Departamento de
Universidade de Aveiro Electrónica, Telecomunicações e Informática,
2008

Rui Manuel Fernandes Pereira

Pesquisa Semântica e Selecção de Imagens para Visão Robótica
Semantic Image Retrieval and Subset Selection for Robot Vision

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

o júri / the jury

presidente / president

Doutor Paulo Jorge dos Santos Gonçalves Ferreira
Professor Catedrático da Universidade de Aveiro

vogais / examiners committee

Doutor Jorge Manuel Miranda Dias
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Doutor Luís Filipe de Seabra Lopes
Professor Auxiliar da Universidade de Aveiro (Orientador)

Doutor Augusto Marques Ferreira da Silva
Professor Auxiliar da Universidade de Aveiro (Co-orientador)

agradecimentos / acknowledgements

É com muito gosto que aproveito esta oportunidade para agradecer a todos os que me ajudaram durante a realização deste trabalho. Começo por agradecer aos meus orientadores, Prof. Luís Seabra Lopes e Prof. Augusto Silva, pelo apoio no trabalho realizado e pelas sugestões para a correcção e realização desta dissertação.

Desejo também agradecer aos colegas de equipa, Luís Ribeiro e Frederico Valente, pelo excelente trabalho desenvolvido bem como pelo apoio prestado. Agradeço ainda ao nosso colega de laboratório, Aneesh Chauhan, pela ajuda que prestou e disponibilidade demonstrada.

Finalmente, quero agradecer e dedicar este trabalho à minha namorada Liliana, pais Domingos e Gina, padrinhos Resende e Licínia e ao meu irmão Mário. Foram eles em conjunto que, embora entrando na minha vida em diferentes alturas, me deram a maior compreensão e ajuda que foram cruciais ao longo de todo o meu percurso académico. Sem eles, nada disto teria sido possível.

Palavras-chave

Semantic Image Retrieval, Subset Selection, Robot Vision, Clustering, Object/Scene Representation and Classification, SIFT, Shape Context, Similarity Measures, SRVC

Resumo

Pesquisa semântica e selecção de sub-conjuntos de imagens são um instrumento valioso na produção de bases de conhecimento para reconhecimento de objectos e visão semântica em robótica. Estas técnicas são também úteis para catalogar e organizar grandes colecções de imagens, tanto para uso pessoal como profissional.

Foi desenvolvido um sistema que, dada uma lista de categorias de objectos, tem a capacidade de recuperar da Web imagens potencialmente representativas dessas categorias e subsequentemente tentar identificar as mais representativas, descartando as restantes. Também foi realizado algum trabalho no uso das imagens seleccionadas como boas representações das categorias como exemplos de treino para reconhecimento de objectos em cenas complexas.

O objectivo principal do trabalho foi assim produzir parte de um sistema mais amplo que, quando completo, seria capaz de participar no *Semantic Robot Vision Challenge* e, apoiando-se apenas numa lista de nomes de categorias, construir os seus modelos e reconhecê-las no palco da competição.

Usou-se uma abordagem baseada em agrupar as imagens recuperadas segundo similaridades par a par, usando descritores globais de forma para categorias gerais e características locais SIFT para categorias específicas. Ambas as representações são também usadas no processo de classificação.

Aprendemos que é possível obter boa selecção de sub-conjuntos usando como critério o número de vezes que uma imagem aparece no maior grupo, admitindo que existem suficientes boas representações de categorias em cada conjunto. Concluiu-se também que *Shape Contexts* podem ser usados como descritores globais, com bons resultados para categorias gerais, e que SIFT é melhor para categorias específicas.

Adicionalmente, ao integrar este trabalho no sistema global, verificou-se que, combinando classificadores baseados em diferentes descritores globais de forma, usando um sistema de votos, se obtêm melhores resultados do que com os classificadores individuais.

Os resultados da avaliação deste trabalho expõem os seus pontos fortes e fracos, bem como das suas partes individuais.

Keywords Semantic Image Retrieval, Subset Selection, Robot Vision, Clustering, Object/Scene Representation and Classification, SIFT, Shape Context, Similarity Measures, SRVC

Abstract Semantic image retrieval and subset selection are valuable tools in producing knowledge bases to support object recognition. These techniques are also useful for cataloging and organizing large image collections, whether for personal or corporate use.

We have developed a system that, given a list of object categories, is able to retrieve potentially representative images of those categories from the Web and subsequently try to separate the useful images from the coexisting non-representative images. Some work was also performed in using the images selected as good category instances as training examples for object recognition in complex scenes.

The main objective of the work was to produce part of a wider system that, upon completion, would be able to enter the Semantic Robot Vision Challenge, where, relying only on a list of category names, would build their models and recognize them in the competition scenario.

We used an approach based on unsupervised clustering of the retrieved images according to pairwise similarity, using Shape Contexts as global descriptors for general categories and SIFT local features for specific categories. Both types of representations are also used in the classification process.

We learned that it is possible to achieve good subset selection using as a criterion the number of times an image appears in the biggest cluster, providing there are sufficient good representative images in each set. We also concluded that Shape Context can be used as a global descriptor (Global Shape Context - GSC), with good results for general categories and that SIFT is better for specific categories.

Additionally, by integrating this work in the global system, we also learned that combining the Global Shape Context classifier with a Roy's Shape Representation (RSR) classifier, via a voting system, yields better results than the individual classifiers.

The results on the evaluation of this work outline its strong and weak points, both as whole and as a coherent set of individual parts.

Contents

1 Introduction **1**
- 1.1 Semantic image retrieval . . . 1
- 1.2 Object recognition . . . 2
- 1.3 Semantic robot vision . . . 3
- 1.4 Semantic Robot Vision Challenge . . . 3
- 1.5 Objectives . . . 4
- 1.6 Overview of UA@SRVC . . . 4
- 1.7 Dissertation structure . . . 6

2 Previous work **9**
- 2.1 Content based image retrieval . . . 9
  - 2.1.1 CBIR using category based indexing . . . 9
  - 2.1.2 CBIR using color, texture and shape features . . . 11
  - 2.1.3 CBIR using geometrical shape of objects in image . . . 13
  - 2.1.4 CBIR using multiple shape descriptors . . . 14
  - 2.1.5 An image semantic retrieval system design and realization . . . 16
  - 2.1.6 Scene image classification and segmentation with quantized local descriptors and latent aspect modeling . . . 17
  - 2.1.7 Keywords to visual categories: multiple-instance learning for weakly supervised object categorization . . . 18
- 2.2 Data clustering and subset selection . . . 19
  - 2.2.1 Data clustering algorithms . . . 19
  - 2.2.2 Unsupervised feature subset selection (UFSS) . . . 20
- 2.3 Object representations and recognition . . . 22
  - 2.3.1 Object representations . . . 22
  - 2.3.2 Object recognition . . . 23

3 Representations and similarity measures **25**
- 3.1 General and specific categories . . . 25
- 3.2 Shape representations and measures . . . 26
  - 3.2.1 Shape Context . . . 27
  - 3.2.2 The Shape Context as a global descriptor . . . 30
  - 3.2.3 Roy's Shape Representation (RSR) . . . 33
- 3.3 Scale Invariant Feature Transform (SIFT) . . . 34
  - 3.3.1 Scale-space extrema detection . . . 34
  - 3.3.2 Keypoint localization . . . 36

3.3.3 Orientation assignment 36
3.3.4 Keypoint descriptor 37
3.3.5 Matching 37
3.3.6 Implementation 39
3.4 Category representation 40

4 From Web search to classification **41**
4.1 Image retrieval 41
4.2 Pre-processing 42
4.2.1 Object extraction for shape analysis 42
4.2.2 Filtering 46
4.3 Subset selection 46
4.3.1 Subset selection through clustering 46
4.3.2 Similarity matrices 46
4.3.3 Object clustering algorithm 47
4.3.4 Object ranking 48
4.4 Classification 49
4.4.1 The matching strategy 50
4.4.2 SIFT classifier 51
4.4.3 Shape Context and RSR classifiers 51
4.4.4 Voting system 51

5 Performance evaluation **53**
5.1 Subset selection of the Web retrieved objects 53
5.1.1 Shape-based subset selection 53
5.1.2 SIFT-based subset selection 57
5.2 Classification 60
5.2.1 General categories classification 60
5.2.2 Specific categories classification 63
5.3 The SRVC scenario 65

6 Conclusions and future work **69**
6.1 Overview 69
6.2 Conclusions 69
6.3 Future work 70

A Performance tables **73**

List of Figures

1.1 Google search for "hammer" . . . 2
1.2 System overview. . . . 5

2.1 Hiremath and Pujar' system overview . . . 11
2.2 Bipartite graph showing 4 tiles of two images. . . . 12
2.3 Image similarity computation based on MSHP principle. . . . 12
2.4 Adnan et. al' system architecture . . . 13
2.5 Sarfraz and Rhida area ratios. . . . 15
2.6 A semantic network . . . 16
2.7 A hyponymy relation. . . . 17
2.8 Pyramid match kernel . . . 23

3.1 Shape category . . . 26
3.2 SIFT category . . . 26
3.3 Image contours of a hammer . . . 27
3.4 Shape Context of a point . . . 27
3.5 Shape Context and matching . . . 28
3.6 Hungarian algorithm . . . 30
3.7 Shape Context as a global descriptor . . . 31
3.8 Rotation of the Shape Context histogram . . . 32
3.9 Roy's Shape Representation . . . 33
3.10 Linear scale-space representations for different scales . . . 34
3.11 Difference-of-Gaussians . . . 35
3.12 Local extrema detection . . . 36
3.13 Image gradients . . . 37
3.14 Keypoint descriptor creation . . . 38
3.15 SIFT matching . . . 40

4.1 Segmentation example . . . 43
4.2 Extraction cases . . . 43
4.3 Contour aggregation . . . 44
4.4 Example of contour aggregation . . . 45
4.5 Similarity matrix . . . 47
4.6 *K-means* clustering . . . 48
4.7 Classification process . . . 50

5.1 Examples of bad shape representations . . . 54

5.2 Shape: good images before and after selection without object extraction . . . 55
5.3 Shape: good subset images in function of good initial images without object extraction . . . 55
5.4 Shape: good images before and after selection with object extraction . . . 56
5.5 Shape: good subset images in function of good initial images with object extraction . . . 57
5.6 Good and bad images for SIFT modeling . . . 58
5.7 SIFT: Good images before and after selection . . . 59
5.8 Aerius: two different, valid brandings . . . 59
5.9 SIFT: good subset images in function of good initial images . . . 60
5.10 Shape classifiers and voting system on general categories . . . 61
5.11 Shape-based classifiers, SIFT and voting system on general categories . . . 62
5.12 SIFT on specific categories . . . 64
5.13 Shape classifiers, SIFT and voting system on specific categories . . . 65
5.14 Our results (2nd place) . . . 66
5.15 1st place results . . . 67

List of Tables

2.1 Image categories as proposed by Wardhani and Thomson 10

5.1 General categories list for subset selection 54
5.2 Specific categories list for subset selection 58
5.3 General categories list for classification 61
5.4 Specific categories list for classification 63
5.5 List of categories to be found in the SRVC'08 65

A.1 Analysis of shape-based subset selection without object extraction 74
A.2 Analysis of shape-based subset selection with object extraction 75
A.3 Analysis of SIFT-based subset selection 76

Chapter 1

Introduction

Summary

In this first chapter, a brief introduction to the tackled problem, field and motivations is presented. We start by clarifying the concepts of object recognition and semantic image retrieval and proceed to the motivations to this work. Next, we explore the motivations of semantic robot vision, allied to the possibility of enrolling the *Semantic Robot Vision Challenge*. Finally, we present the structure upon which the rest of this dissertation is organized.

1.1 Semantic image retrieval

According to the dictionary, semantics is "*the meaning, or an interpretation of the meaning, of a word, sign, sentence*". Semantic image retrieval is then the ability to, given a list of characters grouped together as words, determine their meaning by associating an image to them. These words are not comprehensible to a computer, because grammar is a conceptualization: words don't have meaning *per se*, they're just a group of characters or sounds, to which we humans attribute a significance.

In our work, the system doesn't have any previous knowledge about the world. All the information available is a list of names of categories to be found in the Internet, i.e., we will take a category name and provide meaning to it by relying on the Web to retrieve images from the same category and use these images to build a model for it, effectively learning what that category is. We modified an abstraction to something the computer can understand and use for recognizing it again in the future. The problem with this approach is that most current web services are not semantic and web searching is basically syntactic-based[9]. That, allied to the exponential growth of image content in the Internet, makes it not possible to straightforwardly do a web search and use all the obtained information to build a category model. Because of the quantity of unrelated results we get, as can be seen on Figure 1.1, the model wouldn't be faithful. In the following chapters we'll discuss the techniques to tackle this problem.

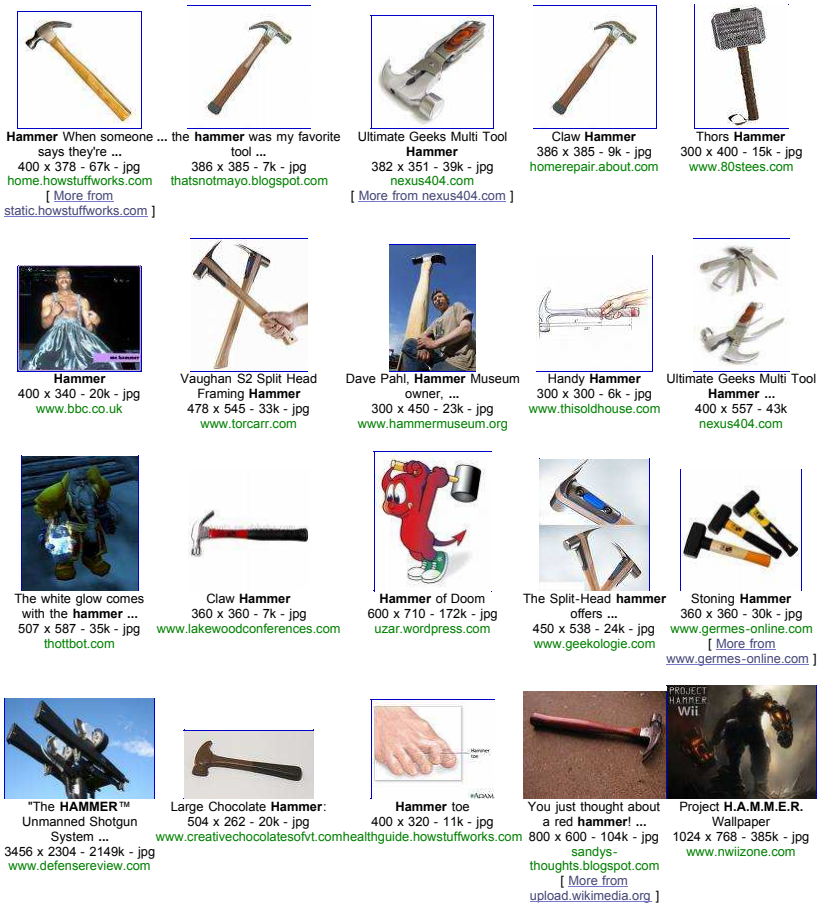

Figure 1.1: Google search for "hammer". Not all results are canonical hammers.

1.2 Object recognition

In computational terms, object recognition is the task of, given an input image, being able to equate and discern that image within previously constructed models. Specifically, it means the ability to categorize a given object in the input image or pinpoint where in the image the desired object is.

The disparities that two-dimensional representations (images) of a three-dimensional object can have, such as viewpoint discrepancies, scale, occlusion, etc., imply that a transformation is needed to neutralize the variation[17] and make the recognition possible. Even then, such other factors as ambient or direct lighting and background clutter make the task even more complicated.

Tackling these challenges, with more or less complex methods, we developed a system that after an initial phase, where the learning takes place and category models are created, will be able to recognize objects in a complex scene belonging to those categories.

1.3 Semantic robot vision

Computer vision and perception offer unique and desirable prospects to the field of Artificial Intelligence and all people in general. Sight is arguably the most important human sense and it's of unrivaled importance to the comprehension and interaction with the world that surrounds us. Applications in the fields of medicine, autonomous vehicles, astronomy and military are vast and will soon enter everyday life.

The motivation to engage in this sphere of research, allied to the incentive that was the possibility of attending and participating in the SRVC (described below), helped to focus on the objective of building components of an integrated system which, upon completion, would be able to recognize objects using a database of self constructed models.

The integration of both semantic image retrieval and subset selection with object recognition, or classification, is a semantic robot vision system, to which we called ***UA@SRVC***.

1.4 Semantic Robot Vision Challenge

This dissertation documents the work of the author produced in the development of a system for the software-only league of the Semantic Robot Vision Challenge [1], which was held in Anchorage, Alaska, in association with the CVPR[2]' 2008. This year's edition counted with sponsorships by Google and NSF[3].

In the SRVC, participating teams, that can be either hardware/software teams or software-based teams, are given a list of objects to be found in a given environment. After being presented with the list of the objects, the teams have to rely on the Internet and publicly accessible data to retrieve images from these same object categories and build models of them. Any previous knowledge embedded in the contestant agents is forbidden and, as such, this web search is the only method of learning that can be used.

In the hardware/software league (robot league), agents are free to roam the environment taking pictures of any desired location and using them to pinpoint the objects.

In the software league, however, images of the environment are not controlled by the participants: instead, the competition organizers provide several images of this environment. These images may or may not contain any or several of the desired items because pictures of the set are taken as randomly as possible. As in the robot league, the images are used to locate the objects.

Since the participation was on a team-level, some work that wasn't carried out by this author is also documented, because of its relevance to the subjects exposed in this document.

[1] *http://www.semantic-robot-vision-challenge.org/*
[2] Computer Society Conference on Computer Vision and Pattern Recognition
[3] National Science Foundation

1.5 Objectives

The objectives of our work were, given a name of a category and relying only in Web searching, to be able to build a correct model, or abstraction, of that category and find an instance of it in a scene. This can be decomposed in two sub-objectives: **learning** and **recognizing**. The work by the author was focused in the learning sub-objective, with the objectives of:

- Search and retrieve images from the Web given a list of objects.
- Pre-process the retrieved images to extract the depicted objects and remove noise.
- Filter out bad representations of the categories or junk images.
- Build models for the categories that are invariant to scale, rotation and deformation

Some work on the recognition phase was also carried out, namely in the object classification part.

1.6 Overview of UA@SRVC

The complete system integrates both the learning and recognition sub-objectives. In the learning phase, the system takes as input a list of categories' names and starts by searching and retrieving images from the Web, using queries with each category's name. These images suffer a process of object extraction, in which images that contain more than one object are segmented into sub-images where, ideally, each of those contains only one object. In this process we also try to ignore possible noise. After this initial pre-processing step, we separate the images into two subsets, where one has the images that contain the desired category and the other does not. The images that do not contain the category are to be discarded. The learning phase is concluded by building models for the categories that are invariant to scale, rotation and deformation. In the recognition phase we must use the previously gained knowledge to identify and pinpoint objects of the desired categories in complex scenes. This phase takes as input a list of images of complex scenes that may or may not contain the object. It starts by segmenting these images using a color saliency detector and then tries to extract the detected objects while ignoring noise. The learned models, built in the previous phase, are used to match each extracted object and identify the one that best depicts the desired category. Finally, the identified objects are marked by bounding-boxes in the original images.

In Figure 1.2 is shown a basic block diagram of the developed system. The architecture uses *C/C++* compiled code for the main programs and interpreted *bash* scripts to control the flow and invoking the applications. Image processing makes use of *Intel OpenCV*[4], a free cross-platform computer vision library. *Imagemagick*[5], a software for image manipulation, is used for some image format conversions. Image fetching uses a *Perl* script with the *WWW::Google::Images* module.

[4] *http://opencvlibrary.sourceforge.net*
[5] *http://www.imagemagick.org*

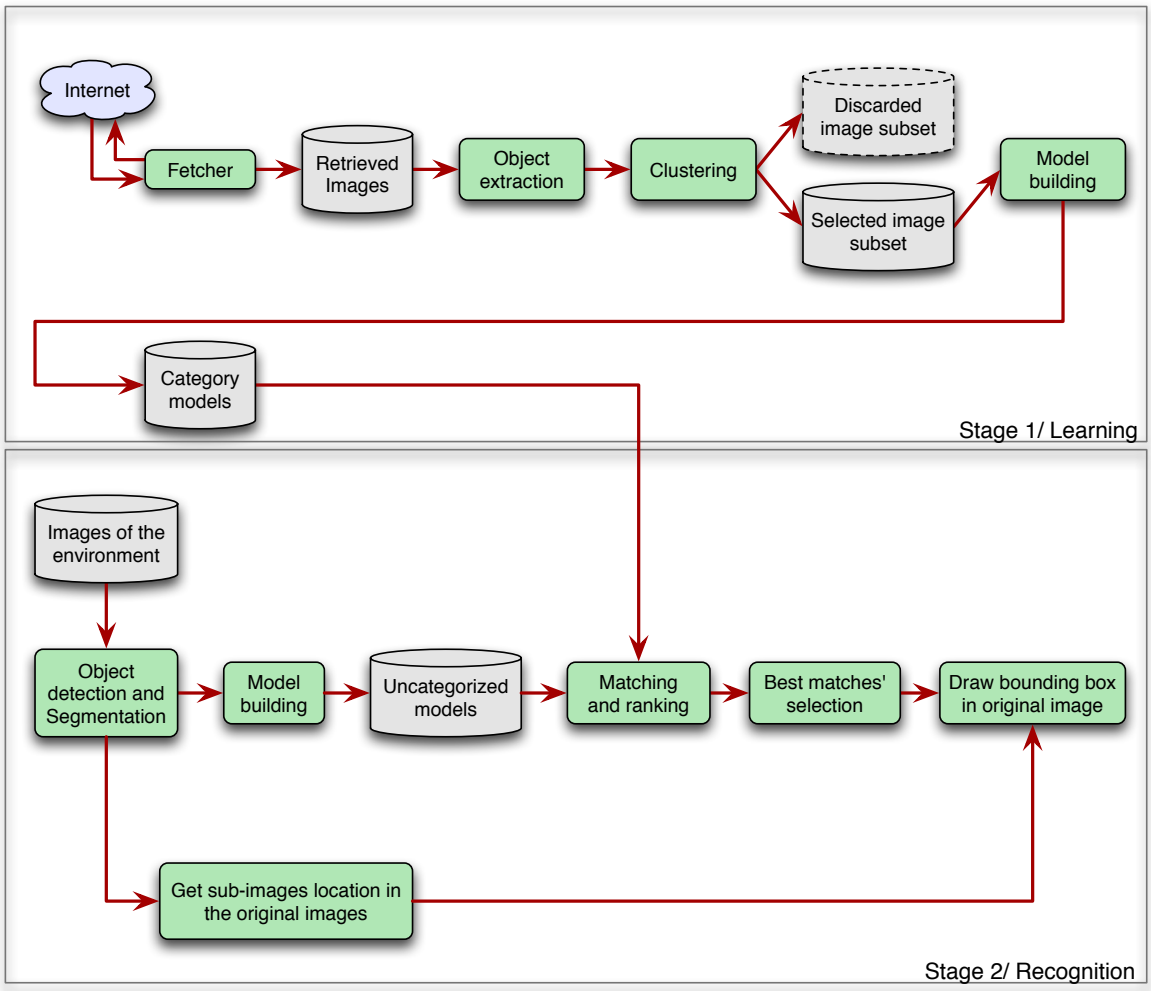

Figure 1.2: System overview.

An implementation of David Lowe's[34] SIFT feature detector and matching by Rob Hess[6] is also used, with some small changes, for both model building and image classification.

The work of the author was primarily focused in the learning part, including image retrieval, subset division and selection and the model building, but other aspects in the recognition, such as matching the created models with the real world scenes, were also approached.

[6] *http://web.engr.oregonstate.edu/~hess/*

1.7 Dissertation structure

In order to enhance the clarity of this document, the remaining information here contained is organized in five chapters, in addition to the present introductory chapter, as follows:

- **Chapter 2 - Previous work -** In this chapter, we present some of the previous research work in the fields of content-based image retrieval, feature subset selection and object recognition. In it we also bring forward the fundamental concepts on which the work was founded.

- **Chapter 3 - Representations and similarity measures -** In this chapter, we present an in-depth explanation of the methods used to represent objects and measure similarities between them. The similarity measures are what allow us to classify objects in different categories. For this we have two distinct approaches: one based on the shape of the object, retrieved from contour analysis, using Global Shape Contexts (GSC); and another based on local features, using an approach known as SIFT: Scale Invariant Feature Transform. Also, we present and explain Roy's Shape Representation (RSR), which though not implemented by the author, is part of the system developed for the SRVC. Specific categories are represented using SIFT. More specifically, a category is represented by the concatenation of the SIFT features of each individual object belonging to that category. General categories are represented by the agglomeration of individual shape histograms (GSC and RSR). Each of these histograms is a representation of an object belonging to those generally described categories.

- **Chapter 4 - Subset selection and classification -** After selecting the object representation and similarity measures, as described in the previous chapter, it's necessary to develop an automated image retrieval and category representation and classification system. Since images are obtained without human interference from the Web, good instances of a category will be accompanied by inevitable junk. Hence the need to select the subset of these images, which can be used to build a model, and discard the rest.

- **Chapter 5 - Performance evaluation -** In this chapter we present an evaluation of the methods used in our work. We perform a series of tests in both the subset selection and classification processes. In subset selection we visually analyze the images that were downloaded from the Internet, marking good and bad representations. Then, after running the subset selection, we check how many good representations were selected. This allows us to infer if there are improvements in automatically selecting a sub-group of the initial set or if this type of automated selection is disadvantageous. Tests are done to specific (SIFT based) and general (shape based) categories, where in the latter we also compare the results of running object extraction first or not. In the classification process we use *40* categories of objects, each one with several images, to automatically benchmark the shape based classifiers (GSC, RSR and the voting system) and the SIFT based classifier. A group of images of each category is used as the category model and all of the remaining images are matched to these models and automatically classified. Finally, we present the results obtained on the Semantic Robot Vision Challenge edition of 2008.

- **Chapter 6 - Conclusions -** This final chapter concludes the dissertation. It starts with a recapitulation of the work and exposes the conclusions to which we came during

its development and evaluation. We also explore the possibilities of expansion and improvement that could be achieved in related future work.

Chapter 2

Previous work

Summary

In this chapter, we present some of the previous research work in the fields of content-based image retrieval, feature subset selection and object recognition. In it we also bring forward the fundamental concepts on which the work was founded.

2.1 Content based image retrieval

Content Based Image Retrieval (CBIR) is the ability to retrieve certain images from an image database, given queries that describe their contents without the need of manual annotation. These queries can either be textual or in the form of images. From section 2.1.1 to 2.1.5 we present some systems developed by several researchers to tackle this task.

2.1.1 CBIR using category based indexing

Wardhani and Thomson[55] argue that, in order to handle an arbitrary collection of images, one should not rely on a single analysis method, because different images require different approaches. For instance, a shape retrieval is not suitable for texture-rich, complex, irregular images. To solve this problem they propose a partition of the whole image collection into different categories (Table 2.1) based in the dominant attributes of the image:

- *Semantic* (natural scenes, people and geometric objects)
- *Syntactic* (single, multiple objects)
- *Statistic* (smooth, textural)

The first step in achieving this categorization is extracting the **dominant region** of the image, using multi-level color segmentation. Then, based in extracted regions' features of color, texture, shape and relation between regions, the image is categorized. Clustering images into categories allows for large sets to be organized into groups based on the features of the image content, providing for an easier and faster navigation within the results.

Category Name	Features
Natural Scene	*Green & blue spatial relation*
People	*Human skin hue*
Geometric objects	*Man made objects*
Single object	*Figure/ground (F/B)*
Multiple objects	*Non F/B*
Mainly smooth	*Smooth color*
Mainly textural	*Large Variance*

Table 2.1: Image categories as proposed by Wardhani and Thomson

To extract the dominant region, Wardhani and Thomson analyze the feature space . They use color segmentation adopting the **Mean Shift Algorithm**, that estimates density gradients:

1. Choose a radius r for a search window for the initial feature vector density estimation.
2. Choose the initial location of the window.
3. Compute the Mean Shift Vector, i.e., the vector difference between the local mean and the centre of the window (Equation 2.1, where x is the p-dimensional feature vector, $p(x)$ is the probability density function of x and $\bigtriangledown p(x)$ is the gradient of x).
4. Translate the search window by the shift amount
5. Repeat till convergence

$$MSV = \frac{r^2 \bigtriangledown p(x)}{p + 2p(x)} \tag{2.1}$$

Segments in the image correspond to high density, or significant, regions in the feature space. Because segmentation can produce too many or too few segments, this phase results are further reduced by multi-level processing to obtain the dominant region. Images suffer variable scaling, by beginning at the smallest defined resolution. The image is continuously segmented and its size increased until it produces the required number of segments (*2* to *6* regions), while at the same time smaller regions are merged with the larger ones.

By isolating the dominant regions in the image, we can use information such as the presence of a background, number of regions, region size, standard color and statistical properties to weight for the corresponding category in the image. For example, the presence of the skin color would weigh in for "*people*", while such colors as green and blue would weigh in for "natural scene".

Wardhani and Thomson use the Gestalt figure/ground principle[56] for removing the background regions: the determination of the largest region completely surrounding objects. If the remainder is one region, the image is classified as an F/B image.

This approach allows for both image automatic categorization with semantic context and object extraction as well.

2.1.2 CBIR using color, texture and shape features

Hiremath and Pujari[25] present a framework for combining color, texture and shape information to produce better retrieval results.

Color and texture information is captured by partitioning the image in *24* (*4* × *6* or *6* × *4*) tiles of same size. Gabor features and color moments of these tiles provide the local descriptor of texture and color respectively. This info is obtained by a multi-level approach where *2* resolutions and *2* grid layouts provide different details of the same images. Figure 2.1 shows the image scaling and decomposition into size *M/2* × *N/2*, where M is the number of rows and N is the number of columns. The number of tiles for each level of decomposition doesn't change so that all have the same weight.

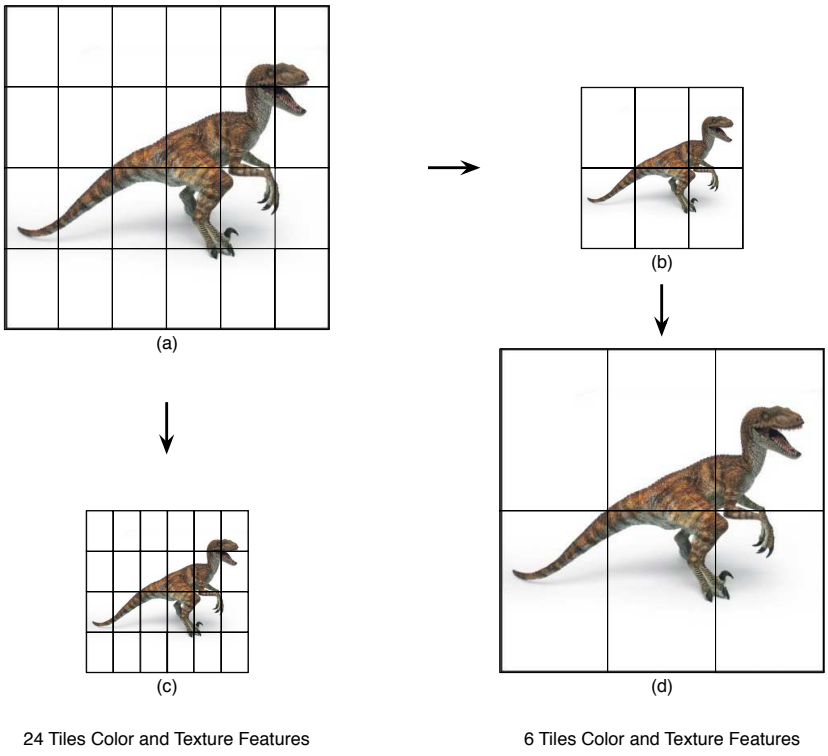

Figure 2.1: Hiremath and Pujar' system overview. (a) *32* × *32* tiles. (b) *32* × *32* tiles. (c) *16* × *16* tiles. (d) *64* × *64* tiles.

The problem of matching a tile with only another, minimizing the total cost, is an instance of the bipartite graph problem, as seen in Figure 2.2.

The authors introduce a method for matching where each tile of a query image and a target image can be used only once, i.e., tiles from the query image can be matched to tiles in the target image, but only once. After being matched once they cannot enter the matching process again. This process of matching is done using **Most Similar Highest Priority** (MSHP) principle and the adjacency matrix of a bipartite graph of the tiles of the target image and query image. In MSHP, when a matching between tiles i and j is done, the

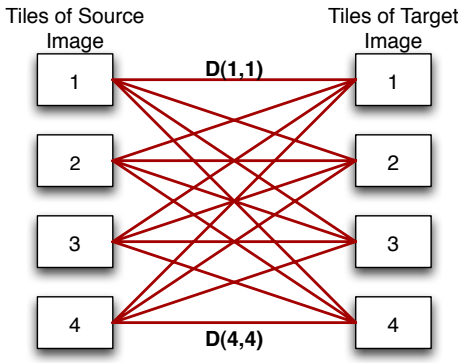

Figure 2.2: Bipartite graph showing *4* tiles of both the images.

corresponding cell in row i, column j) of the adjacency matrix is replaced by a high value (Hiremath and Pujari suggest *999*) preventing it from being matched again (see Figure 2.3 for an example). This process is repeated until all tiles are matched, with complexity $O(n)$, where n is the number of tiles. A brute force approach would have a complexity of $O(n^2)$.

The minimum cost of matching two images, q and t, is given by:

$$D_{qt} = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{i,j} \ , \tag{2.2}$$

where $d_{i,j}$ is the best-match distance between tile i of query image q and tile j of target image t and D_{qt} is the distance between q and t.

4.87	2.56	14.88	3.71
1.67	9.45	2.43	3.39
5.54	9.28	13.29	4.78
2.67	18.67	25.33	7.81

(a)

999	2.56	14.88	3.71
999	999	999	999
999	9.28	13.29	4.78
999	18.67	25.33	7.81

(b)

999	999	999	999
999	999	999	999
999	999	13.29	4.78
999	999	25.33	7.81

(c)

999	999	999	999
999	999	999	999
999	999	999	999
999	999	25.33	999

(d)

Figure 2.3: Image similarity computation based on MSHP principle. (a) First pair of matched tiles *i=2, j=1*. (b) Second pair of matched tiles *i=1, j=2*. (c) Third pair of matched tiles *i=3, j=4* (d) fourth pair of matched tiles *i=4, j=3*. They yield the integrated minimum cost match distance *34.34*.

Shape information is captured in terms of edge images of the gray scale equivalent of every image in the database, computed using **Gradient Vector Flow** (GVF) fields. GVF is used to obtain the edge image and it's computed as a diffusion of the gradient vectors of a gray scale or binary image: After converting an image to grayscale, this image is blurred by

making use of a Gaussian filter. The gradients present in this grayscale, blurred image are then used to compute the GVF. Strong edge responses are filtered and discarded, originating a shape representation of the original object in form of an edge image.

Hiremath and Pujari determine that the combination of color, texture and shape features provides a robust framework for CIBR.

2.1.3 CBIR using geometrical shape of objects in image

Adnan *et al.*[1] consider geometrical shapes as contents of an image. The major object in an image is extracted and then its geometrical shape is estimated and compared with a predefined database of shapes. Images are cataloged according to the number of objects and their geometrical shapes. These features will be the basis for retrieval and searching within a set.

When a query is done to the database it can either be an image or text describing what features to look for. In the former case, features are extracted from the query image and compared to a set of features in a database, as illustrated in Figure 2.4.

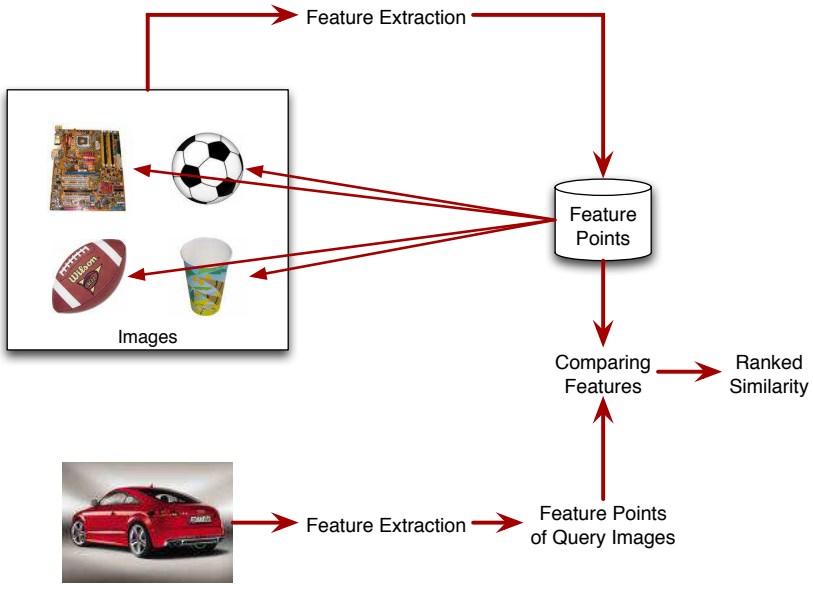

Figure 2.4: Adnan et. al' system architecture

To extract features of the images' main objects and the target object itself, segmentation is used. Afterwards the shape of the main objects is approximated and recognized, being stored along with its geometrical parameters. When the number and shape of the objects has been identified, their location is determined in the image. The number of objects, shape and its geometrical parameters, location and visual properties (color and texture can also be used) together make the "contents of the image".

The object extraction and understanding, meaning, the identification of its shape, starts with a color based segmentation where each segment is considered to be an independent ob-

ject. The edges of these objects are detected and made smooth, by making use of the average, and then are identified by the drawing of their outline.

Keypoints in the outline are detected by locating the points where curve slope is different from its surroundings, using the 2^{nd} order derivative of the curve. The curve that unites two different keypoints is estimated using the least-square method. Only curves of the first three polynomial orders are considered, whereas higher order curves are divided into lower order, smaller curves. Keypoints and curves are then optimized by combining similar curves or lines.

The object boundary is then a group of concatenated lines and curves. The relationship between different lines/curves' angles, different slopes at joining points and ration between boundary and area is used to do a shape estimation of the object.

The feature database contains links to the images as well as the information on the major objects, such as shape, color, texture, etc.

Some heuristics are used to relate the information on the object with basic shape categories, such as circles and ellipses, polygonal shapes and other shapes. By using the number of major objects present in the image and approximating their geometrical shapes to these basic categories, the retrieving system can successfully classify the query.

2.1.4 CBIR using multiple shape descriptors

Sarfraz and Rhida[45] conducted a test with several shape descriptors (or categories of features):

- *11* **moment invariants**: Moment invariants are some weighted averages (moments) of pixels' intensities, or functions of those moments that have desirable properties such as scale and rotation invariance. The authors use the moment invariants presented by Hu[27] and J. Flusser and T. Suk[20].

- **Area ratios** (*3* concentric ring based and *8* sector based): these features represent the ratios between the area of the object in certain regions and the total area[58]. Authors use concentric circles and sectors, as seen in Figure 2.5.

- **Simple Shape Descriptors** (SSD) based in different ratios:

  - Eccentricity is the ratio between the object major axis to the minor axis[28].
  - Compactness is the ratio of the area of the object to the area of a circle with the same perimeter[28].
  - Convexity is the ratio of the object's convex hull[1] to the object's perimeter[28].
  - Rectangularity is the ratio of the area of an image object and the area of the minimum surrounding rectangle[28].
  - Solidity is a measure of density and is the ratio of the area of an image object and the area of the object convex hull.

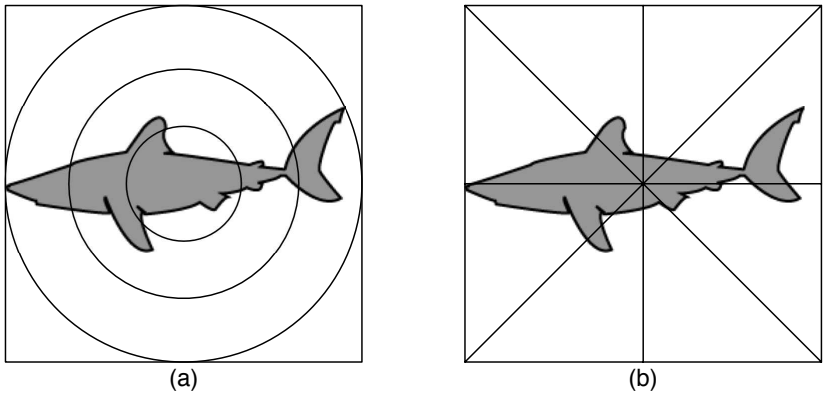

Figure 2.5: Sarfraz and Rhida area ratios. (a) *3* concentric circles based region. (b) *8* sector based region.

Retrieval works by having an image query and set of images in a database that are returned based on the similarity between the features of the query image and the features of the images in the database. Similarity is measured using both **Euclidean distance** and a **cosine correlation coefficient** for each category of features previously referred.

Euclidean distance is actually a measurement of dissimilarity. The smaller the distance, bigger the similarity and *vice-versa*. It's the geometric distance between two vector values:

$$d(Q,I) = \sqrt{\sum_{j=1}^{n}(f_j^Q - f_j^I)^2} \tag{2.3}$$

Cosine correlation coefficient is the magnitude-free coefficient between the feature and query vector, meaning that it doesn't change when the vector is multiplied by a constant factor. It measures the cosine of the angle between the vectors as a similarity measure:

$$\cos\theta = \frac{\sum_{j=1}^{n} f_j^Q f_j^I}{\sqrt{\sum_{j=1}^{n}(f_j^Q)^2}\sqrt{\sum_{j=1}^{n}(f_j^I)^2}} \tag{2.4}$$

The image retrieval process works as follows:

1. The image given as query has its noise removed by a median filter.
2. The largest object of the query image is taken as the query object.
3. Query object's features are computed.
4. Similarity of those features with those of the objects in the database is measured.

[1]Convex hull is the minimum convex boundary that covers a given finite set of points in a plane

5. Images with highest similarity are returned.

The authors conclude, after testing, that for both similarity measures (Euclidean distance and cosine correlation coefficient), SSD have the best performance, followed by area ratios which have an overall better performance than moment invariants. It's worth to notice, although, that cosine correlation coefficient degrades the performance of SSD for *9.0%* but improves the one of area ratios in *0.3%*. Moment invariants increase *24.6%* which despite being a great improvement it's still not sufficient to surpass any of the others shape representations.

A combination of both SSD and area ratios while using Euclidean distance improves the performance by *3.3%* compared to the best previous performance held by SSD also with Euclidean distance.

2.1.5 An image semantic retrieval system design and realization

Tong-Zhen and Yong-Gang[51] propose a method for automatic semantic annotation and retrieval with user feedback. Users submit their queries and the system calculates the similarity between the queries and the images in the database. Once a ranked list of results is presented to the user, he/she can provide feedback for the results to add, delete, reinforce or bate semantic information. Because large collection of images are almost impossible to manually annotate, due to the quantity of labor involved, automatic annotation is highly desirable, relieving the user from this part.

Each image in the system is represented by its feature vector and a **Semantic Network** with associated keywords (that have a weight between *0.0* and *1.0*). A Semantic Network is composed by a group of images that are connected to one or multiple keywords. This is a weighted relation, in the sense that there are keywords that are more descriptive of certain images than others. Image similarities can therefore be computed in function of the similarity of different keyword sets. In Figure 2.6 an illustration of a Semantic Network is provided.

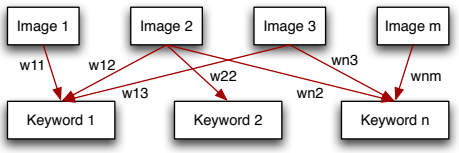

Figure 2.6: A semantic network.

Similarity between images is measured taking into account both their Semantic Networks and low level features. The Semantic Network is given more importance because it's the best and most exact representation of semantics that can be used. Low level features are also used, as a second order measure, because until a certain point they do reflect the image semantics. They have a more important role at the early stages because the Semantic Network is still very poor in information or when new images are added that have no keyword description.

Measuring the similarity between Semantic Networks is done with the help of *WordNet*, that provides relation lookup between different keywords. This system, developed by George

Miller in Princeton, enables the creation of a linguistic ontology[2] of **semantic associations**, that helps to compute the semantic distance between two keywords. This is important because while two images can have two different keywords, they can have the same meaning, i.e., be semantically very close. For example, a "cup" and a "coffee mug" are two distinct keywords that are synonyms. In *WordNet*, each noun has several meanings and each meaning correlates with different synonyms. There are *3* different types of relations in *WordNet*: *IS_A*, *MEMBER_OF* and *PART_OF*. In Figure 2.7 we exemplify an *IS_A* hierarchy (hyponymy).

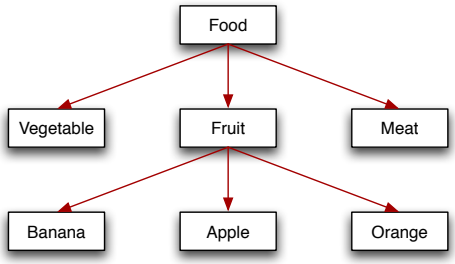

Figure 2.7: A hyponymy relation. *IS_A* is a hyponymy, meaning that represents the quality of belonging to a lower class. Apple, banana and orange are fruits and a fruit is a food.

Image labeling can be manual, but it can also be automatic. Automatic annotation uses keywords retrieved from the image filename, HTML *alt* tag or title of the webpage that contains the image, or by attaching the keywords in a query to the result images, updating the semantic network in the database in function of the query. In this last case, if some returned images are labeled as relevant during the user relevance feedback stage, then the query keywords are added to those images' semantic networks. Furthermore, if several images are returned, they exchange semantic information (or keywords) between themselves.

The authors also propose a system of **seed spreading** and **new image registering**. New image registering occurs when new images are introduced in the database collection. Their features are computed and recorded, and if there are keywords' attachments to the image, then they're also added to the Semantic Network. Seed spreading is useful when the Semantic Network is very poor in information at the early stages of the system: the administrator can throw determined images, denominated *semantic seeds*, into the database to enrich the Semantic Network. These images are rich in keyword information and when the system retrieves them for a query, their keywords propagate rapidly between the rest of the images in the repository.

2.1.6 Scene image classification and segmentation with quantized local descriptors and latent aspect modeling

Quelhas[40] uses local descriptors for scene and object classification, while exploring the combination between text modeling methods and local invariant features. Similarities between discrete scene representations are studied, comparing the sparsity and semantics of two types of representations: bag-of-visterms (histograms of quantized local visual features), or

[2] An ontology is an accurate and comprehensive organization of hierarchical knowledge, containing all relevant items and their relations.

BoV, and bag-of-words (text representation that doesn't take into account the order of the words), or BoW. Texture and color information are also used for natural scene classifications. Probabilistic Latent Semantic Analysis (PLSA) is used to investigate the possibility that latent space models can be used in the classification part and to discover patterns of visual co-occurrence.

2.1.7 Keywords to visual categories: multiple-instance learning for weakly supervised object categorization

Sudheendra Vijayanarasimhan and Kristen Grauman[52] propose a method for multiple-instance learning of discriminative category models from images with associated keywords.

Multiple-instance learning[15] is a type of supervised learning. In this approach, the objective is to learn a description from data that consists of a sequence of instances, each described as a set of vectors that are labeled positive or negative. A set of vectors, or instance, is described as positive if at least one of the vectors belongs to the intended concept, and negative if none of the vectors belongs to the concept. The task is to learn an accurate description of the concept from this information.

Given a list of category names, their method retrieves from the Web groups of images that potentially belong to each category. Multiple search engines with queries in multiple languages are used to collect these noisy positive bags of instances. Negative bags are collected from random samples in already labeled data sets from categories with different names from the category of interest, or from unrelated searches.

Each image is represented as a bag of visual words, that is, a histogram of how many times each of a given number of prototypical local features occurs in the image.

The learning method uses the text that surrounds the images, be it on the webpage HTML or in meta-tags, while already accounting for anticipated noise and ambiguity.

It's assumed that images retrieved from the Web contain at least one category depiction, thus forming a positive bag. The sparse multiple-instance learning (sMIL) classifier can discriminate the true positive instances from the negatives, even if their sparsity in the positive training bags is high. The classifier is then refined by iteratively updating the representation of the training bags: stronger positive instances have more weight on the decision, while those that are probably false positives have less weight.

To discriminate between positive and negative bags, an objective function is used to determine a decision boundary, while also accounting that the positive bags can be randomly sparse. This means that the objective function can't assume that there's a determined minimum number of positive example images in each positive bag. Because the sMIL classifier will result in sparse representations of each positive bag, an iterative refinement step is used, to make the true positives represent the bag as much as possible. This optimization adapts a standard support vector machine (SVM) to accommodate the multi-instance setting.

According to the authors, by modeling a visual category from a rich group of learning examples, richer category models can be the result. This approach was also presented by the authors in the SRVC'08 with good results in the image retrieval phase.

2.2 Data clustering and subset selection

Subset selection is the ability to divide a set of items into smaller subsets/groups/clusters by their similarity. Specially interesting for us is the case of image clustering. When a determined number of images represent a smaller number of categories (e.g.: *100* images of only *3* categories, like "landscapes", "cars", and "people"), clustering the images according to their similarity will result in a self organization of the set. The objective is to have images from the same category grouped together. We start by presenting some data clustering algorithms and proceed to refer some previous work and research made in unsupervised feature subset selection.

2.2.1 Data clustering algorithms

Data clustering is the process of partitioning of a data set into different subsets or clusters. The partition aggregates the data items into groups that share the same characteristics and hence, have a determined similarity between them, measured by a predetermined distance measure. The following are some of the algorithms for clustering items into groups:

Expectation-Maximization[12] (EM) is an algorithm to estimate maximum likelihood of parameters in probabilistic models. It's composed of two steps, the first being the Expectation step, where initial distribution parameters are estimated and from these current parameters, the expected values of the latent or unknown variables are also estimated. The second step is the Maximization, where the distribution parameters found in the E step are re-estimated and maximized, using the previously estimated values of the unknown variables. Parameters found in the M step are then used to begin another E step and the process is repeated until convergence is achieved. The EM clustering algorithm computes probabilities of cluster memberships based on one or more probability distributions. The goal of the clustering algorithm then is to maximize the overall probability or likelihood of the data, given the (final) clusters.

K-means, an algorithm invented in *1956*[48], whose most common form uses an iterative refinement heuristic called Lloyd's algorithm[32], is used to cluster n objects into k partitions based on their attributes. It's similar to the EM algorithm, in the sense that both use an initial seed and refine the centers of the clusters based in their member data: initially, k centroids are either randomly seeded or by following some heuristic. An iterative process starts where the remaining n-k items are grouped with their nearest centroids and then the centroids are refined using the data of their clusters until convergence is attained. Because we use K-means in our work, we provide a more thorough explanation of the algorithm in 4.3.1. The **Canopy clustering**[35] algorithm is a lighter version of K-means: the data is initially cheaply partitioned into overlapping clusters called *canopies* and then the more expensive clustering is performed only within the canopies. This is useful for large data sets because

the number of instances that is compared in each step is reduced. Another clustering algorithm similar to K-means is an extension to Lloyd's algorithm: the **Linde-Buzo-Gray**[31] algorithm. It uses either a known probabilistic model or a long training sequence of data to derive a good book from vector quantization.

Artificial Neural Networks are a significantly improved and more sophisticated version of the first neuron mathematical model devised by McCulloch and Pitts[36] in *1943*. ANNs map the believed behavior of the brain neurons: they're an adaptive system that changes its structure based on the information that flows in the network during a learning phase. This adaptation can be used to model complex relationships between inputs and outputs or to find patterns in data, which can provide for data clustering. **Self-Organizing maps**[24] (SOM) are a type of ANN used to produce a lower-dimensional representation of the training set. Note that, for a small number of nodes or neurons, SOMs behave similar to K-means.

Multi-Dimensional Scaling[6] are statistical techniques used for searching multidimensional configurations of items in respect to observed similarities among these items. Given a matrix with item-item similarities, it uses those similarities to assign a location for each item in a lower-dimensional space.

A more recent clustering method uses fuzzy clustering. This means that each item can belong to several clusters, by having each item associated with a set of membership levels. **FLAME**, or Fuzzy clustering by Local Approximation of MEmberships[22], defines clustering by mapping the behavior of an item to the behaviors of its neighbors. Pairwise or item-item similarity measures are used to approximate the fuzzy membership of each item from its neighbors' memberships.

Most of these algorithms can be used subjected to **Constrained clustering**[54]. Constrained clustering algorithms are semi-supervised: they incorporate a previous set of rules that define a relationship between two data instances. These are mainly *must-link* and *cannot-link* constraints, which force the clustering algorithm to either always or never cluster together two determined instances.

A procedure that isn't really a clustering method but rather a feature reduction method is **Principal Component Analysis** (PCA), invented in 1901 by Karl Pearson[39]. PCA transforms a number of correlated (or not) variables into a smaller number of uncorrelated variables called principal components. Its main objectives are to discover or reduce the dimensionality of the data set and identify meaningful underlying variables. This can also be useful for a clustering approach if we take into account that previously to the clustering we can extract the important variables. Since it's also a common technique for finding patterns in data of high dimensions it can be used to define clusters.

2.2.2 Unsupervised feature subset selection (UFSS)

Feature selection is the technique of selecting a subset of relevant features from a bigger set and use that information for building robust learning models. It can be either supervised, where from an exhaustive search the optimal set is selected, or unsupervised where the result is a satisfactory set of the features. In this document we present some of the previous research

in unsupervised selection.

Unsupervised subset selection algorithms can fall into three categories:

- Wrappers: these are algorithms that search through the space of possible features and evaluate the produced subsets by running models on them. In UFSS these feature subsets are evaluated according to the performance of the unsupervised learning algorithm on the original data, projected onto the features in the produced subsets.

- Filters: these algorithms are similar to wrappers in the search approach, but instead of evaluating against a model, a simpler filter is used. In UFSS the produced features or feature subsets are assessed from data exclusively, ignoring the subsequent unsupervised learning algorithm.

- Embedded: in this case, the used techniques are embedded and specific to a model.

Luis Talavera[49] provided empirical comparison between wrapper and filter methods of feature selection for clustering, confirming the superiority of wrapper methods. Nevertheless, dependency based filters are, according to him, a reasonable feature selection alternative. The tests used EM-WFS (EM wrapper with forward search) for the wrapper benchmark and EM-PWDR (EM pairwise dependency ranker) for the filter.

There are several methods used for feature searching (*exhaustive search, best fit first, genetic algorithms, greedy hill climbing, etc.*), similarity metrics for evaluating subsets (*correlation, mutual information, class separability, etc.*) and clustering (*K-means, Expectation-Maximization (EM), etc.*).

Manoranjan Dash and Huan Liu[11] report a poor performance of the K-means clustering algorithm with high-dimensional data. Their approach then used a feature selection hill-climbing method, where images were ranked before clustering. First, the features are ranked using an entropy-based measure, ranking the features according to their importance on clustering. Then a subset of features is selected using a criterion-function that is invariant with respect to differences in the number of features, solving the problem with the K-means performance with high-dimensional data. Pabitra Mitra *et al.*[37] describe an UFSS algorithm also for large datasets based in a similarity between features. The similarity is pairwise and is measured a maximum information compression index. This approach doesn't require search, so the algorithm has a superior speed and performance.

Søndberg *et al.*[47] use a hybrid filter-wrapper approach with the number of clusters or classes previously known, using naive Bayes models for unsupervised learning and an objective function for the filter/wrapper that scores the relevance of every feature. Selection uses a hill climbing strategy referred as sequential selection. This can be either sequential backwards selection (SBS), where the whole set is selected and then the least rewarding feature is removed, or sequential forward selection, where initially no feature is chose and then the most rewarding feature is added. The hybrid approach is then formed by a wrapper approach combined with SBS on top of a filter approach combined with SFS.

Mark Devaney and Ashwin Ram[13] use COBWEB[19], a conceptual clustering system, for unsupervised learning. The conceptual clustering system is responsible for generating a conceptual description for each generated class, although they usually exist indirectly in the data and are often used to evaluate the final output of the system. The authors also implemented an *attribute incrementation* concept (An attribute-incremental learner adds, or removes, attributes to an existing concept structure) creator based on COBWEB to improve the efficiency of the feature selection process. Jennifer G. Dy and Carla E. Brodley[16] also researched a wrapper framework where the number of clusters is unknown. To determine the number k of clusters in the data a feature subset selection wrapped around EM clustering is implemented. The authors concluded that when the number of clusters is unknown, results in both simulated Gaussian data and real data sets show that incorporating the search for k within the feature selection procedure yields better "class" accuracy than fixing k to the number of classes.

Volker Roth and Tilman Lange[42] use an approach that combines clustering and a wrapper strategy for feature selection. Features are directly selected by efficiently optimizing the used partitioning algorithm with guaranteed local convergence property. This feature selection is based on a Gaussian Mixture Model optimized by EM. GMM is a model in which independent variables are fractions of a total and the results do not depend on the number of variables but rather in the fraction of each variable. For the EM algorithm to incorporate feature selection, the M step is reformulated as a linear discriminant analysis (LDA) problem, which makes use of the labels estimated in the previous E step. The identity of LDA and linear regression are then used to restate the M step.

2.3 Object representations and recognition

Object representations are computer models built upon characteristics of an object, such as shape, texture, color(s), local features, etc. either individually or by a combination of different types of characteristics. With these models it's possible to correctly identify/classify/recognize the same object or one that belongs to the same class in other images by computing their own models and matching them against the first ones.

2.3.1 Object representations

There are several features that can be retrieved from an image and used to build object representations. Methods based in the object shape have been explored by D.K. Roy[43] and S. Belongie *et al.*[2][3]; David G. Lowe[34] explored the use of scale invariant keypoints as image descriptors; J. Vogel and B. Schiele[53] use a grid layout and extract texture and color based information from each block of the grid; I. Biederman[4] explored a modeling approach using recognition by components; *etc.*

Features can either be used as bag-of-words (BoW), where their spatial location is ignored or as part-based models, where the location and relation between the features is important. Part-based models can yet be divided in constellation models, based on the relative positions in which the components are, and non-constellation models, where the use of small parts has the objective to build an algorithm that can detect/recognize an item.

2.3.2 Object recognition

For BoW models, there are several approaches to object categorization, some of which are now exposed:

C. Dance *et al.*[10] use a naïve Bayes classifier, assuming that each category has its own identifiable *word* distribution. Methods such as probabilistic latent semantic analysis (pLSA), by T. Hoffman[26] and latent Dirichlet allocation (LDA), by D. Blei *et al.*[5] are more refined methods for analyzing co-occurrence data.

There are also some approaches based on discriminative models: T. Serre *et al.*[46] use Adaboost, a machine learning algorithm by Y. Freund and R. Schapire[21]. C. Dance *et al.*[10] investigate on the use of support vector machines.

Grauman and Darrel[23] provide a method for comparing BoW models, called Pyramid match kernel. Histograms are compared at several, ever decreasing resolutions and their intersection at each level is computed (Figure 2.8).

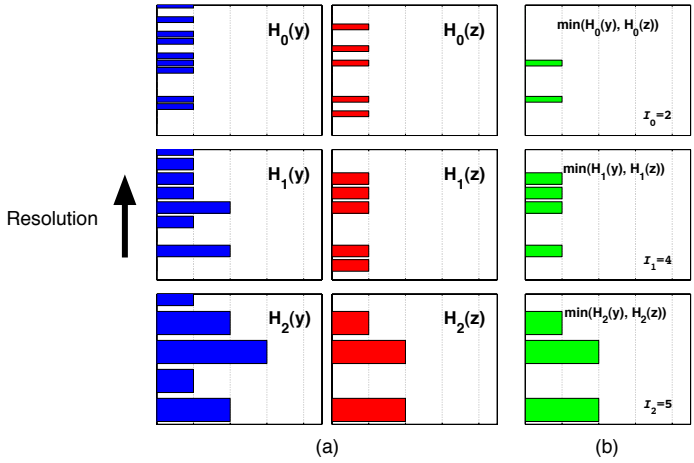

Figure 2.8: Pyramid match kernel. (a) Histograms to be compared. (b) Intersections at each resolution.

In part-based models, there are constellation and non constellation models. The constellation model was originally introduced by M. Fischler and R. Elschlager[18] and has since been explored by many others, such as M. Burl an P. Perona[8]. Non constellation models have been explored by researchers such as A. Yuille *et al.*[57], T. Poggio and R. Brunelli[7], A. Lanitis *et al.*[29] and B. Leibe *et al.*[30].

Chapter 3

Representations and similarity measures

Summary

In this chapter, we present an in-depth explanation of the methods used to represent objects and measure similarities between them. The similarity measures are what allow us to classify objects in different categories. For this we have two distinct approaches: one based on the shape of the object, retrieved from contour analysis, using Global Shape Contexts (GSC); and another based on local features, using an approach known as SIFT: Scale Invariant Feature Transform. Also, we present and explain Roy's Shape Representation (RSR), which is part of the system developed for the SRVC, implemented by teammate Luís Ribeiro[41]. Specific categories are represented using SIFT. More specifically, a category is represented by the concatenation of the SIFT features of each individual object belonging to that category. General categories are represented by the agglomeration of individual shape histograms (GSC and RSR). Each of these histograms is a representation of an object belonging to those generally described categories.

3.1 General and specific categories

We decided to separate categories in two main groups: general categories, whose concept is more abstract, and can exist in a wide variety of forms (e.g.: chair, table, garbage bin,etc.) and specific categories, whose characteristics are regular and well defined (e.g.: a specific book, a determined music album, etc.). The former are to be analyzed according to their shape and the latter according to their SIFT features. Common nouns identify general categories whereas proper nouns and quotes identify specific categories.

The reason for this separation is because previous tests indicated that a SIFT based classification didn't present good results for simple objects. The simple artifacts produced while rotating and saving a JPG image, for example, made the SIFT classifier think some of the rotated versions of the same object belonged to different categories. Simple objects, from general categories, are much more identifiable by their shape. Nevertheless, for complex images from specific categories, rich in textures, SIFT proved to be a good approach.

Figures 3.1 and 3.2 show category examples of a general and a specific category. The decision heuristic is based in the category name string given in the SRVC. All specific categories in the competition had either quotes or capital letters in their name, or both. The rest are general categories. In the figures, objects (a-d) are good representations and (e) is noise. If we assume these were retrieved from the Web, we want an automatic method that is able to build a model from the (a-d) images while ignoring (e).

Figure 3.1: Shape category - teapot. (a-d) Good representations of the object. (e) Junk image. Images have high intra-category variation.

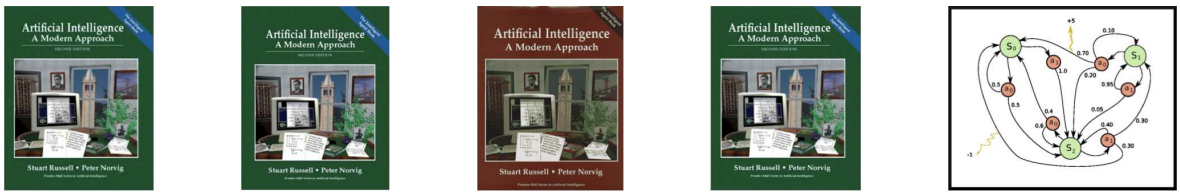

Figure 3.2: SIFT category - Book "Artificial Intelligence: A Modern Approach". (a-d) Good representations of the object. (e) Junk image. Images have low or nil intra-category variation.

3.2 Shape representations and measures

This approach uses image edges to build a representation of an object. The edges of an object are, most of the time, representative of the shape of that object. For a human, it would be easy to associate a line drawing with the object that it's supposed to represent. Even though the corresponding points that constitute two identical shapes are not spatially present in the exact same locations, it is possible through deformation (or morphing) to match different images. This concept is called **deformable matching**[50]. The distance between two shapes will be the cost of matching its corresponding points[44].

In our implementation the image is first smoothed by recurring to a Gaussian filter to eliminate possible noise and then a Canny edge detector is used to find the edges. The result will be a black and white image on which only the contours of the object and its components are visible (refer to Figure 3.3 for an example). On a first stage, the shape of the object is thus represented by the coordinates of the points that comprise the contours.

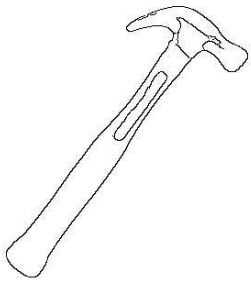

Figure 3.3: Contours of a hammer, as obtained by the use of a Gaussian filter in the original image followed by a Canny edge detector.

3.2.1 Shape Context

A Shape Context is a description of the arrangement of points relatively to a reference point (Figure 3.4). This means that a Shape Context is not a property of a shape but rather of all of its individual points. An image is thus described by a collection of Shape Contexts.

Matching two shapes implies the discovery of corresponding points in both images, i.e., the ones with the most similar shape contexts. This is an optimal assignment problem (an instance of bipartite graph matching), on which similarities are maximized while affecting each pixel in the first shape to its homologous in the second[2][3].

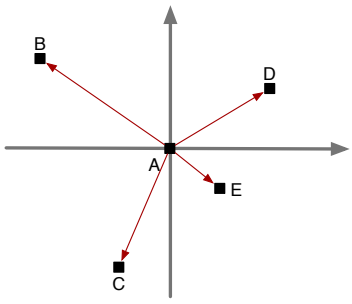

Figure 3.4: Shape Context of point A. The origin of the referential is centered in the point. This implies that a Shape Context of a point is a description of the arrangement of the rest of the points relatively to it. In our case, this description will be composed of log-polar coordinates.

Having a full set of points representing the shape will inevitably lead to an over detailed shape description, with an also over detailed agglomerate of Shape Contexts. A meticulous description may seem like a good thing, since the most scrupulous it is, the better is the object representation. However, in a category, such a comprehensive evaluation contributes to undesired variations in shape descriptions. It's preferable to sample the shape points, as it serves as a more sound and succinct, while very discerning, shape descriptor. The computational advantages are also evident.

Each point p_i on the shape will give origin to a granular histogram h_i of the relative

coordinates of the remaining *n-1* points,

$$h_i(k) = \#\{q \neq p_i : (q - p_i) \in bin(k)\} \tag{3.1}$$

This histogram is defined to be the Shape Context of the point p_i (see Figure 3.5(d-f)) for examples of Shape Contexts). The bins in the histogram use a log-polar coordinate system, making it more sensitive to variations in its immediate vicinity sample points than to those that are farther away. An example of this is shown in Figure 3.5(c).

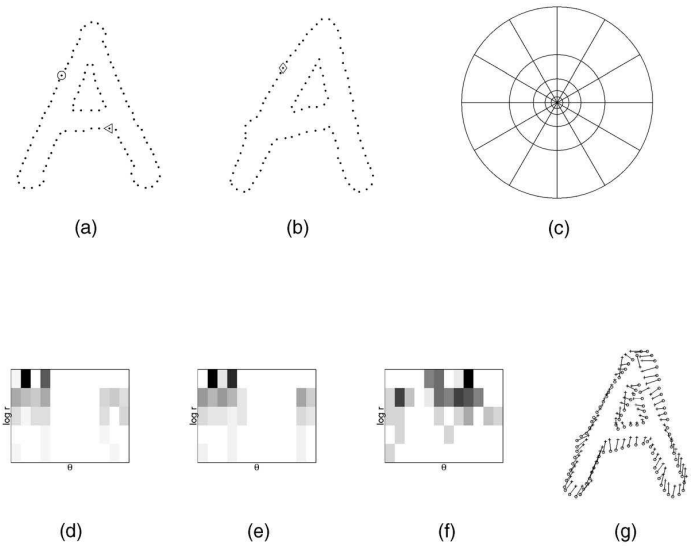

Figure 3.5: Shape Context and matching. (a,b) Samples of two shape contours. (c) Log-polar histogram used for the Shape Contexts. (d) Shape Context of the point marked with a circle (○) in (a). (e) Shape Context of the point marked with a triangle (◁) in (a). (f) Shape Context of the point marked with a rhombus (◇) in (b). Each of these Shape Contexts is a log polar representation of the distance and angle from the sample point to all the other points in the shape, meaning that the sample point is the origin in (c). Note the similarity between (d) and (e), which represent Shape Contexts for points that are near the same location in two homogeneous shapes (g) Correspondences between (a) and (b).

The vectors originating in a point to all the other *n-1* sample points in shape, describe the arrangement of the complete shape relative to the reference point. As n becomes large the description of the shape becomes exact.

Shape matching

Shape matching is the process of finding, for each point p_i on the first shape, the best matching point q_i on the second shape.

Let p_i and q_j be two points, each in a different shape, and $C_{ij} = C(p_i, q_j)$ denote the cost of matching these two points. Matching with Shape Contexts is the process of, 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, minimize the total cost of matching,

$$H(\pi)=\sum_i C(p_i, q_{\pi(i)}) \tag{3.2}$$

subject to the constraint that the matching be one-to-one, i.e., π is a permutation. We obtain a matrix that represents the costs of matching each point in the first shape to each point in the second. The problem will be to find the one-to-one assignment that minimizes the total cost of matching. This is an instance of weighted bipartite matching.

The χ^2 distance is used to represent the distance between the histograms:

$$C_{ij}=\frac{1}{2}\sum_{k=1}^{K}\frac{[h_i(k)-h_j(k)]^2}{[h_i(k)+h_j(k)]} \tag{3.3}$$

where $h_i(k)$ and $h_j(k)$ denote the kth bin of the normalized histogram at p_i and q_j. A brute-force algorithm for solving the assignment problem involves generating all possible assignments in the matrix C, calculating the total cost of each assignment, and finding the assignments with the minimal-sum of costs. The complexity of this method is driven by the number of elements in an $n \times n$ matrix. There are n choices for the first assignment, n-1 choices for the second assignment and so on, giving $n!$ possible assignment sets. Therefore, this approach has a complexity of $O(n!)$, i.e. a factorial/combinatorial complexity.

This problem can however be solved in $O(N^3)$ time using the Hungarian method[38]. The combinatorial optimization algorithm is described in Figure 3.6 and is used to solve assignment problems.

In the Shape Context case, each cell of the cost matrix represents the cost C_{ij} of matching two histograms, $h_i(k)$ of a first shape and $h_j(k)$ of a second shape. We want each histogram in the first shape to be assigned to its correspondent in the second, i.e., find the optimal assignment, or the assignment that minimizes the sum of costs C_{ij}.

For example, if the following matrix denotes the cost of matching the 3 histograms on the first shape (columns) with the 3 histograms in the second shape (rows),

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{pmatrix}$$

then, the best assignment would be

$$\begin{pmatrix} 1 & 2 & \mathbf{3} \\ 2 & \mathbf{4} & 6 \\ \mathbf{3} & 6 & 9 \end{pmatrix}$$

if we consider that the assignment must be one-to-one. The Hungarian algorithm converges to this solution in polynomial time.

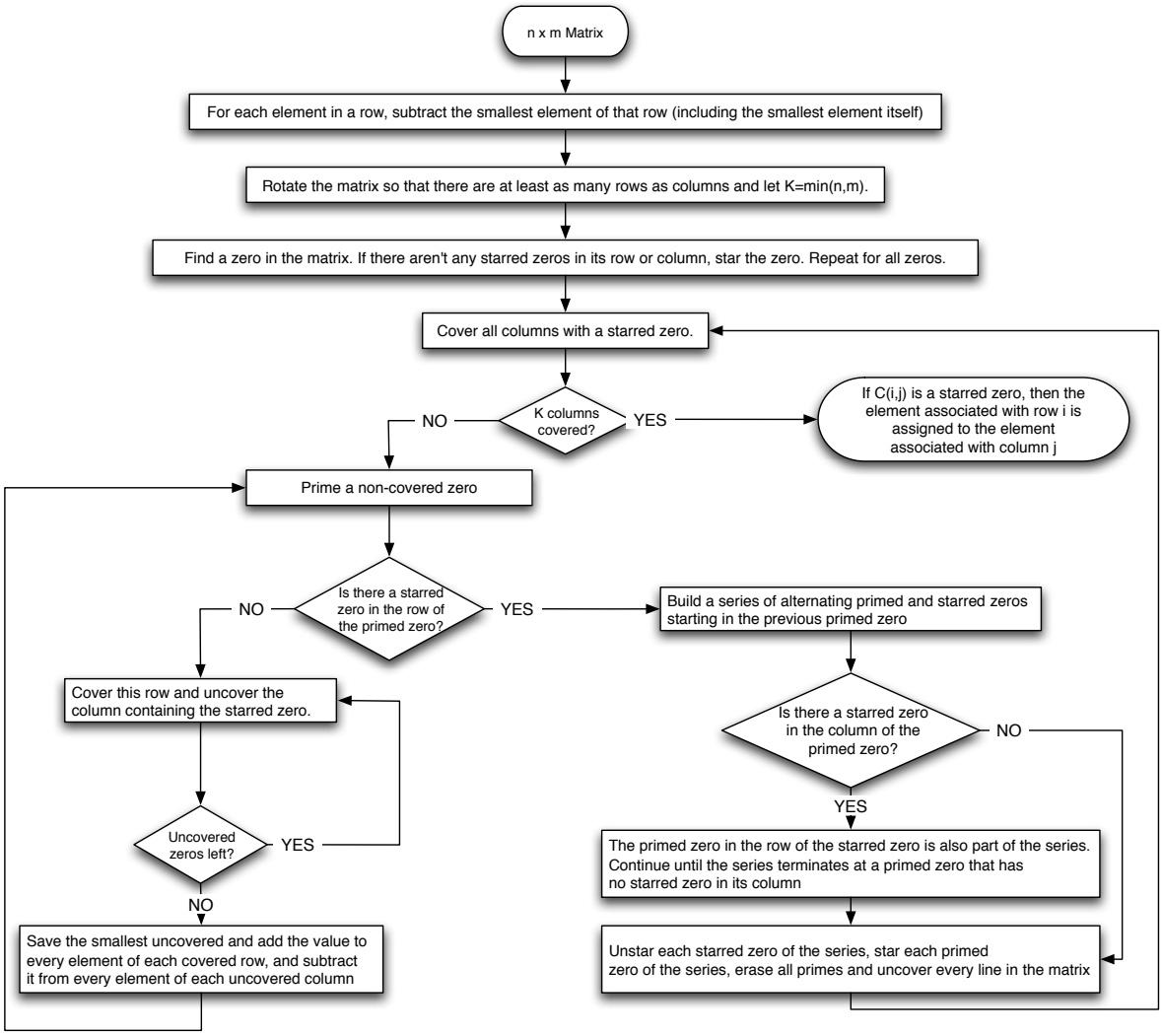

Figure 3.6: Hungarian algorithm.

3.2.2 The Shape Context as a global descriptor

In this section we describe an adaptation of the previous Shape Context technique designed and developed specifically for the UA participation in the SRVC'2008. In the original approach, each edge point of an object had a Shape Context. In other words, the Shape Context is used as a local descriptor. This implies that shape matching is a computationally heavy operation. In our case, where we can have images whose edges are formed by an arbitrarily large number of pixels, this would pose a limitation to the executability of the program in a useful time.

Instead, we determine the center of mass of the object and set the origin of the referential on that point, as seen on Figure 3.7. Thus, we use the Shape Context as a global descriptor (Global Shape Context, or GSC). A similar approach has already been explored by Seabra Lopes and Chauhan[33], where for each image there are two histograms: one based in dis-

tances to the center of mass and another based in the angles relative to a referential centered in the center of mass.

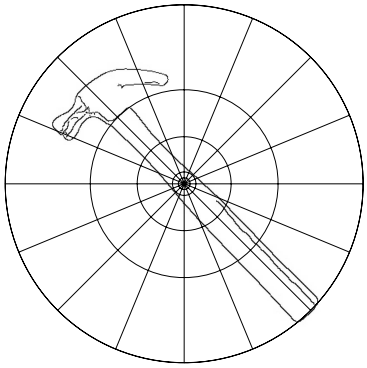

Figure 3.7: Shape Context as a global descriptor. Now the Shape Context isn't referring to a single point in an object, but to the object itself. It continues to be a log-polar histogram, but now has its origin in the center of mass of the object.

Determination of the center of mass

After some image cleansing and segmentation, we use flood filling to color the interior pixels of the object. The Center of Mass is then:

$$CM = \frac{1}{N}\sum_{i}^{N}(x_i, y_i), \tag{3.4}$$

where (x_i, y_i) are coordinates of the colored pixels.

Histogram construction and rotation/scale invariance

The coordinates of the points that make the contour are stored in a list and the polar coordinates from the center of mass to these points are calculated, meaning that both the geometric distance and the angle are stored. These points will occupy their respective angle/distance bins in the histogram.

Rotation and scale invariance are necessary if we are to compare objects that may be out of scale or in any orientation or position. Consider that the histogram is divided in d distance bins $\times$ a angles bins:

Scale invariance is obtained by making the radius of the histogram the maximum measured distance[1]. All intermediate distances will then be relative to the maximum, but this alone

[1]Note that, as in the original Shape Context, these distances all use a logarithmic scale

doesn't solve the scale problem. Bigger shapes will be composed of more points, presenting yet another difficulty in matching. The solution is the normalization of the histogram: in each bin, there won't be a counting of points that fit in it, but rather the ratio of points in a bin relative to the total number of points (*number of pixels that fit in the bin/total pixels in shape*). The histogram is built in *O(n)* time, where n is the number of points, because each point only has one angle and one distance associated to it, both relative to the center of mass.

The histogram itself is not invariant to rotation. Even though its horizontal and vertical axis are always parallel to the image borders, there's no guarantee that the object represented will be in the same position in two matched images. To make possible having a rotation invariant matching, we rotate one of the histograms A times while comparing two shapes (see Figure 3.8). The rotation that provides the lowest cost is the one used to calculate the similarity between the two shapes.

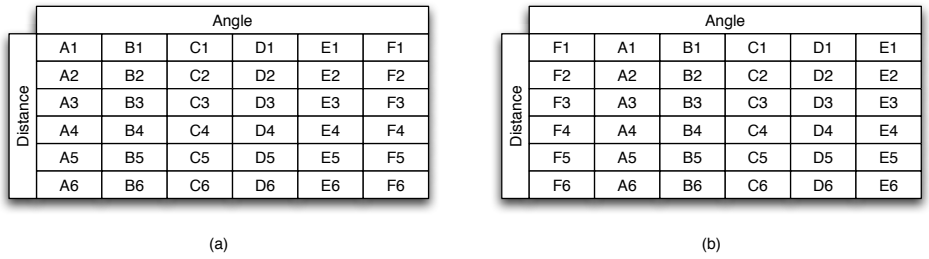

Figure 3.8: Rotation of the Shape Context histogram. To rotate the histogram and achieve rotation invariance, the columns are shifted. (a) The original histogram. (b) The histogram rotated. If the columns are shifted as many times as the angle divisions (which is the same as the total number of columns) then we have rotation invariance.

Matching

The determination of the similarity between two shapes is done by computing the costs of matching their two Shape Contexts. Since in our method an object has only one Shape Context, computed from the center of mass and used as a global descriptor, the χ^2 distance, used to calculate distances between histograms, is used as the cost of matching.

As it was mentioned in the previous point, the comparison between two histograms is made by rotating one of them as many times as there are angle divisions and use the one which gives the lower cost. The similarity S is given in function of this cost c, as follows:

$$S(c) = 1/c \quad if \quad c > 0 \tag{3.5}$$

For $a \times d$ histograms, where a is the number of angle bins and d the number of distance bins, matching the histograms has a complexity of *O(a(a×d))* because we calculate the χ^2 distance a times (for each histogram rotation).

3.2.3 Roy's Shape Representation (RSR) [2]

Roy's Shape Representation[43] uses tangents to object edges to compute a representation of an object. A Canny edge detector is used to obtain the edges from an object. Then, the coordinates of all the pixels that compose the edges are computed and kept. In the next step, the angle δ and distance d (Figure 3.9) between all of these points is also computed. For an object with n points, $\frac{n(n-1)}{2}$ distances and angles have to be determined, giving a complexity of $O(n^2)$ to this approach.

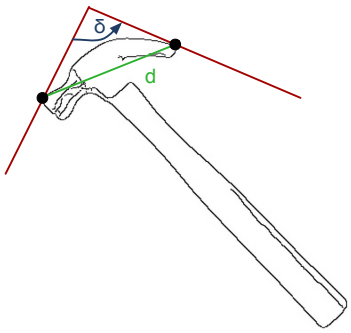

Figure 3.9: Roy's Shape Representation. RSR uses distances (d) and angles (δ) between all points of the image edges to compute a rotation and scale invariant descriptor for an object.

After the calculation of the Euclidean distance d and angle δ between the points, a two-dimensional histogram of both distance and angle is computed. This histogram has a 32×32 resolution, which was decided based on experimentation[41] showing it was better for our objectives than the 8×8 suggested by Roy[43]. Since the angle is calculated between the tangents to the edge points, it's a relative measurement and therefore rotation invariance is achieved. To obtain scale invariance, we simply normalize all distances relatively to the distance between the edge points that are farther away from each other.

To calculate the angle δ between two tangents, we use the difference between their angles π_i with the horizontal axis. This is easily obtained after the tangent has been computed, by simply using its slope m_i:

$$\pi_i = arctan(m_i) \tag{3.6}$$

The tangent of a pixel is considered to be equal to the linear regression of the neighboring pixels[41], so its slope can be determined by the regression line.

[2] ***Note**: Although this shape representation was not implemented by the author of this thesis, it is an integrating part of the jointly developed system. It was implemented by Luís Ribeiro[41], and as such, is also mentioned and explained.*

Matching

For every two points we therefore have an angle and a distance. These pairs are accumulated in a *32×32* histogram, where *32* is the number of divisions in angles and distances. Matching uses the χ^2 distance, like in the Global Shape Context case, with $O(n \times n)$ complexity, where $n \times n$ represents the size of the histogram, i.e., its total number of bins.

3.3 Scale Invariant Feature Transform (SIFT)[3]

SIFT[34] uses highly distinctive image features whose properties make them propitious for matching objects with different scales and orientations, as well as with some variations in illumination. This method is also computationally efficient, because the more expensive operations are only applied to locations that pass an initial test.

When matching a new object with SIFT, we need to compare it with category representations to determine to which one it belongs. These representations are databases of features extracted from previous sets of images that contain an instance of a category. Matching requires comparing each feature vector from the new object with these databases to find candidate matches based in Euclidean distance, using a nearest-neighbor criterion. If the matches agree on the object, location, scale and orientation, then they are good possible matches. These consistent clusters are found using the Hough transform and each group of more than *3* of these features will be kept and used to make an approach to the object pose.

3.3.1 Scale-space extrema detection

This stage uses a Difference-of-Gaussians function to identify locations and scales that can be repeatably assigned under differing views of the same same object. It searches for stable features that are scale and rotation invariant across all scales, using a scale space function. An example of a continuous scale space function applied to an image can be seen in Figure 3.10.

Figure 3.10: Linear (Gaussian) scale space representations for different scales. (a) Scale-space representation $L(x,y,\sigma)$, with scale $\sigma = 0$, corresponding to the original image I. (b) Scale $\sigma = 6$. (c) Scale $\sigma = 30$. (d) Scale $\sigma = 60$.

The scale space of an image is defined as a function, $L(x,y,\sigma)$, that is produced from the convolution of a variable scale Gaussian blur $G(x,y,\sigma)$

[3]We use an implementation of the SIFT feature detector and matching by Rob Hess - *http://web.engr.oregonstate.edu/∼hess/*

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{3.7}$$

with an input image, *I(x,y)* such that

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{3.8}$$

To find stable keypoint locations, blob detection is used. These blobs are regions that are either brighter or darker than their surroundings (extrema). This is done by using a Difference-of-Gaussians function convolved with the image *D(x,y,σ)* on the scale-space. The DoG is the difference of two nearby scales separated by a constant multiplicative factor k, i.e., the difference between the Gaussian-blurred images at scales $k_i\sigma$ and $k_j\sigma$:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{3.9}$$

By subtracting two scales, the information that lies between the two blurred images is stored and random noise is removed. The preserved information corresponds to stable keypoints. The DoG is calculated between pairs of adjacent Gaussian-blurred images in each octave, as seen in Figure 3.11. An octave corresponds to doubling the value of σ, and the value of the scale k_i of each image is selected to obtain a fixed number of blurred images per octave.

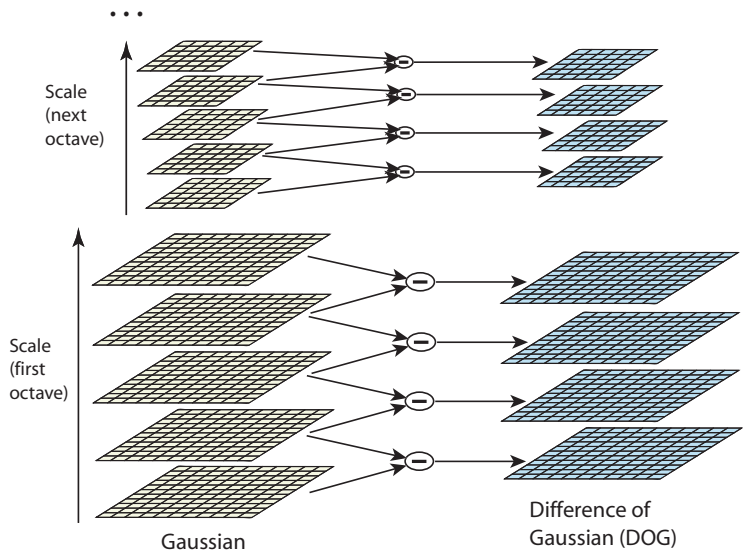

Figure 3.11: Difference-of-Gaussians. To compute *D(x,y,σ)*, the initial image is sequentially blurred by a constant $k = 2^{1/s}$, producing a sequence of a fixed number s of image scales per octave. Adjacent scales are subtracted to obtain the DoG images. After each octave has been processed, the scale that has twice the initial value of σ is down-sampled, by removing every second pixel in the horizontal and the vertical. This produces the first scale of the next octave.

Stable keypoints are the local minima/maxima (extrema) of the DoG across scales. To identify these keypoints, we need to compare each pixel of the DoG image to its surroundings. A pixel is surrounded in its own scale by *8* neighbors and by *18* neighbors in the adjacent scales: *9* in the previous and another *9* in the following. If the pixel value is either the maximum or minimum among all compared pixels, it is selected as a candidate keypoint. This process is illustrated in Figure 3.12.

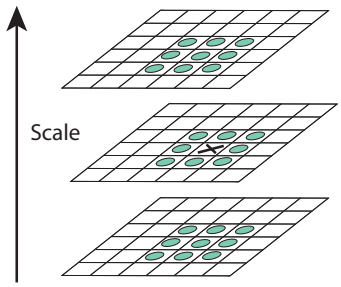

Figure 3.12: Local extrema detection. To detect extrema of $D(x,y,\sigma)$, each sample point is compared to its *26* neighbors in a 3×3 *3D* region at the current and adjacent scales. It is selected only if it's smaller or larger than all of them.

3.3.2 Keypoint localization

After the previous step, on which scale space extrema are detected, it's necessary to discard low contrast keypoints and of the remaining, poorly localized ones, i.e., points localized in edges. For this effect, location, scale and ratio of principal curvatures must be determined.

For each of the candidate keypoints previously found the position is determined by interpolating the location of the nearby maxima, which confers stability and increased matching confidence to the process.

Points with low contrast are rejected, but this alone is not enough. The DoG will have a strong response in edges, even if the point location is poorly determined and therefore is unstable to noise. These unstable and poorly located points are consequently to be also discarded by eliminating the ones with high edge response. This is accomplished by selecting the points whose ratio of principal curvature is below a certain threshold.

3.3.3 Orientation assignment

The orientation of the keypoints will be assigned according to local image gradients. By representing the keypoint descriptor relative to the image gradient, we obtain rotation invariance. All future operations will be performed on data that is transformed relative to this gradient, achieving scale and location invariance besides to the rotation invariance.

The local image gradients are selected on the Guassian smoothed image whose scale is closest to the scale of the keypoint found in the former step. This makes the computation scale invariant, because it will occur in the several scales on which keypoints were detected.

The gradient orientations of the region around the keypoint are used to build an orientation histogram. Peaks in the histogram correspond to dominant directions of local gradients. Histogram highest peaks (which are selected by choosing the highest peak and all others above *80%* of it) are used to create a representation of the keypoint. It's important to note that a location can have several dominant directions of similar magnitude, a fact that will give place to multiple keypoints in the same location, but with different orientations.

The keypoint representation, at this point, is a collection of gradient orientations (Figure 3.13)

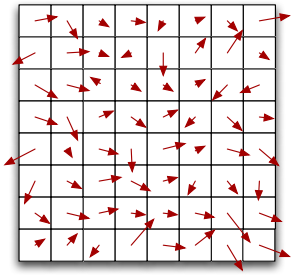

Figure 3.13: Image gradients. A keypoint is represented by the orientation of the gradients around it. Whereas in the implementation a *16×16* grid is used, we illustrate an *8×8* grid for simplicity purposes.

3.3.4 Keypoint descriptor

Previously, representations were determined for the local keypoints relative to local image gradients that are location, scale and rotation invariant. It's now necessary to compute a representation that is invariant to changes in illumination or *3D* viewpoint.

The local gradients around each keypoint are to be grouped to form a descriptor. These gradients (Figure 3.13) are sampled in *4×4* groups from a *16×16* grid, weighted by a Gaussian window, as shown in Figure 3.14(a) where the sample is done *4* by *4* in an *8×8* grid. This makes the gradients closer to the keypoint more relevant to the computation of the descriptor and less sensible to the ones that are far.

After the sampling, keypoints are grouped into orientation histograms, like the ones in Figure 3.14(b), where the length of the arrows represent the sum of the keypoints that were nearer to each of the *8* directions, in a *4×4* window. A descriptor is thereby formed by a *4×4×8* histogram, or in other words, resulting in an *128*-dimension feature vector.

At this point, descriptors for the local image regions, that are stable to changes in shape and illumination and yet highly distinctive, have been computed.

3.3.5 Matching

The first step in matching an object with a category is to retrieve its features, as described previously. Then, these features are individually matched against a previous database of SIFT

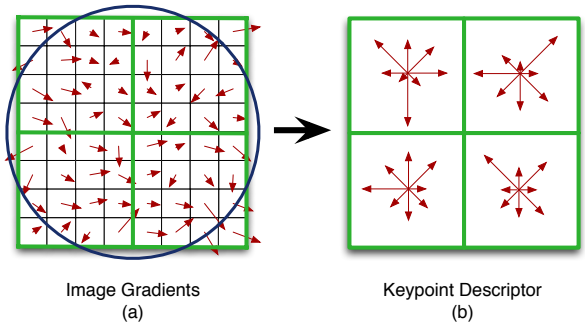

Figure 3.14: Keypoint descriptor creation. Shown is the creation of a Keypoint Descriptor, where from an 8×8 grid, gradients are grouped 4 by 4 forming a 2×2 descriptor with 8 directions. The length of the arrows in the right represents the sum of the gradients which were nearer each of the 8 directions. In the SIFT implementation, a 4×4 descriptor with also 8 directions is computed from a grid of 16×16, where gradients are grouped 4 by 4. This smaller version is used for clarity purposes.

features extracted from a training set. The model set (the set of SIFT representations that serve as models for each specific category) is composed of various databases, each one for a different category, and the matching process will attribute the new object to the correct database.

Object/category matching is done by computing the Euclidean distance between the features, using a nearest-neighbor approach (the nearest neighbor is the feature with the lowest Euclidean distance). Instead of using a global threshold to discard matches, the distance to the nearest neighbor and to the second nearest is compared. If the ratio between the former and the latter is greater than 0.8, the matching is rejected. This is done because the reliable correct match must have the closest neighbor much closer than the best false match. This discards many false matchings. It's also good to note that Lowe suggests using an approximate algorithm, called Best-Bin-First (BBF) instead of exhaustive search because it significantly reduces the computation time.

Because we will still have false matchings, derived from background object matches, it's necessary to cluster those that belong to the desired object and reject the others. This is done by a Hough Transform, that identifies clusters of features that are consistent with the object pose.

The Hough transform is used to detect certain class of shapes by a voting procedure. It uses an accumulator where votes are casted, whose dimension (or number of bins) is equal to the number of parameters that are trying to be estimated for the type of edge being detected (lines, circles, ellipses, etc.). For example, the linear Hough transform problem has two unknown parameters: m and b ($y = mx + b$). In the Hough transform each pixel and its neighborhood is used to determine the likelihood of an edge in that area. If there's evidence of an edge, its parameters are calculated and the bins of the accumulator where those parameters fall into are incremented. By finding the bins with the highest values, the most likely lines can be extracted.

In the SIFT case, the keypoints (in both the training and input image) are used for deter-

mining *4* edge parameters: 2D location (x and y - two parameters), scale, and orientation.

Each matched keypoint has a record of the keypoints parameters relative to the training image in which it was found, so it can predict (or vote on) the model location, scale, and orientation. Bins with *3* or more votes are identified as candidate clusters. Note that, when these clusters vote for the same pose of an object, the probability of a correct match is higher than for a single feature.

For each candidate cluster, a least-squares solution for the best estimated affine projection parameters (scale, orientation and position) relating the training image with the input image is computed, by minimizing the sum of the squares of the distances from the projected model locations to the corresponding image locations. If a projection of the keypoint lies within half the error that was used in Hough transform bins, the keypoint match is kept. If we have more than *3* keypoint matches then the object match is also kept.

3.3.6 Implementation

We adapted a SIFT feature detection and matching implementation by Rob Hess, from Oregon State University, USA. In this implementation, matching doesn't use the Hough Transform. Instead it finds SIFT features in two images, detects the correct matches based on the ratio between the nearest and second-nearest neighbors and ends there. We changed his match implementation to allow importing previously extracted SIFT features from files and to allow comparing not only two, but a batch of files that are passed through the command line as arguments. In this matching, the first file contains SIFT features from a group of objects belonging to a category and the rest are matched against this file to analyze how many positive matches each one produces. For a match to be correctly marked and counted, the ratio between nearest and second nearest neighbor was also modified to *0.35* from the original Hess's value of *0.49*.

Matching is not symmetrical, i.e., it's different to match image A with B and B with A. This has to do with the nearest neighbors: if we have a feature i in the image A and its nearest and second nearest neighbors in image B are, respectively, j and k with a ratio between them lower than the threshold, then the match is valid. But if we compare the images the other way around and feature j in B has a nearest neighbor i and second nearest g in A, and these have a ratio superior to the threshold, then the matching is discarded. So, having a successful matching between features may also depend in the order of matching.

Hess's original implementation finds the nearest neighbors of the *static* features, i.e. the features that serve as a category model, on the feature sets that are currently being tested. In our adapted version, we continue to use this direction of matching. This means we find matches for the features of the category model in the features of the objects that are to be classified. We could also compute the matching in both directions and use the average or biggest number of correct matches, but this wasn't implemented, so we can't attest on the advantage or disadvantage of doing so.

For every pair of images for which we're computing the similarity, we compare their SIFT features and match them to their nearest neighbor. That is, we pair features according to

Figure 3.15: SIFT matching between two images. The matches are nearest-neighbor features of the two images.

their Euclidean distance, as described in section 3.3.5. As it was explained, a ratio between the nearest neighbor and second nearest neighbor is used as a heuristic to select or discard feature matches. We use a ratio of *0.35* and keep matches that are below this threshold. In the end, we count the total number of feature matches for each pair of images. An example of SIFT features matching is portrayed in Figure 3.15.

3.4 Category representation

To represent a category we must start from the initially computed object models to arrive at a broader description (that encompasses all the characteristics of individual objects that are perceived as belonging to that category).

Since the initial models are computed from objects retrieved through a Web search, some will be noise or irrelevant. It's now necessary to discard them. The models are to be matched among themselves and clustered into groups, discarding the ones that are deemed to be less similar to the dominant group after a series of runs of the *K-means* algorithm, as described in the next chapter.

Category models were created according to the type of category they represent: either specific, well defined categories that are modeled after their SIFT features; or more abstract ones, modeled according to their shape. A category representation is built upon the models that are kept:

a) General categories are represented by the set of shape based models (histograms) of individual objects that are known to belong to them.

b) Specific categories are represented by the concatenation of SIFT local features of all images known to contain objects that belong to them.

Chapter 4

From Web search to classification

Summary

After selecting the object representation and similarity measures, as described in the previous chapter, it's necessary to develop an automated image retrieval and category representation and classification system. Since images are obtained without human interference from the Web, good instances of a category will be accompanied by inevitable junk. Hence the need to select the subset of these images, which can be used to build a model, and discard the rest.

4.1 Image retrieval

The image retrieval step uses a *Perl* script, supported by the *WWW::Google::Images* module. It downloads a series of images from the Internet using the *Google* search engine. The process was configured as follows:

- Only JPG images are retrieved.
- The maximum width of an image must be less or equal to *1200* pixels.
- The maximum number of images downloaded for a specific category is *20* and *40* for a genera category. These values are a tradeoff between maximizing the number of good images and minimizing the number of bad images that come along. Because specific categories are associated to more specific queries, it is natural that there is a better *good images/bad images* ratio, hence needing less retrieved images than general categories to maximize the chance of building good representations.
- General categories are searched with a query beginning in *allinurl:*, meaning that all the search terms must be present in the image URL. Since, for the most part, general categories don't have many words in their name, we maximize the chances of retrieving good images if we force the name to be in the URL. This happens because if the image filename (or its directory, etc.) is (or contains) the name of the category, it's much more likely to be a good image than if it's retrieved by analyzing the surrounding text, for example.

- If a search for a specific category doesn't retrieve the maximum number of images and if the query contains quotes, then a new search is conducted, removing the quotes from the query.
- Since we don't use color information, when a search for a general category doesn't return the maximum number of images and the category name has a color in it (e.g. "red apple"), a new search is conducted, removing the color information from the query.
- When searching for a general category, if the maximum number of images is not reached, a new search is conducted, removing *allinurl:* from the beginning of the query. Note that this can happen after the removal of the color information from the query.

4.2 Pre-processing

Before the subset selection and classification *per se*, images of the shape categories are pre-processed. This step is next described.

4.2.1 Object extraction for shape analysis [1]

In both the learning and recognition phase, when dealing with images containing objects for which we want to build a shape representation, we must take into account the possibility that the image contains several other objects. This is not problematic if the we want a SIFT-based model, for two reasons:

a) We use SIFT for specific categories, so the results do not suffer as much from noise within the image. This happens because the query that is passed to the Web search engine is also more specific, contributing to better results. It's very unlikely to have a very specific query return something that has no relation with the desired category.

b) A useful SIFT image may well be composed of several objects that we don't want to segment, because it is the agglomeration that forms the desired representation. That can be the case of a book or CD cover, for example, where any type of design is possible.

But when evaluating the shape of an object we need to check if the image has more than the intended, because outlining the shape of an image with noise will lead to misleading representations. Images with several objects, for example, produce a shape representation that can't be used for anything useful, unless we wanted to identify those objects in the same scene, in the same position. Admitting then that shape categories have images with clutter and possibly even several instances of the same object, it's necessary to produce another set of images from these, where each image ideally represents a single object (Figure 4.1). This is accomplished by means of segmentation and extraction.

We take the Canny edge images, which is a group of white pixels over a black background and follow the contours trying to isolate individual objects. After detecting a pixel that is

[1] ***Note**: Although this step was not implemented by the author of this thesis, it is an integrating part of the jointly developed system which was explored by other members of the UA@SRVC team, and as such, is also mentioned and explained.*

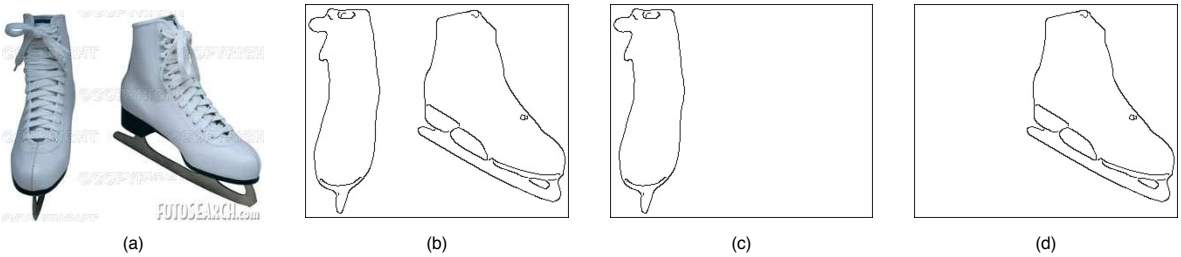

Figure 4.1: Segmentation example. (a) Original Image. (b) Original after Canny Edge Detector. (c-d) Segmented images produced from (a). Note that colors are inverted in the contour images.

part of a contour (white pixels), we use region growing, to extract the shape. The neighboring pixels of the detected pixel area are scrutinized and we determine if each of these pixels should be added to the object being extracted, i.e., if it's also part of the contour. If we get to a bifurcation, its localization is saved and we follow one of the paths. When this path terminates, we backtrack to the last found bifurcation and take the unexplored route.

After the contour is established, then we determine its extreme coordinates (bottom, top, left and right). If a pixel is determined to be inside the contour, i.e., it has contour pixels below, above and at both sides, then it's also marked as white. In a next step, pixels that are inside the same window but are black are also added to the shape if the majority of their *8* neighbors is white. This is useful because a shape is not always defined by a complete and continuous contour. In fact, there are several possibilities for a contour to be a candidate object or object part, as seen in Figure 4.2.

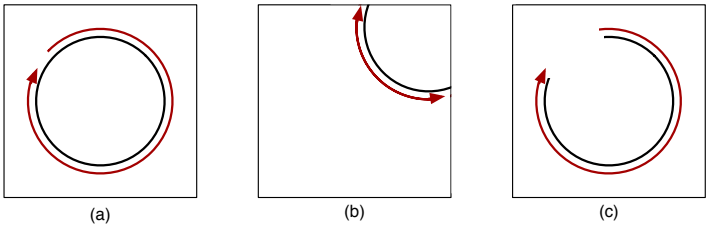

Figure 4.2: Extraction cases. These are the basic cases we might encounter while extracting the objects. (a) The object has a defined an continuous contours. (b) The object contours touch the image edges and are continuous. (c) The contours are not continuous.

If we start in a point and end up in the vicinity of the same point (Figure 4.2(a)), if the contour leads to the image edges (Figure 4.2(b)) or if it is not continuous but defines an area (Figure 4.2(b)) then it's possible that this contour defines an object or a component of one.

The last step is the aggregation of contours. Since one contour doesn't necessarily define an object, then contours are grouped together according to the average distance D of the contour pixels to their geometric center GC and the distance between contours.

This is an iterative process: given the list of contours, for each pair of contours we determine whether they can be aggregated. If two contours are aggregated, then we merge them by

creating a new contour and remove the ones that originated it. The total number of contours is decremented and for the new formed contour we compute its new geometric center, list of border pixels and their new average distance to the geometric center. This process is repeated until all pairs of contours have been tested. The aggregation criterion is explained next.

For contours a and b, with geometric centers $GC_a(x, y)$ and $GC_B(x, y)$, where T is the distance between the latter:

$$T = \sqrt{(GC_a(x) - GC_b(x))^2 + (GC_a(y) - GC_b(y))^2}, \tag{4.1}$$

if the following, where D denotes the average distance of the contour points to their geometric center, is true:

$$T < k \times (D_a + D_b), \tag{4.2}$$

i.e., the distance between the contours (as measured by the distance between their geometric centers) is smaller than the sum of the average distance of them to their own GCs multiplied by a constant factor k[2], then the contours are aggregated and count as an object shape. For an illustration, see Figure 4.3.

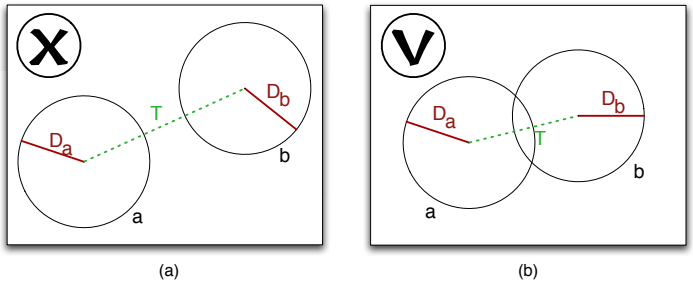

Figure 4.3: Contour aggregation. (a) T is bigger than D_a+D_b. Contours are not aggregated. (b) T is smaller than D_a+D_b. Contours are aggregated. Note that in a circle, the average distance D of the contour points to the center of mass is the radius.

In Figure 4.4 we can see a real example of the aggregation of contours. The leftmost object is extracted alone because the distance T of its geometric center to the geometric center of the middle object is bigger than the sum of both their average distances D to their own geometric center. The rightmost object is aggregated and extracted together with the middle object because the distance T of its geometric center to the geometric center of the middle object is smaller than the sum of both their average distances D to their own geometric center.

Object extraction pseudo code

Algorithm 1 is a simplified description of the object extraction process.

[2]In the current version, the multiplicative factor k has the value of *1*.

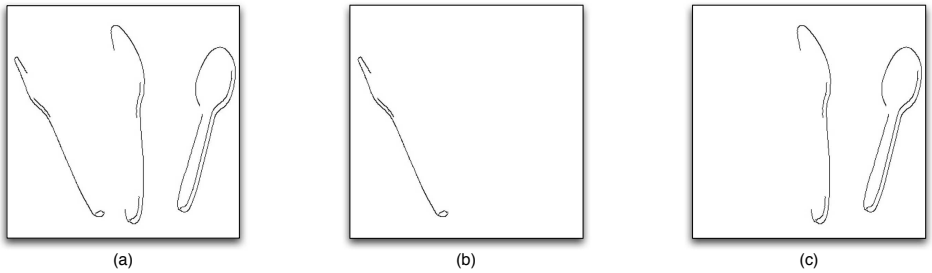

Figure 4.4: Example contour aggregation. (a) Original image. (b) The leftmost object is not aggregated. (c) The center and rightmost objects are aggregated.

Algorithm 1 Object extraction

```
smoothImage(image);
image = extractCannyContours(image);

n_objects = 0;
do {
  objectExtracted = false;

  for (i=1; i<image->width-1; i++) {
    for (j=1; j<image->width-1; j++) {

      if (image->point[i][j] == white ) {
        objectExtracted = true;
        object = regionGrowingObjectExtraction(image->point[i][j]);
        whiteMask(object);
        object->centerOfMass = getCM(object);
        object.avgDistanceToCM = getAvgDist2CM(object);

        object++;
        n_objects++;
        break;
      }
    }

    if (objectExtracted)
      break;
  }
} while (objectExtracted);

for (i=0; i<n_objects-1; i++) {

  for (j=i+1; j<n_objects); j++) {
    dist = getDist(object[i],object[j]);

    if (dist < object[i].avgDistanceToCM + object[i].avgDistanceToCM) {
      newObject = merge(object[i],object[j]);
      newObject.cm = getCM(newObject);
      newObject.avgDistanceToCM = getAvgDist2CM(newObject);

      remove(object[i]);
      remove(object[j]);
      add(newObject,object);

      n_objects--;
    }
  }
}
```

4.2.2 Filtering

In the object extraction step (described in the previous subsection) images are segmented, originating several sub-images for which we compute their Global Shape Contexts.

Before using the GSCs to build the category model in the learning phase, a filtering step is applied. Since images were subjected to a Canny edge detector, they are now composed only of contours. By removing images with too many edges, we will likely discard clutter images, such as highly complex scenes that are rich in noise.

Images are scrutinized according to the number of pixels N_{img} that compose their edges. This process is done by evaluating both the average number of edge pixels N_{avg} and maximum number of edge pixels N_{max} in the category array of images and reject the ones where

$$N_{img} > \frac{N_{avg} + N_{max}}{2} \tag{4.3}$$

4.3 Subset selection

Image sets retrieved from the Web will always have an amount of noise. We want to discard some of this noise, ignoring images that are not faithful representations of the target category, building a category model from the individual models of the images that do represent the object.

In the following sections we will explain how a category is represented, while trying to minimize the amount of noise included in the representation. This noise reduction is more efficient as there are more good images in the set, because the only way to identify junk is by matching it with good images. If there's more noise than good images, then the latter may be considered as being junk.

4.3.1 Subset selection through clustering

In this section we describe how we keep up with the similarities between images, how this similarity is computed by means of matching and how we use this similarity to group and select a subset of images, using the *K-means* clustering algorithm.

Lowe suggests that SIFT matching should be done against a previously built training knowledge base[34], as we explained in section 3.3.5. But because all of our images were retrieved from the Web, we don't possess a trustworthy training dataset. That's what we want to achieve, so we'll filter the initial array of images by computing similarities among themselves. The same also applies for the shape categories' case.

4.3.2 Similarity matrices

The similarity score between several images is kept using a *Similarity Matrix*. This is a common structure to hold similarities in both shape and SIFT matching and it's the starting

point for using the clustering algorithm. In Figure 4.5 we can see an example of a Similarity Matrix. A,B,C and D are images and in the matrix cells are the similarities scores between them. Note that only the top diagonal matrix is necessary.

	A	B	C	D
A	MAX	10.334	53.082	01.453
B	10.334	MAX	77.439	14.700
C	53.082	77.439	MAX	83.029
D	01.453	14.700	83.029	MAX

Figure 4.5: Similarity matrix of images A,B,C and D. The diagonal represents the similarity of an image with itself and thus can be ignored.

.

The similarity scores in the cells of the matrix for shape and SIFT categories are respectively the χ^2 distance and the number of SIFT matches.

4.3.3 Object clustering algorithm

The objective of clustering the objects, or the images, is to filter out the images that don't fit in, and thus are probably junk. Grouping images according to their similarity causes the most homogeneous ones to aggregate together.

Retrieved images are clustered admitting that a good percentage of them are accurate representations of the desired object. This clustering is then a means to separate into groups the true object representations from the noise images. Object clustering is done using the ***K-means*** algorithm. *K-means* clustering was invented in 1956 by H. Steinhaus[48] and we use an iterative refinement heuristic known as Lloyd's algorithm[32]. Lloyd's algorithm and K-means are often used synonymously, but in reality Lloyd's algorithm is a heuristic for solving the K-means problem.

This algorithm starts by selecting, either randomly or using some heuristic function, a k number of points, or centroids, of the complete set. The remaining points are grouped around their closest centroid, partitioning the problem space in k sets. Then the algorithm iterates, and in each iteration, two things are done:

a) Recalculate centroids coordinates for each of the new formed clusters as median points of the region.

b) Redistribute remaining points around their closest centroid.

Iterations stop when the centroids no longer change or when the points stop changing clusters. When the algorithm converges, the problem space has been partitioned around the k median points or centroids. An example of the algorithm is shown in Figure 4.6.

Our implementation of the *K-means* algorithm works by randomly seeding an initial number k of centroids across the group of N images. After selecting this initial group of clusters, the Similarity Matrix obtained in the previous steps is used as the distance measure to group the remaining images to their closest (most similar) centroid. Every time an image joins a

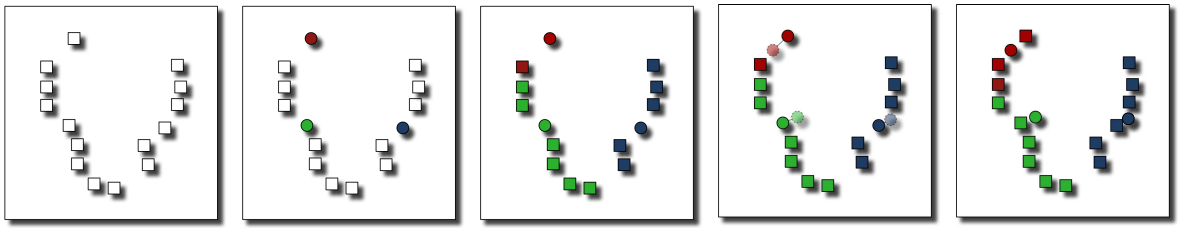

Figure 4.6: *K-means* clustering. Circles (◯) represent centroids, squares (□) represent the elements being clustered. Initial centroids are randomly selected from the initial set. Elements are clustered with the closest centroid and the centroids are then recalculated for the new clusters. The algorithm repeats until no element changes its cluster.

cluster, the similarity s_c of the other images to that cluster c changes and it's given by the arithmetic mean of the similarities to the images in the cluster, i.e., it's a mean of the n similarities s_i to the images that are contained in the cluster.

$$S_C = \frac{\sum_{i=1}^{i \leqslant n} s_i}{T}, i \in c, n \subseteq N \tag{4.4}$$

When the algorithm finishes we have a group that contains most of the images. If there isn't a cluster with more images than all the rest, i.e., if there are several main collections, then we simply select the first main group and compute the cost of moving any image from the other remaining sets to it. The image with the maximum similarity to that main group will moved to it, thus making it the biggest cluster.

Our algorithm (using Lloyd's approach) uses a variable number of clusters: it starts with *1/2* of the total number N of images in the category and incrementally gets to *1/4*. The number of times X_i each object appears in the main cluster in each of these runs is kept and is cumulatively incremented at the end of each run. The images are to be filtered according to this number X_i.

4.3.4 Object ranking

Because the initial randomness leads to variations in the final groups of each run, we run the algorithm *K=100* times, to provide a reliable sampling. The total number of runs is then *100×(1/2×Total images - 1/4×Total images)*. This number of runs allied to the variation of the number of clusters in each of the *100* loops provides more significant results.

At the end of each run of K-means, we count the number of items in the main cluster and add to an accumulator A. At the end of the *100×(1/2×N - 1/2×N)* runs, A will have the cumulative number of elements that were in the main group. Each image also keeps how many times X_i it were in the main group. Since we have the images sorted according to this value, we will keep the images as long the following verifies

$$\sum_{i=0}^{i<N} X_i < 0.85 \times A \tag{4.5}$$

When this limit is exceeded, the current and next images in the sorted list are discarded. These are considered to be the noise images.

Algorithm pseudo-code

Algorithm 2 is a simplified description of our approach.

Algorithm 2 Image clustering

```
for (k = 1; k <= 100; k++) {
  for (i = 1/2 total images; i < 1/4 total images; i++) {

    selectInitialCentroids();             //randomly select a number of images as the initial centroids
    do {
      for (j = 1; j < total images; j++) {
        image[j].group = nearestgroup();  //assign each item to the same group as its closest centroid
                                          //distance to a cluster centroid is the average distance to
                                          //the images in the cluster
      }
    } while (images change groups);

    biggestGroup = determineBiggestGroup();
    totalElementsEverInBiggestGroup += getElementsOf(biggestGroup);

    for (j = 1; j < total images; j++)
      if (image[j].group == biggestGroup)
        image[j].timesInMainGroup++;
  }
}

sortImagesByTimesInMainGroup(image);

accumulator = 0;
j = 0;
while (accumulator < 0.85 * totalElementsEverInBiggestGroup) {  //images are ordered now
  accumulator += image[j].timesInMainGroup;
  print(image[j].name);
  j++;
}
```

For the most part, the Web retrieved images had a good percentage of accurate or usable object representations. By clustering the images with this approach, we group these images together, leaving out the possible garbage.

This concludes the subset selection phase, where we automatically determined which of the retrieved images were good representations of the object.

4.4 Classification

In addition to the semantic image retrieval, there was some work made in the classification process. The process of classification can be defined as the process in which individual objects

are retrieved from complex scenes and correctly identified as belonging to a previous category. In Figure 4.7 we illustrate what a successful classification looks like, though the Figure may not represent the actual performance of the classifiers.

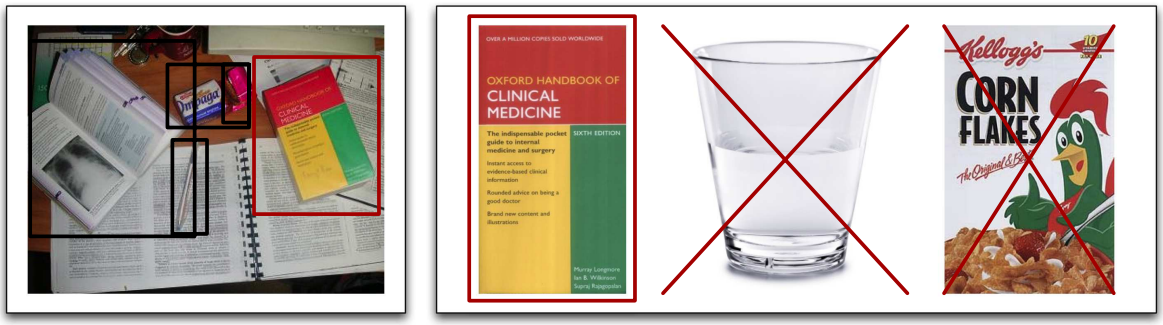

Figure 4.7: Classification process. (a) Individual objects are identified. (b) The previously existent categories. In the figure the book in (a) is correctly retrieved and identified as belonging to the correct category in (b).

The images that contain the objects to be classified are photographs of everyday scenes. Therefore, they may or may not contain the object, and with high probability they contain noise. There might even be several images for the same object.

To extract the objects from these scenes, the images are segmented, based in color saliency detection[41], where saliencies are grouped accordingly to their proximity, forming one or several new images from the original. These new sub images will be used for matching against the shape and SIFT based databases that were built from the images retrieved from the Web. A bounding box is formed in the original picture window to locate the zone that originated the successful matching, hence, pointing to the object in the image.

4.4.1 The matching strategy

Every previous existent model is compared to all models of the real world images. For every model, a rank is produced, according to the similarity between the matches. If there are several images from the real world that lie on the top of several rankings they are likely attractors and are discarded.

Since shape analysis uses GSC and RSR, there will be two rankings produced for each shape category. Of these two, a final ranking is produced, based in a voting system.

There is nothing preventing an image from the workset to be representative of several categories, though. An image can be ranked at the top in a SIFT category and in a Shape category at the same time. Since the methods are different it makes no sense discarding one of the images. Maybe the image has the two objects, or maybe just one of them is the right one, but since there's no way to select which one is wrong in this case, both are used.

For each category, the top ranked images from the workset represents the best matching and the image that is selected.

4.4.2 SIFT classifier

A category is represented by the keypoints of the all the images that compose the category. The candidate SIFT models, computed from the test images, are to be matched against this static database. For every candidate model, the number of feature matches, based in counting Euclidean nearest neighbors below a modified ratio between nearest and second neighbor of 0.35, are counted and used as similarity measure. This will origin a series of scores for each category, representing the similarity between it and all the candidate images.

For each category these scores are ranked, producing a list where at the top are the best candidates for the given category.

After the ranking, the scores are updated taking into account the size of the images. We give a smaller weight to the size and recalculate the scores, where the image size ($width \times height$) is used as a penalty score. Hence, bigger the image, bigger the penalty and lower the score.

4.4.3 Shape Context and RSR classifiers

A category is represented as a series of histograms (either GSCs or RSRs) of the images that were previously retrieved from the Web and selected. These histograms are written in a file and when we compare a category with the real world image, what we do is comparing each of these histograms with the category.

We considered matching the individual histograms and select the matching with most similarity. What's being done is basically compare the test image with each of the images that compose the category and select the best match. This best match has the largest similarity between the test image and that category.

Another way of matching is, instead of using the best match as a similarity score, use the average of the matches with the category histograms. We end up using best matching because average is disturbed by bad results when there can be a good match of just one of the image categories and that is enough to consider that the test image belongs to the category.

Since there are several test images, there will be several similarity scores for each category: one for each test image. These scores are to be ranked, like previously explained.

4.4.4 Voting system

When analyzing shape, we use both the RSR and GSC classifiers. Because they perform differently (see Chapter 5) their combination is desirable, as the performance of an aggregation is often superior to the performance of individual classifiers[14]. The combination of classifiers is done recurring to a weighted voting system: Each of the shape classifiers casts a variable number (*1* to *3*) of votes to their top *3* ranked images. The first image in the rank gets *3* votes, the second *2* and finally the third *1*. In the end of the voting process, the image with the most votes is selected.

Voting is used only in the classification of general objects (by shape analysis) but could as well be extended to the classification of specific objects. The reason this is not done is

because in the developed system only SIFT was used to examine these types of images.

This voting approach makes the system very scalable, because we can insert any number of additional classifiers as long as they vote with the same rules.

Chapter 5

Performance evaluation

Summary

In this chapter we present an evaluation of the methods used in our work. We perform a series of tests in both the subset selection and classification processes.

In subset selection we visually analyze the images that were downloaded from the Internet, marking good and bad representations. Then, after running the subset selection, we check how many good representations were selected. This allows us to infer if there are improvements in automatically selecting a sub-group of the initial set or if this type of automated selection is disadvantageous. Tests are done to specific (SIFT based) and general (shape based) categories, where in the latter we also compare the results of running object extraction first or not.

In the classification process we use *40* categories of objects, each one with several images, to automatically benchmark the shape based classifiers (GSC, RSR and the voting system) and the SIFT based classifier. A group of images of each category is used as the category model and all of the remaining images are matched to these models and automatically classified.

Finally, we present the results obtained on the Semantic Robot Vision Challenge edition of 2008.

5.1 Subset selection of the Web retrieved objects

Images retrieved from the Web are matched among themselves and clustered using the *K-means* algorithm to select the subset that correctly represents the desired object. We perform a manual, visual analysis of the retrieved images before this selection and count how many of them are representations that can be used to build a correct model for the object.

We compare the percentage of good images in the original set with the percentage of good images in the selected subset to determine if improvement exists.

5.1.1 Shape-based subset selection

In this case, we decide that an image represents an object if it's possible to use it to build a good shape model for the object. Images with highly stylized objects (see Figure 5.1(b)), depicting several instances of the desired object (see Figure 5.1(a)), that do not contain the object, or with high noise around the contours (see Figure 5.1(c)) are not good representa-

tions for a shape-based analysis. For instance, an image with several telephones is not a good telephone representation, but if after the extraction and selection we obtain each individual telephone from this image, then the resulting images are useful representations of the object.

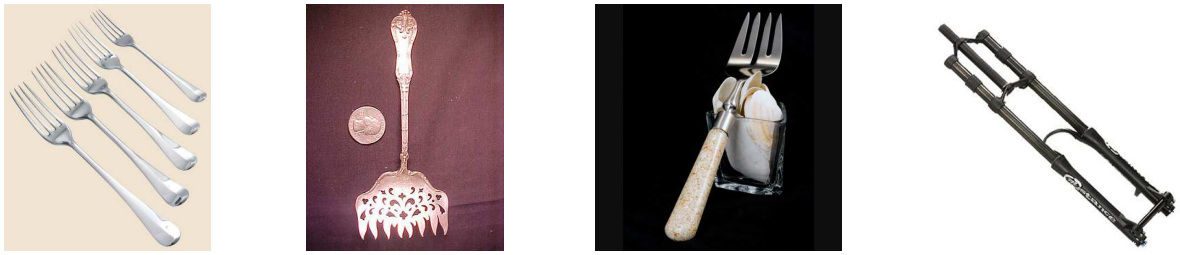

Figure 5.1: Examples of bad shape representations of a fork. (a) Several forks on the same image. (b) Highly stylized fork. (c) Noise around the contour. (d) A different definition of "fork", which is not the required.

We selected *32* categories (some of which were used in SRVC editions) to benchmark the subset selection based on shape analysis using Global Shape Contexts, as it can be seen in both Table A.1 and Table A.2 (pages 74 and 75, respectively). For these *32* categories, several images were retrieved and the good ones marked and counted. We'll present and discuss the results for a shape-based analysis with both object extraction enabled and disabled.

For reading convenience, in Table 5.1 we present the general categories used in these tests, associated to their respective numerical identifiers.

1	bicycle_helmet	9	banana	17	ice_skates	25	rolling suitcase
2	spoon	10	desk	18	telephone	26	red plastic cup
3	trashcan	11	table	19	calendar	27	blue dry erase marker
4	pear	12	eyeglasses	20	hammer	28	tape dispenser
5	orange	13	stapler	21	clothes hanger	29	computer mouse
6	clothes_hanger	14	coffee_cup	22	electric iron	30	chair
7	scientific_calculator	15	whiteboard	23	green apple	31	nokia cellphone
8	fork	16	compact_disc	24	red bell pepper	32	baseball hat

Table 5.1: General categories list for subset selection

Without object extraction

When we competed in the SRVC'08, the UA@SRVC agent wasn't yet using object extraction, so this is an analysis of that version. Without object extraction there are no new images generated. This approach makes it impossible for a good image to be decomposed into several bad images, but has the disadvantage of a bad image being unable to be improved. Each image in the original set originates its corresponding contour image, so after selecting the good images in the initial group, we check the selection to see which ones made through that group and check if there's an improvement in the ratio of good images from the initial set.

If we look at Figure 5.2, we see that there's an improvement in several categories. It appears that a high percentage of good images at the beginning, hence a better Signal-to-Noise

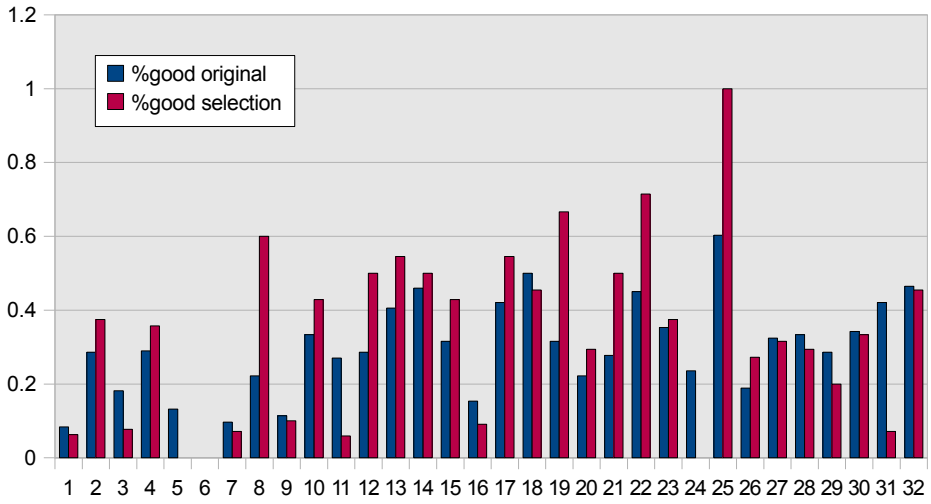

Figure 5.2: Shape subset selection without object extraction: Percentages of good images before (blue) and after (magenta) the subset selection. The vertical axis represents the percentages and the horizontal the categories' numerical identifiers.

Ratio, gives place to a higher percentage of good images after the selection.

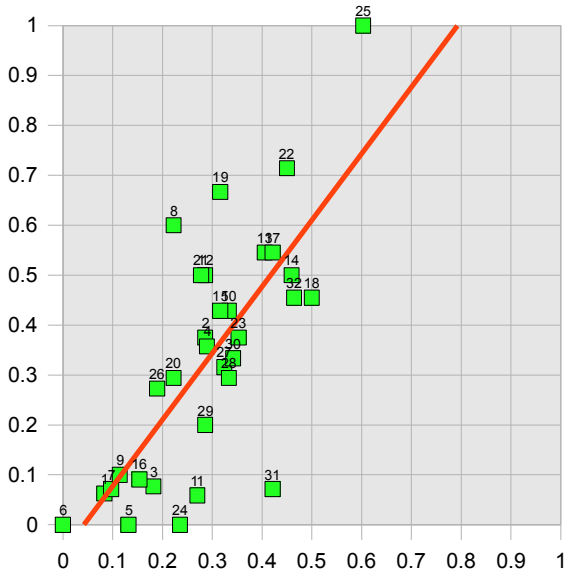

Figure 5.3: Shape subset selection without object extraction: Percentages of good images after the subset selection (yy axis) plotted in function of the respective percentages of good images in the initial set (xx axis). Also shown is the linear regression line, of equation $f(x) = 1.33x - 0.06$, for the plotted points.

In Figure 5.3, where we represent the percentage of good images in the selection as a function of the percentage of good images in the initial set, we get a linear regression line of equation

$$f(x) = 1.33x - 0.06 \tag{5.1}$$

We can therefore conclude that object extraction provides advantage to the automated subset selection of the Internet retrieved images. For $x \gtrsim 0.2$, the regression line has the property of $f(x) > x$, meaning there's improvement. The selection based in K-means clustering improves the initial set.

With object extraction

Object extraction was an improvement introduced in the UA@SRVC system after the competition. In this case, we run object extraction on the initial set, a step that originates a new group of images. Each original image is segmented while extracting objects, creating a set, bigger than the first, of Canny-edge images with the objects' edges. This approach solves the problem where an image has several depictions of the same object: in spite of being possible for a good image to be segmented into bad images, the individual objects in the same image can also be extracted to individual images.

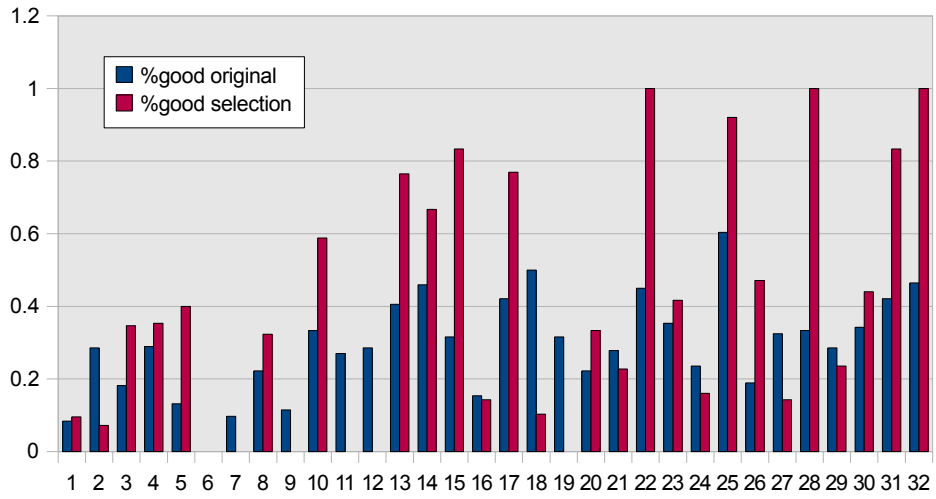

Figure 5.4: Shape subset selection with object extraction: Percentages of good images before (blue) and after (magenta) the subset selection. The vertical axis represents the percentages and the horizontal the categories' numerical identifiers.

In Figure 5.4 we show the number of good images before and after selection. We can see that it varies, although we can also see that for the most part, when there is a good number of good initial images, then the selection effect is positive. We can see this better in Figure 5.5, where we show the percentage of good images after the subset selection as a function of the percentage of good initial images. The linear regression line equation is:

$$f(x) = 1.64x - 0.09 \tag{5.2}$$

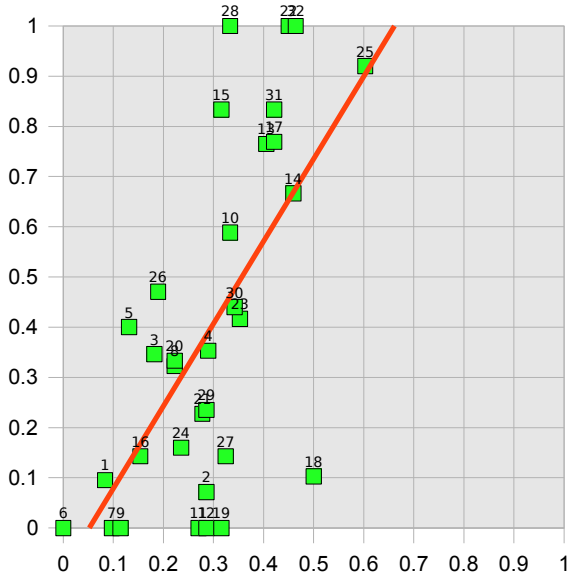

Figure 5.5: Shape subset selection with object extraction: Percentages of good images after the subset selection (yy axis) plotted in function of the respective percentages of good images in the initial set (xx axis). Also shown is the linear regression line, of equation $f(x) = 1.64x - 0.09$, for the plotted points.

The slope of the regression line fundaments our conclusion. We can see that as soon as $x \simeq 0.15$ the regression line has the property of $f(x) > x$, i.e., the values of the percentages of good images in the selected subset are superior to the initial percentages. This line, compared to the one in Figure 5.3, has a bigger slope, which means this method is better in improving the selected subset.

5.1.2 SIFT-based subset selection

Building a good SIFT model for an object doesn't require such a strict selection as is necessary for a good shape model. First, since SIFT is used for specific categories, it will already start with some advantage. Furthermore, because SIFT is tolerant to occlusion, noise, etc., an image can have a partially hidden, deformed or not alone object and still be a good candidate for building a model. Images with more than one instance of a category are also not a problem, because we actually want to maximize the features that characterize a category, and having more than one in an image will not do any harm. This is because our models will be a concatenation of local features, so, having one object per image or more than one will culminate in the same result. In shape model building, having more than one object would be harmful because the Global Shape Context histogram obtained would not represent the object.

In Figure 5.6 we present good and bad examples of images for SIFT modeling.

We selected *25* categories for this test and, as before, counted the initial number of good images and the same number after the subset selection. We determine the percentage of these images in the initial and final set and use this data to draw conclusions on the performance of the SIFT-based automated subset selection.

(a) (b) (c) (d)

Figure 5.6: Good and bad images for SIFT modeling. (a) The image has significant noise, but the branding is clearly visible, so it's a good candidate for modeling. (b) A picture of the entire candy-bar without noise, even though it has a seal in the upper right corner, which is irrelevant for this method. Another good candidate. (c) The image depicts the candy bar, but it's not what we would want to recognize (branding), so we do not consider as a good image. (d) This image represents a candy bar, which although similar, is not the desired (the candy bar package is not the real, but an imitation), and since we would want to tell apart one from the other in a possible recognition scene where both would be present, it's also considered a bad image.

The object categories are identified by numbers in Figures 5.7 and 5.9, and can be matched to their respective names by referring to Table 5.2. The complete results can be seen in Table A.3, on page 76.

1	Coca-Cola can	14	Colgate Total
2	Apple iPhone	15	Dasani water
3	Butterfinger candy bar	16	book "Artificial Intelligence: A Modern Approach"
4	Dvd 300	17	The Economist magazine
5	CD "Final_Straw" by Snow Patrol	18	cd "Hey Eugene"
6	Cadbury Caramilk	19	Kellogg's Corn Flakes
7	Twinings Earl Grey Tea	20	Pringles Can
8	Nintendo Wii box	21	Fanta Orange Can
9	CD "Look-Alikes_Jr." by Joan_Steiner	22	Sony AIBO
10	Ritter Sport Dark Chocolate	23	Ryobi drill
11	Aerius Allergy	24	book "Probabilistic Robotics"
12	CD "Introducing Joss Stone"	25	Crayola Crayons box
13	DVD "Madagascar"		

Table 5.2: Specific categories list for subset selection

As seen in Figure 5.7, the performance of subset selection with SIFT for specific categories is very good. These images, which are rich in features, provide a solid basis for the clustering algorithm to filter out the noise. Only one category - *Aerius Allergy* (Category number *11*) - makes this approach give worst results in the final, filtered set than in the initial group of images.

It's also worth to note that, ignoring the *CD "Look Alikes Jr." by Joan Steiner* category (Category number *9*, which had *0* good images initially, hence being impossible to improve), the category *Aerius Allergy* is the one with the lowest percentage of good images in the initial group. Also, this query returns two inconsistent good instances of the category, that are both considered good candidates because we don't know which one would be the most common, as in the retrieved set they are both equally represented. These are shown in Figure 5.8. Because we have few good representations of this category and the ones we do have aren't uniform, the clustering and subset selection doesn't perform so well. We can also see that many categories originated a subset where *100%* of the images are good representations.

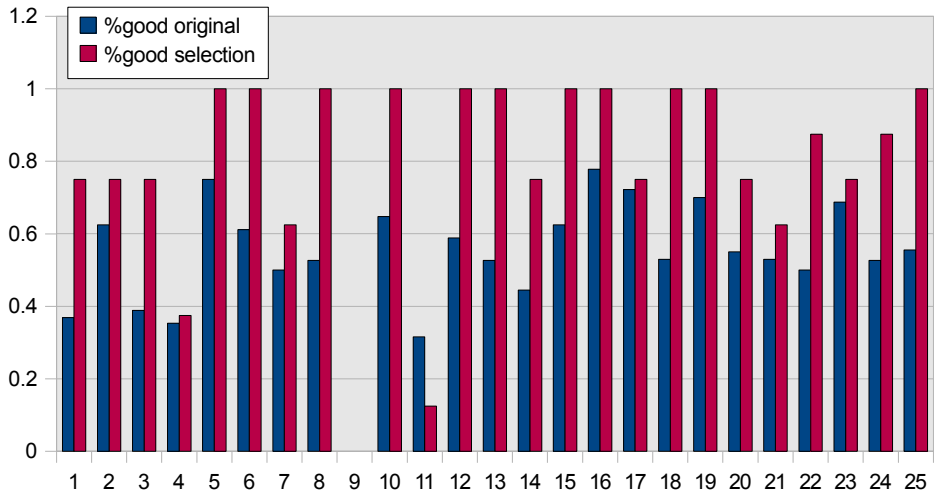

Figure 5.7: Subset selection with SIFT: percentages of good images before (blue) and after (magenta) the subset selection. The vertical axis represents the percentages and the horizontal the categories' numerical identifiers.

(a) (b)

Figure 5.8: Aerius: two different, valid brandings.

In Figure 5.9, the percentage of good images in the subset (yy axis) is plotted as a function of the percentage of good images in the initial set (xx axis). The figure also shows a linear regression line of equation

$$f(x) = 1.29x + 0.1, \tag{5.3}$$

where from the beginning ($x = 0$) the values of $f(x)$ are always higher than x, meaning a significant improvement of the subset relatively to the initial group. The slope is inferior to the one of the linear regression line in Figure 5.5, that shows shape-based subset selection with object extraction, even though we have a significantly better improvement in this case. This fact can be explained by the quantity of categories located in the lower values of x, i.e., categories with poor percentages of good images. Here, this quantity is smaller, a basis that also accounts for the better performance.

In Table A.3 we present the data retrieved from the experiment.

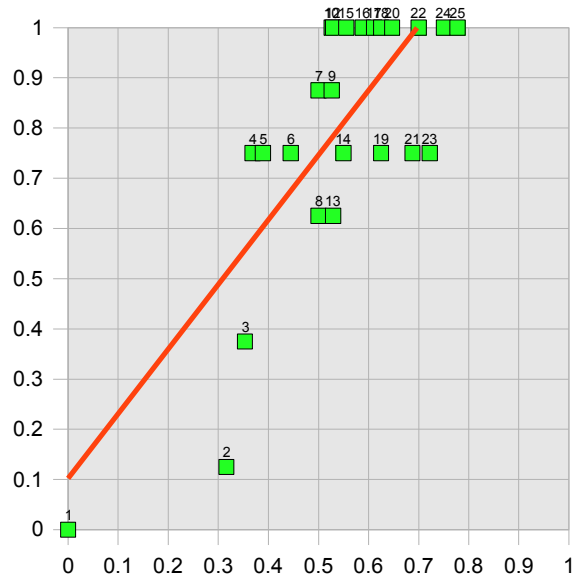

Figure 5.9: Subset selection with SIFT: percentages of good images after the subset selection (yy axis) plotted in function of the respective percentages of good images in the initial set (xx axis). Also shown is the linear regression line, of equation $f(x) = 1.29x + 0.1$, for the plotted points.

5.2 Classification

To test the classifier we used two different groups of categories. One group is comprised of general categories, i.e., categories that are more encompassing and comprehensive and therefore more heterogenous. The other group is made of specific categories, that are relatively unique and well defined.

Classification tests are conducted by selecting *3* images of each category for learning/model creation and measuring the success of correctly classifying the remaining images using the:

- GSC classifier *(1)*
- RSR classifier *(2)*
- SIFT classifier *(3)*
- Voting system between the classifiers (either between *(1)* and *(2)*, or *(1)*, *(2)* and *(3)*)

For both the specific and general groups there are *40* categories to be classified. Tests of the classifiers start with only *2* categories, where the chance of random correct classification is *50%*, and increment up to this maximum number of *40* categories, where by chance alone, the correct classification probability of each image is *1/40*, or *2.5%*.

5.2.1 General categories classification

The *40* general categories are listed in Table 5.2.1. There are approximately *100* images for each category, giving a total count of about *4000* individual images. These images have been collected in experiments on the framework of the LANGG project and do not have

background noise, occlusion or deformation. It's worth to note that because of abstraction of generic names, different objects can belong to the same category. As a consequence, besides the same object translated and rotated several times in each category, occasionally we have different objects that also are rotated and translated several times.

General Categories			
Letter A	Coffee Spoon	Lighter	Remote
Battery	Cup	Mobile	Screw
Bottle Top	Duster	Mouse	SD
Box	Floppy	Nail	SF
Boy	Glove	One	Stapler
CD	Glue Bottle	Passbook	Staple Remover
Cig Box	Horse	Pen	Sugar Packet
Circle	Can	Pencil	Table Fork
Circuit Board	Ink Remover Bottle	Penguin	Table Knife
Coffee Cup	Key	Post It	Table Spoon

Table 5.3: General categories list for classification

In a first test, we compare the two shape-based classifiers, RSR and GSC, and the Voting System with these two classifiers. We will determine how much the increase in the number of categories impacts each of these classification methods.

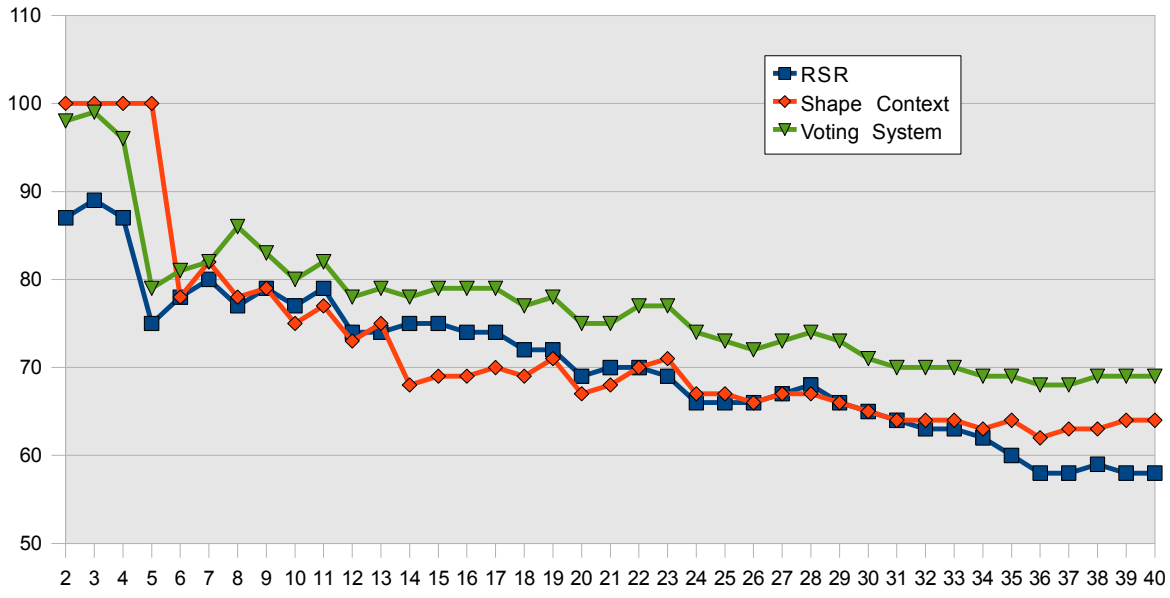

Figure 5.10: Shape-based classifiers and voting system on general categories. Performance of the shape classifiers RSR (□) and GSC (◇), and the voting system (▽) using these two. The xx axis represents the number of categories and the yy axis is the percentage of successful classification. In each iteration a new category is added, based on alphabetical order.

In Figure 5.10 we can visually attest that the GSC classifier outperforms both the RSR

classifier and the voting system for up to *5* different categories, with a *100%* efficient classification. For this small number of categories, the usage of a voting system is actually harmful. But for *6* categories or more, the voting system performs better than the individual classifiers with its lowest scores being around *70%*. This corroborates the conclusions of Dietterich[14], who asserts that the performance of an aggregation of classifiers is often superior to the performance of individual classifiers.

The individual classifiers have a linear decrease in performance with the increase of the number of categories, after a somewhat pronounced break around the *4-6* categories interval. For *6-40* categories, they overlap and intermittently outperform each other, with the GSC classifier declining more rapidly until *14* categories, after which it's more stabilized, and the RSR with a more gradual decline since *7* categories.

In a second test, we introduce the SIFT classifier and change the Voting System to allow votes from all the *3* individual classifiers.

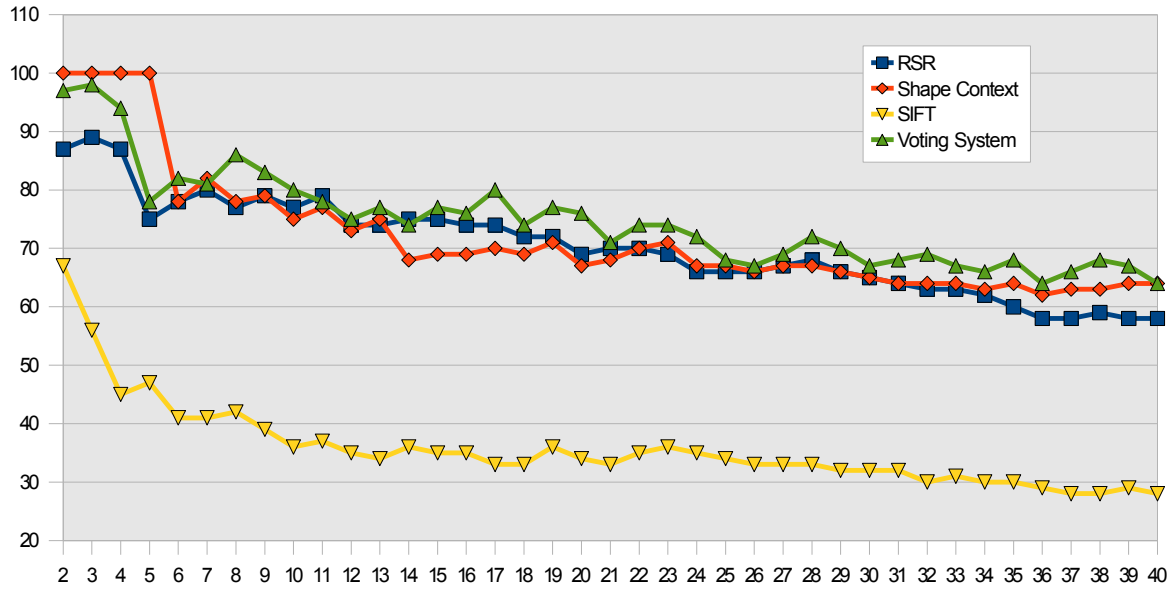

Figure 5.11: Shape-based classifiers, SIFT and voting system on general categories. Performance of the shape classifiers RSR (□) and GSC (◇), SIFT (▽), and the voting system using all the three mentioned individual classifiers (△). The *xx* axis represents the number of categories and the *yy* axis is the percentage of successful classification. In each iteration a new category is added, based on alphabetical order.

In Figure 5.11 we see that SIFT-based classification has a poor performance on general categories. This is because SIFT works better with categories that have highly distinctive features. These generic categories are very simple and do not have such features. Besides, since the categories are generally described, sometimes there are different objects in the same category, making the category heterogenous. With few and sometimes different features in objects belonging to the same category, it's hard for SIFT to correctly match them.

The classifier can correctly perform its work when the number of categories is low, be-

cause even though there are few features, they are still different from one category to another. When this number increases, these features become repeated in different categories and are not so distinctive anymore.

The Voting System doesn't suffer a great change, because the majority (the shape-based classifiers) still votes on the correct classifications, being that only in few cases, the SIFT classifier vote actually made any difference. But when this happened, there was a decrease in the score.

We can therefore conclude that, for generic categories, because of high intra-category variance and few distinctive keypoints, SIFT-based classification is not advisable, while shape-based classification provides very good results that can be improved even further with an agglomeration of classifiers and a voting strategy.

5.2.2 Specific categories classification

The *40* specific categories are listed in Table 5.4. There are *20* images for each category, accounting for a total *800* individual images. They can have any orientation or localization inside the image.

Specific Categories	
Aquafresh Toothpaste	Mars
Ariel Detergent	Mountain Dew Can
AI: a Modern Approach Book	Nesquik Cereal Box
Blue Smint	Nokia 3310
Camel Tobacco Box	Nokia 6100
Chocapic Cereal Box	Nokia N95
Coca-cola Can	Organics Shampoo
Colgate Toothpaste	Persil Detergent
Corn Flakes Box	Portugues Tobacco Box
Dasani Bottle	Reeses Pieces
Fanta Can	Serra Da Estrela Bottle
Harry Potter DVD	Sin City DVD
HS Shampoo	Skip Detergent
Icedtea Can	Snickers
In Between Dreams CD	Sprite Can
I Robot DVD	The Fellowship Of The Ring Book
L&M Tobbaco Box	Tide Detergent
Linic Shampoo	Twix
M&M's	Vitalis Bottle
Maltesers	X-Men 3 DVD

Table 5.4: Specific categories list for classification

We first conduct a test using only the SIFT-based classifier. As seen in Figure 5.12 this

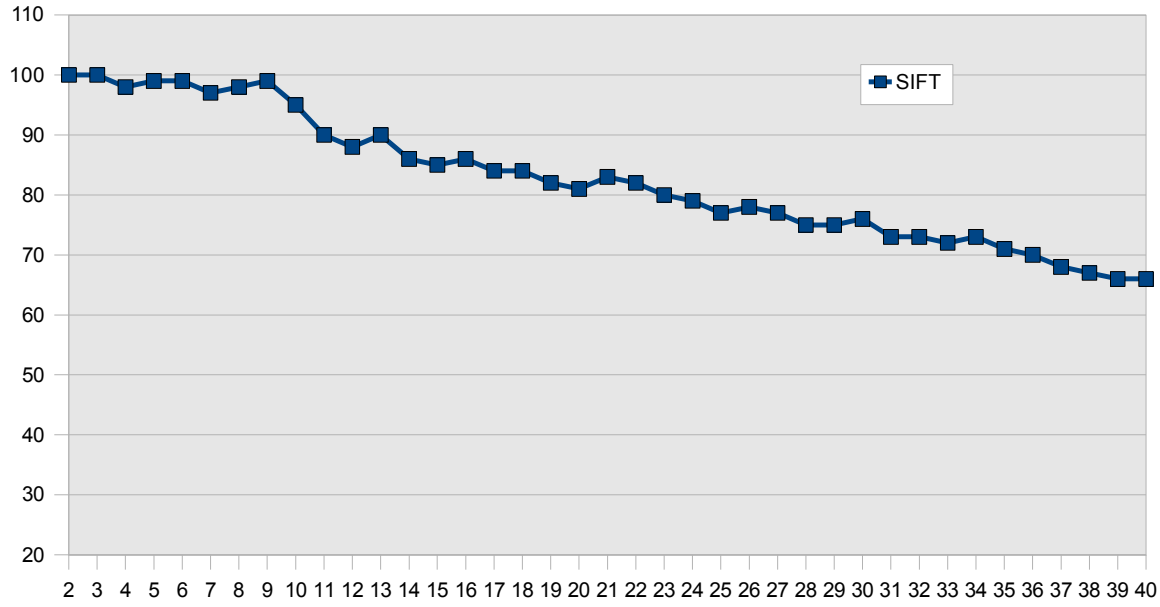

Figure 5.12: SIFT on specific categories. The *xx* axis represents the number of categories and the *yy* axis is the percentage of successful classification. In each iteration a new category is added, based on alphabetical order.

classifier now has much better results than with generic categories. With this new group of categories, objects are rich in highly distinctive features, such as a branding or a design that repeats itself over the entire category. It has a better performance with up to *13* categories with correct classifications of *90%* or more. Even with more categories it has a performance comparable to the voting system with RSR and GSC in general categories (see Figure 5.10).

If we introduce the shape-based classifiers and the voting system where all three individual classifiers (SIFT, RSR and GSC) cast their votes, it doesn't provide any performance increase. In Figure 5.13 we see that shape-based classifiers perform very badly with these types of categories.

Initially, because the shapes of few categories are still very distinguishable the classifiers can correctly identify to which category the objects belong. But as more categories with similar shapes and very rich in contours are added, their performance significantly deteriorates, being even worst than SIFT for general categories (Figure 5.11).

Democracy also doesn't help, because the majority (shape-based classifiers) pulls the performance down by casting the majority of votes in the wrong categories. We don't test the voting system with shape only, because first, as established previously, it wouldn't make much of a difference in the votes and second because it's obvious that a shape analysis isn't a good approach for this case.

It's then easy to determine that when the categories are specific, our best bet is to use only the SIFT-based classification.

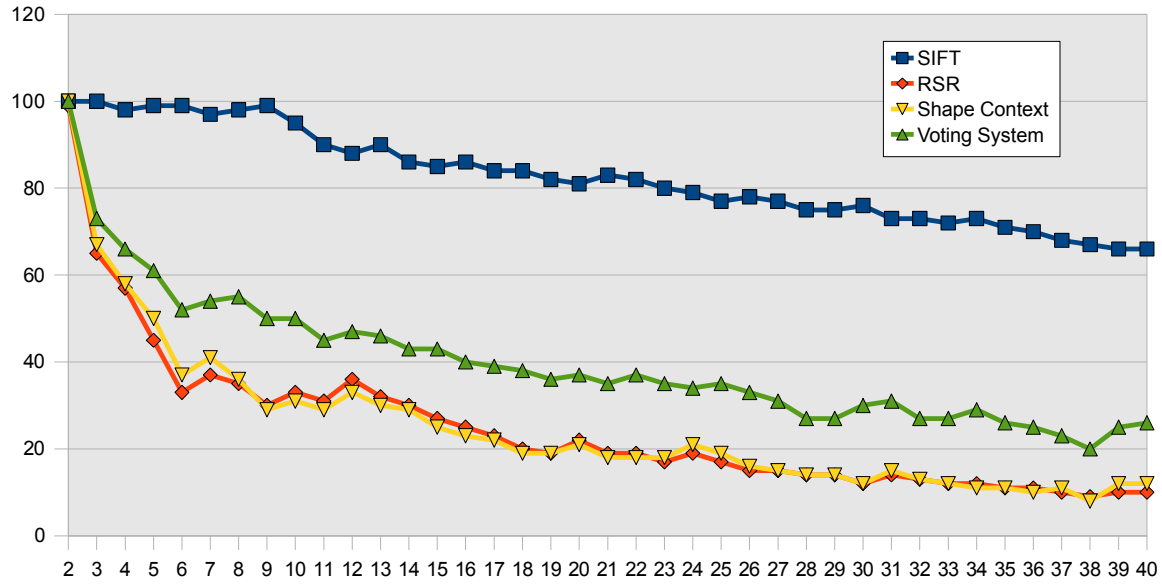

Figure 5.13: Shape classifiers, SIFT and voting system on specific categories. Performance of the shape classifiers RSR ($\diamond$) and GSC ($\triangledown$), SIFT ($\square$), and the voting system using all the three mentioned individual classifiers ($\triangle$). The *xx* axis represents the number of categories and the *yy* axis is the percentage of successful classification. In each iteration a new category is added, based on alphabetical order.

5.3 The SRVC scenario

In this section we present the results obtained in our participation in the software league of the Semantic Robot Vision Challenge '08. Note that, in this stage, the SRVC@UA agent didn't use the RSR classifier nor the voting system. The results were obtained with the GSC and SIFT classifiers only. The list of *20* categories that were supposed to be found in the competition round is detailed in Table 5.5.

CD "Retrospective" by Django Reinhardt	book "Paris to the Moon" by Adam Gopnik
apple	saucepan
remote control	Spam
digital camera	Ritz crackers
upright vacuum cleaner	book "Big Book of Concepts"
DVD "I, Robot"	banana
DVD "300"	game "Crysis"
eyeglasses	Doritos Blazin' Buffalo Ranch
fax machine	ulu
Kiwi Strawberry Snapple	frying pan

Table 5.5: List of categories to be found in the SRVC'08

In the set, the category "eyeglasses" was not present in the environment, while there were the following *3* distracting categories: a Dole Pineapple, a package of Reese's Butter Cups and Bounty paper towels.

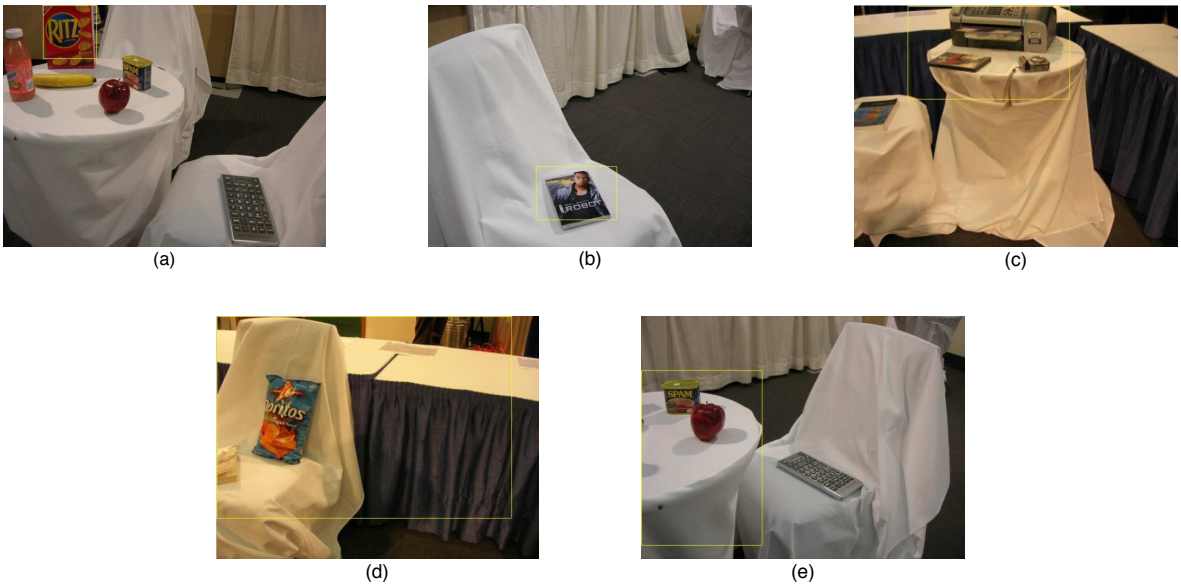

Figure 5.14: Our results (*2nd* place). (a) Ritz Crackers - *3* points. (b) I, Robot - *2* points. (c) Fax machine - *1* point. (d) Doritos - *0* points. (e) Spam - *0* points.

There were *3* teams both in the hardware and software league. The hardware winning team was the University of British Columbia and our competitors in the software league were the National University of Singapore (*1st* place) and the University of Texas at Austin (*3rd* place). Our team got the *2nd* place, with a total of *5* categories found (Figure 5.14), or with a more detailed performance of:

- *33* minutes of Internet search time + *5* minutes of run time.
- *1* instance of a general category found: "fax machine".
- *4* instances of specific categories found: "Doritos", "Spam", "Ritz", "I, Robot".
- *6* points of score - not all of our detections got full points, because of the size of the bounding box (see Figure 5.14 for a detailed viewing of the found categories and their score).

These results were just a little bit worse than in the trial run, where we identified *2* instances of general categories and *4* instances of specific categories. There were more specific than general categories found because the former used SIFT-based classification. This classifier gave much better results in the competition reality because SIFT is more robust to background clutter, while the GSC-based classification suffered with the background noise that object extraction wasn't able to completely remove from the photos of the challenge.

Comparison with other teams

The team in the first place got *8* categories, where *6* of which were scoring detections and *2* were non-scoring detections (see Figure 5.15 for more details). The University of Texas

detected *1* object only, thus rendering it the *3rd* place. Even in the hardware league, the biggest number of detected categories was *6* (UBC), so the performance of the UA@SRVC agent was very satisfying.

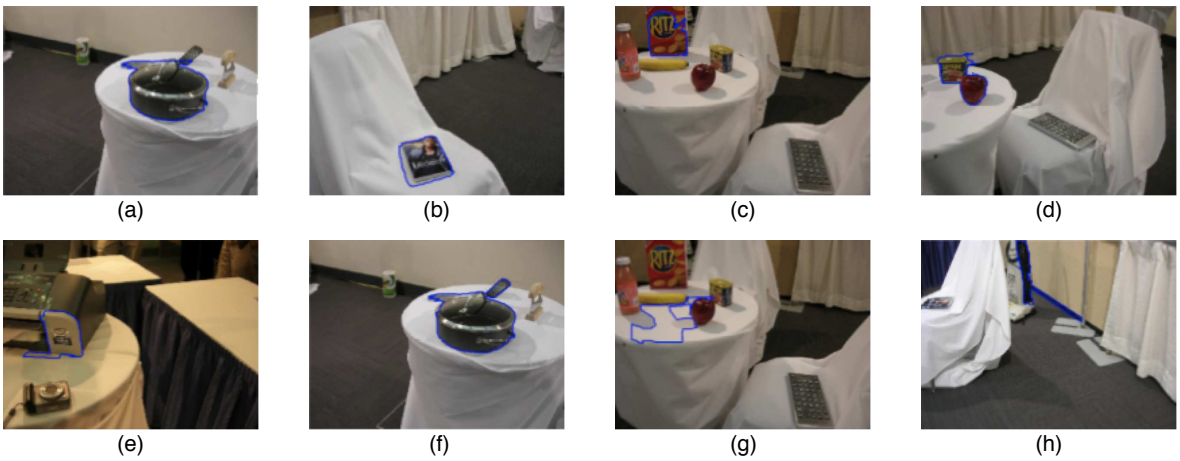

Figure 5.15: *1st* place results. (a) Sauce Pan - *3* points. (b) I, Robot - *3* points. (c) Ritz - *3* points. (d) Spam - *1* point. (e) Fax - *1* point. (f) Ulu - *0* points (ulu is the object at the sauce pan's right). (g) Banana - *0* points. (h) Vacuum - *1* point.

Our running time was also several times faster than the other teams. This happened because of their reliance on interpreted code (using *Matlab*) while we used compiled code (*C/C++*).

Chapter 6

Conclusions and future work

Summary

This final chapter is a short and concise conclusion to this dissertation. It starts with a recapitulation of the work and exposes the conclusions to which we came during its development and evaluation. We also explore the possibilities of expansion and improvement that could be achieved in related future work.

6.1 Overview

In this work, a semantic image retrieval, subset selection and classification system is described. We investigated the use of shape contexts as global shape descriptors (GSC) for general categories and SIFT descriptors for specific categories. We use these descriptors to determine pairwise similarity scores between images and use this information for clustering the images retrieved for each category set. The top ranked images in each category set, determined according to the number of times they appear in the main cluster, are selected as the subset of good category representations.

Classifiers based both on SIFT local features and global shape descriptors were also developed. These classifiers match representations of objects in the environment with category models and pinpoint potential objects that belong to each category within these images. Category models are formed by the SIFT features and GSC descriptors of the images selected as category representative instances.

Finally, we measure the performance of both the image subset selection environment and classification parts. Tests were also conducted where image classification based on Roy's Shape Representations is introduced. The results of both the individual classifiers and of their combination using a voting system are finally compared.

6.2 Conclusions

The analysis of the performance of both the individual parts and of the integrated system can be enumerated in the following main points, each with its own advantages, disadvantages and/or characteristics:

1. Retrieving images from the Web using only keyword based searches results in sets of images that cannot be directly used for category representations.

2. The use of Shape Contexts as global shape descriptors provides a faster and simpler shape descriptor than the original Shape Context approach. By making each shape described as only one Shape Context centered in its center of mass instead of the use of one Shape Context for each point makes both the histogram creation and matching much faster. Rotation invariance can be achieved simply by rotating the obtained histogram. On the other hand, by selecting the Shape Context as a global descriptor we necessarily lose the better representation accuracy that the original approach delivers. Nevertheless, for our case, the selected approach had very satisfying results.

3. We have determined that, for specific categories, the best type of representation and classification is based on the SIFT features of that category. That happens because the most discriminative features of these categories are not related to their shape but to their richness in gradients. Shape is useful for general categories, since it's an identifying trait for these types of categories.

4. For SIFT-based classification and subset selection, relying only on the number of correct features' matches as a similarity measure yields mixed results. Although it performs rather well on specific categories it's possible that other methods can bring SIFT to the same performance level as the shape based approaches in some general categories. It's also possible that it can make SIFT more robust to the increase of the number of acquired categories in the case of specific categories, as seen in Figure 5.12

5. Because our work ignores color information, such categories as "*green apple*" or "*red pepper*" do not benefit from this distinguishing feature.

6. By using K-means clustering on the Web retrieved images, we select a subset of those images to build a suitable category model based on how often they appear in the main cluster, after a determined number of runs. This subset is, in most cases, composed of the best images of the category, provided there is a sufficient number of good images in the initial set. A low signal-to-noise ratio on this set, however, significantly contributes to the degradation of the final selected subset.

7. A voting system relying on the agglomeration of shape based classifiers (in our case, GSCs and RSRs) for generally described categories yields better global results than each individual classifier. Also, its performance is also more stable, suffering an inferior decline with the increase in the number of categories than the voting classifiers. Because SIFT-based classification performs significantly worse in general categories and shape-based classification in specific categories, combining these two methods is ineffective in the sense that doesn't improve global performance.

6.3 Future work

Future work built upon this document and the system it describes can benefit and/or tackle on some points that were left aside during the development of this work or thought after the analysis of its performance:

- Classification based on GSCs presently matches the target image to each GSC in each category model and uses as a similarity score the best match. Other similarity measures, such as average match or the average of the top N matches can also be explored.
- The current version of the UA@SRVC currently only uses *Google* as a search engine for retrieving images. Because sometimes in the initial set there are few or even none good category representations, the use of more search engines (*Yahoo*, *Altavista*, etc.) as well as online image services (*flickr*,*photobucket*,etc.) can improve this number.
- Color information can be used, together with the already used SIFT and shape based methods, weighting it in both the subset selection and classification phases. A part-based model using color information can also be another new type of object representation and matching.
- The unsymmetrical matching problem in SIFT can be better explored. For example, match two images in both directions and choose the highest or average number of correct matches as a similarity score. Also, infer on the performance change when Hough transform is also used, as described by Lowe.
- Improve the segmentation algorithm and inquire on the possibility of a better method to discriminate two different shapes while extracting objects: not always two different shapes are extracted individually and are sometimes aggregated together. A better way to remove noise is also desirable.
- Use other object representations methods and investigate if they improve the performance. Also, some other learning methods can be tried, namely Multiple Instance Learning (MIL). Like described in 2.1.7, Vijayanarasimhan and Grauman[52] used this method and obtained good results. However, since they only use SIFT features we may benefit by continuing to use more than one type of object representation and matching.
- Integrate the system in a hardware platform for entering the hardware league of the SRVC.

As it was put in the SRVC '08, "*the ultimate goal is to have a robot that will be able to get your slippers for you*".

Appendix A

Performance tables

In this appendix we present the complete data resulting from the subset selection experiments described in Chapter 5.

- Table A.1, on page 74, contains the results of the analysis of shape-based subset selection without object extraction.
- Table A.2, on page 75, contains the results of the analysis of shape-based subset selection with object extraction.
- Table A.3, on page 76, contains the results of the analysis of SIFT-based subset selection.

n	Category	Initial Set		Selection Set		Percentages (%)		
		Total Imgs	Good Images	Total Imgs	Good Images	Good Original	Good Selection	Variation
1	bicycle_helmet	36	3	16	1	8.33	6.25	-2.08
2	spoon	35	10	16	6	28.57	37.5	8.93
3	trashcan	33	6	13	1	18.18	7.69	-10.49
4	pear	38	11	14	5	28.95	35.71	6.77
5	orange	38	5	9	0	13.16	0	-13.16
6	clothes_hanger	39	0	22	0	0	0	0
7	scientific_calculator	31	3	14	1	9.68	7.14	-2.53
8	fork	36	8	15	9	22.22	60	37.78
9	banana	35	4	10	1	11.43	10	-1.43
10	desk	39	13	14	6	33.33	42.86	9.52
11	table	37	10	17	1	27.03	5.88	-21.14
12	eyeglasses	35	10	16	8	28.57	50	21.43
13	stapler	37	15	11	6	40.54	54.55	14
14	coffee_cup	37	17	14	7	45.95	50	4.05
15	whiteboard	38	12	14	6	31.58	42.86	11.28
16	compact_disc	26	4	11	1	15.38	9.09	-6.29
17	ice_skates	38	16	11	6	42.11	54.55	12.44
18	telephone	40	20	11	5	50	45.45	-4.55
19	calendar	38	12	12	8	31.58	66.67	35.09
20	hammer	36	8	17	5	22.22	29.41	7.19
21	clothes hanger	36	10	12	6	27.78	50	22.22
22	electric iron	40	18	14	10	45	71.43	26.43
23	green apple	34	12	16	6	35.29	37.5	2.21
24	red bell pepper	34	8	13	0	23.53	0	-23.53
25	rolling suitcase	63	38	11	11	60.32	100	39.68
26	red plastic cup	37	7	11	3	18.92	27.27	8.35
27	blue dry erase marker	37	12	19	6	32.43	31.58	-0.85
28	tape dispenser	39	13	17	5	33.33	29.41	-3.92
29	computer mouse	35	10	10	2	28.57	20	-8.57
30	chair	38	13	15	5	34.21	33.33	-0.88
31	nokia cellphone	38	16	14	1	42.11	7.14	-34.96
32	baseball hat	28	13	11	5	46.43	45.45	-0.97

Table A.1: Analysis of shape-based subset selection without object extraction

n	Category	Initial Set		Selection Set		Percentages (%)		
		Total Imgs	Good Images	Total Imgs	Good Images	Good Original	Good Selection	Variation
1	bicycle_helmet	36	3	21	2	8.33	9.52	1.19
2	spoon	35	10	14	1	28.57	7.14	-21.43
3	trashcan	33	6	26	9	18.18	34.62	16.43
4	pear	38	11	17	6	28.95	35.29	6.35
5	orange	38	5	15	6	13.16	40	26.84
6	clothes_hanger	39	0	2	0	0	0	0
7	scientific_calculator	31	3	11	0	9.68	0	-9.68
8	fork	36	8	31	10	22.22	32.26	10.04
9	banana	35	4	10	0	11.43	0	-11.43
10	desk	39	13	17	10	33.33	58.82	25.49
11	table	37	10	19	0	27.03	0	-27.03
12	eyeglasses	35	10	18	0	28.57	0	-28.57
13	stapler	37	15	17	13	40.54	76.47	35.93
14	coffee_cup	37	17	18	12	45.95	66.67	20.72
15	whiteboard	38	12	12	10	31.58	83.33	51.75
16	compact_disc	26	4	14	2	15.38	14.29	-1.1
17	ice_skates	38	16	13	10	42.11	76.92	34.82
18	telephone	40	20	39	4	50	10.26	-39.74
19	calendar	38	12	38	0	31.58	0	-31.58
20	hammer	36	8	21	7	22.22	33.33	11.11
21	clothes hanger	36	10	22	5	27.78	22.73	-5.05
22	electric iron	40	18	11	11	45	100	55
23	green apple	34	12	24	10	35.29	41.67	6.37
24	red bell pepper	34	8	25	4	23.53	16	-7.53
25	rolling suitcase	63	38	25	23	60.32	92	31.68
26	red plastic cup	37	7	17	8	18.92	47.06	28.14
27	blue dry erase marker	37	12	35	5	32.43	14.29	-18.15
28	tape dispenser	39	13	11	11	33.33	100	66.67
29	computer mouse	35	10	17	4	28.57	23.53	-5.04
30	chair	38	13	25	11	34.21	44	9.79
31	nokia cellphone	38	16	12	10	42.11	83.33	41.23
32	baseball hat	28	13	17	17	46.43	100	53.57

Table A.2: Analysis of shape-based subset selection with object extraction

n	Category	Initial Set		Selection Set		Percentages (%)		
		Total Imgs	Good Images	Total Imgs	Good Images	Good Original	Good Selection	Variation
1	Coca-Cola can	19	7	8	6	36.84	75	38.16
2	Apple iPhone	16	10	8	6	62.5	75	12.5
3	Butterfinger candy bar	18	7	8	6	38.89	75	36.11
4	Dvd 300	17	6	8	3	35.29	37.5	2.21
5	CD “Final_Straw” by Snow Patrol	16	12	8	8	75	100	25
6	Cadbury Caramilk	18	11	8	8	61.11	100	38.89
7	Twinings Earl Grey Tea	18	9	8	5	50	62.5	12.5
8	Nintendo Wii box	19	10	8	8	52.63	100	47.37
9	CD “Look-Alikes_Jr” by Joan_Steiner	32	0	8	0	0	0	0
10	Ritter Sport Dark Chocolate	17	11	8	8	64.71	100	35.29
11	Aerius Allergy	19	6	8	1	31.58	12.5	-19.08
12	CD “Introducing Joss Stone”	17	10	8	8	58.82	100	41.18
13	DVD “Madagascar”	19	10	8	8	52.63	100	47.37
14	Colgate Total	18	8	8	6	44.44	75	30.56
15	Dasani water	16	10	8	8	62.5	100	37.5
16	book “Artificial Intelligence: A Modern Approach”	18	14	8	8	77.78	100	22.22
17	The Economist magazine	18	13	8	6	72.22	75	2.78
18	cd “Hey Eugene”	17	9	8	8	52.94	100	47.06
19	Kellogg’s Corn Flakes	20	14	8	8	70	100	30
20	Pringles Can	20	11	8	6	55	75	20
21	Fanta Orange Can	17	9	8	5	52.94	62.5	9.56
22	Sony AIBO	20	10	8	7	50	87.5	37.5
23	Ryobi drill	16	11	8	6	68.75	75	6.25
24	book “Probabilistic Robotics”	19	10	8	7	52.63	87.5	34.87
25	Crayola Crayons box	18	10	8	8	55.56	100	44.44

Table A.3: Analysis of SIFT-based subset selection

Bibliography

[1] Adnan, A., Gul, S., Ali, M., and Dar, A. H. Content based image retrieval using geometrical-shape of objects in image. In *Emerging Technologies, 2007. ICET 2007. International Conference on* (2007), pp. 222–225.

[2] Belongie, S., and Malik, J. Matching with shape contexts. In *Content-based Access of Image and Video Libraries, 2000. Proceedings. IEEE Workshop on* (2000), pp. 20–26.

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

[4] Biederman, I. Recognition by components: A theory of human image understanding. 115–147.

[5] Blei, D. M., Ng, A. Y., Jordan, M. I., and Lafferty, J. Latent dirichlet allocation. *Journal of Machine Learning Research* (2003).

[6] Borg, I., and Groenen, P. *Modern Multidimensional Scaling: Theory and Applications (Springer Series in Statistics)*. Springer, 1996.

[7] Brunelli, R., and Poggio, T. Face recognition: features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15* (1993), 1042–1052.

[8] Burl, M. C., and Perona, P. recognition of planar object classes. In *In Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn* (1996), pp. 223–230.

[9] Celik, D., and Elgi, A. 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* (2005), pp. 675–687.

[10] Dance, C., Willamowski, J., Fan, L. X., Bray, C., and Csurka, G. Visual categorization with bags of keypoints. In *Proceedings of ECCV International Workshop on Statistical Learning in Computer Vision* (2004), pp. 1–22.

[11] Dash, M., and Liu, H. Feature selection for clustering. In *4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications* (2000), 110-121, Ed., Springer-Verlag.

[12] Dempster, A., Laird, N., and Rubin, D. Likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B 39* (1977), 1–38.

[13] Devaney, M., and Ram, A. Efficient feature selection in conceptual clustering. In *In Proceedings of the Fourteenth International Conference on Machine Learning* (1997), Morgan Kaufmann, pp. 92–97.

[14] Dietterich, T. G. Ensemble methods in machine learning. In *SpringerVerlag* (2000), pp. 1–15.

[15] Dietterich, T. G., Lathrop, R. H., and Pérez, T. L. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence 1–2*, 89 (1997), 31–71.

[16] Dy, J. G., and Brodley, C. E. Feature subset selection and order identification for unsupervised learning. In *In Proc. 17th International Conf. on Machine Learning* (2000), pp. 247–254.

[17] Edelman, S., and Bulthoff, H. H. Modeling human visual object recognition. In *Neural Networks, 1992. IJCNN., International Joint Conference on* (1992), vol. 4, pp. 37–42 vol.4.

[18] Fischler, M. A., and Elschlager, R. A. The representation and matching of pictorial structures. *IEEE Transactions on Computers C-22*, 1 (1973), 67–92.

[19] Fisher, D. Knowledge acquisition via incremental conceptual clustering. *Machine Learning 2* (1987), 139–172.

[20] Flusser, J., and Suk, T. Pattern recognition by affine moment invariants. *Pattern Recognition 26*, 1 (1993), 167–174.

[21] Freund, Y., and Schapire, R. E. A brief introduction to boosting. In *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* (1999), Morgan Kaufmann, pp. 1401–1406.

[22] Fu, L., and Medico, E. Flame, a novel fuzzy clustering method for the analysis of dna microarray data. *BMC Bioinformatics 8* (2007), 3.

[23] Grauman, K., and Darrel, T. The pyramid match kernel: Discriminative classification with sets of image features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2005), pp. 1458–1465.

[24] Haykin, S. *Neural networks - A comprehensive foundation, 2nd edition.* Prentice-Hall, 1999, ch. 9. Self-organizing maps.

[25] Hiremath, P. S., and Pujari, J. Content based image retrieval using color, texture and shape features. In *Advanced Computing and Communications, 2007. ADCOM 2007. International Conference on* (2007), pp. 780–784.

[26] Hoffman, T. Probabilistic latent semantic analysis. In *In Proceedings of Uncertainty in Artificial Intelligence, UAI 99* (1999), pp. 289–296.

[27] Hu, M. Visual pattern recognition by moment invariants. *IRE Trans. Information Theory IT-8* (1962), 179–187.

[28] Jamil, N., Bakar, Z., and Sembok, T. Image retrieval of songket motifs using simple shape descriptors. In *Geometric Modeling and Imaging - New Trends* (2006), pp. 171–176.

[29] Lanitis, A., Hill, A., Cootes, T. F., and Taylor, C. J. Locating facial features using genetic algorithms. In *Proceedings of the 27th International Conference on Digital Signal Processing* (1995), pp. 520–525.

[30] Leibe, B., Leonardis, A., and Schiele, B. Robust object detection with interleaved categorization and segmentation. *Internation Journal of Computer Vision 77*, 1–3 (2008), 259–289.

[31] Linde, Y., Buzo, A., and Gray, R. An algorithm for vector quantitizer design. In *IEEE Transactions on Communications* (1980), vol. 28, pp. 84–94.

[32] Lloyd, S. Least squares quantization in pcm. In *Information Theory, IEEE Transactions on* (1982), vol. 28, pp. 129–137.

[33] Lopes, L. S., and Chauhan, A. Open-ended category learning for language acquisition. *Connection Science, 8(4), (in press)* (2008).

[34] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60* (2004), 91–110.

[35] Mccallum, A., Nigam, K., and Ungar, L. H. Efficient clustering of high-dimensional data sets with application to reference matching. In *In Knowledge Discovery and Data Mining* (2000), pp. 169–178.

[36] McCulloch, W., and Pitts, W. A logic calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics 5* (1943), 115–137.

[37] Mitra, P., Murthy, C., and Pal, S. K. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2002), 301–312.

[38] Papadimitriou, C. H. *Combinatorial optimization.* Prentice-Hall, New Jersey, 1982. Christos H. Papadimitriou, Kenneth Steiglitz 496 p.

[39] Pearson, K. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine 2 6* (1901), 559–572.

[40] Quelhas, P. *Scene Image Classification and Segmentation with Quantized Local Descriptors and Latent Aspect Modeling.* PhD thesis, École Polytechnique Fédérale de Lausanne, 2007.

[41] Ribeiro, L. S. Object recognition for semantic robot vision. Master's thesis, Universidade de Aveiro, 2008.

[42] Roth, V., and Lange, T. Feature selection in clustering problems. In *In Advances in Neural Information Processings Systems 16* (2004).

[43] Roy, D. K. *Learning Words from Sights and Sounds: A Computational Model.* PhD thesis, MIT, 2000.

[44] Russell, S. J., and Norvig, P. *Artificial intelligence: A modern approach*, 2nd ed. Prentice Hall series in artificial intelligence. Prentice-Hall, Englewood Cliffs, New Jersey, 2001. Stuart J. Russell, Peter Norvig 25 cm.

[45] Sarfraz, M., and Ridha, A. Content-based image retrieval using multiple shape descriptors. In *Computer Systems and Applications, 2007. AICCSA '07. IEEE/ACS International Conference on* (2007), pp. 730–737.

[46] Serre, T., Wolf, L., and Poggio, T. Object recognition with features inspired by visual cortex. In *In CVPR* (2005), pp. 994–1000.

[47] Søndberg-madsen, N., Thomsen, C., and Peña, J. M. Unsupervised feature subset selection. In *In Proceedings of the Workshop on Probabilistic Graphical Models for Classification* (2003), pp. 71–82.

[48] Steinhaus, H. Sur la division des corp materiels en parties. *Bulletin L'Acadmie Polonaise des Science IV*, C1. III (1956), 801–804.

[49] Talavera, L. An evaluation of filter and wrapper methods for feature selection in categorical clustering. In *IDA : international symposium on intelligent data analysis No6* (2005), vol. 3646, pp. 440–451.

[50] Thompson, D. W. *On Growth and Form*. Cambridge University Press, Cambridge, UK, 1917.

[51] Tong-Zhen, Z., and Yong-Gang, F. An image semantic retrieval system design and realization. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on* (2005), vol. 9, pp. 5284–5289 Vol. 9.

[52] Vijayanarasimhan, S., and Grauman, K. Keywords to visual categories: Multiple-instance learning for weakly supervised object categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2008).

[53] Vogel, J., and Schiele, B. Natural scene retrieval based on a semantic modeling step. In *Proceedings of the International Conference on Image and Video Retrieval* (2004), pp. 207–215.

[54] Wagstaff, K., Cardie, C., Rogers, S., and Schroedl, S. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning* (2001), pp. 577–584.

[55] Wardhani, A., and Thomson, T. Content based image retrieval using category-based indexing. In *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on* (2004), vol. 2, pp. 783–786 Vol.2.

[56] Wardhani, A. W. *Application of psychological principal to automatic object identification for CBIR*. PhD thesis, Griffith University, 2001.

[57] Yuille, A., Cohen, D., and Hallinan, P. Feature extraction from faces using deformable templates. In *IEEE Computer Society Conference on CVPR* (1989), pp. 104–109.

[58] Zhang, D., and Lu, G. Review of shape representation and description techniques. *Pattern Recognition 37*, 1 (2004), 1–19.