to solve the problem (designed polysemy-like phenomenon), it is required human aid to confirm if a latent topic is relevant to that category or not. The images in the training set are ranked according to the words nearby the respective web page.

For the top ranked images, three visual cues are computed. A voting with the three visual cues and the previous word cue is performed. In the end of the process the images with higher number of votes are selected.

For all categories tested, this technique outperforms Google text search in classification.

Schroff *et. al.* [25] developed a multi-modal approach that combines text, meta-data and visual features to gather relevant images from Internet.

It starts by downloading images and the respective web-page from Internet. The retrieved images are ranked using a Bayes estimator trained with the meta-data extracted from the text surrounding the image. No visual features are used in this first step.

In a second step, the top ranked images are used as training data for a SVN classifier. This classifier is then applied to the initial set, removing unrelated images.

The tests proved that this approach performs as well as state of the art systems like [17], while outperforming Google image search.

Vijayanarasimhan *et. al.* present a novel approach to learn discriminative categories from images associated with keywords [26]. Given a list with the names of the categories, the algorithm will provide discriminative models to distinguish them.

Taking into account the noise environment of the Internet, it is proposed an unsupervised method for multiple-instance visual category learning that explicitly acknowledges and accounts for their ambiguity.

For each category name, a set of training images are returned from multiple image search services and using multiple languages queries. The returned training sets of images are treated as positive bags. A positive bag is a bag that contains an unknown number of positive example bigger than zero. On the other hand, a negative bag contains only negative examples. Negative bags are collected from random samples in already labelled sets of categories with different names.

It's assumed by the authors that at least one image retrieved from the Internet contains one category depiction, forming then a positive bag. Each image is represented as a bag of visual words, i. e., a histogram of how many times each of the given number of prototypical local features occur in the image.

The sparse multiple instance learning (sMIL) classifier, through iterative refinements, can discriminate the true positive instances from the negative instances. The tests proved that the system performs as good as state of the art unsupervised approaches.

Pereira *et. al.* [9, 7] present a novel method to train a visual classification system without human interaction, using only web-services to gather images related to the category name. Initially images are searched and gathered using Google Images. As it was previously mentioned, most of the Internet services are syntax-based and not semantics-based. For this reason, several of the gathered images are irrelevant or contain poor representations for that category. Other than poor representations for the category, it is highly probable that each image has more than one object. This factor is more evident for generic categories.

Each image is pre-processed with an object extraction algorithm. The subset selection algorithm does not depend on the object representation or nature. It can be used with all representations if there exists a method that, given two representations returns the distance between them. The pre-processing for object extracting depends much on the representation used and will no be considered at this point.

After segmenting the image and extracting all relevant objects, their respective representations are computed. The objects will be clustered according to their similarity.

It is proposed that the cluster with more elements has a higher probability to better

represent the category. The higher the percentage of good representations in the initial set, the more correct and precise will be the extracted subset. The clustering is done using k-means algorithm. To prevent ties between clusters, after the clustering process a post-processing step guarantees that only one cluster has the highest number of elements assigned.

Taking into account the random nature of k-means and the fact that its performance also depends on the number of clusters multiple runs are used. It starts with $\frac{N}{4}$ clusters and inclemently gets to $\frac{N}{2} - 1$. For each number of clusters, the process is repeated k times to provide a reliable sampling. The total number of runs is then $K \times (\frac{N}{2} - \frac{N}{4})$. The number of times, X_i, an object i appears in the cluster with more elements is updated at each run. At the end, the objects are ranked according to X_i. The final selection is determined by the condition

$$\sum_{s=1}^{S} X_{i(s)} < \eta \times \sum_{i=1}^{N} X_i \tag{2.7}$$

where S is the number of images included in the selection, s is a rank position, $i(s)$ identifies the object in rank position s, and $\eta \in [0, 1]$ is the reject threshold. When the selection process terminates, the remaining $N-S$ objects are assumed noisy/irrelevant and therefore discarded.

2.3 Generic object representation and recognition

Generic categories usually don't have highly textured features that allow robust recognition. The most salient and distinctive feature that allows robust recognition is the shape of the object.

This section describes the most relevant representations based on shape features.

2.3.1 Shape Context

Belongie *et. al.* propose a descriptor based on the shape of the object, the Shape Context [27]. The Shape Context is a property of a point rather than of the whole object. An object is described by a set of Shape Contexts.

For a point p_i a coarse histogram h_i of the relative coordinates of the remaining $n - 1$ points is computed, $h_i(k) = \{q \neq p_i : (q - p_i) \in bin(k)\}$. This histogram is called the Shape Context of the point p_i. The used histogram is log-polar, which means the descriptor is more sensitive to positions nearby the sampling point. An example of this process is illustrated in figure 2.9.

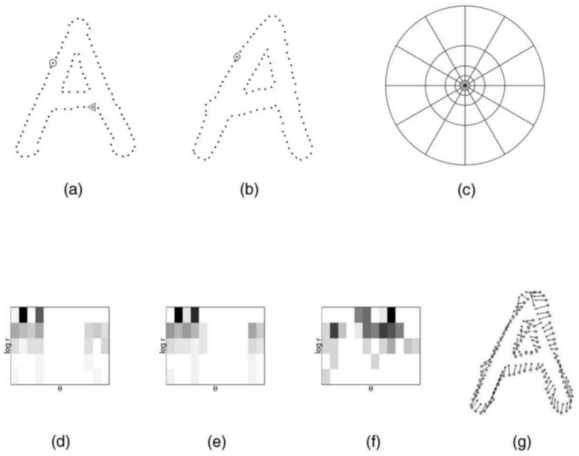


Figure 2.9: Shape Context and matching. Images a and b represent sampled points used to extract Shape Contexts. Image c represents the log-polar grid used to compute the each Shape Context. Images d, e and f represent Shape Contexts extracted form the marked points in image a and b. Image g represents the correspondences of points between images a and b.

It is not necessary to compute Shape Contexts for all points. The points will be sampled and the shape context will be computed just for a subset of all the edge points that define the shape of the object. Intuitively, since the histogram of a point describes all the other points, Shape Contexts for all points will lead to an over detailed representation. The computational advantages of using a subset of points are also evident.

Matching two objects is the process of finding, for each sampled point p_i on the first object, the best matching point q_i on the second object. Let $C_{ij} = C(p_i, q_j)$ denote the cost of matching the point p_i with the point q_j. Given the set of costs C_{ij} between all pairs of points p_i on the first shape and q_j on the second shape, matching the two shapes is given by minimising $H(\pi) = \sum_i C(p_i, q_{\pi(i)})$ subject to the constraint that the matching be one-to-one, i.e, π is a permutation. This is an instance of the weighted bipartite matching problem,which can be solved in $O(N^3)$ time using the Hungarian method [28].

Matching two histograms is achieved using χ^2 distance as follows

$$C_{ij} = \frac{1}{2}\sum_{k=1}^{K}\frac{[h_i(k) - h_j(k)]^2}{[h_i(k) + h_j(k)]} \tag{2.8}$$

where $h_i(k)$ and $h_j(k)$ denote the k^{th} bin of the normalized histogram at p_i and p_j.

2.3.2 Global Shape Context (GSC)

Pereira *et. al.* [7, 9, 8] present the Global Shape Context, a descriptor comparable to the Shape Context. This approach has some advantages. Since only one descriptor is used to

characterize all the shape of the object, the functions to extract the descriptors and match two descriptors are computationally more efficient, which is ideal for real-time processing. On the negative side, because it is a global descriptor, it cannot handle as well as Shape Context or other local descriptor, partial occlusion and differences in perspective.

In the conventional Shape Context, descriptors are computed for a group of sampled points. In the GSC, the geometric center is computed and a single descriptor is computed for this point. After the segmentation of the object, it is used a flood filling algorithm to color the interior of the object. Then the geometric center is defined as $GCM = \frac{1}{N}\sum_{i}^{N}(x_i, y_i)$, where (x_i, y_i) are the coordinates of the interior points. It is not trivial to compute the geometric center since it is difficult to define the interior of a noisy shape. This problem is tackled by the flood filling algorithm. Since the GSC is a global descriptor, the grid used is a polar grid, not the log-polar grid used for the standard Shape Context, as seen on Figure 2.10.

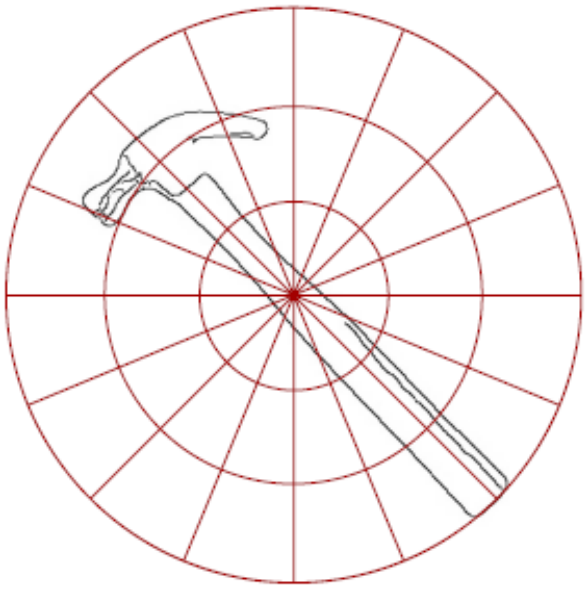


Figure 2.10: Shape Context as a global descriptor. Now the Shape Context is not referring to a single point in an object, but to the object itself.

Scale invariance is obtained by normalizing the histogram, so each cell represents the ratio of shape points in that cell and not the absolute value. Rotation invariance is obtained during matching by rotating the histogram as many times as angle bins.

Since, for each object, there is only one descriptor, the similarity between two global shape representations is $S(c) = \frac{1}{c}$, where c is the minimum χ^2 distance between them. The distance is computed angle bins times, the number of times that is required to rotate the histogram.

2.3.3 Roy's Shape Representation (RSR)

Deb Kumar Roy presents a representation of objects [29], that uses the tangents to the borders of the object. Given an edge map of an object, that can be obtained using any edge detector, and for each pair of edge points, the euclidean distance d and the angle δ between the respective tangents are computed as illustrated in figure 2.11.

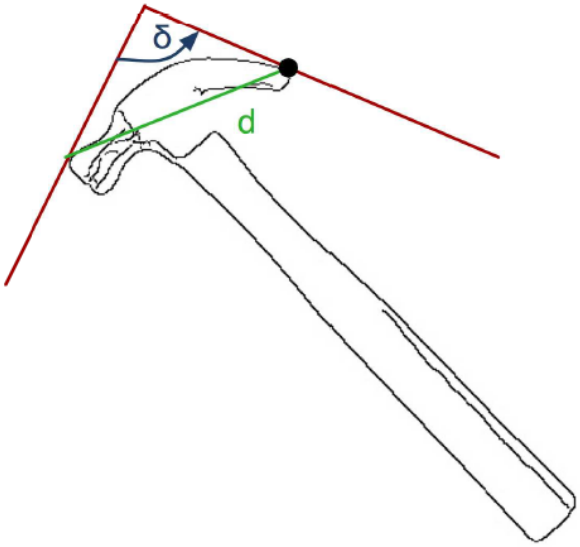


Figure 2.11: Roy's Shape Representation.

After the calculations of the euclidean distance d and angles δ between all pairs of edge points, a two dimensional histogram is created. This representation is invariant to rotation, since the angle between tangents is a relative measure. It is also invariant to scale if the distance between edge points are normalized relatively to the greater distance found.

The distance between objects can be obtained using χ^2 distance (equation 2.8) similar to the approaches used in Shape Context and GSC.

2.3.4 Object Recognition Using Junctions

Wang *et. al.* [30] present a new set of shape-driven features based on corner points or junctions. Unlike the representations previously presented, this method needs a robust corner detector. Instead of the standard Harris Corner Detector, the detector of this paper is recommended [31].

While in the Shape Context (section 2.3.1), the sampled points are just any edge points, this method only uses well defined junctions to produce the representation. A contour C with n junction points can be described as $C = (J_1, J_2, ..., J_n)$, where J_i is the i^{th} junction in the contour C. Two types of features F_1 and F_2 are presented. For every junction J_i, a

feature $F_1(J_i)$ is computed based on its connected contour segments using a shape context like approach. Unlike the Shape Context, only the points in the contour segment are considered.

The contour segments $e_{i-1,i}$ and $e_{i,i+1}$ between J_{i-1} and J_i, and J_i and J_{i+1}, respectively are the path to J_i, denoted as $P(J_i)$. The path $P(J_i)$ is used to characterize the junctions J_i as represented in Figure 2.12. For each sampled point p_t it is computed a feature $h(p_t)$ based on 50 densely sampled points on path $P(J_i)$. This feature is the same type of histogram used in Shape Context 2.3.1. So the feature for junction J_i is described as $F_1(J_i) = (h(p_1^{(i)})), ..., (h(p_{10}^{(i)}))^T$.

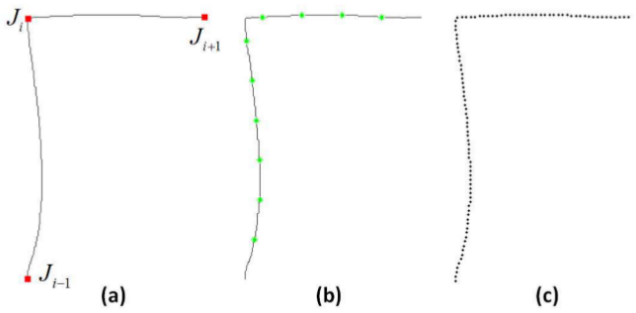


Figure 2.12: First junction feature. This feature characterizes the shape information of a junction.

Another feature is extracted, which characterizes the shape information of the contour segment $e_{i,i+1}$ using the same approach as the first feature. An example of this features is given in Figure 2.13. 10 points are sampled at equal space in shape contour $e_{i,i+1}$, $(p_1^{(i,i+1)}, ..., p_{10}^{(i,i+1)})$. For each sample point $p_t^{i,i+1}$ the shape context is computed based on 50 equally space points between $e_{i-1,i}$ and $e_{i+1,i+2}$. The final feature to define a contour segment is $F_2(e_{i,i+1}) = (h(p_1^{(i,i+1)}), ..., h(p_{10}^{(i,i+1)}))^T$.

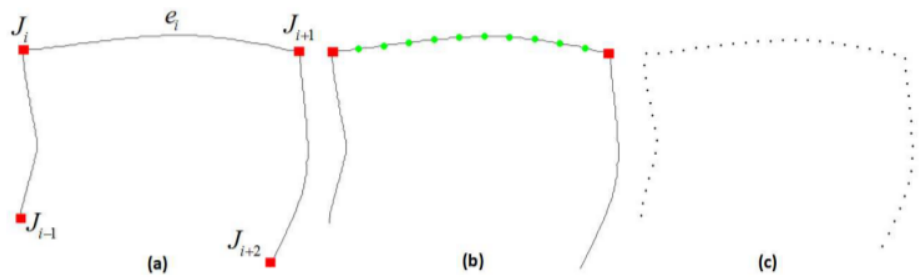


Figure 2.13: Second junction feature. This feature characterizes the shape information of a contour segment.

The matching is a two step process. First the $F1$ feature is used to classify all junctions in the object using a K-Nearest Neighbour scheme. It is used the same distance measure as

in Shape Context, equation 2.8. Based on the results of the kNN classifier, a set of groups of junctions are created G'_i. A graph model is created with the group of junction as vertices. The weight of each edge is computed with dissimilarity provided by the feature $F2$. A shortest path algorithm is used to localize the object, based on the previous weights. The sum of all edge weights, on the shortest path, represents the confidence in the classification.

Chapter 3

Object extraction and clutter removal

Completely automated extraction of objects from an image is a very hard task. The most successful approaches either require active participation from a human user or some information regarding the context of the object in the image. In the absence of context, the segmentation process can only rely on information derivable from pixel intensity. This chapter describes a methodology to automatically derive the context of the object from pixel intensity values. More specifically, a high level processing layer has been designed that works on the edge-based counterpart of the original image. This layer firstly identifies groups of edges as parts of object candidates. A validation and aggregation stage is required for reliable object segmentation. Therefore, in the second processing stage, an algorithm has been designed to coherently aggregate the group of edges that belong to the same real object. but have been identified as separate objects.

These functionalities will be used for two main purposes. On one hand, they will be used to segment and extract objects from images with multiple objects, as those obtained from the Internet. On other hand they will be used to discard clutter from images expected to contain a single object. The source of the images will either be the Internet or an object detector.

Images retrieved from Internet,by searching for some category name, will often contain more than one object and the image background will not be very complex. The images provided by the user for classification and the images processed by our object detector have a high probability of containing only one object, but a much more complex background.

The extracted objects will then be described by shape features. For this reason, it is important to preserve the contours that lie inside the object boundary. These inner contours, in combination with the object boundary, give a detailed description of the object's shape. This extra information about the shape can lead to better object recognition.

The proposed high-level layer has been designed to be robust against background noise, support extraction of multiple object and preserve the relevant details of the shape of an

object.

3.1 From edges to contours

The first step is to aggregate the edge pixels of an edge map (result of running Canny edge detector [10] on the original image) into rudimentary object candidates based only on pixel neighbourhood connectivity.

Classic neighbourhood border tracing methods, such as Moore's neighbour tracing algorithm [4, pp. 192] and Theo Pavlidis' algorithm [32], are not suited for this task. In fact, while searching for the neighbourhood connectivity, they only look at 8 neighbouring pixels, or even less. In the cases where edge maps have been poorly extracted and the edges of the contour have gaps greater than 1 pixel, the resulting contour extraction will be poor. In the current work, it is highly likely that the edge map will be noisy, with gaps and various other imperfections.

To tackle this problem, a more relaxed boundary tracing algorithm has been developed (Algorithm 1). This algorithm is based on Moore's neighbour tracing algorithm [4, pp. 192], with the following two differences:

1. For each identified edge, the algorithm begins with a search window of size 3×3, which corresponds to a neighbourhood of 8 pixels. If no unexplored edge is found, the size of the search window is increased by 2 (the search windows needs to be odd square matrices). This process is repeated until an unexplored edge is found or the maximum allowed size (11×11 in implementation of the algorithm) of the search window has been reached. In contrast, the regular Moore's neighbour tracing algorithm uses a fixed search window of size 3×3.

2. All the edges found in a search window are added to a stack and not just the first edge. This guarantees the preservation of contours that lies inside the border. In contrast, the regular Moore's neighbour tracing algorithm that extract only the external border of one object. As previously stated, the shape of one object benefits from the fine details that the inner contours provide.

The edge map is swept from the top until an edge point is found. Then this edge point is used as a starting point for the proposed contour tracing algorithm. Figure 3.1 provides an illustration of the growth of the search window around the currently explored edge. The search window grows until a new unexplored edge is found and added.

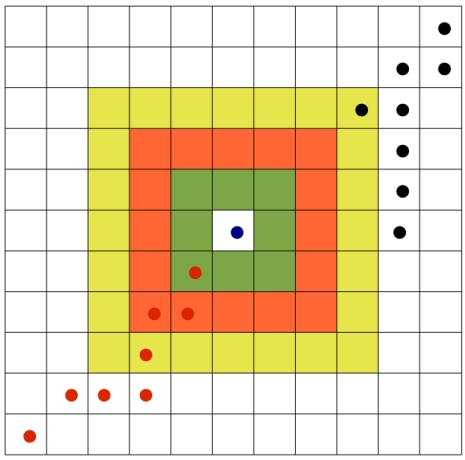


Figure 3.1: Proposed algorithm for border tracing. Edges are represented by circles. Red circles represents explored edges, blue circle represents the current explored edge, black circles represents unexplored edges. Around the blue edge the search windows are represented.

A simplified version of the algorithm is given in Algorithm 1.

Algorithm 1 Proposed contour tracing algorithm

```
Input
start - the edge where the algorithm starts

Variables
stack - stack with the visited edge points with unexplored neighbourhood, initially empty;
contour - set of visibly edges, initially empty;

stack.push(start)
contour.add(start)
while not stack.empty( do
  currentPoint = stack.pop()
  found ← false
  for wSize = 3; wSize ≤ maximumWSize && found = false; wSize ← wSize + 2 do
    if unexplored point found in the current window size then
      stack.push(unexploredPoint)
      contour.add(unexploredPoint)
      found ← true
    end if
  end for
end while
return currentContour
```

3.2 Contour aggregation metrics and criteria

The second step in the segmentation process is to aggregate the detected contour segments.

The previous object extractor, developed for SRVC'08 [1, 9], used only a distance criterion to decide if two contour segments should be aggregated or not. The new proposed method uses three items of information to represent a contour segment: the geometric center, the color histogram of the respective contour region, and the minimal bounding box. This information is required for applying the rules for contour aggregation. Following two rules have been proposed for identifying contours that can be aggregated:

1. If the bounding box of one contour fully contains the bounding box of another contour, these two are merged together.

2. If the geometric centres of two contours are close enough, then the colour histograms of the respective regions are compared. If the difference between the histograms is less than a certain threshold, the candidates are aggregated.

In the following three subsections these rules will be explained in details.

3.2.1 Bounding box criterion

It is a reasonable assumption that if a contour is completely inside another contour, these contours are part of the same object (see Figure 3.4 for an example). To identify whether a contour is inside another contour is to check if all the points of one of the contours lie inside the other. However, this approach can be computationally expensive. Therefore, a more efficient and computationally inexpensive approach was devised. This approach is based on the bounding box criterion, where instead of finding out if one contour is inside another, it is found if the bounding box of one contour fully contains the bounding box of another contour.

Figure 3.2 illustrates the bounding box criterion.

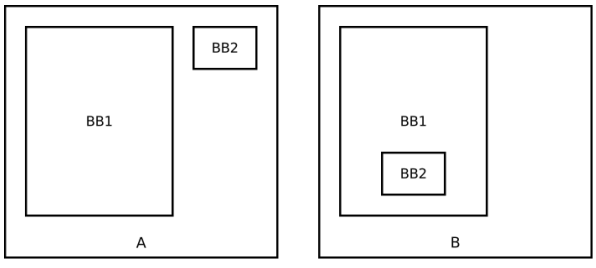


Figure 3.2: Bounding box criterion. Image A shows two bounding boxes that doesn't satisfy the criterion. Image B shows two bounding boxes that satisfy the criterion.

Given the points that form the contour, the minimal bounding box can be computed with Algorithm 2:

Algorithm 2 Minimal bounding box

```
Input
contour - contour points

Variables
upLeftPoint - the upper left point of the bounding box
downRightPoint - the downer right point of the bounding box
rect - the bounding box

upLeftPoint ← contour.points[0]
downRightPoint ← contour.points[0]
for i ← 1; i < contour.nPoints; i ← i + 1 do
  if upLeftPoint.x < contour.points[i].x then
    upLeftPoint.x ← contour.points[i].x
  end if
  if upLeftPoint.y < contour.points[i].y then
    upLeftPoint.y ← contour.points[i].y
  end if
  if downRightPoint.x > contour.points[i].x then
    downRightPoint.x ← contour.points[i].x
  end if
  if downRightPoint.y > contour.points[i].y then
    downRightPoint.y ← contour.points[i].y
  end if
end for
return rect(upLeftPoint,downRightPoint)
```

3.2.2 Distance

The geometric center distance criterion was the only criterion used on the previous object extractor. Each pair of contours that matched this criterion were aggregated. In the new algorithm, if this criterion is matched, it is still needed a match in the colour criterion to allow an aggregation.

A flood fill algorithm is used to fill the contour. After this process, the geometric center of the object, GC, is found as:

$$GC = \frac{1}{N}\sum_{i}^{N}(x_i, y_i) \tag{3.1}$$

where N is the number of points inside the contour, and (x_i, y_i) are the positions of the points.

After computing the geometric center for each contour, it is computed the euclidean distance for all the pairs of geometric centres. For two geometric centres GC_a and a GC_b, the

euclidean distance D is given by:

$$D = \sqrt{(GC_a(x) - GC_b(x))^2 + (GC_a(y) - GC_b(y))^2} \tag{3.2}$$

If the euclidean distance D is smaller than the sum of average distances of the contour edges to their own geometric centers multiplied by a constant factor k (equation 3.3), then the two contour candidates will be aggregated. The algorithm was implemented with $k = 1$.

$$D < k \times (ACD_a + ACD_b) \tag{3.3}$$

Figure 3.3 explains the distance criterion. The image to the left does not match the criterion because the euclidean distance D is bigger than the average distance of the contour points to their own center $(ACD_a + ACD_b)$. In contrast, the distance D is smaller than the average distance of the contour points to their own center $(ACD_a + ACD_b)$ in the image on the right, which matches the criterion.

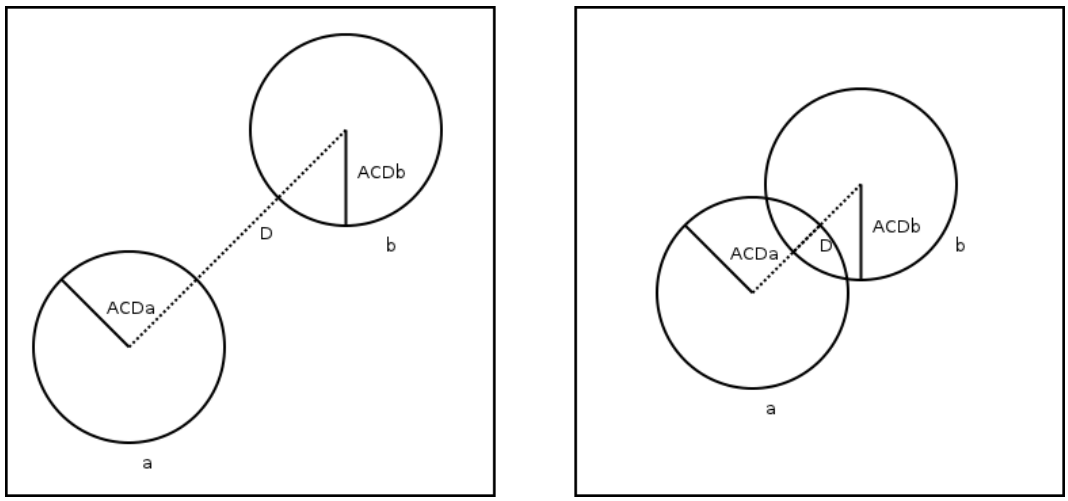


Figure 3.3: Distance criterion. The image on the left doesn't match the distance criteria, while the image on the right matches.

3.2.3 Colour Disparity

For the area inside each contour, a 3D histogram with colour information is extracted. Each dimension stores information of one color channel (the colour space used is RGB). Using the contour edges as delimiters, a color histogram of the object candidate is computed.

The disparity between two colour histograms is computed using the χ^2 distance (equation

3.4). This metric is widely used to compare histograms (section 2.3).

$$D_{ij} = \frac{1}{2} \sum_{r=1,g=1,b=1}^{R,G,B} \frac{[h_i(r,g,b) - h_j(r,g,b)]^2}{h_i(r,g,b) + h_j(r,g,b)}, \tag{3.4}$$

Applying this concept to the current problem, D_{ij} can be interpreted as the disparity in colour between two contour segments. A color disparity value under a set threshold (0.35 in current implementation) identifies whether contour segments can be merged or not.

3.3 Contour aggregation algorithm

The previous object extraction algorithm aggregated the contours with an iterative process that stopped when no more contour satisfied the distance criterion (see Section 3.2.2). This technique leads to overgrowing of the final contours, because after an aggregation two contour will result in a new contour with different properties. These intermediate aggregates influence the results of the remaining contours. In the new proposed algorithm, a square matrix with the same size as the number of available contours is created. For each row (or column), each cell in the column (or row) identifies whether the corresponding contours will be aggregated or not. An example of an aggregation matrix is show in Table 3.1.

Contours	A	B	C	D	E
A	-	Match	-	-	-
B	Match	-	Match	-	-
C	-	Match	-	-	-
D	-	-	-	-	Match
E	-	-	-	Match	-

Table 3.1: Example of one aggregation matrix with 5 elements. Contours A and B, B and C, and D and E are marked to be aggregated with each other.

After comparing each pair of contours and filling the aggregation matrix, the matrix is reduced to allow a single step aggregation. Each of the final clusters is aggregated in just one step, and not iterative as in the previous object extractor. Because of this the aggregation matrix is reduced to the minimal non redundant number of aggregations needed to obtain the final contours. For example, in Table 3.1, contours A and B, B and C, and D and E follow the proposed rules. Computationally, aggregating candidates A and B, and then aggregating B and C is the same as aggregating A and B and then the resulting contour AB and C. Table 3.2 shows the reduced aggregation matrix derived from the matrix in Table 3.1. Here A, B and C, and D and E will be aggregated in a single aggregation step.

Contours	A	B	C	D	E
A	-	Match	Match	-	-
B	-	-	-	-	-
C	-	-	-	-	-
D	-	-	-	-	Match
E	-	-	-	-	-

Table 3.2: Example of one simplified aggregation matrix with 5 elements. Contour A, B and C, and D and E are marked to be aggregated with each other.

The algorithm used to reduce the aggregation matrix could, at the same time, aggregate the respective contours. However, one of the objectives of this dissertation was the development of generic modules that can be easily reused. Taking this into account, the reducing matrix function was separated from the aggregation function. Allowing the use of this aggregation technique in other components.

This technique provides two main advantages. First, the contours are compared in equal terms and only the properties of the individual contours are taken into account for the aggregation process. Second, there is no need to recalculate properties that are used only for the aggregation process for the new contours, since they are formed in a single step. That is, the single-step aggregation approach avoids the problem of overgrowing intermediate object candidates.

3.4 Clutter Removal

After aggregating the selected pairs of contours, a process of selection is needed. There is no guarantee that all the aggregated contours indeed correspond to objects. As mentioned earlier, very little information is available about the context of the image. Therefore, no external information is available to separate good candidates from the poor ones. This section discusses the removal of poor object candidates.

The removal of noise is based on the hypothesis that objects inside an image will cover greater area than noise or clutter. Using this hypothesis, the candidates are sorted by area and all candidates with an area greater than a determined threshold are considered good object candidates and returned.

To resolve the problem of finding the optimal threshold, an automatic thresholding method [33, pp. 83] is used. The algorithm was adapted for a more generic scenario. For a given population of candidates that has been sorted on the basis of a score (area, in this case), the algorithm iteratively computes the optimal score. Based on this optimal score, the noisy contours are separated from the relevant ones.

Using this algorithm allows our detector to have a dynamic threshold for contour selection. This technique has been shown to work very well on bimodal histograms [4, pp. 180]. This algorithm tries to aggregate the object candidates such that in the end of the process they can be divided into two very distinct groups (clutter against the relevant contours).

The adapted version of the algorithm is the following Algorithm 3:

Algorithm 3 Automatic thresholding

```
Input
elements - set of elements that are ranked by a score

Variables
initialThreshold - the initial threshold
finalThreshold - refined threshold computed after multiple iterations
avgG1 - average score of elements above initial threshold
avgG2 - average score of elements bellow initial threshold
nG1 - number of elements with score above initial threshold
nG2 - number of elements with score bellow initial threshold

initialThreshold ← average(elements)
finalThreshold ← 0.0
done ← false
while not done do
  avgG1 ← 0, nG1 ← 0
  avgG2 ← 0, nG2 ← 0
  for i ← 0; i < elements.size; i++ do
    if elements[i].score > initialThreshold then
      avgG1 ← elements[i].score + avgG1
      nG1++
    else
      avgG2 ← elements[i].score + avgG2
      nG2++
    end if
  end for
  avgG1 ← avgG1 / nG1
  avgG2 ← avgG2 / nG2
  finalThreshold ← (avgG1 + avgG2) / 2.0
  if initialThreshold = finalThreshold then
    done ← true
  else
    initialThreshold ← finalThreshold
  end if
end while
return finalThreshold
```

3.5 Application example

In this section a detailed example of the proposed object extraction approach is presented. Later, two examples that express the limitation of the approach are also presented and explained.

In figure 3.4 the principal steps of the algorithm are visually represented. The process begins with an input image, from which an edge map is computed by applying Canny edge detector. From the edge map, contour candidates are extracted (the bounding boxes are represented by a red rectangle and the geometric centres are represented by a blue dot). Following the proposed aggregation rules described earlier, the object contours are aggregated. At the end of the aggregation step, the "bottle" object gains more detail. Finally, computing the optimal threshold based on the area of the contours, the clutter is discarded and only the bottle shape is returned.

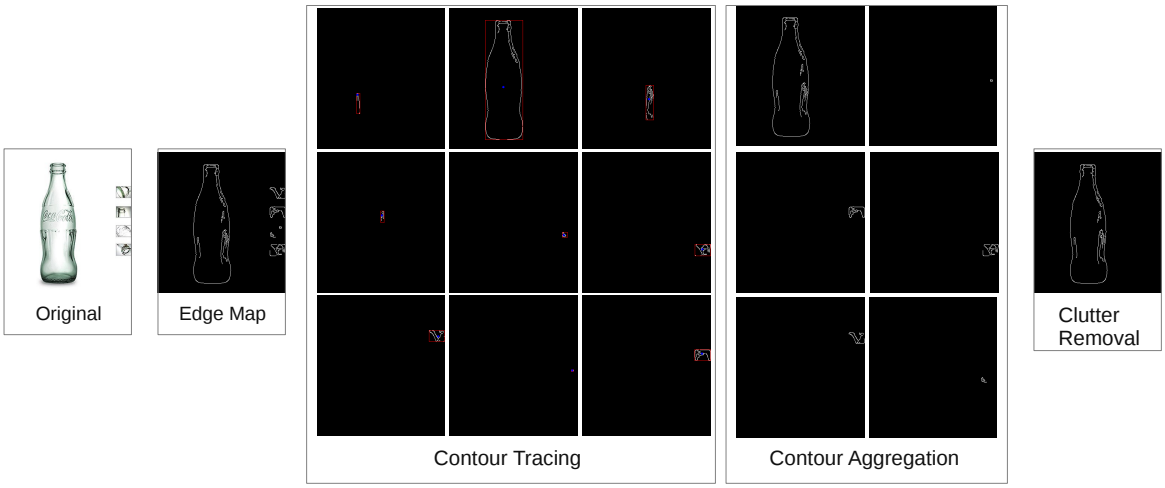


Figure 3.4: An complete example of the proposed algorithm.

Although performing better than the previous algorithm the new approach still has some limitations. The first limitation is related to the contour tracing algorithm and is illustrated in Figure 3.5. In this figure, the last two bananas are so close that the contour tracing algorithm considers them part of a single contour. This situation can be controlled by modifying the maximum allowed size of the search window in the very first step. A small window size will produce more contours, which in turn will require more processing time. On the other hand, a big window size will produce less contours which can lead to aggregation of real objects.

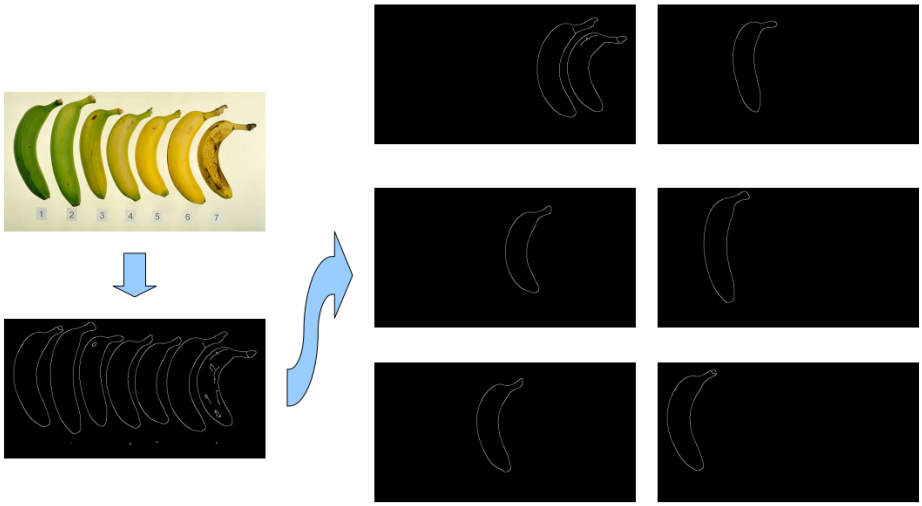


Figure 3.5: Example of poor contour extraction because of pixel neighbourhood.

The second limitation is related to using colour as a distinctive feature. Figure 3.6 provides one example. The two bottles in the figure have similar colour histograms. Since their geometric center is close enough to be aggregated, the algorithm returns only one object (formed by both bottles). It is possible to control the threshold of colour disparity. With a high threshold, contours with higher colour disparity will aggregate, whereas, a lower threshold is more selective in the process of aggregation.

Taking in account the possible combinations of colours in objects it is not possible to define an optimal threshold without previous information. Without previous information, a static threshold is defined based in the results of the contour segmentation tests.

Figure 3.6: Example of inappropriate aggregation of contours.

In absence of context, segmentation of objects becomes a very hard problem. For this reason, it is very difficult to tune the algorithm to optimal values that produce good contour extraction. But, on an average, this approach produces interesting results when compared with other similar approaches (without compromising the computational performance).

Chapter 4

Category learning

Recognizing an object consists on extracting a representation from the object and comparing it with representations of instances of known categories. A category is composed by a set of Global Shape Context descriptors. The process of recognition normally relies on knowledge acquired on a training phase. In most works, the training phase is completely manual or supervised by a human user. In this chapter an unsupervised category learning phase, that requires only the category names, is presented.

The technique presented here is an improvement over the algorithm presented by Pereira *et. al* [9, 7] (see Section 2.2). Given a list of category names, the algorithm begins by retrieving images from a public web-service. As previously stated, the web-services are syntax-based, so the images retrieved are mostly noisy. Other important factor is that the images returned are not ranked according to their visual relevance. For this reason, a ranking and selection algorithm is necessary to create a subset of relevant objects (extracted from Internet images), which can be used to create a faithful category representation.

This chapter is divided into four sections. In the first section, the module developed to search and retrieve images from the Internet is presented. The next section summarizes the object extraction and the representation method. Later, the unsupervised object relevance evaluation algorithm is described and the final section details the object selection algorithm.

4.1 Internet-based image retrieval

The approach previously used in UA@SRVC was based on a *Perl* script provided by the SRVC organizers[1]. One of the objectives of this dissertation was rebuild the UA@SRVC agent to reduce external library dependencies and improve performance. Because this script depends on *Perl* libraries and was not very efficient, in the context of this dissertation, a new module

[1] http://search.cpan.org/ grousse/WWW-Google-Images-0.6.4/lib/WWW/Google/Images.pm

in C programming language, using the cURL[2] library, was developed. This module has two main components, as illustrated in figure 4.1:

Web-service Query: given a category name this module queries a web-service, processes the response and extracts the image links using regular expressions;

File Download: given a link to a file, create a connection and either retrieve the file or return a "time out error".

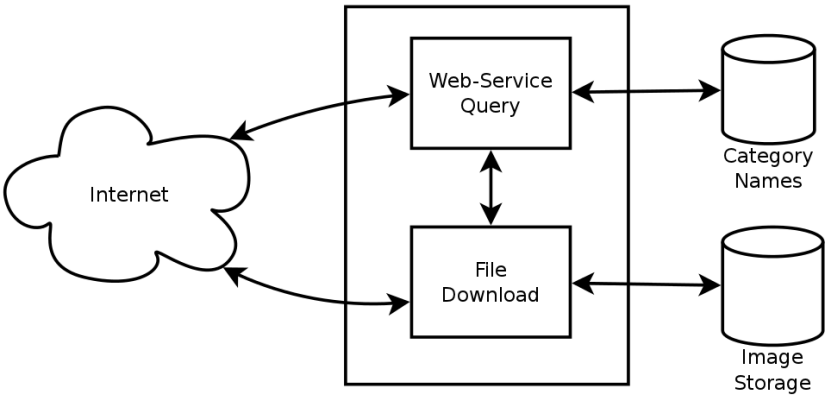


Figure 4.1: Architecture of the Internet search module

The retrieved images have to follow the following conditions: for now only JPG, PNG and BMP image types are supported, because they are the most commonly used image types; and the size of the image should lie in the range $[6, 200]$KB. The user has the flexibility to define the number of images to be retrieved. There can be cases where the web-service returns less than the specified number of images for a given category name. In such circumstances, only these images are used for building category representations.

This new module, besides improved efficiency, provides more flexibility to the agent: more than one web service (e.g. use of multiple search engines) for image search, and to exploit multiple web-service applications (e.g. the use of translation services to translate search queries into multiple languages. Eventually, the aim is to design these services for improving the quality of category representations.

4.2 From images to object models

After retrieving the images from Internet, it is necessary to build models for the objects present in them. The first step involves segmentation of the retrieved images to extract the shape information of the objects present in them. The method described in Chapter 3 for

[2]http://curl.haxx.se/

object extraction is used here. In short, Canny edge detector is used to create an edge map for each image. From each edge map, one or more coherent object contours are extracted using a set of rules that describe the distance and colour disparity between object candidates.

For each extracted object, a GSC [7, 9, 8] descriptor is computed. This representation was chosen because is was already implemented in the UA@SRVC agent. Based on the evaluation produced by Pereira *et. al.* [7, 8] it has a good performance to represent and recognize generic categories, while it is not computational heavy. Given the shape information of an object, its geometric center is computed using a flood fill algorithm. A 2D polar grid is placed on the geometric center based on which a 2D histogram is filled with the information of the shape points. The histogram is normalized with respect to the maximum distance between the geometric center and an edge point. A more detailed summary of the GSC descriptor is presented in Section 2.3.2.

The input of the object relevance evaluation algorithm is a square matrix, where each cell contains the distance between each pair of objects. The algorithm is generic and can be used with any type of object representation. The responsibility of extracting objects and their representations are delegated to a higher level. An example of a distance matrix is given in Table 4.1.

Objects	A	B	C	D	E
A	0	$D_{A,B}$	$D_{A,C}$	$D_{A,D}$	$D_{A,E}$
B	$D_{B,A}$	0	$D_{B,C}$	$D_{B,D}$	$D_{B,E}$
C	$D_{C,A}$	$D_{C,B}$	0	$D_{C,D}$	$D_{C,E}$
D	$D_{D,A}$	$D_{D,B}$	$D_{D,C}$	0	$D_{D,E}$
E	$D_{E,A}$	$D_{E,B}$	$D_{E,C}$	$D_{E,D}$	0

Table 4.1: Example of a distance matrix for five objects

4.3 Unsupervised object relevance evaluation

Images retrieved from the Internet by searching for a given category name will always have an amount of noise. Some of this noise will be segmented as false objects, or false instances of the category.

For this reason, the objects are ranked based on their relevance and only the top ranked objects are used for representing a category. This section will describe the process of evaluating the relevance of the objects.

A human can use previous knowledge to establish semantic relations between a category name and the relevant instances. However, in the state of the art, this task is very difficult for an artificial agent. Assuming that at least a small number of the extracted objects are

relevant instances of the category, and that these objects are more similar between themselves than irrelevant objects, the relevance evaluation can build upon a similarity function. The relevance evaluation can be achieved through clustering algorithms, as proposed by Pereira *et. al* [9, 7]. Clustering is the placement of a set of observations into disjoint subsets (designated clusters) so that the observations in the same cluster are similar in some sense. Based on the produced clusters, the relevance of each object will be evaluated.

4.3.1 Object clustering

Pereira *et. al* proposed using K-means algorithm [34, 35] to cluster the objects extracted from the images retrieved from the Internet. Given a population of n elements $(x_1, x_2, ..., x_n)$, these elements can be grouped in k sets $S = \{S_1, S_2, ..., S_k\}$, such that, the within-cluster sum of squares (WCSS) is minimized:

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2 \tag{4.1}$$

where μ_i is the center of the cluster i.

This is the mathematical problem that the K-means clustering algorithm attempts to resolve. Using the distance matrix computed previously, the object representations are clustered based on their similarity. In standard K-means implementations the seeding process is random, and can be done with linear complexity using Durstenfeld shuffle algorithm [36]. After the initial seeding process, all the remaining elements are added to the closest respective clusters. The algorithm continues with an adaptation loop that iteratively assigns elements and updates clusters until no more assignments can be made. The adaptation loop is described in Algorithm 4.

The conventional K-means implementation suffers from two major theoretical drawbacks.

Firstly, it has been proved that in the worst case, the running time of the algorithm is super-polynomial in relation to the input size [37]. Usually the learning phase does not need to be in real time. But for a system that performs continuous learning, a clustering algorithm with a worst case super-polynomial algorithm is not a good candidate.

Secondly, the approximation found can be arbitrarily bad with respect to the objective function compared to the optimal clustering. This occurs because, when the initial seeding is completely random, the final clusters can be very distant from the optimal distribution. This problem is usually tackled with multiple repetitions of the clustering algorithm, which again is not a convenient feature for a real time system when a repetition takes super-polynomial time.

Algorithm 4 K-means adaptation loop

```
Input
elementClusters - array that specifies in what cluster the element is inserted; initialized by
the seeding process
nElements - number of elements to cluster
clusters - set of clusters; initialized by the seeding process
distMatrix - square matrix with the distances between all elements

Variables
oldCluster - temporarily stores the previous cluster of one element
change - boolean variable that indicates if there were changes in the clusters

repeat
  change ← false
  for i ← 0; i < nElements; i ← i + 1 do
    oldCluster ← elementsClusters[i]
    elementClusters[i] = closestCluster(i, distMatrix, clusters)
    if oldCluster ≠ elementsClusters[i] then
      removeElementFromCluster(clusters[oldCluster], i)
      addElementToCluster(clusters[elementsCluster[i]], i)
    end if
  end for
until change
return clusters
```

Because of this drawback, in the context of this dissertation the unsupervised object relevance evaluation algorithm was implemented with K-means++ algorithm [37]. This clustering algorithm tackles the previous drawbacks by improving the initial seeding process. In fact, the only difference between K-means and K-means++ is the initial seeding process. The initial seeding proposed by K-means++ places only one random cluster center and, from that point beyond the cluster centres are spread as far away as possible. Which means, the algorithm converges rapidly to the final clustering configuration, with an improvement in the error between the clustering produced and the optimal clustering.

The seeding process is implemented as follows:

1. Choose one center uniformly at random from the data points.
2. For each data point x, compute $D(x)$, the distance between x and the nearest center that has already been chosen.
3. Add one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$.
4. Repeat Steps 2 and 3 until all remaining k cluster centers have been chosen.

After the initial seeding process, the main loop of K-means++ is the same as K-means, described in Algorithm 4.

4.3.2 Object scoring

As previously stated, Pereira *et. al.* proposed that the cluster with more elements has higher probability of being composed of correct category instances. A voting system was developed based on that. Since the K-means algorithm only provides an approximation, a better result is achieved with multiple runs and multiple values of the number of clusters, k (even with the implementation of the more reliable K-means++). The algorithm starts with $k = \frac{Total\ extracted\ objects}{4}$ and increases to $k = \frac{Total\ extracted\ objects}{2}$, and for each k value the clustering is repeated 100 times.

Pereira proposed that the cluster with more elements has a higher probability to better represent the category. But if only a few instances in the set are relevant, the biggest cluster has a higher probability of containing clutter. Because of this, in the context of this dissertation, a different measure of relevance is used. Based on the equation 4.1, we can compute the *compactness* of each cluster. For example, for cluster i the compactness measure will be defined as:

$$compactness_i = \sum_{x_j \in S_j} \|x_j - \mu_i\|^2 \tag{4.2}$$

The compactness of each cluster, in this context, measures the similarity between the objects that constitute that cluster.

A cluster with only one element has maximum compactness, but is considered noise. An important cluster is a cluster with maximum compactness but at least with two objects.

The measure proposed in this dissertation can be summarized as following: in each run, the objects belonging to the most compact cluster, gain 2 points; objects that constitute clusters with only 1 element, lose 1 point.

4.4 Object selection and category model building

The objects are firstly sorted by their relevance. Next problem is to choose an optimal division between the relevant and non relevant objects. This problem is conceptually similar to the problem of selecting the ideal threshold that correctly separates real object candidates from clutter (explained in object extraction and clutter removal algorithm Section 3.4). Because of this, the same automatic thresholding algorithm (Algorithm 3) can be and was used to determine the optimal division between relevant and non relevant objects.

Figure 4.2 shows an example of unsupervised subset selection. After searching Internet

with the query "apple", 10 images were retrieved (image A). From this initial set, 10 objects were extracted. Four of them are the actual fruit apple, while the remaining six are the brand logo of the company Apple (image B). After the object relevance evaluation and selection, a subset of four objects is determined (image C). From this subset, three objects are good representations of the fruit apple, while only one object represents the brand logo. The initial set of objects had 40% of relevant objects, and after the selection process the returned subset had 75% of relevant objects.

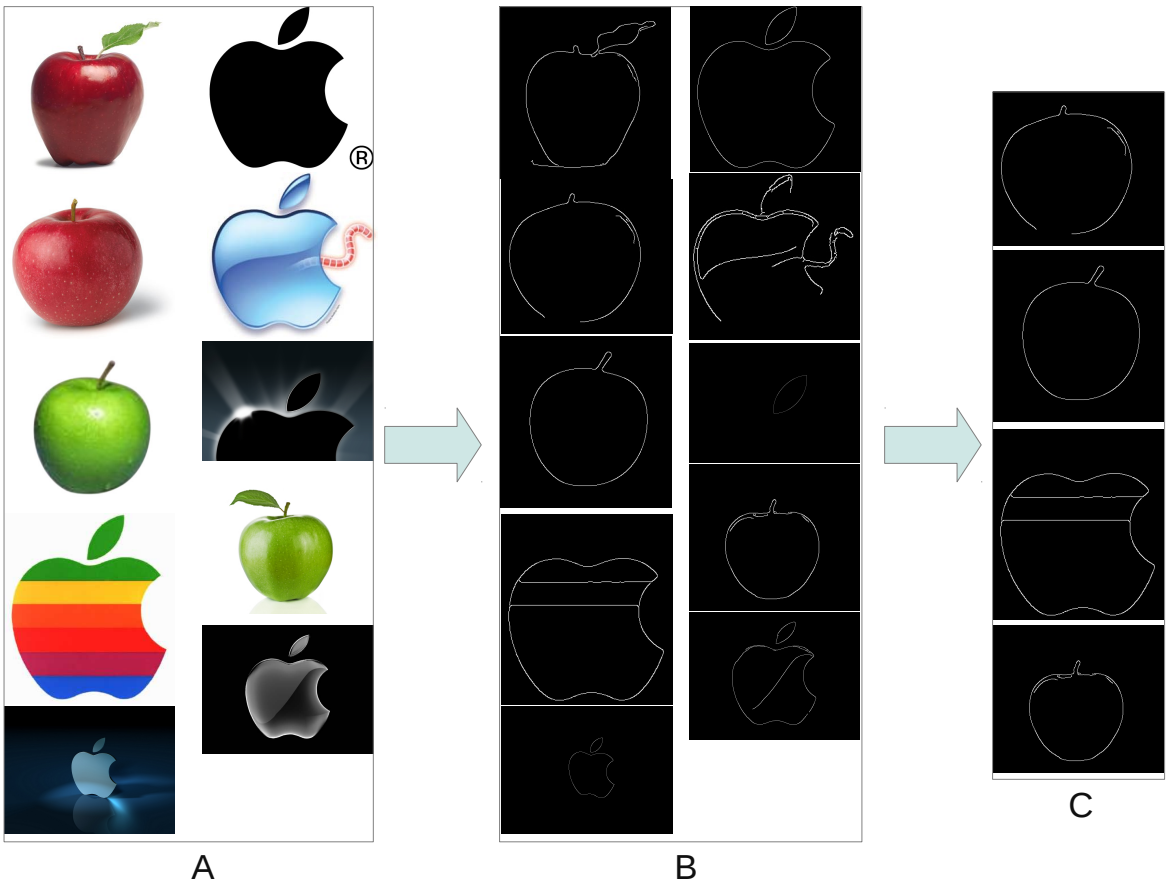


Figure 4.2: Example of unsupervised subset selection. Image A shows a set of 10 images retrieved from Internet when searching for "apple". Image B shows the objects extracted from the set of images retrieved from Internet and ready to be ranked. Image C shows the result of the unsupervised sub selection. From the four returned images, three are correct.

After determining the subset of relevant objects from the initial set of extracted objects, a category is created. Following the previous design of the UA@SRVC agent, a category is described by the set of GSC histograms of each of the selected objects. The main reason for this approach is that only one instance can not represent all the fine details needed to robustly recognize a large diversity of other representations. An example of this is the category

bottle,for which there is a great diversity of shapes. Figure 4.3 shows five different bottles. To allow robust recognition, a descriptor of each bottle shape should be contained in the category. In practical terms, it is not feasible to store each possible instance to describe a category. But storing a feasible number of instances improves the robustness of the category representation.

Figure 4.3: Example of bottle diversity.

It is important to mention here that the categories created with this method will have a variable and uncontrolled number of instances. The number of instances depends greatly on the category and the images retrieved from Internet. The classification system must take this into account.

Chapter 5

Object recognition

The agent, to be able to recognize new instances from the previously learned categories, needs a classification system. This chapter describes the classification system developed for the agent.

This chapter first describes category membership measures used to determine the membership of one object to one category. Second, classification algorithms that support category representations with multiple instances are described.

5.1 Category membership measures

Section 2.3.2 describes the Global Shape Context (GSC) representation, used by the developed agent. A GSC descriptor is a 2 dimensional histogram, that captures the shape of one object. It is a global descriptor, which means, that an object is represented by only one histogram. This property makes the matching process very efficient. This section presents the membership measures used to compare one object GSC histogram with the set of GSC histograms in a category and returning the measure of membership of that object to that category.

5.1.1 Maximum inverse χ^2 distance

The distance between two histograms is evaluated by the χ^2 distance (see Equation 2.8). If two histograms have zero distance, this means that they are the same (maximum similarity). A similarity measure can be obtained using the inverse of the distance as follows:

$$S_{pq} = \frac{1}{D_{pq}} \tag{5.1}$$

where D_{pq} is the χ^2 distance between objects p and q, and S_{pq} is the similarity value between the two objects.

One important aspect is that for GSC, rotation invariance is easily obtained in the matching process. One of the histograms is rotated angle bin times. For each rotation the distance to the other histogram is computed. The final distance is the minimal distance over all rotation iterations, which corresponds to the highest similarity. Figure 5.1 illustrates the rotation of the GSC histogram.

	Angle			
Distance	A1	B1	C1	D1
	A2	B2	C2	D2
	A3	B3	C3	D3

	Angle			
Distance	D1	A1	B1	C1
	D2	A2	B2	C2
	D3	A3	B3	C3

Figure 5.1: Rotation of the GSC histogram. To rotate the histogram and achieve rotation invariance, the columns are shifted. The image on the left shows the original histogram. The image on the right shows the same histogram rotated.

As previously explained each category is composed of multiple instances. The measure of membership of one object to one category is the maximum similarity between the object and one of the category instances. Algorithm 5 shows the algorithm to compute this measure:

Algorithm 5 Maximum inverse χ^2 distance

```
Input
object - GSC histogram of the object
category - set of GSC histograms describing one category

Variables
membership - stores the best membership found; initialized with -1
instance - stores the instance whit highest membership

for i ← 0; i < category.NumberInstances; i ← i + 1 do
  for j ← 0; j < angleBins; j ← j + 1 do
    if membership < 1/chiSquareDistance(object, category[i]) then
      membership ← 1/chiSquareDistance(object, category[i])
      instance ← category[i]
    end if
    rotateHistogram(object)
  end for
end for
return {membership, instance}
```

5.1.2 Penalized membership

The second membership measure is similar to the maximum inverse χ^2 distance with only one difference. The membership value takes into account the positions of the category instances in the relevance ranking (see Chapter 4), e. g., the first elements of the ranking have a higher weight than last elements. This measure was conceived because the unsupervised category learning ranks and sorts the instances by their relevance. Algorithm 6 is used to compute this measure:

Algorithm 6 Penalized membership

```
Input
object - GSC histogram of the object
category - set of GSC histograms describing one category

Variables
membership - stores the best similarity found; initialized with -1
instance - stores the instance with highest similarity
currentSimilarity - stores the similarity for one iteration

for i ← 0; i < category.NumberInstances; i ← i + 1 do
  for j ← 0; j < angleBins; j ← j + 1 do
    currentSimilarity ← 1/chiSquareDistance(object, category[i])
    if membership < (currentSimilarity × (category.NumberInstances -
    i))/category.NumberInstances then
      membership ← (currentSimilarity × (category.NumberInstances -
      i))/category.NumberInstances
      instance ← category[i]
    end if
    rotateHistogram(object)
  end for
end for
return {membership, instance}
```

5.2 Instance-based classification

In the previous section, two different category membership measures were described. Using those category membership measures, three instance-based classification algorithms were developed. These algorithms receive an object and a set of categories and find the category that better represents the object. The system has to take into account that each category is composed of a variable number of instances (all the instance-based classification algorithms are based on the k-Nearest-Neighbour rule (k-NN) [38]).

5.2.1 Nearest Neighbour

The first instance-based classification algorithm is based on Nearest Neighbour search, or similarity search. This algorithm is a special case of the k-Nearest-Neighbour rule, described later, when $k = 1$. The main idea is to find the category that provides higher similarity.

The developed nearest neighbour classifier is given in (Algorithm 7):

Algorithm 7 Nearest neighbour classifier

```
Input
object - GSC histogram of the object
categories - set of categories, each category is described by a set of GSC histograms

Variables
membership - stores the best membership value found; initialized with -1
currentMembership - temporarily stores the membership value for one iteration
category - stores the best category for the object
instance - stores the instance with highest similarity

for i ← 0; i < categories.Size; i ← i + 1 do
  {currentMembership, instance} ← membershipMeasure(object, categories[i])
  if membership < currentMembership then
    membership ← currentMembership
    category ← categories[i]
  end if
end for
return category
```

5.2.2 k-Nearest-Neighbour (k-NN)

The second instance-based classification algorithm is based on the k-NN rule [38]. It uses k rounds. In each round the category that has the higher similarity with the object earns one vote. The winning instance of each round is not eligible in the following rounds. In the end of the process the object will be recognized as belonging to the category that earned more votes.

Unlike the previous classifier, this type of classification is theoretically more robust when there is not full confidence in the "prior" knowledge. In chapter 4 we presented an unsupervised method for training categories based only on their names. The result of this method is highly dependent on the information retrieved from the Internet. Because of this there can not be full confidence on all instances selected for each category.

The k-NN classifier was implemented as follows (Algorithm 8):

Algorithm 8 k-NN classifier

```
Input
object - GSC histogram of the object
categories - set of categories, each category is described by a set of GSC histograms
k - number of nearest neighbours

Variables
membership - stores the best similarity found; initialized with -1
currentMembership - temporarily stores the membership for one iteration
instance - stores the instance with highest similarity
currentInstance - temporarily stores the instance for one iteration
category - stores the category with higher membership
votes - set of votes for each category; initialize as empty

for i ← 0; i < k; i ← i + 1 do
  for j ← 0; j < categories.Size; j ← j + 1 do
    {currentMembership, currentInstance} ← membershipMeasure(object, categories[j])
    if membership < currentMembership and currentInstance.selected = false then
      membership ← currentMembership
      instance ← currentInstance
      category ← categories[j]
    end if
  end for
  votes[category] ← votes[category]+1
  instance.selected ← true
end for
category ← select category with higher votes
return category
```

5.2.3 Weighted k-NN

The final instance-based classification algorithm is based on the Distance-Weighted k-NN rule [39]. This algorithm is similar to the previous, with only one difference. Instead of using equally weighted votes, it will use variable weight to vote. In this implementation a linear decreasing progression was used $(k; k-1; ...3; 2; 1)$. Which means that the closest neighbour have more weight.

This minimal change has two important consequences. First, while k-NN classification is highly affected by ties in the votes, this system has a lower probability of producing ties. Second, since each category has a variable number of instances, selected automatically by the unsupervised learning system, the k-NN classification algorithm could lead to misclassification. For example, consider two categories learned, "key" with two instances and "dog" with three instances, and using a 5-NN classification system. Given a real key object, it will be

classified as dog since key only has two instances. But using the proposed weighted k-NN classification system the two key instances will have higher number of points and allowing a correct classification. This reason makes this classification system more stable.

The weighted k-NN classifier was implemented as follows (Algorithm 9):

Algorithm 9 weighted k-NN classifier

```
Input
object - GSC histogram of the object
categories - set of categories, each category is described by a set of GSC histograms
k - number of nearest neighbours

Variables
membership - stores the best membership measure found; initialized with -1
currentMembership - temporarily stores the membership for one iteration
instance - stores the instance with highest similarity
currentInstance - temporarily stores the instance for one iteration
category - stores the category with higher membership
votes - set of votes for each category; initialize as empty

for i ← 0; i < k; i ← i + 1 do
  for j ← 0; j < categories.Size; j ← j + 1 do
    {currentMembership, currentInstance} ← membershipMeasure(object, categories[j])
    if membership < currentMembership and currentInstance.selected = false then
      membership ← currentMembership
      instance ← currentInstance
      category ← categories[j]
    end if
  end for
  votes[category] ← votes[category]+k-i
  instance.selected ← true
end for
category ← select category with higher votes
return category
```

Chapter 6

Performance evaluation

This chapter presents an evaluation of the models developed in this dissertation. It will be divided in four sections. First, it is evaluated the object extraction and clutter removal algorithm (Chapter 3) comparing it with the object extraction previously used in the UA@SRVC agent. Second, it is evaluated the performance of the unsupervised object selection algorithm proposed in this dissertation (Chapter 4), comparing it with the algorithm proposed by Pereira *et. al.* [7, 9]. Third, the instance-based classifiers integrated in the agent (Chapter 5) are evaluated on two different tests. The overall performance is tested with a standard k-fold cross validation. Then a test similar to the one proposed by Pereira *et. al.*, with an increasing number of categories, is performed allowing direct comparison. Finally a last test is presented, where the new agent will learn categories from the Internet without any kind of human intervention and then tries to recognize a set of objects.

Appendix A contains the complete tables with the results of the performance evaluation.

6.1 Basic evaluations measures

Two basic performance measures will be used in the analysis of the results, namely, *precision* and *recall*. These performance measures can be combined to produce F-measure (or F-score). These measures are defined as follow:

$$precision = \frac{|\{\text{relevant items}\} \cap \{\text{retrieved items}\}|}{|\{\text{retrieved items}\}|} \tag{6.1}$$

$$recall = \frac{|\{\text{relevant items}\} \cap \{\text{retrieved items}\}|}{|\{\text{relevant items}\}|} \tag{6.2}$$

$$F = 2 \times \frac{precision \times recall}{precision + recall} \tag{6.3}$$

6.2 Contour-based object extraction

A set formed by 75 images was created using a regular Internet search engine and the following search keywords: banana, bottle, plastic objects, can, pencil sharpener, bell pepper, pen and soccer ball. The Internet environment is noisy and unpredictable, but since this test is to evaluate the performance of contour-based object extraction, the selection of images was manually carried out by the author enforcing two properties: a human can without doubt segment all the objects in the image; and no object has hidden parts.

The performance of the proposed object extraction algorithm is compared with the algorithm originally used in UA@SRVC agent. Both object extraction algorithms process the mentioned set of images. The results are examined and counted by the author.

The analysis of the performance is based on computing *precision* and *recall*. In this experiment the relevant items correspond to the objects recognizable by a human and the retrieved items are the contours extracted automatically. Table 6.1 presents the results obtained with the original algorithm developed for the UA@SRVC agent, and the results obtained with the new algorithm. Comparing the results, the proposed the new algorithm nearly doubles the performance obtained with the original UA@SRVC algorithm. The original UA@SRVC algorithm obtained 30% in F-measure, while the new algorithm obtained 56%.

Measures	Original algorithm	Proposed algorithm
Precision	50%	73%
Recall	21%	46%
F-measure	30%	56%

Table 6.1: Performance analysis of the original and the proposed contour-based object extraction algorithms.

The proposed algorithm isn't a general solution for contour-based object extraction problems, but proved to work very well with the Internet images in which there is almost no context about the objects present.

6.3 Unsupervised object subset selection

For this evaluation, the generic categories presented in the three editions of the Semantic Robot Vision Challenge (2007, 2008, 2009) were used (see Table 6.2). For each category, 30 images were retrieved using the Internet module presented in Section 4.1. The images are stored locally and all the evaluation is executed with this set.

1	Banana	2	Bottle	3	Digital Camera
4	Dinosaur	5	Electric Iron	6	Eyeglasses
7	Fax Machine	8	Fork	9	Frying Pan
10	Green Apple	11	Laptop	12	Orange
13	Pumpkin	14	Red Bell Pepper	15	Red Ping Pong Paddle
16	Red Plastic Cup	17	Remote Control	18	Rolling Suitcase
19	Saucepan	20	Scientific Calculator	21	Toy Car
22	Ulu	23	Upright Vacuum Cleaner	24	White Soccer Ball

Table 6.2: Generic categories from the three editions of SRVC.

This evaluation follows the same evaluation scheme adopted by Pereira *et. al.* in his work [7]. A training phase, similar to the training phase used in the SRVC competition, was performed with the obtained images, and then the percentage of good training objects in the initial set is compared with the percentage of good training objects in the selected subset. The performance of the unsupervised subset selection algorithm is also evaluated using the F-measure expression. The baseline for comparison is an agent running the original UA@SRVC modules for object extraction and unsupervised subset selection.

In the context of these experiments, the relevant items for computing precision and recall expressions (see Equation 6.3) are good training objects, as assessed by the author.

6.3.1 Original UA@SRVC2008 object extraction and selection modules

Figure 6.1 presents the ratio of good objects after the object extraction step with the ratio of good objects after running the unsupervised object subset selection algorithm. Figure 6.2 presents the improvement of the subset selection as a function of the respective percentage of good training objects in the initial set. On average, before the unsupervised subset selection, 35.11% of the objects could be considered good training objects. After subset selection, the ratio increased to 45.77%, which represents an improvement of 10.66%. The improvement depends highly on the quality of the initial set. In this evaluation four categories end up with zero relevant instances (see categories 4, 12, 13, and 16 in Figure 6.1).

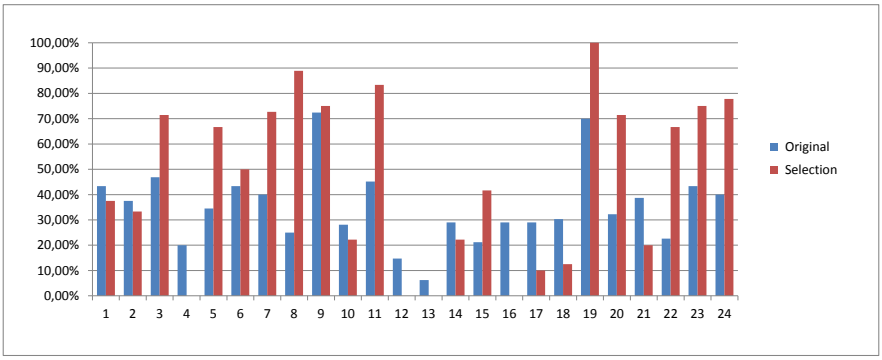


Figure 6.1: Ratio of good objects before and after the unsupervised subset selection, for the different categories in table 6.2, in the first experiment.

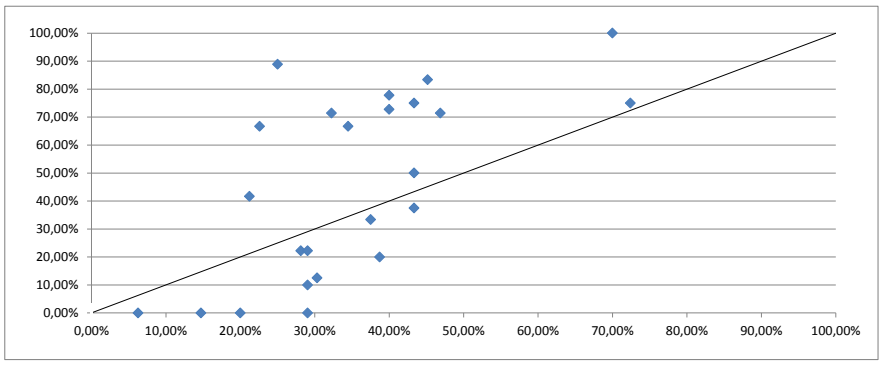


Figure 6.2: Percentages of good training objects after the subset selection (yy axis) plotted as a function of the respective percentages of good training objects in the initial set (xx axis), in the first experiment.

The previous figures present an analysis based on precision only. In order to obtain a more global evaluation, F-measure was used. The selection algorithm obtained 42.93% and 33.59% for precision and recall respectively. This corresponds to an F-measure of 37.69%.

6.3.2 Original extraction module combined with the new selection module

In a second experiment, the agent was modified to include the proposed unsupervised object subset selection algorithm. The previous test with the exact same initial set was then repeated. Figure 6.3 presents the ratio of good training objects, before and after the subset selection. Figure 6.4 presents the improvement of the subset selection as a function of the respective percentage of good training objects in the initial set. On average, there were 35.11% of good objects in the initial set. After running the new subset selection algorithm the ratio improved to 64.77% (an improvement of 29.66% over the initial set and 19% over the previous technique).

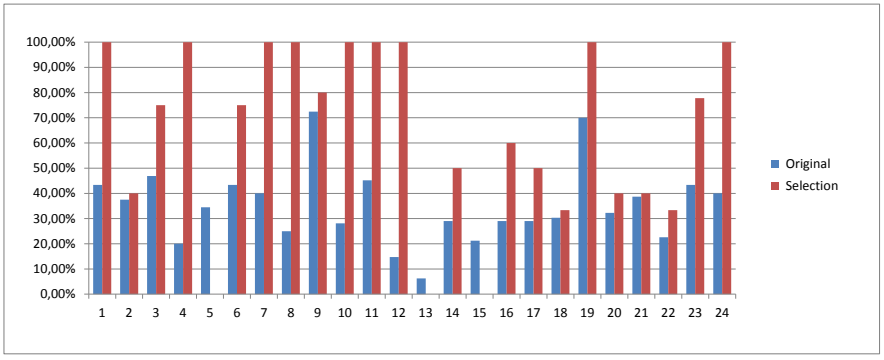


Figure 6.3: Ratio of good objects before and after the unsupervised subset selection, for the different categories in table 6.2, in the second experiment.

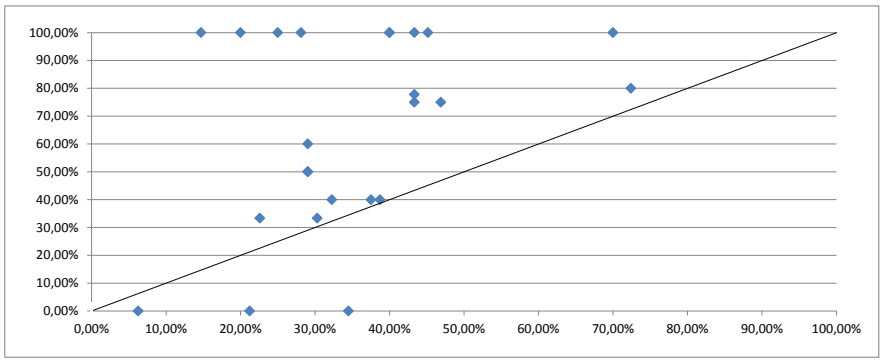


Figure 6.4: Percentages of good training objects after the subset selection (yy axis) plotted in function of the respective percentages of good training objects as a the initial set (xx axis), in the second experiment.

This algorithm obtained 65.22% and 28.63% for precision and recall respectively. This corresponds to an F-measure of 39.79%. While, in F-measure, the proposed algorithm did not improve much (only 2.1%) in precision it shows an improvement of 22.29% which is significant. For this technique it is more important the precision than the recall. Since the training phase is unsupervised, it is more important to have a high confidence in the learned categories than having categories built from all good training objects and some possible false objects.

6.3.3 Using the developed modules for object extraction and selection

Finally, it was integrated the proposed contour-based object extraction module in the agent and the results were compared with the previous methods. Figure 6.5 presents the ratio of good training objects before and after the unsupervised subset selection. Figure 6.6 presents the improvement of the subset selection as a function of the respective percentage of good training objects in the initial set. Since the segmentation algorithm is different, the initial ratio of good training objects is 39.59%, which is a little better than the previous object extraction method. On average after the subset selection, the percentage of good objects is 74.35%, which represents an improvement of 34.66% over the initial set and 28.58% over the original selection algorithm. Another important aspect is that none of the categories was created without at least one good training object, even the categories with lower ratio.

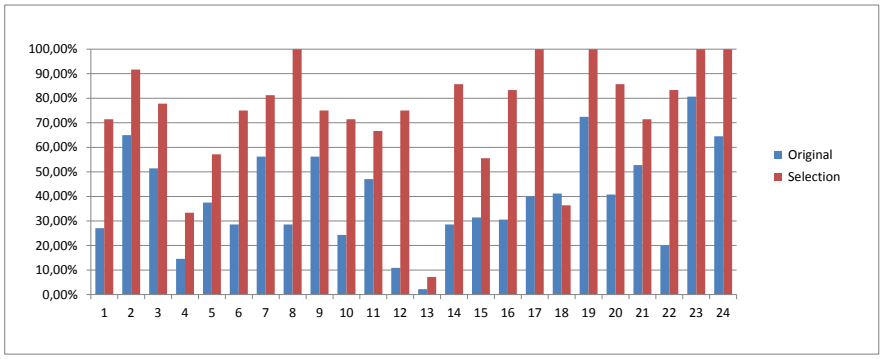


Figure 6.5: Ratio of good objects before and after the unsupervised subset selection, for the different categories in table 6.2, in the third experiment.

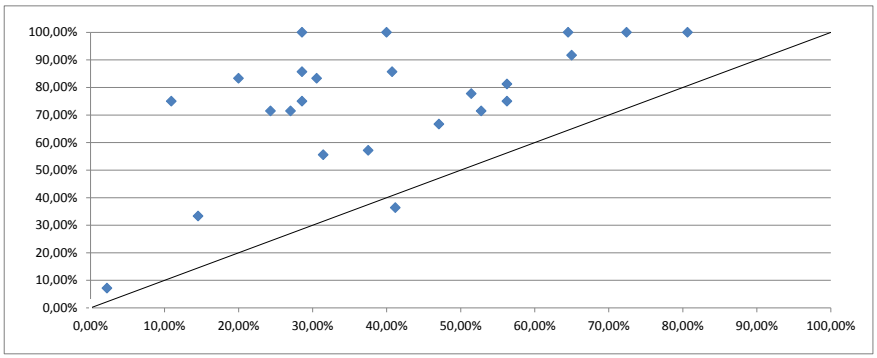


Figure 6.6: Percentages of good training objects after the subset selection (yy axis) plotted as a function of the respective percentages of good training objects in the initial set (xx axis), in the third experiment.

The combination of the contour-based object extraction and unsupervised subset selection modules produced 70.48% and 48.63% of precision and recall respectively. This corresponds to an F-measure of 57.55%. This is an improvement of 19.86% over the original technique, used in UA@SRVC'08 agent (see Section 6.3.1). And an improvement of 17.76% over the agent with the new unsupervised selection algorithm and the original object extractor (see Section 6.3.2). In both there was an improvement in precision and recall. This result shows the importance of a good object extraction in a recognition system. The improved contour-based object extractor improves the quality of the object shape which improves the recognition and as a consequence improves the clustering of objects based on their similarity.

6.4 Classifiers

To evaluate the performance of the classifiers presented in this dissertation, a set of images from the LANGG project [40] was used. It consists of 7536 images from 68 homogeneous

categories. These images can be regrouped in terms of 49 heterogeneous categories. A homogeneous category is a category that is defined by the shape of one object. For example, a category that represents a specific stapler. If another different stapler is added to the system a new category is created. On the other hand, a heterogeneous category is a category that is defined by shapes of different objects. For example, a category stapler will be defined by the shapes all known staplers, independently of their specific shape. Tables 6.3 and 6.4 shows the names of the 68 homogeneous and 49 heterogeneous categories respectively, present in the LANGG image set. In general, the objects in the images are easy to segment.

Two sets of experiments were carried out, namely standard 10-fold cross validation tests, and experiments with increasing number of categories. In the different sets of experiments, the three used classifiers are evaluated, namely Nearest-Neighbor (NN), k-Nearest-Neighbor (k-NN), and weighted k-Nearest-Neighbor (w-k-NN) classifiers. For the k-NN and w-k-NN, $k = 5$ was used. An odd value of k minimizes the tie problem. As previously stated (see Section 5.2.3), k-NN classifiers can lead to false classifications when it is used more rounds than instances within a category. Since some results will be compared with the final evaluation of the agent (see Section 6.5) it is important to choose a value of k that allows this comparison. Training the agent using the unsupervised subset selection algorithm proposed in this dissertation, the category with lower number of instances was created with only 5 instances.

For this evaluation only the maximum inverse χ^2 distance membership measure was used, since the instances used in the learning step are randomly selected and not sorted by relevance.

1	A	2	Battery	3	Bottle top	4	Boy
5	CD	6	Cigarette Box	7	Circle	8	Circuit Board
9	Duster	10	Floppy	11	Glove	12	Horse
13	Icetea can	14	Key	15	Spanner	16	Lighter
17	Mouse	18	Nail	19	One	20	Passbook
21	Pen	22	Pencil	23	Postit	24	Remote control
25	Star fish	26	Staple remover	27	Sugar packet	28	Tape
29	Teddy bear	30	Three	31	Toy bike	32	Toy car
33	Toy jeep	34	Toy saw	35	Toy scissor	36	Twenty cent
37	Ubuntu cd cover	38	USB pen	39	Train top	40	Train
41	Coffee mug	42	Tilted coffee mug	43	Tilted cup	44	Water cup
45	Coffee cup	46	Cup	47	Table knife	48	Table fork
49	Table spoon	50	Coffee spoon	51	Penguin	52	Penguin sitting
53	Stapler1	54	Stapler2	55	Stapler3	56	Water bottle
57	Ink remover bottle	58	Glue bottle	59	Screw	60	Toy screw
61	Tractor	62	Tilted tractor	63	Screw Driver	64	Toy screw driver
65	Box	66	Box2	67	Toy mobile	68	Mobile

Table 6.3: The 68 homogeneous categories in the LANG image set.

1	A	2	Battery	3	Bottle top	4	Boy
5	CD	6	Cigarette Box	7	Circle	8	Circuit Board
9	Duster	10	Floppy	11	Glove	12	Horse
13	Icetea can	14	Key	15	Spanner	16	Lighter
17	Mouse	18	Nail	19	One	20	Passbook
21	Pen	22	Pencil	23	Postit	24	Remote control
25	Star fish	26	Staple remover	27	Sugar packet	28	Tape
29	Teddy bear	30	Three	31	Toy bike	32	Toy car
33	Toy jeep	34	Toy saw	35	Toy scissor	36	Twenty cent
37	Ubuntu cd cover	38	USB pen	39	Train	40	Cup
41	Cutlery	42	Penguin	43	Stapler	44	Bottle
45	Screw	46	Tractor	47	Screw driver	48	Box
49	Mobile						

Table 6.4: The 49 heterogeneous categories in the LANG image set.

6.4.1 Cross validation with all available images

The initial set of images is randomly divided in roughly 10 equal parts. In each run, nine parts are used for training and the remaining part is used for testing. This process is repeated 10 times, so all parts will be used for training and testing. Tables 6.5 and 6.6 present the results from the 10-fold cross validation evaluation for 68 homogeneous categories and 49 heterogeneous categories respectively.

With a "prior" knowledge nine times bigger than the test set, and with objects easy to segment (selected by a human user), and without false positives, the best classifier was the NN for both sets of categories. On average in the test with 49 heterogeneous categories, all classifiers improve 1%. This suggests that categories composed by multiple instances have a higher performance for heterogeneous categories.

Fold iteration	Nearest Neighbour	K-Nearest Neighbour	Weighted K-Nearest Neighbour
1	91,77%	89,38%	90,44%
2	91,50%	89,97%	90,64%
3	91,63%	89,82%	90,93%
4	91,43%	89,51%	90,74%
5	91,31%	89,30%	90,52%
6	91,17%	89,22%	90,46%
7	91,22%	89,02%	90,31%
8	91,33%	89,19%	90,49%
9	91,28%	89,29%	90,51%
10	91,05%	88,98%	90,26%
Average	91,37%	89,37%	90,53%
Standard Deviation	0,22%	0,32%	0,20%

Table 6.5: 10-Fold cross validation of classifiers for 68 homogeneous categories.

Fold iteration	Nearest Neighbour	K-Nearest Neighbour	Weighted K-Nearest Neighbour
1	92,43%	90,17%	90,84%
2	93,23%	91,37%	92,83%
3	92,03%	89,91%	90,84%
4	90,57%	88,71%	90,04%
5	92,70%	89,64%	91,24%
6	92,30%	91,37%	92,56%
7	92,96%	90,70%	92,56%
8	92,30%	91,10%	91,50%
9	91,90%	91,90%	92,16%
10	90,86%	90,86%	91,26%
Average	92,13%	90,57%	91,58%
Standard Deviation	0,85%	0,96%	0,92%

Table 6.6: 10-Fold cross validation of classifiers for 49 heterogeneous categories.

6.4.2 Cross-validation with increasing number of categories

For the second set of experiments, n categories are randomly chosen and trained using a small number of instances randomly taken from the image set (4 instances for the test with 68 homogeneous categories and 10 for test with 49 heterogeneous categories). This evaluation is similar to the evaluation performed by Pereira *et. al.* [8], allowing a direct comparison between classification algorithms with the set of 68 homogeneous categories. The key differences in this evaluation are the contour-based object extractor and the new classification algorithms. The test with 49 heterogeneous categories will be compared with final evaluation of the agent (see Section 6.5). on average the unsupervised subset selection algorithm produces categories with 10 instances, this is the reason why in the second test are used 10 images to train.

After that, the learned categories are tested with 3 different instances of the same categories. This process is repeated 5 times. Everything is repeated for increasing number of categories, starting in 5 up to the maximum (65 for the set with 68 homogeneous categories and 45 for the set of 49 heterogeneous categories).

Plotting the results for increasing number of categories allow to infer the scalability trend of the system. Figures 6.7 and 6.8 plots the classification precision as a function of the number of categories. The performances of the classifiers have a gradual decrease with the increase of learned categories. For 68 homogeneous and 49 heterogeneous categories, the best classifier was the NN classifier. The fact that the categories are trained with objects selected and verified by a human user, which implies that there is a high confidence in the learned knowledge, explains why the best classifier was the classifiers based on the Nearest-Neighbour

rule.

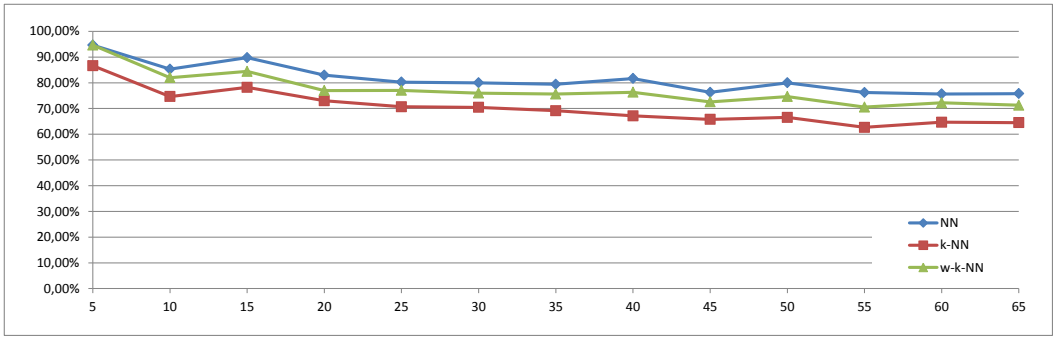


Figure 6.7: Performance of the classifiers for 68 homogeneous categories. The xx axis represents the number of categories and the yy axis represents the percentage of successful recognitions.

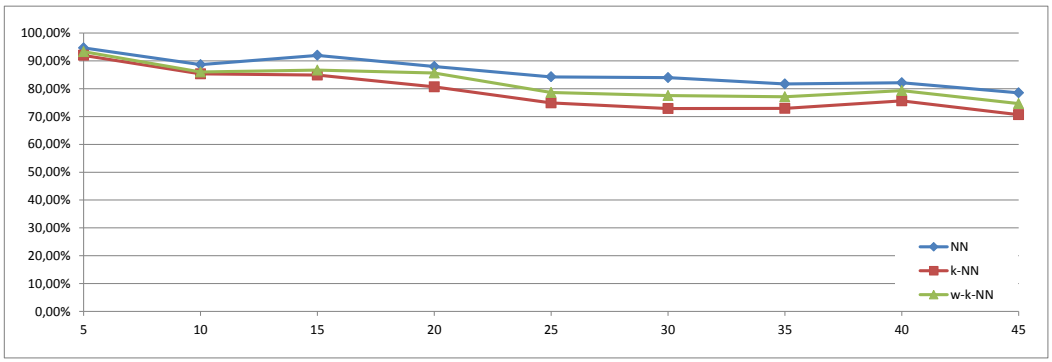


Figure 6.8: Performance of the classifiers for 49 heterogeneous categories. The xx axis represents the number of categories and the yy axis represents the percentage of successful recognitions.

On average the NN classifier correctly recognized $81.40 \pm 3.56\%$ objects. Pereira *et. al.* presented a similar evaluation, with the difference that the number of categories incremented only to 40, the classification system correctly recognized 78.3% objects. On average, for increments until 40 categories, the NN classifiers correctly recognized $84.27 \pm 3.72\%$ objects. Thus, this was an improvement over the original UA@SRVC classification system. The evaluation on the 49 heterogeneous categories has $86 \pm 3.38\%$ of recognition for the NN classifier. This value is not comparable with the previous evaluation since the number of categories in the system is different and the number of instances used to train is bigger in this case. As previously stated, the main purpose of this test was to compare with the final test of the agent. This will allow comparing the unsupervised training method with a supervised training method.

6.5 Semantic vision agent

The final evaluation of the agent is similar to the cross validation with increasing number of categories presented in the previous section, with a key difference: the training phase uses the unsupervised subset selection algorithm and not images from the LANGG set. Since it is not easy to search homogeneous categories in the Internet, this test was only performed over the 49 heterogeneous categories.

Based on the category names the agent retrieved 30 images from the Internet. For each image the present objects were extracted, ranked and selected as good training object or not. Since the categories were trained with the unsupervised subset selection method, the instances that compose one category are sorted by its relevance. Because of this the penalized membership measure can and was be used in the testing.

6.5.1 Maximum inverse χ^2 distance

Figure 6.9 plots the classification precision as a function of the number of categories, using maximum inverse χ^2 distance membership measure. The first conclusion is that a completely unsupervised technique to learn generic categories does not compare with a supervised one. It is important taking into account that for each category in the set there are around 150 object images, and only three instances are randomly selected to test at each round. The probability of selecting an object image that can be recognized with the categories learned automatically is low.

Unlike the previous tests the best classifier was w-k-NN with an average of 11.77% correct recognitions. It is possible to conclude that when there is not full confidence on the "prior" knowledge a classifier based on the weighted k-Nearest-Neighbour rule is more effective.

Figure 6.10 plots the ratio between the performance of this system and a random system, proving that for a scenario with 10 categories or more the developed system is more efficient.

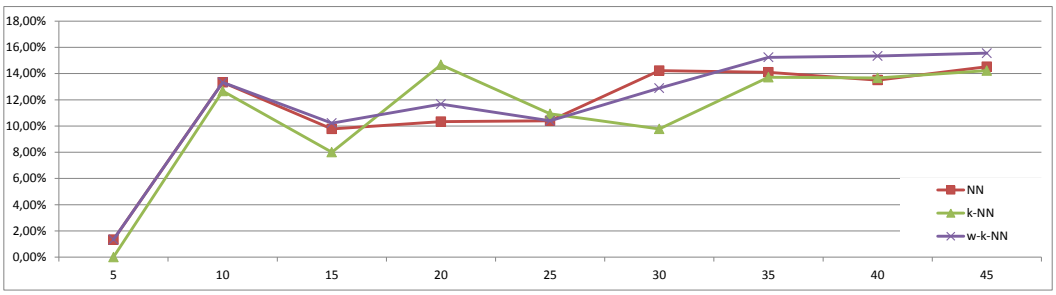


Figure 6.9: Performance of the agent, using maximum inverse χ^2 distance membership measure. The xx axis represents the number of categories and the yy axis represents the percentage of successful recognitions.

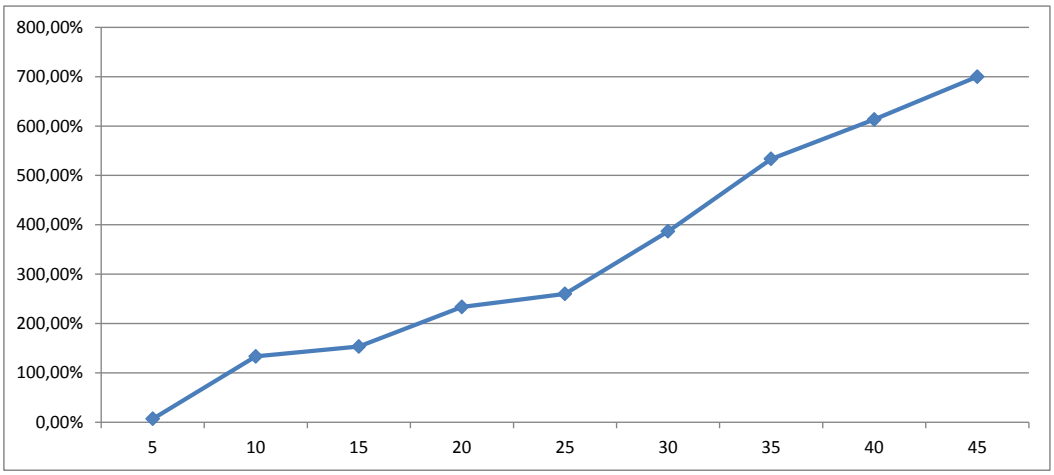


Figure 6.10: Learning efficiency of the developed agent, given by the ratio between the precision of the agent and the precision of a random classifier. The xx axis represents the number of categories and the yy axis represents the ratio.

One unexpected result in this test is the increase of performance when the system learns more categories. After analysing in detail the categories created by the unsupervised subset selection, only a small number of them had instances similar to the objects in the set of training images. So for a small number of categories, the probability that of one of those is a category with good instances is low.

6.5.2 Penalized membership

Figure 6.11 plots the classification precision as a function of the number of categories, using penalized membership measure. For all the categories the classifiers performed worst with the penalized membership measure than with maximum inverse χ^2 distance membership measure. This can be explained by taking into account that the objects extracted from the Internet images are ranked by their similarity, but for very heterogeneous categories (for example bottle) an unusual shape, that is completely correct, will have a lower score in the ranking system. Because of this there was a degradation of performance when the penalized membership measure was used.

Still the best classifier was w-k-NN with an average of 8.8% correct recognitions. Which reinforces the idea that, when there is not full confidence in the "prior" knowledge, a classifier based on the weighted k-Nearest-Neighbour rule is more effective.

Figure 6.10 plots the ratio of the performance of this system with a random system.

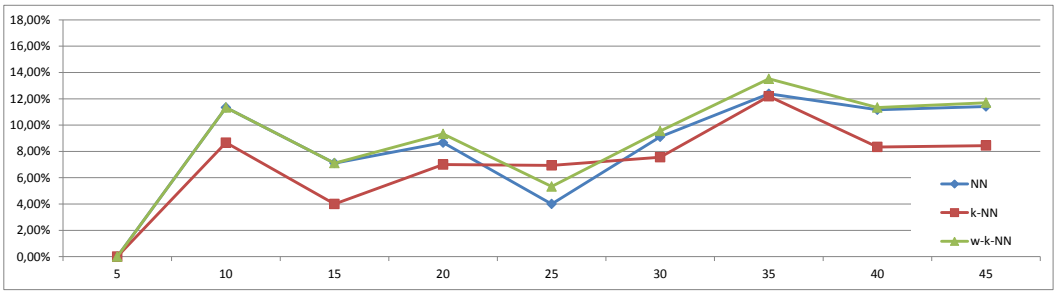


Figure 6.11: Performance of the agent, using penalized membership measure. The xx axis represents the number of categories and the yy axis represents the percentage of successful recognitions.

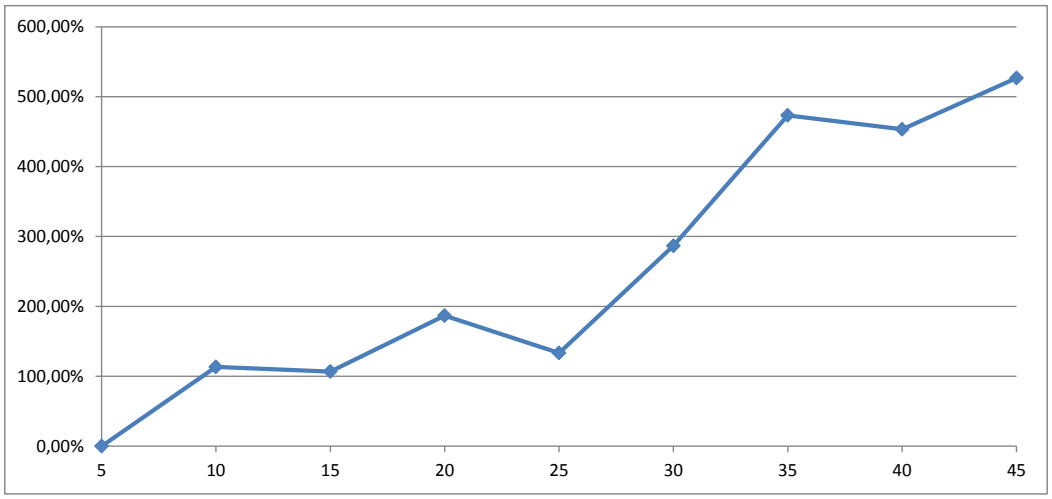


Figure 6.12: Learning efficiency of the developed agent, given by the ratio between the precision of the agent and the precision of a random classifier. The xx axis represents the number of categories and the yy axis represents the ratio.

Chapter 7

Conclusion

7.1 Overview

In this dissertation, a semantic vision agent was developed and evaluated. This final chapter summarizes the work produced and presents the major conclusions of this dissertation. It ends with the author's opinion about which path to follow for improving the agent.

A novel technique was presented to extract objects from images with complex scenes, e. g., images retrieved from the Internet. Without high level information about the object, except for the category name, the process of extracting its shape can only rely on simple heuristics. The proposed algorithm uses a simple heuristic that combines colour disparity and distance between contour segments to agglomerate or not contours and extract a coherent shape from the image.

A technique that allows unsupervised learning of visual categories was developed. Relying only on the category name, images related to it are retrieved from the Internet using a public web service. From these images objects are extracted using the segmentation algorithm presented in this dissertation. It is proposed that the most similar objects correctly represent the category. The relevance of each object to the category is obtained from similarity. The objects are ranked using a clustering algorithm, which will use the similarities between objects to form clusters. The top ranked objects will be used by the agent to learn a new category.

Even with the unsupervised subset selection process described in this dissertation, the categories learned without human intervention are noisy. Classifiers based in K-Nearest Neighbour or the weighted K-Nearest Neighbour presented in this dissertation minimize misclassification.

The proposed algorithms were designed to be as generic as possible, which means, the classifiers and the unsupervised subset selection algorithm do not depend on the descriptor used. The agent was developed with Global Shape Context descriptor, since the focus of the agent was on generic categories. Finally the performance of each algorithm was measured, as

well as the performance of the agent as a whole.

7.2 Conclusions

The conclusions of this dissertation can be summarized into some points:

1. Without a 3^{rd} dimension it is not possible to resolve the problem of object segmentation in 2D images. However, very good approximations can be achieved adding a high level layer on top of an edge detector that, based on several rules, can extract and aggregate coherent contours.

2. Because public web services are syntax-based and not semantic-based, the majority of the retrieved images are noise. To achieve autonomous category learning on the basis of information available on the Internet, it is necessary to filter the relevant information in order to build robust models of categories.

3. Unsupervised learning is a loop problem. To segment and extract good features from one object, high level information about the object is needed. However the type of this information required depend form its features.

4. Using the similarity measures proposed in this dissertation, unsupervised clustering techniques were used to group the most similar instances. It was shown that the most compact of these clusters (found using the compactness criteria) contained the most relevant instances belonging to a category (rest of the clusters being relatively noisy). This is in contrast to the method proposed by Pereira *et. al.* where the instances of the cluster with the highest cardinality were chosen as being the most relevant ones. Our compactness criteria was shown to give more reliable and consistent results in comparison to Pereira's selection approach.

5. A classifier based in weighted K-Nearest Neighbour proved to be more efficient when there is less confidence in the categories, for example, when the categories are learned without human supervision. On the other hand, when the learning was supervised by a human user classifiers based in Nearest Neighbour proved to be the most efficient.

6. The descriptor used by the agent (Global Shape Context) relies only on the shape of the objects. So certain categories like white soccer ball or red bell pepper don't benefit from the colour features.

7.3 Future work

Finally we outline some points that we think represent the correct direction for the work presented in this dissertation. Future developments should focus on:

1. Segment objects from complex scenes is a basic step in object detection. The algorithm presented in this dissertation uses a simple heuristic because the only information known about the category is its name. Using some web service to gather more information about the category should improve the efficiency of the segmentation. For example main dominant colour, circularity, etc.

2. Currently the agent relies only on one Web service (Google Images) and searches only in English. Taking into account the noise environment of the Internet and the exponential growth in terms of content, using multiple sources and multiple languages should improve the quality of the final results.

3. For categories learned without human supervision, a weighted K-Nearest Neighbour classifier proved to be most efficient. In this dissertation a linear decreasing function was used to assign votes. Better performance could be achieved with other decreasing curves, logarithm functions for example. Or by a function that relates the current round and the similarity measure to compute the correct vote value.

4. The descriptor used was Global Shape Context which is a global descriptor that relies only in the object shape. Other descriptor should be used or combined with this relying in other features other than shape.

5. Since Semantic Robot Vision Challenge is in hiatus much of the knowledge presented in this dissertation can be used in other projects. For example, CAMBADA team needs a method to detect and recognize a generic official soccer ball in real time. A modified version of the descriptor used in this dissertation is being currently tested.

Appendix A

Performance tables

This appendix will include the complete data results from the contour-based object extraction, unsupervised subset selection algorithm, and classification algorithms.

The first section presents 2 tables: the first table shows the results from the object extraction algorithm present in UA@SRVC agent; the second table shows the results from the colour-based object extraction presented in this dissertation.

The second section presents 3 tables: the first table shows the results from the unsupervised subset selection algorithm developed by Pereira [7, 9]; the second table shows the results from the unsupervised subset selection presented in this dissertation; the third table shows the results from the unsupervised subset selection combined with the colour based segmentation presented in this dissertation.

The third section presents 4 tables: the first table shows the results from the recognition evaluation with 68 homogeneous categories, trained with human supervision; the second table shows the results from the recognition evaluation with 49 heterogeneous categories, trained with human supervision; the third table shows the results from the recognition evaluation with 49 heterogeneous categories, trained without human supervision, using Maximum inverse χ^2 distance; the forth table shows the results from the recognition evaluation with 49 heterogeneous categories, trained without human supervision, using penalized membership.

A.1 Contour-based object extraction

idImages	idObject	Real	Detected	idImages	idObject	Real	Detected	idImages	idObject	Real	Detected
0	0	TRUE	TRUE	16	2	TRUE	FALSE	39	4	FALSE	TRUE
1	0	TRUE	TRUE	16	3	FALSE	TRUE	40	0	TRUE	FALSE
2	0	TRUE	TRUE	17	0	TRUE	FALSE	40	1	TRUE	FALSE
3	0	TRUE	FALSE	17	1	TRUE	FALSE	40	2	TRUE	FALSE
3	1	TRUE	FALSE	17	2	TRUE	FALSE	40	3	FALSE	TRUE
3	2	TRUE	FALSE	17	3	FALSE	TRUE	41	0	TRUE	FALSE
3	3	TRUE	FALSE	18	0	TRUE	FALSE	41	1	TRUE	FALSE
3	4	TRUE	FALSE	18	1	TRUE	FALSE	41	2	TRUE	FALSE
3	5	TRUE	FALSE	18	2	TRUE	FALSE	41	3	TRUE	FALSE
3	6	TRUE	FALSE	18	3	FALSE	TRUE	41	4	TRUE	FALSE
3	7	FALSE	TRUE	19	0	TRUE	FALSE	41	5	TRUE	FALSE
4	0	TRUE	TRUE	19	1	TRUE	FALSE	41	6	TRUE	FALSE
5	0	TRUE	TRUE	19	2	TRUE	FALSE	41	7	TRUE	FALSE
6	0	TRUE	FALSE	19	3	FALSE	TRUE	41	8	FALSE	TRUE
6	1	TRUE	FALSE	19	4	FALSE	TRUE	42	0	TRUE	TRUE
6	2	TRUE	FALSE	20	0	TRUE	TRUE	43	0	TRUE	TRUE
6	3	FALSE	TRUE	21	0	TRUE	FALSE	44	0	TRUE	TRUE
7	0	TRUE	FALSE	21	1	TRUE	FALSE	45	0	TRUE	FALSE
7	1	TRUE	FALSE	21	2	TRUE	FALSE	45	1	TRUE	FALSE
7	2	TRUE	FALSE	21	3	FALSE	TRUE	45	2	TRUE	FALSE
7	3	TRUE	FALSE	22	0	TRUE	FALSE	45	3	FALSE	TRUE
7	4	TRUE	FALSE	22	1	TRUE	FALSE	46	0	TRUE	TRUE
7	5	FALSE	TRUE	22	2	TRUE	FALSE	47	0	TRUE	TRUE
8	0	TRUE	FALSE	22	3	TRUE	FALSE	48	0	TRUE	TRUE
8	1	TRUE	FALSE	22	4	TRUE	FALSE	49	0	TRUE	TRUE
8	2	TRUE	FALSE	22	5	TRUE	FALSE	50	0	TRUE	TRUE
8	3	FALSE	TRUE	22	6	TRUE	FALSE	51	0	TRUE	TRUE
9	0	TRUE	FALSE	22	7	TRUE	FALSE	52	0	TRUE	FALSE
9	1	TRUE	FALSE	22	8	TRUE	FALSE	52	1	FALSE	TRUE
9	2	FALSE	TRUE	22	9	FALSE	TRUE	53	0	TRUE	TRUE
10	0	TRUE	FALSE	23	0	TRUE	FALSE	54	0	TRUE	FALSE
10	1	TRUE	FALSE	23	1	TRUE	FALSE	54	1	TRUE	FALSE
10	2	TRUE	FALSE	23	2	FALSE	TRUE	54	2	TRUE	FALSE
10	3	TRUE	FALSE	23	3	FALSE	TRUE	54	3	FALSE	TRUE
10	4	TRUE	FALSE	24	0	TRUE	TRUE	55	0	TRUE	FALSE
10	5	TRUE	FALSE	25	0	TRUE	TRUE	55	1	TRUE	FALSE
10	6	TRUE	FALSE	26	0	TRUE	FALSE	55	2	TRUE	FALSE
10	7	TRUE	FALSE	26	1	TRUE	FALSE	55	3	FALSE	TRUE
10	8	TRUE	FALSE	26	2	FALSE	TRUE	56	0	TRUE	TRUE
10	9	TRUE	FALSE	27	0	TRUE	FALSE	57	0	TRUE	FALSE
10	10	TRUE	FALSE	27	1	TRUE	FALSE	57	1	TRUE	FALSE
10	11	TRUE	FALSE	27	2	FALSE	TRUE	57	2	TRUE	FALSE
10	12	TRUE	FALSE	25	0	TRUE	TRUE	57	3	TRUE	FALSE
10	13	TRUE	FALSE	29	0	TRUE	TRUE	57	4	TRUE	FALSE
10	14	TRUE	FALSE	30	0	TRUE	TRUE	57	5	TRUE	FALSE
10	15	TRUE	FALSE	31	0	TRUE	TRUE	57	6	FALSE	TRUE
10	16	FALSE	TRUE	32	0	TRUE	FALSE	58	0	TRUE	FALSE
11	0	TRUE	FALSE	32	1	TRUE	FALSE	58	1	TRUE	FALSE
11	1	TRUE	FALSE	32	2	TRUE	FALSE	58	2	TRUE	FALSE
11	2	TRUE	FALSE	32	3	TRUE	FALSE	58	3	TRUE	FALSE
11	3	TRUE	FALSE	32	4	FALSE	TRUE	58	4	FALSE	TRUE
11	4	TRUE	FALSE	33	0	TRUE	TRUE	59	0	TRUE	TRUE
11	5	TRUE	FALSE	34	0	TRUE	TRUE	60	0	TRUE	TRUE
11	6	TRUE	FALSE	35	0	TRUE	TRUE	61	0	TRUE	TRUE
11	7	TRUE	FALSE	36	0	TRUE	FALSE	62	0	TRUE	TRUE
11	8	TRUE	FALSE	36	1	TRUE	FALSE	63	0	TRUE	FALSE
11	9	TRUE	FALSE	36	2	TRUE	FALSE	63	1	FALSE	TRUE
11	10	TRUE	FALSE	36	4	FALSE	TRUE	63	2	FALSE	TRUE
11	11	FALSE	TRUE	37	0	TRUE	TRUE	64	0	TRUE	FALSE
12	0	TRUE	FALSE	38	0	TRUE	FALSE	64	1	FALSE	TRUE
12	1	TRUE	FALSE	38	1	TRUE	FALSE	64	2	FALSE	TRUE
12	2	TRUE	FALSE	38	2	TRUE	FALSE	65	0	TRUE	TRUE
12	3	FALSE	TRUE	38	3	TRUE	FALSE	66	0	TRUE	TRUE
13	0	TRUE	FALSE	38	4	TRUE	FALSE	67	0	TRUE	TRUE
13	1	TRUE	FALSE	38	5	TRUE	FALSE	68	0	TRUE	TRUE
13	2	TRUE	FALSE	38	6	TRUE	FALSE	69	0	TRUE	TRUE
13	3	TRUE	FALSE	38	7	TRUE	FALSE	70	0	TRUE	TRUE
13	4	TRUE	FALSE	38	8	TRUE	FALSE	71	0	TRUE	TRUE
13	5	TRUE	FALSE	38	9	TRUE	FALSE	72	0	TRUE	FALSE
13	6	FALSE	TRUE	38	10	TRUE	FALSE	72	1	FALSE	TRUE
14	0	TRUE	TRUE	38	11	TRUE	FALSE	72	2	FALSE	TRUE
15	0	TRUE	FALSE	38	12	TRUE	FALSE	73	0	TRUE	TRUE
15	1	TRUE	FALSE	38	13	FALSE	TRUE	74	0	TRUE	FALSE
15	2	TRUE	FALSE	39	0	TRUE	FALSE	74	1	TRUE	FALSE
15	3	FALSE	TRUE	39	1	TRUE	FALSE	74	2	FALSE	TRUE
16	0	TRUE	FALSE	39	2	TRUE	FALSE				
16	1	TRUE	FALSE	39	3	TRUE	FALSE				

Table A.1: Results from the original contour-based object extraction algorithm.

idImages	idObject	Real	Detected	idImages	idObject	Real	Detected	idImages	idObject	Real	Detected
0	0	TRUE	TRUE	15	2	TRUE	FALSE	39	1	TRUE	TRUE
1	0	TRUE	TRUE	16	0	TRUE	TRUE	39	2	TRUE	FALSE
2	0	TRUE	TRUE	16	1	TRUE	TRUE	39	3	TRUE	FALSE
3	0	TRUE	TRUE	16	2	TRUE	FALSE	40	0	TRUE	TRUE
3	1	TRUE	TRUE	17	0	TRUE	TRUE	40	1	TRUE	TRUE
3	2	TRUE	TRUE	17	1	TRUE	TRUE	40	2	TRUE	FALSE
3	3	TRUE	TRUE	17	2	TRUE	TRUE	41	0	TRUE	TRUE
3	4	TRUE	FALSE	18	0	TRUE	TRUE	41	1	TRUE	TRUE
3	5	TRUE	FALSE	18	1	TRUE	TRUE	41	2	TRUE	FALSE
3	6	TRUE	FALSE	18	2	TRUE	TRUE	41	3	TRUE	FALSE
3	7	FALSE	TRUE	19	0	TRUE	FALSE	41	4	TRUE	FALSE
4	0	TRUE	TRUE	19	1	TRUE	FALSE	41	5	TRUE	FALSE
5	0	TRUE	TRUE	19	2	TRUE	FALSE	41	6	TRUE	FALSE
6	0	TRUE	TRUE	19	3	FALSE	TRUE	41	7	TRUE	FALSE
6	1	TRUE	TRUE	20	0	TRUE	FALSE	41	8	FALSE	TRUE
6	2	TRUE	TRUE	20	1	FALSE	TRUE	41	9	FALSE	TRUE
7	0	TRUE	FALSE	21	0	TRUE	FALSE	42	0	TRUE	TRUE
7	1	TRUE	FALSE	21	1	TRUE	FALSE	43	0	TRUE	TRUE
7	2	TRUE	FALSE	21	2	TRUE	FALSE	44	0	TRUE	TRUE
7	3	TRUE	FALSE	21	3	FALSE	TRUE	45	0	TRUE	TRUE
7	4	TRUE	FALSE	22	0	TRUE	TRUE	45	1	TRUE	TRUE
7	5	FALSE	TRUE	22	1	TRUE	TRUE	45	2	TRUE	TRUE
8	0	TRUE	FALSE	22	2	TRUE	FALSE	46	0	TRUE	TRUE
8	1	TRUE	FALSE	22	3	TRUE	FALSE	47	0	TRUE	TRUE
8	2	TRUE	FALSE	22	4	TRUE	FALSE	48	0	TRUE	TRUE
8	3	FALSE	TRUE	22	5	TRUE	FALSE	49	0	TRUE	TRUE
9	0	TRUE	FALSE	22	6	TRUE	FALSE	50	0	TRUE	TRUE
9	1	TRUE	FALSE	22	7	TRUE	FALSE	51	0	TRUE	TRUE
9	2	FALSE	TRUE	22	8	TRUE	FALSE	52	0	TRUE	FALSE
10	0	TRUE	TRUE	22	9	FALSE	TRUE	52	1	FALSE	TRUE
10	1	TRUE	TRUE	23	0	TRUE	FALSE	53	0	TRUE	TRUE
10	2	TRUE	TRUE	23	1	TRUE	FALSE	54	0	TRUE	FALSE
10	3	TRUE	TRUE	23	2	FALSE	TRUE	54	1	TRUE	FALSE
10	4	TRUE	TRUE	24	0	TRUE	TRUE	54	2	TRUE	FALSE
10	5	TRUE	FALSE	25	0	TRUE	FALSE	54	3	FALSE	TRUE
10	6	TRUE	FALSE	25	1	FALSE	TRUE	55	0	TRUE	FALSE
10	7	TRUE	FALSE	25	1	FALSE	TRUE	55	1	TRUE	FALSE
10	8	TRUE	FALSE	26	0	TRUE	TRUE	55	2	TRUE	FALSE
10	9	TRUE	FALSE	26	1	TRUE	TRUE	55	3	FALSE	TRUE
10	10	TRUE	FALSE	27	0	TRUE	FALSE	55	4	FALSE	TRUE
10	11	TRUE	FALSE	27	1	TRUE	TRUE	56	0	TRUE	TRUE
10	12	TRUE	FALSE	28	0	TRUE	TRUE	57	0	TRUE	TRUE
10	13	TRUE	FALSE	29	0	TRUE	TRUE	57	1	TRUE	TRUE
10	14	TRUE	FALSE	30	0	TRUE	TRUE	57	2	TRUE	TRUE
10	15	TRUE	FALSE	31	0	TRUE	TRUE	57	3	TRUE	TRUE
10	16	FALSE	TRUE	31	1	FALSE	TRUE	57	4	TRUE	TRUE
10	17	FALSE	TRUE	32	0	TRUE	TRUE	57	5	TRUE	FALSE
10	18	FALSE	TRUE	32	1	TRUE	FALSE	58	0	TRUE	TRUE
11	0	TRUE	FALSE	32	2	TRUE	FALSE	58	1	TRUE	TRUE
11	1	TRUE	FALSE	32	3	TRUE	FALSE	58	2	TRUE	TRUE
11	2	TRUE	FALSE	32	4	FALSE	TRUE	58	3	TRUE	FALSE
11	3	TRUE	FALSE	33	0	TRUE	TRUE	59	0	TRUE	TRUE
11	4	TRUE	FALSE	34	0	TRUE	TRUE	60	0	TRUE	TRUE
11	5	TRUE	FALSE	35	0	TRUE	TRUE	61	0	TRUE	TRUE
11	6	TRUE	FALSE	36	0	TRUE	FALSE	62	0	TRUE	TRUE
11	7	TRUE	FALSE	36	1	TRUE	FALSE	63	0	TRUE	FALSE
11	8	TRUE	FALSE	36	2	TRUE	FALSE	63	1	FALSE	TRUE
11	9	TRUE	FALSE	36	3	FALSE	TRUE	64	0	TRUE	FALSE
11	10	TRUE	FALSE	37	0	TRUE	TRUE	64	1	FALSE	TRUE
11	11	FALSE	TRUE	38	0	TRUE	FALSE	65	0	TRUE	TRUE
12	0	TRUE	TRUE	38	1	TRUE	FALSE	66	0	TRUE	TRUE
12	1	TRUE	TRUE	38	2	TRUE	FALSE	67	0	TRUE	TRUE
12	2	TRUE	TRUE	38	3	TRUE	FALSE	68	0	TRUE	FALSE
13	0	TRUE	FALSE	38	4	TRUE	FALSE	68	1	FALSE	TRUE
13	1	TRUE	FALSE	38	5	TRUE	FALSE	69	0	TRUE	TRUE
13	2	TRUE	FALSE	38	6	TRUE	FALSE	70	0	TRUE	TRUE
13	3	TRUE	FALSE	38	7	TRUE	FALSE	71	0	TRUE	FALSE
13	4	TRUE	FALSE	38	8	TRUE	FALSE	71	1	FALSE	TRUE
13	5	TRUE	FALSE	38	9	TRUE	FALSE	71	2	FALSE	TRUE
13	6	FALSE	TRUE	38	10	TRUE	FALSE	72	0	TRUE	TRUE
14	0	TRUE	TRUE	38	11	TRUE	FALSE	73	0	TRUE	TRUE
14	1	FALSE	TRUE	38	12	TRUE	FALSE	74	0	TRUE	TRUE
15	0	TRUE	TRUE	38	13	FALSE	TRUE	74	1	TRUE	TRUE
15	1	TRUE	FALSE	39	0	TRUE	TRUE				

Table A.2: Results from the proposed contour-based object extraction algorithm.

A.2 Unsupervised subset selection

N	Category	Initial Set		Selection Set		Percentages (%)		
		Total Objects	Good Objects	Total Objects	Good Objects	Good Original	Good Selection	Variation
0	Banana	30	13	8	3	43,33%	37,50%	-5,83%
1	Bottle	32	12	12	4	37,50%	33,33%	-4,17%
2	Digital Camera	32	15	7	5	46,88%	71,43%	24,55%
3	Dinosaur	35	7	13	0	20,00%	0,00%	-20,00%
4	Electric Iron	29	10	6	4	34,48%	66,67%	32,18%
5	Eyeglasses	30	13	6	3	43,33%	50,00%	6,67%
6	Fax Machine	30	12	11	8	40,00%	72,73%	32,73%
7	Fork	32	8	9	8	25,00%	88,89%	63,89%
8	Frying Pan	29	21	8	6	72,41%	75,00%	2,59%
9	Green Apple	32	9	9	2	28,13%	2,22%	-5,90%
10	Laptop	31	14	6	5	45,16%	83,33%	38,17%
11	Orange	34	5	9	0	14,71%	0,00%	-14,71%
12	Pumpkin	32	2	11	0	6,25%	0,00%	-6,25%
13	Red Bell Pepper	31	9	9	2	29,03%	22,22%	-6,81%
14	Red Ping Pong Paddle	33	7	12	5	21,21%	41,67%	20,45%
15	Red Plastic Cup	31	9	8	0	29,03%	0,00%	-29,03%
16	Remote Control	31	9	10	1	29,03%	10,00%	-19,03%
17	Rolling Suitcase	33	10	8	1	30,30%	12,50%	-17,80%
18	Saucepan	30	21	8	8	70,00%	100,00%	30,00%
19	Scientific Calculator	31	10	7	5	32,26%	71,43%	39,17%
20	Toy Car	31	12	5	1	38,71%	20,00%	-18,71%
21	Ulu	31	7	6	4	22,58%	66,67%	44,09%
22	Upright Vacuum Cleaner	30	13	8	6	43,33%	75,00%	31,67%
23	White Soccer Ball	35	14	9	7	40,00%	77,78%	37,78%
					Average:	35,11%	45,77%	10,65%

Table A.3: Results from the original subset selection algorithm.

N	Category	Initial Set		Selection Set		Percentages (%)		
		Total Objects	Good Objects	Total Objects	Good Objects	Good Original	Good Selection	Variation
0	Banana	30	13	3	3	43,33%	100,00%	56,67%
1	Bottle	32	12	5	2	37,50%	40,00%	2,50%
2	Digital Camera	32	15	4	3	46,88%	75,00%	28,13%
3	Dinosaur	35	7	2	2	20,00%	100,00%	80,00%
4	Electric Iron	29	10	3	0	34,48%	0,00%	-34,48%
5	Eyeglasses	30	13	4	3	43,33%	75,00%	31,67%
6	Fax Machine	30	12	6	6	40,00%	100,00%	60,00%
7	Fork	32	8	4	4	25,00%	100,00%	75,00%
8	Frying Pan	29	21	5	4	72,41%	80,00%	7,59%
9	Green Apple	32	9	3	3	28,13%	100,00%	71,88%
10	Laptop	31	14	4	4	45,16%	100,00%	54,84%
11	Orange	34	5	3	3	14,71%	100,00%	85,29%
12	Pumpkin	32	2	6	0	6,25%	0,00%	-6,25%
13	Red Bell Pepper	31	9	6	3	29,03%	50,00%	20,97%
14	Red Ping Pong Paddle	33	7	2	0	21,21%	0,00%	-21,21%
15	Red Plastic Cup	31	9	5	3	29,03%	60,00%	30,97%
16	Remote Control	31	9	4	2	29,03%	50,00%	20,97%
17	Rolling Suitcase	33	10	6	2	30,30%	33,33%	3,03%
18	Saucepan	30	21	6	6	70,00%	100,00%	30,00%
19	Scientific Calculator	31	10	5	2	32,26%	40,00%	7,74%
20	Toy Car	31	12	5	2	38,71%	40,00%	1,29%
21	Ulu	31	7	6	2	22,58%	33,33%	10,75%
22	Upright Vacuum Cleaner	30	13	9	7	43,33%	77,78%	34,44%
23	White Soccer Ball	35	14	9	9	40,00%	100,00%	60,00%
						35,11%	64,77%	29,66%

Table A.4: Results from the proposed subset selection algorithm.

N	Category	Initial Set		Selection Set		Percentages (%)		
		Total Objects	Good Objects	Total Objects	Good Objects	Good Original	Good Selection	Variation
0	Banana	37	10	7	5	27,03%	71,43%	44,40%
1	Bottle	40	26	12	11	65,00%	91,67%	26,67%
2	Digital Camera	35	18	9	7	51,43%	77,78%	26,35%
3	Dinosaur	55	8	15	5	14,55%	33,33%	18,79%
4	Electric Iron	32	12	7	4	37,50%	57,14%	19,64%
5	Eyeglasses	49	14	16	12	28,57%	75,00%	46,43%
6	Fax Machine	32	18	16	13	56,25%	81,25%	25,00%
7	Fork	35	10	7	7	28,57%	100,00%	71,43%
8	Frying Pan	32	18	12	9	56,25%	75,00%	18,75%
9	Green Apple	37	9	7	5	24,32%	71,43%	47,10%
10	Laptop	34	16	6	4	47,06%	66,67%	19,61%
11	Orange	55	6	4	3	10,91%	75,00%	64,09%
12	Pumpkin	45	1	14	1	2,22%	7,14%	4,92%
13	Red Bell Pepper	35	10	7	6	28,57%	85,71%	57,14%
14	Red Ping Pong Paddle	35	11	18	10	31,43%	55,56%	24,13%
15	Red Plastic Cup	36	11	6	5	30,56%	83,33%	52,78%
16	Remote Control	35	14	10	10	40,00%	100,00%	60,00%
17	Rolling Suitcase	34	14	11	4	41,18%	36,36%	-4,81%
18	Saucepan	29	21	3	3	72,41%	100,00%	27,59%
19	Scientific Calculator	27	11	7	6	40,74%	85,71%	44,97%
20	Toy Car	36	19	7	5	52,78%	71,43%	18,65%
21	Ulu	35	7	6	5	20,00%	83,33%	63,33%
22	Upright Vacuum Cleaner	31	25	9	9	80,65%	100,00%	19,35%
23	White Soccer Ball	31	20	11	11	64,52%	100,00%	35,48%
						39,69%	74,35%	34,66%

Table A.5: Results from the proposed subset selection combined with the contour-based object extraction.

A.3 Classifiers

N categories	NN	StdDev	K-NN	StdDev	W-K-NN	StdDev
5	94,67%	4,99%	86,67%	7,30%	94,67%	4,99%
10	85,33%	5,42%	74,67%	7,18%	82,00%	7,48%
15	89,78%	3,01%	78,22%	2,95%	84,44%	2,43%
20	83,00%	5,52%	73,00%	7,41%	77,00%	8,19%
25	80,27%	2,59%	70,67%	2,39%	77,07%	2,59%
30	80,00%	2,90%	70,44%	2,86%	76,00%	1,13%
35	79,43%	0,97%	69,14%	3,22%	75,62%	2,30%
40	81,67%	4,38%	67,17%	3,14%	76,33%	4,03%
45	76,30%	6,85%	65,78%	5,43%	72,59%	5,20%
50	80,00%	2,39%	66,53%	3,44%	74,67%	0,73%
55	76,24%	3,12%	62,67%	2,62%	70,55%	2,09%
60	75,67%	2,37%	64,67%	3,25%	72,22%	1,69%
65	75,79%	1,76%	64,51%	1,82%	71,28%	1,17%
Average	81,40%	3,56%	70,32%	4,08%	77,26%	3,39%

Table A.6: Results from three classifiers for the 68 categories set.

N categories	NN	StdDev	K-NN	StdDev	W-K-NN	StdDev
5	94,67%	4,99%	92,00%	6,53%	93,33%	5,96%
10	88,67%	6,18%	85,33%	7,77%	86,00%	7,42%
15	92,00%	4,99%	84,89%	4,95%	86,67%	4,22%
20	88,00%	4,64%	80,67%	5,44%	85,67%	4,29%
25	84,27%	1,77%	74,93%	6,28%	78,67%	4,46%
30	84,00%	0,54%	72,89%	4,80%	77,56%	3,01%
35	81,71%	2,36%	72,95%	3,93%	77,14%	4,13%
40	82,17%	1,45%	75,67%	3,05%	79,33%	1,93%
45	78,52%	3,51%	70,67%	3,13%	74,67%	4,43%
Average	86,00%	3,38%	78,89%	5,10%	82,11%	4,43%

Table A.7: Results from three classifiers for the 49 categories set.

N categories	NN	StdDev	K-NN	StdDev	W-K-NN	StdDev
5	1,33%	2,67%	0,00%	0,00%	1,33%	2,67%
10	13,33%	4,22%	12,67%	8,79%	13,33%	7,60%
15	9,78%	6,68%	8,00%	5,73%	10,22%	6,53%
20	10,33%	3,86%	14,67%	3,23%	11,67%	2,79%
25	10,40%	3,42%	10,93%	2,72%	10,40%	2,72%
30	14,22%	2,76%	9,78%	1,47%	12,89%	2,39%
35	14,10%	3,04%	13,71%	2,14%	15,24%	3,01%
40	13,50%	2,38%	13,67%	2,82%	15,33%	2,01%
45	14,52%	2,63%	14,22%	1,78%	15,56%	2,61%
Average	11,28%	3,52%	10,85%	3,19%	11,77%	3,59%

Table A.8: Results from three classifiers for the 49 categories set, with maximum inverse χ^2 distance membership measure. Unsupervised training phase.

N categories	NN	StdDev	K-NN	StdDev	W-K-NN	StdDev
5	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
10	11,33%	5,42%	8,67%	8,59%	11,33%	6,86%
15	7,11%	3,82%	4,00%	2,95%	7,11%	5,14%
20	8,67%	2,87%	7,00%	3,86%	9,33%	2,26%
25	4,00%	2,53%	6,93%	2,72%	5,33%	1,69%
30	9,11%	1,78%	7,56%	1,91%	9,56%	1,33%
35	12,38%	2,00%	12,19%	1,85%	13,52%	2,12%
40	11,17%	1,94%	8,33%	1,75%	11,33%	2,51%
45	11,41%	1,53%	8,44%	1,91%	11,70%	1,44%
Average	8,35%	2,43%	7,01%	2,84%	8,80%	2,59%

Table A.9: Results from three classifiers for the 49 categories set, with penalized membership measure. Unsupervised training phase.

Bibliography

[1] Luís Miguel Saraiva Ribeiro. Object recognition for semantic robot vision. Master's thesis, Universidade de Aveiro, 2008.

[2] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150 –1157 vol.2, 1999.

[3] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

[4] Milan Sonka, Václav Hlavác, and Roger Boyle. *Image processing, analysis and and machine vision (3. ed.)*. Thomson, 2008.

[5] D. Celik and A. Elgi. A semantic search agent approach: finding appropriate semantic web services based on user request term(s). In *Information and Communications Technology, 2005. Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference on*, pages 675 –687, dec. 2005.

[6] Tao Mei, Yong Wang, Xian-Sheng Hua, Shaogang Gong, and Shipeng Li. Coherent image annotation by learning semantic distance. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, june 2008.

[7] Rui Manuel Fernades Pereira. Semantic image retrieval and subset selection for robot vision. Master's thesis, Universidade de Aveiro, 2008.

[8] Rui Pereira and Luís Seabra Lopes. Learning visual object categories with global descriptors and local features. In *Proceedings of the 14th Portuguese Conference on Artificial Intelligence: Progress in Artificial Intelligence*, EPIA '09, pages 225–236, Berlin, Heidelberg, 2009. Springer-Verlag.

[9] Rui Pereira, Luís Seabra Lopes, and Augusto Silva. Semantic image search and subset selection for classifier training in object recognition. In *Proceedings of the 14th Portuguese*

Conference on Artificial Intelligence: Progress in Artificial Intelligence, EPIA '09, pages 338–349, Berlin, Heidelberg, 2009. Springer-Verlag.

[10] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679 –698, nov. 1986.

[11] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(6):583 –598, jun 1991.

[12] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 1(4):321–331, 1988.

[13] D.R. Martin, C.C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(5):530 –549, may 2004.

[14] J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: cue integration in image segmentation. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 918 –925 vol.2, 1999.

[15] Bryan Catanzaro, Bor-Yiing Su, Narayanan Sundaram, Yunsup Lee, Mark Murphy, and Kurt Keutzer. Efficient, high-quality image contour detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2381 –2388, 29 2009-oct. 2 2009.

[16] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 731 –737, jun 1997.

[17] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1816 –1823 Vol. 2, oct. 2005.

[18] Markus Weber, Max Welling, and Pietro Perona. Unsupervised learning of models for recognition. In *Proceedings of the 6th European Conference on Computer Vision-Part I*, ECCV '00, pages 18–32, London, UK, 2000. Springer-Verlag.

[19] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–264 – II–271 vol.2, june 2003.

[20] Robert Fergus, Pietro Perona, and Andrew Zisserman. A Visual Category Filter for Google Images. pages 242–256. 2004.

[21] Li Fe-Fei, R. Fergus, and P. Perona. A bayesian approach to unsupervised one-shot learning of object categories. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1134 –1141 vol.2, oct. 2003.

[22] Josef Sivic, Bryan C. Russell, Alexei A. Efros, Andrew Zisserman, William T. Freeman, Josef Sivic, Bryan C. Russell, Alexei A. Efros, Andrew Zisserman, and William T. Freeman. W.: Discovering object categories in image collections. In *In: Proceedings of the Tenth International Conference on Computer Vision*, 2005.

[23] K. Grauman and T. Darrell. Unsupervised learning of categories from sets of partially matching image features. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 19 – 25, june 2006.

[24] Tamara L. Berg and David A. Forsyth. Animals on the web. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 1463–1470, Washington, DC, USA, 2006. IEEE Computer Society.

[25] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting image databases from the web. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1 –8, oct. 2007.

[26] S. Vijayanarasimhan and K. Grauman. Keywords to visual categories: Multiple-instance learning forweakly supervised object categorization. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, june 2008.

[27] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):509 –522, apr 2002.

[28] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[29] Deb Kumar Roy. *Learning words from sights and sounds: a computational model*. PhD thesis, 1999. AAI0801519.

[30] Bo Wang, Xiang Bai, Xinggang Wang, Wenyu Liu, and Zhuowen Tu. Object recognition using junctions. In *Proceedings of the 11th European conference on Computer vision: Part V*, ECCV'10, pages 15–28, Berlin, Heidelberg, 2010. Springer-Verlag.

[31] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, june 2008.

[32] T. Pavlidis. *Algorithms for Graphics and Image Processing*, chapter 7. Springer, 1982.

[33] L.G. Shapiro and G.C. Stockman. *Computer Vision*. Prentice Hall, 2001.

[34] H. Steinhaus. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1:801–804, 1956.

[35] S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129 – 137, mar 1982.

[36] Richard Durstenfeld. Algorithm 235: Random permutation. *Commun. ACM*, 7:420–, July 1964.

[37] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[38] Evelyn Fix and Jr. Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties. Technical Report Project 21-49-004, Report Number 4, USAF School of Aviation Medicine, Randolf Field, Texas, 1951.

[39] Sahibsingh A. Dudani. The distance-weighted k-nearest-neighbor rule. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-6(4):325 –327, april 1976.

[40] Luis Seabra Lopes and Aneesh Chauhan. Open-ended category learning for language acquisition. *Connect. Sci*, 20:277–297, December 2008.