FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Ein Segmentation-basiertes Objekt Erkennung System für Bilder

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Paraschos Zeimpekos
Matrikelnummer 0426924

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Dr. Allan Hanbury

Wien, 16. Oktober 2016

Paraschos Zeimpekos	Allan Hanbury

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

A Segmentation-based Object Recognition System for Images

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Media Informatics and Visual Computing

by

Paraschos Zeimpekos
Registration Number 0426924

to the Faculty of Informatics
at the TU Wien

Advisor: Dr. Allan Hanbury

Vienna, 16th October, 2016

Paraschos Zeimpekos	Allan Hanbury

Erklärung zur Verfassung der Arbeit

Paraschos Zeimpekos
Matrikelnummer

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 16. Oktober 2016

Paraschos Zeimpekos

Danksagung

Ich möchte meine geliebte Rina bedanken, meine liebende Lebensparterin, für das Dasein, für alle schönen Zeiten und die besseren noch zu kommen.
Ich wollte auch meine Eltern danken, für alles, die mir gegeben haben, für ihre Liebe, Vertrauen und Verstehen.
Letztens, ich möchte meinen Betreuer Dr Allan Hanbury herzlich danken, für alle wertvollen Empfehlungen und die Geduld, die er gezeigt hat.

Paris Zeimpekos

Acknowledgements

I would like to thank my beloved Rina, my loving companion in life, for being there, for all the good times and the far more better to come.
I would also like to thank my parents for all they have given me, all their love, trust and understanding.
Finally, many thanks go out to my supervisor Dr Allan Hanbury, for all the good advice he has given me and the patience he has shown.

Paris Zeimpekos

Kurzfassung

Das Ziel dieser Masterarbeit ist die Anerkennung und Klassifizierung von Objektklassen in Bildern mit der Hilfe vom Bag of Keypoints Algorithmus und Segmentierung Verfahren. Im Einzelnen, zuerst wir übersegmentieren die vorhandenen Bilder, extrahieren von jeder segmentierten Region Farbe Information und gruppieren diese Information mit der Hilfe vom kmeans Verfahren. Als Segmentierungsverfahren benutzen wir die volumic extraction und Wasserfall Verfahren, welche auf der Minimum Spanning Tree Methode basiert sind. Für die Extraktion der Farbeinformation benutzen wir zwei verschiedene Farbräume, die Opponent und HSI Farbräume. Das Gruppierung Ergebnis ist unsere Merkmale - Wortschatz für das Bag of Keypoints. Der nächste Schritt ist Histogramme zu erstellen, die von der Auftretennummer vom jeden Keypoint nach der Gruppierung der Bilder gebildet sind. Schließlich, wir benutzen die Histogramme, um Support Vector Machines tu trainieren, deren Nummer gleich mit der Nummer der Klassen ist, und mit denen unbekannte Objekte in Bildern zu erkennen und klassifizieren. Für das Training und Testen von den Klassifikatoren benutzen wir die PASCAL Datenbank 2006. Für unser Auswertungverfahren wir benutzen die Receiver Operating characteristic, auch bekannt als ROC Kurve. Unsere Ergebnisse sind ausreichend und vergleichbar mit den offiziellen Ergebnissen von der 2006 VOC Challenge.

Abstract

The aim of this master thesis is the recognition and classification of object classes in images using the bag of keypoints algorithm and the segmentation approach. In more detail, we first over-segment the available images, extract from every segmented region color information and cluster this information using the known kmeans approach. As segmentation method we use the volumic extinction and waterfall approaches, which are both based on the Minimum Spanning Tree method. For the color information extraction we use two different color spaces, the opponent and HSI color spaces. The clustering result is our features - vocabulary for the bag of keypoints. The next step is to build histograms formed by the number of occurrences of every keypoint from the clustering in the images. Finally, we use the histograms to train Support Vector Machines equal to the number of the available classes and use them for the recognition and classification of unknown objects in images. For the training and testing of the classifiers we use the PASCAL database 2006. As our evaluation technique we use the Receiver Operating characteristic, also known as ROC curve. Our results are adequate and are comparable with the official Results from the 2006 VOC Challenge.

Contents

CHAPTER 1

Introduction

1.1 Movitation

Over the last decade, there has been a tremendous increase in the use of computers, into almost every job and human activity possible. An extensive use of this new technology is, in some cases, even prompted by advertisements, or has become the new trend. This tremendous technological increase in everyday use is also causing an ever growing number of digital photographs in private or professional collections through the extensive use of equipment such as digital cameras and mobile device cameras. In order to handle and manage such collections high-level information about the content of the images could be useful. Dealing effectively with this would lead to better applications for many different scientific fields such as medical image analysis, human computer interaction and artificial intelligence.

The problem described above is the so called generic visual categorization problem. Visual categorization refers to the process of identifying whether objects of one or more types are present in an image and the capability to sufficiently cope with many object types simultaneously and to extend to new object types, if necessary. The generic attribute refers to the need the above process to be able to handle the variations in view, lighting, imaging, occlusion, which are typical of the real world, as well as the intra-class variations typical of semantic classes of everyday objects.

This thesis presents an object recognition system for images. Further in this chapter we describe Object Recognition, the key element for our work, in section 1.3 we present our contribution and in section 1.4 we give an overview of the thesis.

1.2 Object Recognition

Object recognition is a major task in computer vision. It tries to automatically recognize the identity of any visual object (s) present in a given image. This task can be seen as

the equivalent for the research being made in order to understand the human cognitive system for artificial intelligence systems. Humans possess the ability to immediately recognize a multitude of objects in images with little effort, despite the fact that the image of the objects may vary somewhat in different view points, in many different sizes / scale or even when they are translated or rotated. Objects can even be recognized when they are partially obstructed from view. Computers unfortunately lack this ability. Over the years much research has been done in different fields such as computer vision and image processing, artificial intelligence and neural networks in order to "computerize" this ability, with many interesting and beneficial results, but, in general, this task remains still a challenge for computer vision systems.

Early systems [TP91, Mor80, BMP02, BJ85] focused more on modelling each individual object and recognizing specific instances of that object, taken from different angles and views. All available data for one object was stored in a database as templates for that particular object. For any query image, a comparison is made between this image and every template stored. Any possible correspondence is calculated and the best matching template is found. These methods may produce good results under certain circumstances, but they are computation time intensive and require massive storage capacity for the database. Though it is possible to have a database that has examples of different views of an object under different lighting conditions, this task is very difficult when the number of objects increases.

Object recognition in real-world scenes, where different objects are cluttered together, and in many cases partially occluded requires the use of local image features, and object detection and recognition. By features we mean some interesting points on the object that can be extracted to provide a "feature" description of it. This description extracted from a training image can then be used to identify the object when attempting to locate the object in a test image containing many other objects. It is important that the set of features extracted from the training image is robust to changes in image scale, noise, illumination and local geometric distortion, for performing reliable recognition. On the other hand, the features must also be sufficiently distinctive in order to identify any specific object among all other possible objects present in the test image. Both tasks are a major research field for computer vision, as the best feature detector and descriptor has yet to be found, although in the last years there have been efforts with promising results [Low99, MS04, MCUP02, Low04].

Object recognition has many different usable applications which make its further research and development plausible. Robotics would benefit strongly from it, as robotic movement and interaction with objects would be much easier, even in real time. Content based image retrieval (CBIR) is a similar but more general application to object recognition, where relevant images are retrieved from a collection based on an image query. CBIR would obviously benefit from a better recognition. In general, we could say that, as object recognition is a major part of computer vision, many of its applications would benefit from further object recognition development.

1.3 Our Contribution

In this thesis we present a segmentation based object recognition system for images. Firstly, we perform an over-segmentation on every image. Then, instead of going with the standard interest point detection and description approach, we base our system only on color information extracted from every region of the over-segmented images. From this information we build our features. The next step is to use the bag of keypoints algorithm and build histograms based on the aforementioned features. These histograms are used in order to train our classifiers for the objects present in the images.

1.4 Overview of the thesis

The structure of this thesis is as follows:

In Chapter 2 we give an overview of crucial theoretical background and related work with the matters discussed in this thesis. In section 2.2 the main idea of the bag of keypoints algorithm is given and in section 2.3 the algorithm is described in detail. In section 2.4 interest points are introduced and their goals and properties are given. Point and region detectors for interest points are described in section 2.5 and local descriptors are introduced in section 2.6. As color spaces are crucial for our approach, they are discussed in section 2.7.

Another important issue of our approach is the image segmentation, which is thoroughly discussed in Chapter 3. Its definition along with some examples is given in section 3.2. In section 3.3 the various segmentation methods are described, such as thresholding, edge- and region-based segmentation. The watershed method combines elements of both edge- and region-based methods and is described in detail in section 3.4.

In Chapter 4 we introduce our object recognition system for images. Our approach makes use of the bag of keypoints algorithm. Its main idea is discussed in section 4.2. The main steps of the algorithm are the feature extraction, the visual vocabulary construction and the categorization, which are described in detail in section 4.3. In section 4.4 we present our object recognition system. Just as every object recognition method, so ours can be divided in two main phases, training and testing. At the same time we present how the aforementioned steps of the bag of keypoints adapt to our method, e.g. over-segmentation, feature extraction, clustering and classifier training.

Chapter 5 presents the experiments carried out with our system and the corresponding results. In section 5.2 we introduce the data base used for the training and the testing of the images. In section 5.3 we describe our small probe-experiment on simple figures produced by us and the reasons that led us to this decision. Finally, in section 5.4 the different experiments on the data base images are explained and the distinctive results are presented, along with our evaluation technique.

Chapter 6 summarizes the results produced by our prototype. Some ideas for future work and possible applications are also given.

CHAPTER 2

Theoretical Background

2.1 Introduction

In this chapter we give an overview of significant theoretical background regarding topics that are discussed in the thesis. Firstly, we describe the bag of keypoints algorithm, which we use in this thesis. In section 2.2 we present the idea behind the algorithm and in section 2.3 we describe the algorithm in detail. In section 2.4 we give a short definition of interest points and describe their goals and properties. Interest points are usually detected in an image with the aid of the point and region detectors, which are described in section 2.5. After the process of detection comes the description of the detected interest points. This is the job of the local descriptors, which are presented in section 2.6. Finally, in section 2.7 color spaces in general are briefly discussed and the ones used in this thesis are described.

2.2 Idea behind bag of keypoints

A bag of keypoints corresponds to a histogram of the number of occurrences of particular image patterns in a given image [CDF+04]. The main idea for this approach comes from text categorization [Joa98, TK00, LSTCW01]. There is an analogy between this method and the learning methods using the bag of words representation for the generic visual categorization problem. The thought of adapting text categorization approaches to visual categorization is not new. There has been significant research on this topic, in particular the vector quantization of small square windows, which were called keyblocks [ZRZ02]. It was shown that these features produced more semantics-oriented results than color and texture based approaches, when combined with analogues of the well known vector-, histogram-, and n-gram-models of text retrieval [CDF+04]. The idea of clustering invariant descriptors of image patches has previously also been used for the problem of texture classification [Th99, VZ02, LSP03]. As the problems themselves differ

in their nature, the same technique may be used (e.g. clustering) in a different way for each case with different results.

2.3 The algorithm

The four steps of the bag of keypoints algorithm are:

- Detection and description of image patches
- Assigning patch descriptors to a set of predetermined clusters (a vocabulary) with a vector quantization algorithm
- Constructing a bag of keypoints, which counts the number of patches assigned to each cluster
- Applying a multi-class classifier, treating the bag of keypoints as the feature vector, and thus determine which category or categories to assign to the image.

One should notice that these steps are designed, in the ideal case, to maximize classification accuracy while minimizing computational effort. This translates to extracting descriptors invariant to possible image variations irrelevant to the categorization task, such as lighting and occlusion issues and image transformations, but also rich enough to be able to sufficiently describe and distinguish each category. The vocabulary mentioned in the second step should be large enough to distinguish relevant changes in image parts, but not so large as to distinguish irrelevant variations such as noise [CDF+04].

The concept of using a set of vocabularies is also motivated from text categorization. The quantized feature vectors (cluster centers) are therefore referred to as "keypoints" by analogy with "keywords" in text categorization. For text it is self evident for the key words to have a clear meaning, such as "human" or "dog". For images that is not the case, as keypoints don't necessarily have repeatable meanings, and there doesn't exist an obvious best choice of vocabulary. The aim is instead a rich vocabulary that allows good categorization performance on a given training set. This results in taking the possibility of multiple vocabularies into consideration. Having all these in mind, we conclude to the following steps for the training part:

- Detection and description of image patches for a set of labeled training images
- Constructing a set of vocabularies: each is a set of cluster centers, with respect to which descriptors are vector quantized
- Extracting bags of keypoints for these vocabularies
- Training multi-class classifiers using the bags of keypoints as feature vectors
- Selecting the vocabulary and classifier giving the best overall classification accuracy.

We present and discuss in more detail the above mentioned steps in the following sections.

2.3.1 Feature Extraction

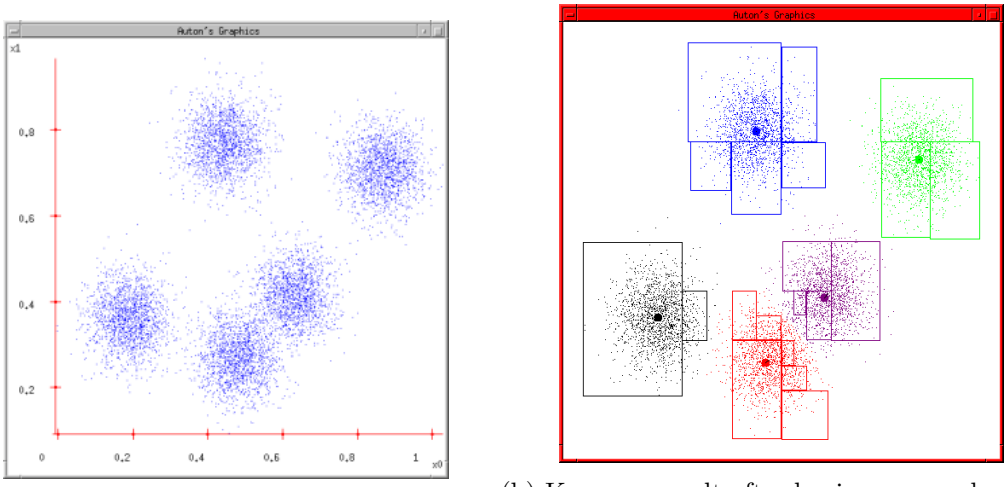
The selection of features is crucial for every application. Although crucial, this task is not easy, as the chosen features should have certain characteristics, and depends also strongly on each application and on the images used for training. There has been much research done [MS02, Low99, MS04] over the last years for this matter in image understanding and local descriptors have proved suitable for recognition and matching problems, as they are robust to background clutter and partial visibility. Moreover, such problems require repeatable descriptors. By repeatable we mean that if there is a transformation between two instances of an object in an image, corresponding points are detected and identical descriptor values are obtained around each [CDF+04]. From the above it is obvious that also the bag of keypoints algorithm requires an adequate feature selection and detection.

2.3.2 Visual vocabulary construction

For the algorithm, the vocabulary is a way of constructing a feature vector for classification that relates "new" descriptors in query images to descriptors previously seen in training [CDF+04]. One ideal method for this would be to compare each query descriptor to every training descriptor. But, given the great number of descriptors involved, this method is rather impractical, as it requires much computational time and space. Another impractical method would be to try to identify a small number of large clusters that are able to distinguish a given class correctly, for the same reasons as above. In practice we find that the best tradeoffs between accuracy and efficiency are gained by using for clustering intermediate number of descriptors.

Most clustering or vector quantization algorithms are based either on iterative square-error partitioning or on hierarchical techniques. Square-error partitioning methods try to find the partition that minimizes the within-cluster scatter or maximizes the between-cluster scatter [CDF+04]. Hierarchical techniques organize the available data in a nested sequence of groups which can be displayed in the form of a tree or a graph. Their main drawback is that they need some heuristics to form clusters and therefore are not used so often as square-error partitioning algorithms in pattern recognition.

The bag of keypoints approach uses the simplest and most common used square-error partitioning technique: k-means [PM00a, PM00b]. The main idea of this algorithm is an attempt to find the centers of natural clusters in the data. The most common form of the algorithm uses an iterative refinement heuristic known as Lloyd's [Llo82] algorithm. Lloyd's algorithm starts by partitioning the input points into k initial sets, either at random or using some heuristic data. It then calculates the mean point, or centroid, of each set. It constructs a new partition by associating each point with the closest centroid. Then the centroids are recalculated for the new clusters, and algorithm repeated by alternate application of these two steps until convergence, which is obtained when the points no longer switch clusters (or alternatively centroids are no longer changed). Other variations exist, but Lloyd's algorithm has remained popular because it converges extremely quickly in practice [DEJ06, SG86]. In terms of performance the algorithm is not guaranteed to return a global optimum. The quality of the final solution depends



(a) Initial data.

(b) K-means result after having a user choosing k = 5.

Figure 2.1: K-means example: From: [PM00a].

largely on the initial set of clusters, and may, in practice, be much poorer than the global optimum. Since the algorithm is extremely fast, a common method is to run the algorithm several times and return the best clustering found. Another main drawback of the algorithm is that it has to be told the number of clusters (i.e. k) to find. Not naturally clustered data may lead to inaccurate results. To overcome this, the algorithm is tested several times with different initializations, e.g. different values for k and different sets of initial cluster centers. The clustering with the lowest empirical risk in categorization is selected, as proposed in [Vap98]. In Figure 2.1 we can see an example of the algorithm. Initially we have some unclustered data, as seen on the left picture. After choosing k = 5 and randomly guessing the k cluster locations, we can see on the right the result of the algorithm, having successfully separated the data into 5 different clusters. This example can be seen in detail and with each step description in [PM00a].

2.3.3 Categorization

Once descriptors have been assigned to clusters to form feature vectors, the problem of generic visual categorization is reduced to that of multi-class supervised learning, with as many classes as defined visual categories [CDF+04]. The system goes through two different modes in order to classify unknown images: training and testing. During training, labeled data is used as input to the system in order to adapt a statistical decision procedure for discriminating categories. We choose among many available classifiers the Support Vector Machine (SVM) [Cri01, Bur98] for its efficiency in high-dimensional problems.

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. They belong to a family of generalized linear classifiers.

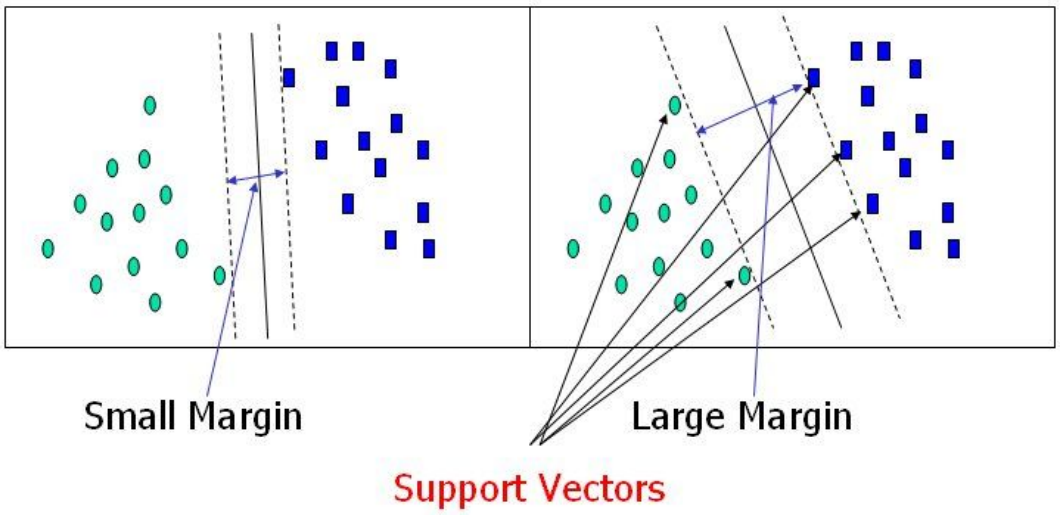


Figure 2.2: Support Vector Illustration: On the left we can a small margin separating the two data sets, and on the right we see the optimum (large) margin separating the sets and the support vectors. From [1].

The SVM classifier finds a hyperplane that separates two-class data with maximal margin. The margin is defined as the distance of the closest training point(s) to the separating hyperplane [CDF+04]. A special property of SVMs is that they simultaneously minimize the empirical classification error and maximize the geometric margin; hence they are also known as maximum margin classifiers. This is visible in Figure 2.2.

A SVM takes as input two vectors, one that contains the – in the bag of keypoints case clustered – data and another with the corresponding labels. For the training data vector $\mathbf{X}$ and corresponding label vector $\mathbf{Y}$, which takes values of $\pm$ 1, we can easily build a classification function *f*,

$$f(x) = sign(w^T \cdot x + b) \tag{2.1}$$

where w and b are the parameters of the hyperplane. For those points $\mathbf{x}$ which lie on the hyperplane satisfy $w \cdot \mathrm{x} + \mathrm{b} = 0$, where $\mathbf{w}$ is normal to the hyperplane, $|\mathrm{b}|/||\mathbf{w}||$ is the perpendicular distance from the hyperplane to the origin, and $||\mathbf{w}||$ is the Euclidean norm of $\mathbf{w}$. Let d_+ (d_-) be the shortest distance from the separating hyperplane to the closest positive (negative) example. Define the "margin" of a separating hyperplane to be $d_+ + d_-$. For the linearly separable case, the support vector algorithm simply looks for the separating hyperplane with largest margin [Bur98]. This can be formulated as follows: suppose that all the training data satisfy the following constraints:

[1] http://www.dtreg.com/svm.htm

$$x_i \cdot w + b \geq +1 \quad \text{for} \quad y_i = +1 \tag{2.2}$$
$$x_i \cdot w + b \leq +1 \quad \text{for} \quad y_i = -1 \tag{2.3}$$

These can be combined into one set of inequalities:

$$y_i \cdot (x_i \cdot w + b) - 1 \geq 0 \quad \forall i \tag{2.4}$$

Now consider the points for which the equality in Eq. (2.2) holds (requiring that there exists such a point is equivalent to choosing a scale for **w** and b). These points lie on the hyperplane H_1: $x_i \cdot$w + b = 1 with normal **w** and perpendicular distance from the origin |1 - b|/||**w**||. Similarly, the points for which the equality in Eq. (2.3) holds lie on the hyperplane H_2: $x_i \cdot$w + b = -1, with again normal w and perpendicular distance from the origin |-1 - b|/||**w**||. Hence $d_+ = d_- = 1/||\mathbf{w}||$ and the margin is simply $2/||\mathbf{w}||$. Note that H_1 and H_2 are parallel (they have the same normal) and that no training points fall between them. Thus we can find the pair of hyperplanes which gives the maximum margin by minimizing $||\mathbf{w}||^2$, subject to constraints [Bur98].

Data sets are in the most cases, not linear separable. The above described method can be generalised in order to solve the problem of non linearity by a straightforward way. Firstly we notice that the only way in which the data appears in the training problem, in the above equations, is in the form of dot products. We can map the data from the original data space to some other (probably infinite dimensional) Euclidean space H', using a mapping called Φ. In this space the data would be linear separable. Then the training algorithm would only depend on the data through dot products in H', i.e. on functions of the form $\Phi(x_i) \cdot \Phi(x_j)$. But in this new space there exists a computational problem because we are working with very large vectors and also a generalization theory problem (curse of dimensionality). We can now introduce the so called "kernel function" K, such that K $(x_i \cdot x_j) = \Phi(x_i) \cdot \Phi(x_j)$. In this way we would only need to use K in the training algorithm, and would never need to explicitly even know what Φ is. Furthermore, if one replaces $x_i \cdot x_j$ with K $(x_i \cdot x_j)$ everywhere in the training algorithm, it will produce a support vector machine which lies in an infinite dimensional space, and furthermore the computation would take roughly the same amount of time it would take to train on the un-mapped data [Bur98]. This can be seen in Figure 2.3.

In the kernel formulation, the decision function f can be expressed as:

$$f(x) = sign(\sum_i y_i a_i K(x, x_i) + b) \tag{2.5}$$

Here $\mathbf{x_i}$ are the training features from the first data space and $\mathbf{y_i}$ is the label of $\mathbf{x_i}$. It is obvious that we avoid computing $\Phi(\mathbf{x})$ explicitly and use the kernel mode. The choice of kernel is problem dependent and always user determined. There exist many possible kernels, some simple and some more complex. Some examples are:

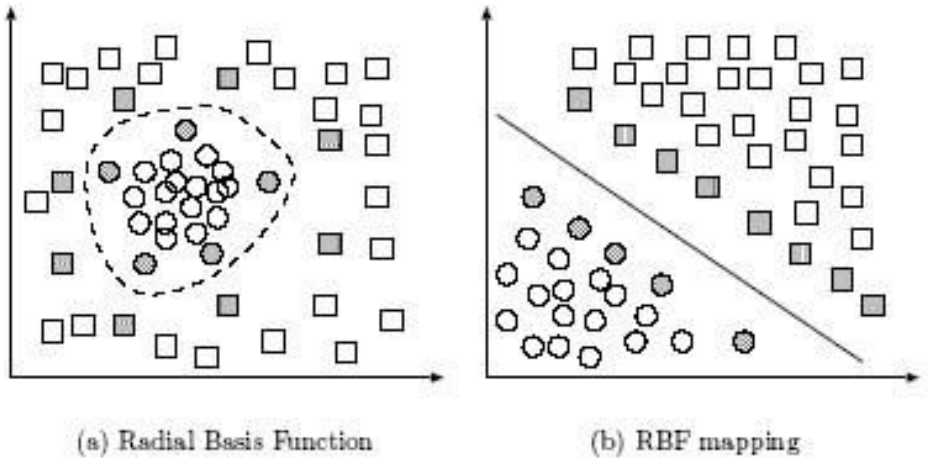


Figure 2.3: Kernel mapping example: (a) original space, not linear separable (b) feature space after mapping using the Radial Basis Function, now linear separable. From [2].

$$K(x, y) = (x, y) \tag{2.6}$$
$$K(x, y) = (x, y)^d \tag{2.7}$$
$$K(x, y) = (x \cdot y + 1)^d \tag{2.8}$$
$$K(x, y) = \mathrm{e}^{-|x-y|^2/2\sigma} \tag{2.9}$$

Finally, the $\mathbf{a_i}$ parameters in equation (2.5) are typically zero for most i. Therefore, the sum can be taken only over a select few of the $\mathbf{x_i}$. These feature vectors are known as support vectors. It can be shown that the support vectors are those feature vectors lying nearest to the separating hyperplane. In order to find the maximal margin these parameters, e.g. the weights, need to be updated until the margin is found. The use of the stochastic Gradient Ascent (sequentially update 1 weight at the time) gives excellent approximation in most cases [Cri01]. The formula is shown below:

$$a_i \leftarrow a_i + \frac{1}{K(x_i, x_i)}(1 - y_i \sum a_i y_i K(x_i, x_i)) \tag{2.10}$$

In the case of visual categorization using the bag of keypoints, the input vectors $\mathbf{x_i}$ are the binned histograms formed by the number of occurrences of each keypoint $\mathbf{v_i}$ from the vocabulary $\mathbf{V}$ in the image $\mathbf{I_i}$. Moreover, in order to apply the SVM to multiclass problems, as in our case, we take the one-against-all approach. Given an m-class problem,

[2] http://www.dtreg.com/svm.htm

we train m SVM's, each distinguishes images from some category i from images from all the other *m-1* categories j not equal to i. Given a query image, we assign it to the class with the largest SVM output [CDF+04].

2.4 Interest points

There appears to be a common strategy in many computer vision approaches [Low99, MS04, Low04, MS01, CWB00], which is the distinction of specific points or regions in the image. Any further action, such as matching, is based upon the processing of these image patches. Locations which have proved to be particularly appropriate are the so called interest points. Although we take another approach in our thesis, we feel that we must also provide a small overview of such an important topic.

There isn't any precise definition of an interest point, but there are some general characteristics that such a point in an image should follow. Firstly, it must possess a clear mathematical definition and its position in the image space should be certain and well defined. Moreover, the local image structure around the point must be rich in terms of local information contents for further use in the application. Finally, an interest point should be stable under certain image transformations and brightness variations, which are described in more detail in the next section. Historically, the concept of interest points derives from the earlier concept of corner detection, as firstly described by Moravec [Mor80]. His method, although later improved, is still the basis for many corner detectors [HS88, MS04]. Hence, we will refer from this point on to extracted corners as interest points.

An important concept that is often met with interest points is that of invariance. An invariant interest point is a point of an image, which can be detected and extracted after various transformations of the specific image. This characteristic is crucial, because in many applications we stumble upon different views of the same image, which need to be compared. There exist many forms of invariance, which are briefly discussed below:

- Spatial invariance: Spatial invariance is the ability of a local interest point detector to detect the same point before and after a translation of an image, or an object in an image [Que07]. An example of a translated image is shown in Figure 2.4.
- Scale invariance: Scale invariance is the ability to track down the same interest point and local area of interest after camera zooming or image resizing [Wit83, Lin94]. An example of a scaled image is shown in Figure 2.5.
- Affine invariance: Affine invariance is a generalization of the scale invariance when the changes in scale are not isotropic [Que07], as in the case when the camera viewing angle changes. Such an image is shown in Figure 2.6.
- Rotation invariance: To obtain a rotation invariant representation we either need an orientation invariant local descriptor or we need to compute a consistent orientation to the local interest area, which remains invariant with respect to a rotation of the local interest area's image content [Que07, Low99]. In Figure 2.7 a rotated image is shown.

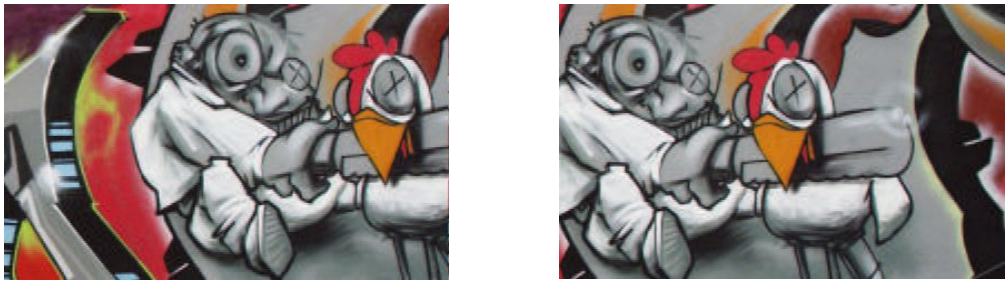


Figure 2.4: Example of a translated image. From: [Que07].

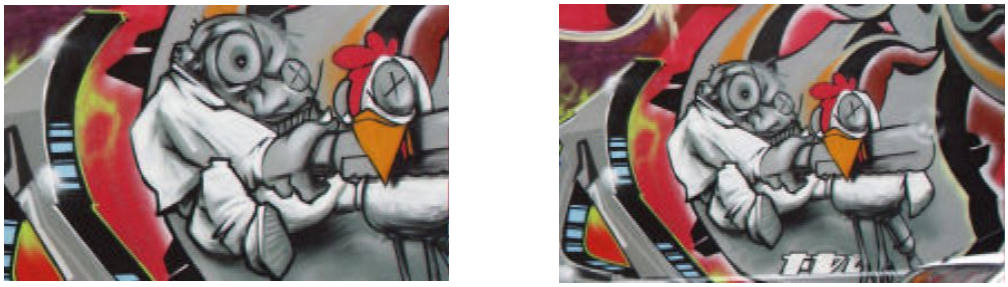


Figure 2.5: Example of a scaled image. From: [Que07].

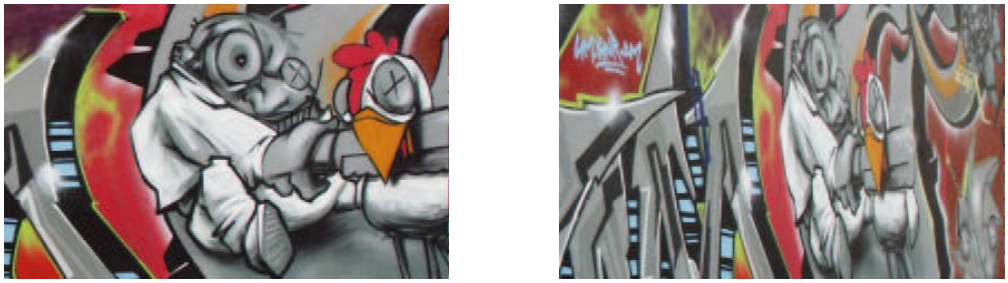


Figure 2.6: Example of an image taken from different angles. From: [Que07].

- There are two major aspects with regard to illumination invariance: the invariance of the interest point location, and the invariance of the local interest area content.

2.5 Local Interest Points Detectors

Local interest point detectors are designed to find points in the image that contain distinctive information in their neighboring area and whose extraction is stable with respect to various transformations, as described in the above section, and noise. In the following sections the most commonly used detectors at present are described.

Figure 2.7: Example of a rotated image. From: [Que07].

2.5.1 Harris Corner Detector

The first approach for interest point detection is through corner detection. As corner we mean the intersection of two edges. Corners have proven to be very interesting interest points and these detected points are more stable than edges and have many valuable properties. As they are used in a variety of applications, such as object recognition and stereo matching, they are very popular.

The first corner detector was developed by Moravec in 1977 [Mor80] for his research involving the navigation of a robotic vehicle through a clustered environment. It was the basis for the further development of many later detectors, such as the Harris corner detector.

Harris and Stephens [HS88] improved upon Moravec's corner detector by taking into account the derivatives at a certain point of the image, instead of using shifted patches. This approach for the detection of local interest points remains the basis for many different improvements that have been proposed.

The original formulation of the Harris corner is based on the second moment matrix M, which is computed from the derivatives of the image at each point and a Gaussian kernel. The exact formulation can be found in [HS88]. The symmetric matrix M is the key for any further calculation. We can gain valuable information by observing its eigenvalues. According to their values at a specific point, we can tell if there exists a corner, an edge or nothing of interest at this point. However, this can be very time consuming and therefore impractical. As an alternative Harris and Stevens proposed in [HS88] a different measure of corner response R, which uses the trace and the determinant of the second moment matrix. In Figure 2.8 a corner detection example using the Harris detector is shown.

The main drawback of the Harris detector is that it is neither scale nor affine invariant. In [MS04] an extension of the standard Harris detector is proposed in order to provide scale invariance. In order to achieve affine invariance, the second moment matrix M can also be used. Instead of the normal Gaussian kernel, Mikolajczyk and Schmid proposed in [MS02] the shape adapted Gaussian kernel. By calculating M in this way, we can obtain affine image patches, and, additionally, the matrix can be generalized to affine scale space, providing even better results, with the use of specific properties as shown in [Lin94].

(a) Original image

(b) The resulting interest points (corners) over imposed on the greyscale of the original image

Figure 2.8: Harris corner detector example. Original image from: [EGW+10]

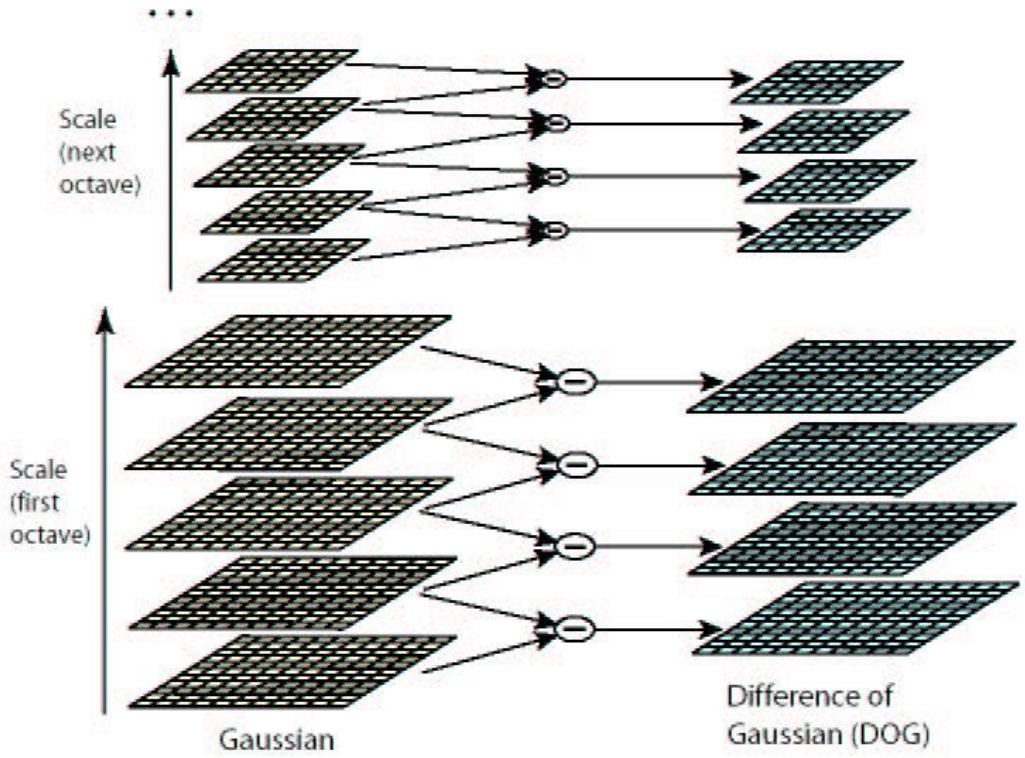


Figure 2.9: Illustration of the DoG scale space computation. The scale space is divided into octaves for which the image size stays the same. From: [Que07].

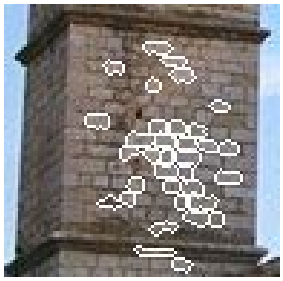
2.5.2 Difference of Gaussians Detector (DoG)

Another detector, which is scale, illumination and orientation invariant, is the Difference of Gaussians (DoG) detector proposed by Lowe. The main idea is to select key locations at maxima and minima of the DoG function applied in scale space [Low99]. This can be done easily be convolving the operator with the image with different smoothing factors and then subtracting them. By continuing this process, we end up having a pyramid like hierarchy, whose new level comes from re-sampling the former one. An illustration of this is shown in Figure 2.9. The resulting key points belong to regions and scales of high variation, making these locations particularly stable for characterizing the image [Que07].

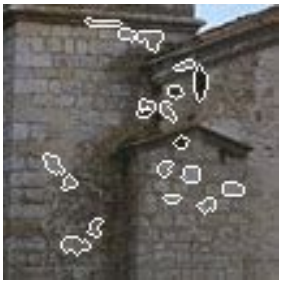
2.5.3 Maximally Stable Extremal Regions (MSER)

Another scale and affine invariant local interest area detector is the one introduced by Matas in [MCUP02], the Maximally Stable Extremal Regions (MSER). Instead of finding the area of interest from certain points in the image, this method detects an area of interest directly. To do so it uses thresholding on grey scale image intensities. Every pixel above every available threshold forms new binary regions. Those regions whose

(a) Original image



(b) Resulting MSER regions



(c) Resulting MSER regions

Figure 2.10: MSER region detector example. From: [MCUP02]

shape remains stable at a range of thresholds are of interest because they are invariant to affine transformations of the image intensity, as pointed out in [MCUP02]. An example of the regions detected using the MSER approach is shown in Figure 2.10.

2.6 Local Descriptors

Local descriptors provide an adequate description for any detected area of interest present in an image. The result is in most cases a feature vector, where information regarding the description is stored. The features should be invariant to changes in the image and resistant to noise, but at the same time highly distinctive [Que07]. The problem that makes this characteristic difficult is that distinctiveness and invariance are two contradicting concepts, so often a trade-off is needed.

2.6.1 Patch Sampling

The most simple and least time consuming local descriptor is based on the idea of patch sampling. It uses for the description of a certain area the luminance values of this area. These values are stored in a feature vector, whose size depends on the actual size of the

area itself. Though simple it this descriptor may be, it possesses none of the desired invariance abilities. There are some cases [LP05] that this simplicity may be taken as advantage, but in general it is a poor descriptor. There exists some ways to produce more stable results against noise and illumination changes, but they are not used in practice, as the distinctiveness of the descriptor becomes much lower [Stő07].

2.6.2 Derivative Description

Another simple and straightforward idea for describing an area of interest is to use different orders of the derivatives at a certain point of the image and take the resulting values as our feature vector. The Gaussian kernel can once again be used for the calculation. In addition, we can take advantage of different abilities that each order possesses, as prompted in [Lin94], in order to make the descriptor stable and invariant. An effective description method was proposed by Koenderink and van Doorn in [KvD87]. Baumberg (and later Schaffalitzky and Zisserman [SZ02]) proposed in [Bau00] the use of a family of Gaussian filters which can be combined in the computation to grant orientation invariance.

2.6.3 SIFT

The Scale Invariant Feature Transform (SIFT) was introduced by Lowe [Low04] in 2004. As the name implies, the features are invariant to image scale and rotation, and are shown to provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination [Low04]. Moreover, the features possess the desired distinctiveness. Although it has some minor frailties in the extraction of the salient points [Stő07, MS05], it remains up today the state of the art local descriptor.

The basic information for the descriptor is gained by using the scale space with the Gaussian function [Lin94]. Possible locations for description are scale space extrema, which are calculated from the difference of Gaussians functions applied to a series of smoothed and resampled images. Then each candidate location is analyzed and compared to each other. Many of them are discarded due to scale space comparisons. Finally, a more stable location is provided by the Taylor expansion of the scale-space function and edge responses are eliminated by using a Hessian matrix.

In order to produce more stable results concerning orientation, the estimated position and the scale of the keypoint is used to select the Gaussian smoothed image with the closest scale, in order to keep all the calculations scale invariant [Low04]. Then for each image the magnitude and orientations are computed using pixel differences. From these orientations a corresponding histogram is built and analyzed, in order to detect the dominant orientations.

The gradient magnitude and orientation at each key point are again used in order to form the feature vector. The resulting values are then weighted with a Gaussian window [Har78] and orientation histograms are built with all the available samples. The descriptor is formed from the above vector containing the values of all the orientation

histograms. After many experiments in [Low04] the best results are taken with a 4x4 array of histograms with 8 orientation bins in each, which leads to a 128 element feature vector for each keypoint. Finally, illumination invariance is gained through normalization and thresholding of the feature vector. An example of the SIFT detector is shown in Figure 2.11.

2.6.4 PCA-SIFT

The Principal Component Analysis (PCA) is a standard technique in image processing for many applications [SK87, SW99]. An interesting approach was introduced by Ke and Sukthankar [KS04] as an alternative to the classical SIFT method. PCA-SIFT uses the Principal Component Analysis to the normalized gradient patch, instead of smoothed weighted histograms [KS04], in order to describe the area of interest. It was experimentally found in [KS04] that 20 dimensions perform well, so this method is significantly more compact, and, in the task of wide-baseline matching [Que07], produces better results than classical SIFT.

2.6.5 SURF

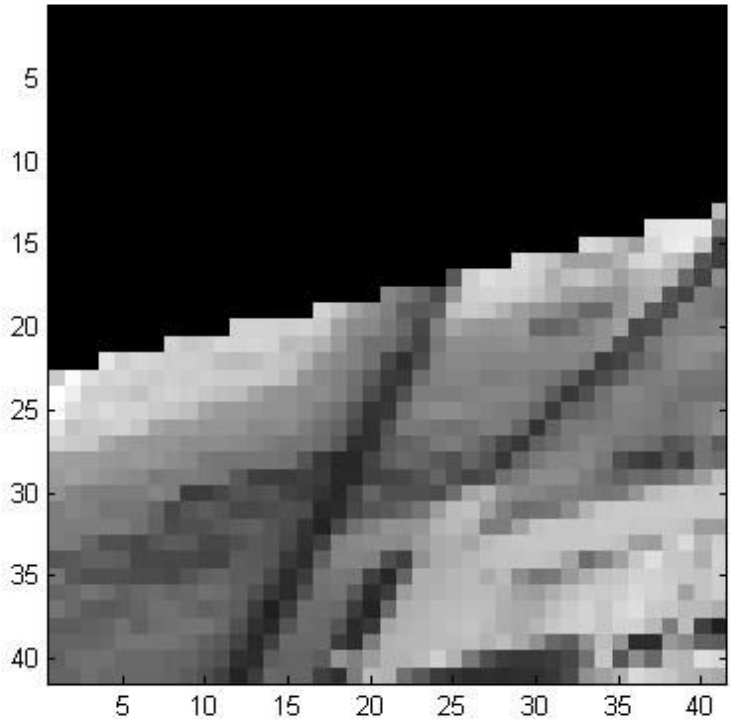
The Speeded up Robust Features (SURF) [BTvG06] is a novel scale and rotation invariant detector and descriptor. It was initially inspired by the SIFT descriptor and its success. The detector is based on the Hessian matrix, using only a very basic approximation which relies on integral images in order to reduce the computation time. The descriptor is based on the distribution of Haar-wavelet responses within the local area of interest. The approach manages to simplify every process, but also to perform strongly, to reduce the time for feature computation and matching and increase simultaneously the robustness.

2.7 Color Spaces

For many years in the field of computer vision color has been seen as superfluous when trying to interpret images, as more attention has been given to luminance [SW06, FSC98] alone. But research [J.06, J.04, SGDvdW06, Stő07] in recent years has shown that it is equally important to other image characteristics and significant information can be extracted also from the colors present in a digital image. There are two major advantages of using color vision. Firstly, color provides extra information which allows the distinction between various physical causes for color variations in the world. Secondly, color is an important discriminative property of objects, allowing us to distinguish between them [J.04].

We can define color as the way the human visual system (HVS) measures a part of the electromagnetic spectrum. Due to certain characteristics (small number of rods) of the HVS described in [BS02] we are not able to see all the possible combinations of the visible spectrum, but we tend to group various spectra into colors. A color space is a notation by which we can specify colors, i.e. the human perception of the visible electromagnetic spectrum [TT03]. For the representation of colors in computers color

(a) Location of the detected region in image having an interest point.



(b) Detailed view of the normalized region after the affine transformation.

Figure 2.11: SIFT example. From: [EGW+10]

models are used. They are abstract mathematical models which describe how colors are represented, usually as three or four values called color components, and commonly provide also a mapping function to the corresponding color space.

There exist many different color spaces, which are suitable for a variety of purposes. The most common is the RGB (Red – Green – Blue) color space. In the following section we are going to describe it, along with two other, the Opponent color space and the Hue Saturation Intensity (HSI) color space, which are widely used for feature description in our approach.

2.7.1 RGB color space

In order to understand the concept how the RGB space was developed, one needs to have a basic knowledge about the human visual system. There are three types of cones in the human eye [WW00], known as the L-, M- and S-cones. The letters stand for Long, Medium and Short wavelength sensitivities. The trichromatic theory, developed by Maxwell, Young and Helmholtz, states that these three types of photoreceptors are approximately sensitive to the red, green and blue region of the spectrum and that the corresponding values are transmitted directly to the brain.

Following this theory, most devices for capturing images have an LMS-fashion light detector. The color is described with three components: R, G and B. These values are depended on the corresponding sensitivity functions and the incoming light:

$$R = \int S(\lambda)R(\lambda)\,\mathrm{d}\lambda \tag{2.11}$$

$$G = \int S(\lambda)G(\lambda)\,\mathrm{d}\lambda \tag{2.12}$$

$$B = \int S(\lambda)B(\lambda)\,\mathrm{d}\lambda \tag{2.13}$$

where $S(\lambda)$ is the light spectrum, $R(\lambda)$, $G(\lambda)$ and $B(\lambda)$ are the sensitivity functions for the R, G and B sensors respectively. How these values are transformed and stored to a three dimensional vector is explained in [Poy95].

RGB color space is the default space used when colors are displayed on computer monitors. It is implemented in different ways, depending on the system and its capabilities used. The most common is the 24-bit implementation, which leaves 8 bits or 256 discrete levels of color per channel. Thus we have million colors. Some implementations use 16 bits per channel, thus 48 bits in total, which leads to a much larger number of different colors. A very common representation of the RGB color space is a three-dimension representation, with each color as an axis. Red as the X axis, Green as the Y axis and Blue as the Z axis. This is visible in the Figure 2.12.

The RGB space is considered the basis for many other color spaces. Although it is so often used, it has some drawbacks, which make it not the best choice for some applications. Its major drawback is its dependence on the capturing device, as is seen

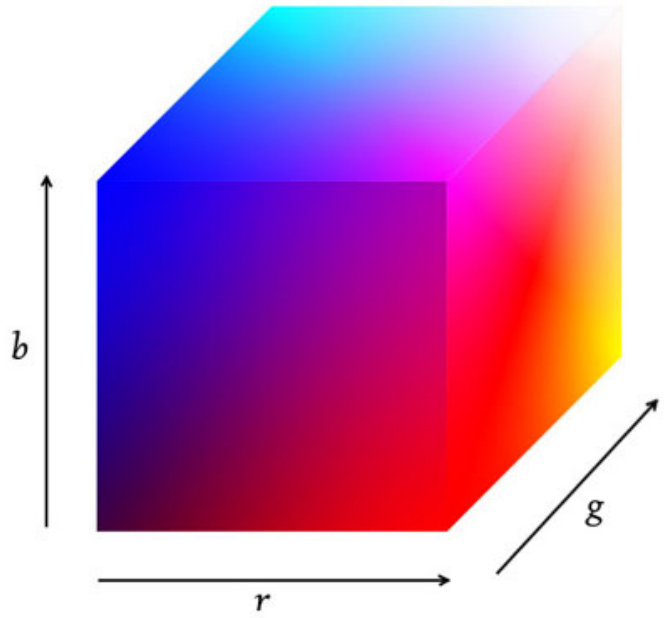


Figure 2.12: 3D representation of the RGB color space. From [3].

from the above equations, and that it is very sensitive to the illumination factor, which makes it unsuitable for applications such as object recognition.

2.7.2 Opponent Color Space

The idea behind this color space comes from the theory with the same name, the Opponent colors theory [D'O96, Fai98, Poy97]. According to this theory, perceived colors are combinations of the luminance factor and the chromatic opponent colors. In addition, there are certain hues, which are never perceived to occur together. A color perception is never described as reddish-green or yellowish-blue, contrary to all other possible combinations [TT03]. This phenomenon has an explanation. Research [KTM85] has shown that there exists a layer in the HVS that converts the LMS responses into an opponent color vector. This vector has an achromatic component (White-Black) and two chromatic components (Red-Green and Yellow-Blue) [TT03]. This encoding helps the human brain to decorrelate the different colors, allowing efficient signal transmission and reducing noise problems. A representation of the opponent color space is visible in the Figure2.13.

There have been many transformations from RGB to the opponent color space proposed, most of them with slight differences. One very good approximation is the one proposed by Joost van de Weijer [J.04], as the opponent colors given by the equations

[3] http://www.codeproject.com/KB/miscctrl/CPicker.aspx

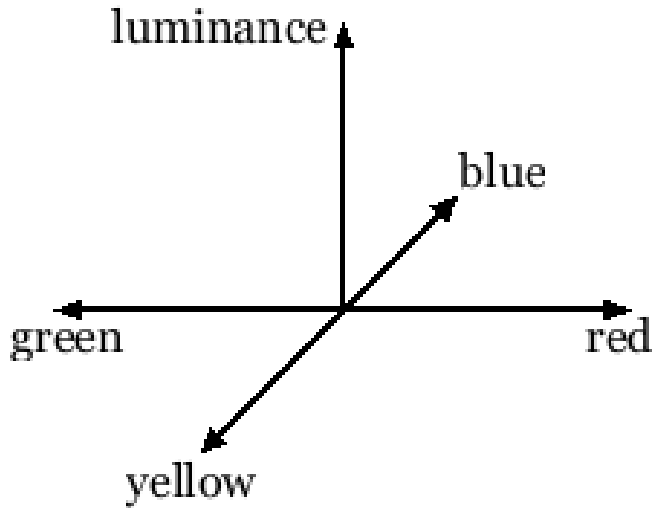


Figure 2.13: Representation of the Opponent color Space. From [4].

below are proven to be specular invariant. These equations are also the ones we are using in our approach. The equations for the transformation are given below:

$$O_1 = \frac{R-G}{\sqrt{2}}, \quad O_2 = \frac{R+G-2\cdot B}{\sqrt{6}}, \quad O_3 = \frac{R+G+B}{\sqrt{3}} \tag{2.14}$$

2.7.3 HSI Color Space

The HSI color space belongs to a group of 3D-polar (cylindrical) coordinate color spaces. The initials stand for Hue, Saturation and Intensity. The last attribute is also often referred to as Lightness (or Luminance), hence HSL color space. The idea behind this space dates back to the years of Isaac Newton, who was the first to arrange colors in a circle [D'O96]. It turns out that this idea is quite correct, as the human brain tends to organize colors by hue, saturation and brightness attributes [D'O96].

The Hue attribute refers to the dominant color seen, e.g. red, blue, green, yellow, etc. The Saturation attribute refers to the degree of dilution, or in a simpler way put, the level of non-whiteness. colors such white or grey have 0% saturation, whilst pure and vivid colors 100%. Sometimes this attribute is also called chroma. It helps to distinguish red from pink, marine blue from royal blue, etc. The brightness attribute refers to the amount of light emitted. It distinguishes the grey levels in different colors. Sometimes this attribute is also called intensity. This leads also to the corresponding name of the space, either HSV (Hue Saturation Value) / HSB (Hue Saturation Brightness) or HSI (Hue Saturation Intensity) / HSL (Hue Saturation Lightness/Luminance). An illustration of the HSI space is seen in the Figure 2.14.

[4] http://www.ryobi-sol.co.jp/visolve/en/colorvision/html

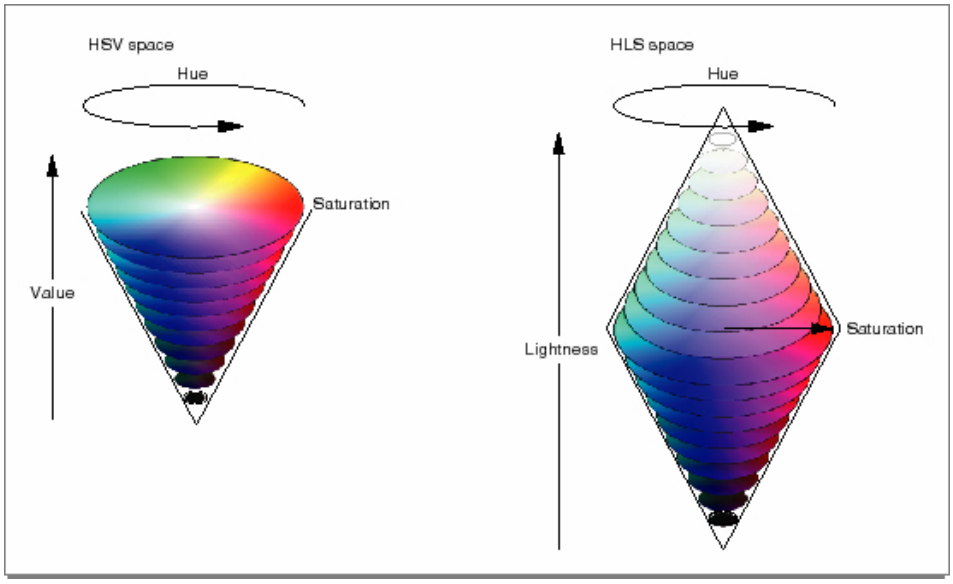


Figure 2.14: Illustrations of the HSV / HSI spaces: (a) on the left, the HSV space as cone (b) on the right, the HSI / HSL space as double cone. From [5].

The transformation of the RGB color space to HSV color space is essentially a conversion from a set of rectangular coordinates to a set of cylindrical coordinates. In the RGB space, each value represents the amount of each red, green and blue in the color. For convenience, we take the values ranging from 0 to 1. Then the valid coordinates for the RGB cube are [0, 1] x [0, 1] x [0, 1]. The idea behind the transformation to the HSI space is to place a new axis between [0, 0, 0] and [1, 1, 1], and to specify the colors in the new space according to the axis. As all points on this axis have the same value for each component (R = G = B), it is called the achromatic axis.

As with the case of the opponent color space, again for the case of the HSI space a lot of transformations have been proposed. The equations for going from the RGB space directly to the HSI space can be found in [FR98]. We again follow in our approach the ones proposed by Joost van de Weijer [J.04], which present a direct transformation from the opponent color space to the HSI, which is actually a polar transformation on the opponent color axis O_1 and O_2 [J.v05]:

$$H = \tan^{-}1(\frac{O_1}{O_2}), \quad S = \sqrt{O_1^2 + O_2^2}, \quad I = O_3 \tag{2.15}$$

[5] http://escience.anu.edu.au/lecture/cg/Color/printNotes.en.html

2.8 Summary

In this chapter we gave an overview of the major issues regarding interest points, which are an important chapter of CBIR. We also discussed the state of the art methods used for their detection and description, such as the Harris Corner Detector and its variations, and the SIFT descriptor. Finally, we gave a description of the different color spaces used in our approach, such as the RGB, the opponent and the HSI color spaces.

CHAPTER 3

Segmentation

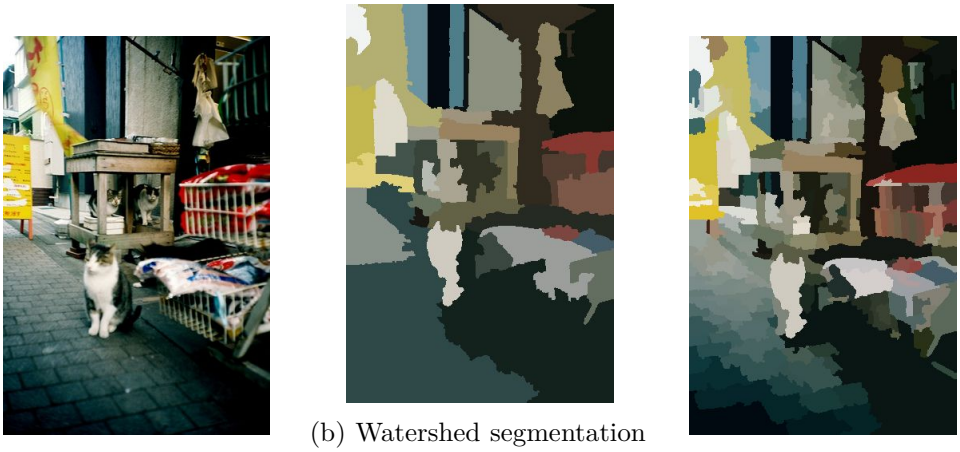
3.1 Introduction

The segmentation of an image is a concept found very often in computer vision and it is used for a variety of applications [BL79, Beu91, BB94, BBY90]. It has been also used for our approach, so it is going to be described in detail in this chapter. In section 3.2 we give the definition of image segmentation, along with its purpose. We then describe some major segmentation methods, such as edge- and region-based segmentation, in section 3.3. Due to its significance, the watershed transform is described separately in section 3.4.

3.2 Definition

The segmentation of an image can be defined as the process of partitioning a digital image into multiple segments or regions. These regions should cover the whole image, but not intersect each other, and the pixels in each region should share a common characteristic, whilst pixels of adjacent regions should be significantly different with respect to the same characteristic. A commonly used criterion is that each region is homogenous with respect to some property, such as texture or color [J.B01]. The whole idea behind this process is to simplify and/or change the representation of the image into something easier to analyze and/or more meaningful. By meaningful we mean that, in the ideal case, we would like the resulting regions to represent or have a major resemblance to regions or objects from the real world contained in the image. A segmentation example is visible in the Figure 3.1.

Although segmentation is widely used in computer vision, is has a "drawback". There exists no "correct" segmentation, as it has not been able to define a perfect segmentation for an image. The reason for this is the fact that a good or bad segmentation is usually judged by the application and the information needed from the image [Han08]. Another



(a) Original image.

(b) Watershed segmentation using synchronous flooding by volume into 40 regions.

(c) Waterfall segmentation.

Figure 3.1: Example segmentations: Original image from: [EGW+10].

problem that makes it difficult to achieve a perfect segmentation is the difference between a segmentation done by a human and an automatic segmentation done by a computer. A human, when asked to segment an image, automatically makes use of the conceptual information depicted in the image, i.e. usually the objects present in the image. An automatic segmentation is not always able to separate all the objects, due to criteria based on homogeneity of image properties and not having any knowledge of the actual objects present in the image. Segmentation is being used in a variety of practical applications, such as medical imaging, face and fingerprint recognition and locating objects in satellite images. Local segmentation may also be used as an effective way to achieve a variety of low level image processing tasks [See02]. This is done by performing segmentation only on a small neighbourhood of pixels and gives information about the local structures of this neighbourhood.

3.3 Major Segmentation Methods

In the following section we will describe the major segmentation techniques used. There have been developed several general-purpose algorithms for image segmentation, which are commonly based on either homogeneity or discontinuity or on both. But, as a general solution for image segmentation isn't possible, these techniques have often to be combined together or with domain knowledge in order to produce adequate results for a specific problem.

3.3.1 Thresholding

Probably the simplest segmentation method is thresholding. Here we define a threshold T such that each pixel x, y where $I(x, y) < T$ is marked as foreground, whilst the other

ones as background. This convention is known as threshold above. Variants include threshold below, which is opposite of threshold above [SS02]; threshold inside [SS02], where a pixel is labeled "object" if its value is between two thresholds; and threshold outside [SS02], which is the opposite of threshold inside. At the end, a binary image is created by coloring each pixel white or black, depending on a pixel's label. As this method is very simple, it is suitable when objects don't touch each other or if their grayscale levels are clearly distinct from the background grey levels.

Correct threshold selection is crucial for successful threshold segmentation. One alternative is to use a fixed threshold chosen independently of the image data. If it is known that one is dealing with very high-contrast images where the objects are very dark and the background is homogeneous and very light, then a constant threshold of 128 on a scale of 0 to 255 might be sufficiently accurate. In most cases the threshold is chosen from the brightness histogram of the image that we wish to segment. The main idea is to smooth the image and then search for minima, using methods such as isodata algorithm or background-symmetry algorithm. This method assumes that there is some average value for the background and object pixels, but that the actual pixel values have some variation around these average values. Another solution would be to use a different threshold for different regions in the image, as a function of local image characteristics. This method is called adaptive or local thresholding [SS02].

Thresholding is computationally inexpensive and fast - it is the oldest segmentation method and is still widely used in simple applications [PP93]. Although simple, this method has significant problems. Firstly, it may be difficult to identify the minima, if the image is noisy. Moreover, we don't take into account spatial information; the method is subject to noise and illumination changes and results in many holes and discontinuities in the segmentation.

3.3.2 Edge-based Segmentation

Edge-based segmentation techniques try to look for borders that separate regions in the image, i.e. using discontinuity as a criterion. This is done by applying edge detection algorithms, using the thought that edges in an image and regions boundaries are closely related, since there is often a sharp adjustment in intensity at the region boundaries. There exist many different algorithms for edge detection. They usually mark image locations of discontinuities in color, grey level, texture, brightness etc.

The main problem with this approach is that the resulting edges can hardly be used as boundaries between regions, as they usually don't form continuous lines, and there occur many false and missed detections. This can be solved with some post-processing steps which connect local edges to produce object contours. A known method is edge linking [CD95], where we link adjacent edge pixels by seeing if they have similar properties. Comparisons are made of the orientation and magnitude values. The sets of linked pixels can be thought of as borders. Other edge refinement methods exist and are often used, such as edge relaxation [SC00]. Here the edge information within a neighbourhood is taken into account. All the image properties, including those of further edge existence, are iteratively evaluated with more precision until the edge context is totally clear - based

on the strength of edges in a specified local neighbourhood, the confidence of each edge is either increased or decreased. Edge relaxation can rapidly improve the initial edge labelling in a few iterations. Moreover, we can use the border local information [PP93]. Supposing there is a clue about where a border should be, then the edges near this border are regarded as part of it. Another method for improving edge-based segmentation is the so called edge image thresholding. In some cases small edge values correspond to non-significant grey level changes resulting from quantization noise, small lighting irregularities, etc. Simple thresholding of an edge image can be applied to remove these small values. Selection of an appropriate global threshold is often difficult and sometimes impossible; p-tile thresholding [See02] can be applied in this case. The threshold is calculated based on the intensity level, so that the desired fraction of the image can be set below this level.

To sum up, regions can be built from detected borders that are complete - by using any of the above described methods. But, unfortunately, detected borders are rarely complete, which leads to the use different approaches, such as region-based segmentation methods, to achieve an adequate segmentation.

3.3.3 Region-based Segmentation

Region-based segmentation techniques construct regions directly, opposed to edge-based techniques described before. Although it is easy to construct regions from their borders, and it is easy to detect borders of existing regions, segmentations resulting from edge-based methods and region- based methods are not usually exactly the same, and a combination of their results may often be a good idea. Region-based techniques are generally better in noisy images than edge-based ones, where borders are difficult to detect [OPR78]. Homogeneity is an important property of regions and is used as the main segmentation criterion in constructing the regions. The criteria for homogeneity can be based on gray-level, color, texture, shape, model (using semantic information), etc. In the following sections we describe the major region-based techniques in detail.

Region Growing

The main idea behind region growing [Zuc76] methods is to build each region starting from a pixel. First, we choose a "seed", which can consist of one or more pixels. Then, we compare it with neighbouring pixels and if they satisfy the pre-defined criteria, we add them to this growing region. The growth for each region stops when none of the adjacent pixels satisfy the similarity criterion [Han08]. Once a region is done, a new seed is chosen, from the yet unlabeled pixels, and the process continues until all the pixels in the image are labelled. The result of region growing usually depends on the order in which pixels are appointed to regions, meaning that the choice of the seed pixel, but also the type of comparison between the pixels is critical.

There are many possibilities for the seed pixel. It may be chosen based on various pixel characteristics, taking also into consideration the similarity criteria used, it may be chosen as one of the four corner pixels of the image, or the choice may be completely

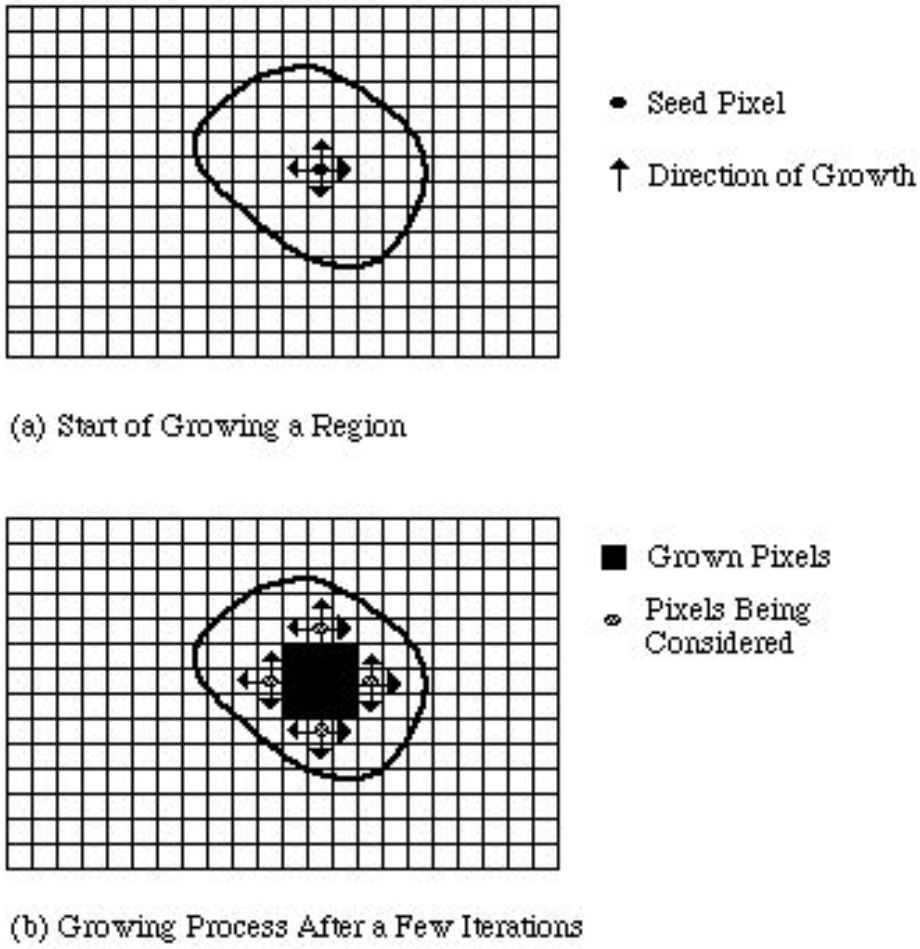


Figure 3.2: Region growing example. From [1].

arbitrary. Another possibility is to initialize the region with not only a single pixel but a small set of pixels to better describe the region statistics. With such initialization, not only a region mean is suggested but the variance as well [Pav77]. These pixels can be given directly through human input or by sampling a small area around the initial pixel. Finally, background information can also be used, as described in [Pav77].

What is maybe more significant than the initial seed choice is the similarity criteria used for each unlabeled pixel in order to grow the region. One obvious similarity measure is to compare individual pixel intensities, with the drawback of being sensitive to noise. We can reduce our sensitivity to noise by comparing neighbourhood characteristics between pixels. Other characteristics usually used are texture, color, grey scale values etc. Comparisons for adding a new pixel in a region can always be done with the seed

[1] http://www.cs.cf.ac.uk/Dave/Vision_lecture/node35.html

pixel. This has the advantage of having a single basis for comparison across all pixels in the region, but, on the other hand, this makes the whole procedure very sensitive to the choice of the initial seed pixel. One way of overcoming this effect is to compare an unlabeled pixel with the neighbouring one already in the region. This approach produces transitive closures of similarity, but also can cause significant drift as one grows farther away from the original seed pixel [Pav77]. A third approach is to compare the candidate pixel to the entire region being built. As the region grows, aggregate statistics are collected and constantly updated, and each candidate pixel is compared to these statistics. Although gradual drift is still possible, the weight of all previous pixels in the region acts as a damper on such drift. Some texts [Por02] refer to this as centroid region growing.

The region growing method has certain advantages compared to edge-based methods. Firstly, it is guaranteed, by definition, to produce coherent regions, i.e. all pixels in the region have sufficient similarity. Linking edges, holes and gaps coming from missing edges are no longer a problem. As it builds regions pixel by pixel, we can immediately know which pixel belongs to which region. But these methods have also one certain drawback, as they cannot detect objects in the image than span multiple disconnected regions. In [MJ97] an improved seeded region growing algorithm is proposed, which retains the advantages of former implementations, but also is pixel order independent. In [PL90] a method is presented, which combines region growing and edge detection for image segmentation.

Region Splitting and Merging

Region splitting and merging - and their combination - are two famous methods for region-based segmentation. Region merging [BFdW01] begins with the whole image and each pixel representing an individual region. Then it starts examining each region, and begins to merge adjacent regions that satisfy a predefined criterion. The process stops when there are no further region to be merged. Region splitting [OPR78] is the opposite of region merging. It begins with the whole image representing as a single region. Then, it starts to sequentially split the region in order to satisfy the homogeneity condition. When no further region needs to be longer split, the procedure stops. Although the above described methods seem to be dual, the results may differ, even if the same homogeneity criterion is used.

A very popular approach is the combination of these methods, in order to use the advantages of them both. The region splitting and merging [LNHT94] is a hierarchical method. It represents the whole image in a pyramid like structure. Regions are square shaped and correspond to elements of the appropriate pyramid level. Again, as in splitting methods, the algorithm starts considering the whole image as a single region. If the homogeneity condition is not satisfied for this region, then it is split into four regions. The condition is again tested on each of the four new regions, and if some of them don't satisfy it, they are further split into four new ones. The procedure continues until all regions in every level of the pyramid satisfy the homogeneity condition. If four regions with the same parent node exist at any pyramid level with approximately the same value

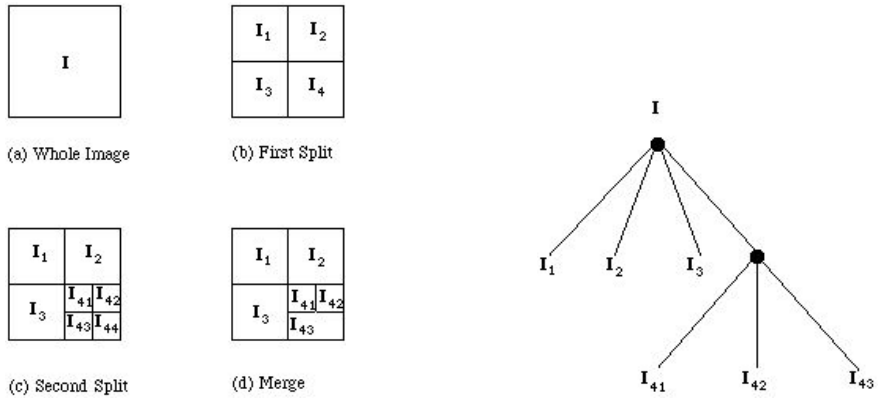


Figure 3.3: Region Splitting and Merging example: (1) Let I be the initial image to be segmented shown in (a). Not all the pixels in I are homogeneous so the region is split into four parts as shown in (b). We assume that all pixels within regions I_1, I_2 and I_3 respectively are homogeneous but those in I_4 are not. Therefore I_4 is split further as shown in (c). Now we assume that all pixels within each region are homogeneous with respect to that region and that after comparing the split regions, regions I_{43} and I_{44} are found to be identical. Hence they are merged as shown in (d). (2) Resulting region splitting and merging tree. From: [2].

of homogeneity measure, they are merged into a single region in an upper pyramid level. The resulting hierarchy can be understood as the construction of a quadtree [KG08], where each leaf node represents a homogeneous region. The result of the algorithm in the image and the corresponding quadtree is shown in Figure 3.3. The algorithm continues splitting and merging recursively until no further regions can be split or merged. An unpleasant drawback of this method is the square-shape assumption of the regions. But the main disadvantage of the algorithm is its sensitivity to image translations [Han08].

3.4 Watershed Transform

The watershed transform combines the two aforementioned approaches of both edge-based and region-based segmentation and it is widely used as a major segmentation tool. It builds the regions around the regional minima of the image (region-based) and the boundaries of adjacent regions are located along the crest lines of the gradient image (edge-based). In this section we describe the basic algorithm along with some methods that make the algorithm more efficient.

3.4.1 Definition

The intuitive idea underlying this method comes from geography: it is that of a landscape or a topographic relief which is flooded by water, watershed being the divide lines of the

[2] http://www.cs.cf.ac.uk/Dave/Vision_lecture/node34.html

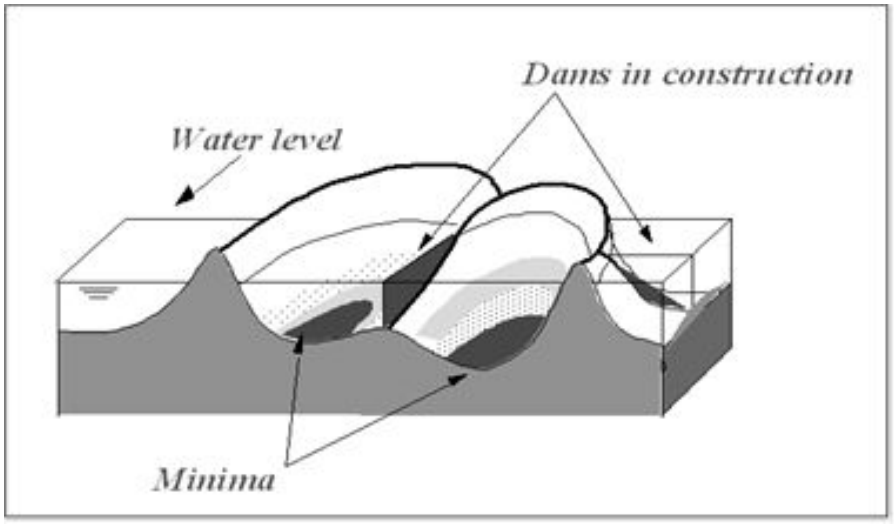


Figure 3.4: Watershed illustration, with minima at the bottom of the catchment basins, and the resulting constructed dams where the water from different basins would meet. The watershed lines are the dam walls dividing the catchment basins. From [3].

domains of rain falling over the region [J.B01]. This is visible in Figure 3.4.

A grey scale image can easily be interpreted as a topographic surface, where the image gray-levels represent altitudes. Following the same process as in topography, a simulated drop of water falling onto the surface will flow in the direction of the steepest gradient to a grey level minimum [Han08]. So each pixel of the image can either be a part of the watershed or a part of a catchment basin. This assignment results to region edges corresponding to high watersheds and low-gradient region interiors corresponding to catchment basins. As far as homogeneity is concerned, catchment basins are homogeneous in the sense that all pixels belonging to the same catchment basin are connected with the basin's region of minimum altitude (gray-level) by a simple path of pixels that have monotonically decreasing altitude (gray-level) along the path [J.B01]. These catchment basins represent the regions of the segmented image.

When dealing with color images, the water drop simulation is not helpful; as such images contain areas of constant value (defined as plateaus). But the following approach solves this problem. Instead of identifying the downstream paths flowing to the image minima, the catchment basins fill from the bottom. In more detail: a hole is punched in each regional minimum and the entire topography is flooded from below by letting water rise through the holes at a uniform level in all dams. The flooding creates lakes at each local minimum e.g. the catchment basins. When rising water level in distinct catchment basins is about the merge, a dam is built to prevent merging. These dam boundaries correspond to the watershed lines and the formed lakes to the segmentation regions. This approach is essentially dual to the first one. This procedure is illustrated in Figure 3.5.

[3] http://www.tele.ucl.ac.be/PEOPLE/OC/these/node23.html

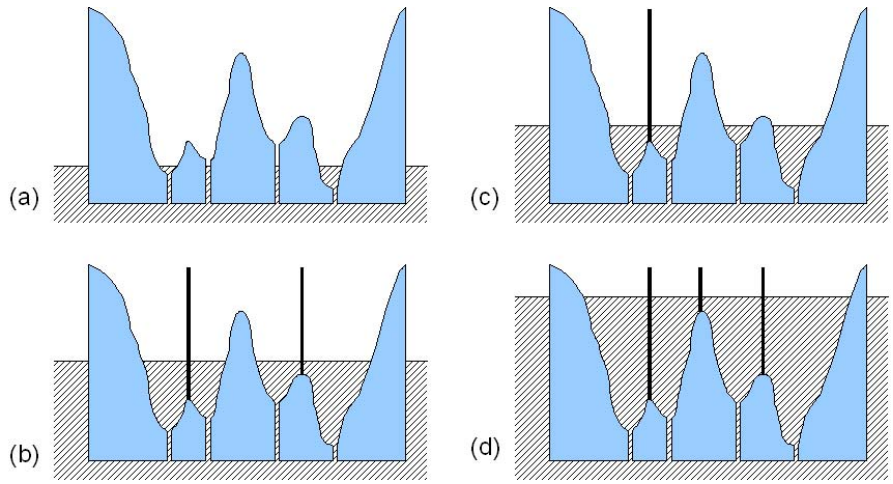


Figure 3.5: Four different stages of watershed construction by flooding. The final watershed lines are given by the black lines. From [DBC06]

The above described watershed algorithm can be easily implemented using morphological geodesic operators, as described in [SAS02]. The gradient operator is very often used for segmentation purposes, as homogeneity of the grey values of the objects is one of the first criteria for segmentation. When other criteria are relevant, other operators may be used, as the distance operator in case of shape based criteria [Beu91]. Beucher and Lantuejoul [BL79] were the first to propose a watershed algorithm based on immersion analogy [VS91]. Luc Vincent and Pierre Soille introduced a fast and flexible algorithm for computing watersheds in digital greyscale images in [VS91], which is used in most practical implementations. Over the years there have been proposed and introduced many different approaches for improving the watershed transform. A more detailed definition of the watershed transform, based on morphological operators, along with a survey of algorithms and parallelisation strategies, can be found in [J.B01].

3.4.2 Watershed Methods

There are two major issues when the watershed transform is being used for images, which affect strongly its effectiveness. Firstly, the standard watershed transform, in an overwhelming majority, results in an over-segmentation. This happens mainly because there is an excess of local minima in the image, with each one of them producing a catchment basin [Han08]. The other problem is that, in many cases, the corresponding landscape of an image is complicated, having many discontinuities, and / or the image itself may be very noisy, thus hindering the watershed transform.

There are three methods used in order to reduce or avoid the aforementioned problems. The first one is pre-processing and filtering, which tries to enhance some image characteristics in order to have a better result. The other two methods take as input the output of the first method, which is usually referred to as the segmentation image.

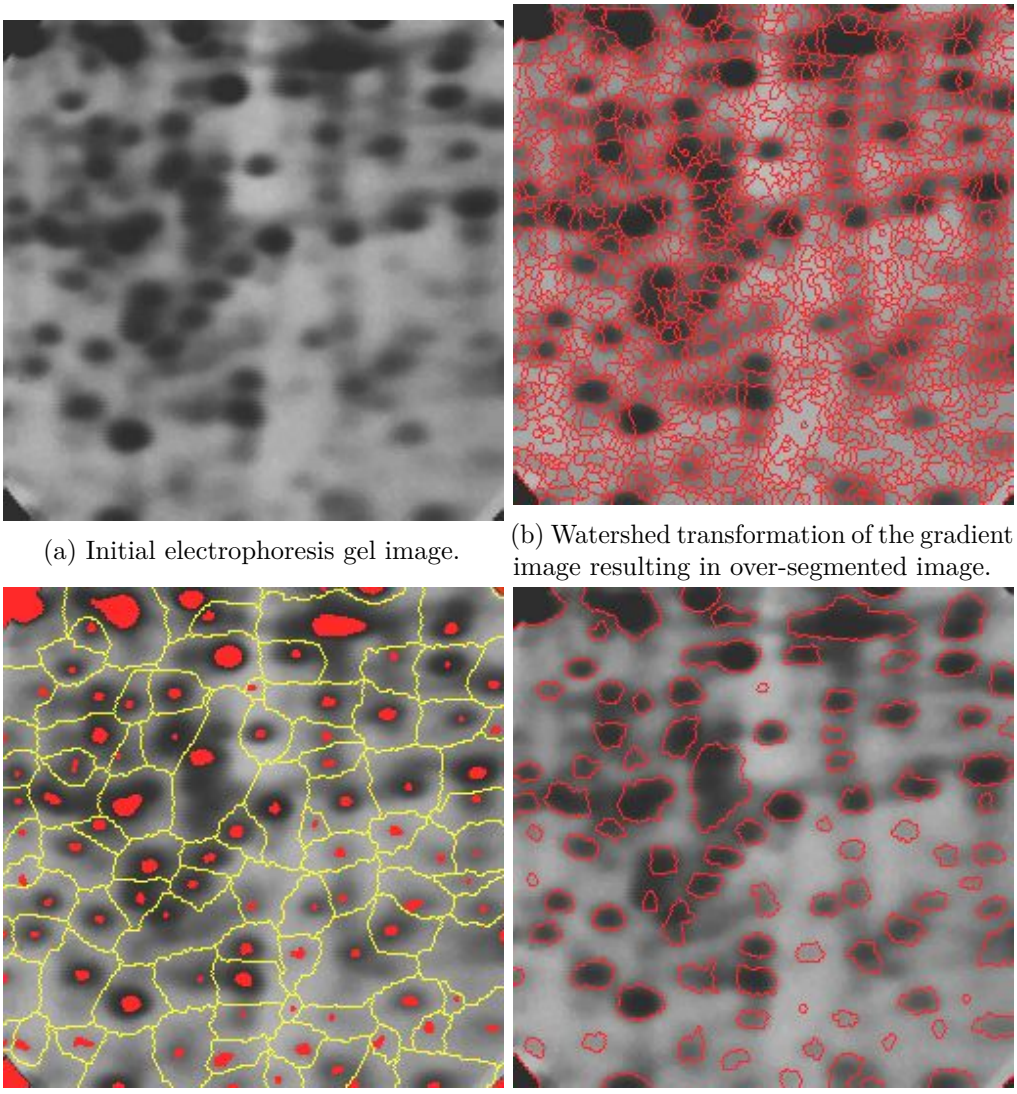
The second method, called watershed with markers, makes use of predefined markers which indicate where the flooding should start. The last method uses a hierarchy of regions, where the top level corresponds to the whole image and the bottom level to the segmented image. These three methods are described in the following sections.

Pre-processing and Filtering

As mentioned before, the intention of the pre-processing and filtering approach is to enhance some image characteristics. This approach is subsequently divided into two steps, the pre-processing step, which tries to enhance the image boundaries, and the filtering step, which aims to filter out the noise that produces the spurious local minima.

The objective of the pre-processing phase is to produce well defined region boundaries marked by grey scale maxima [Han08]. In most cases this is achieved through various morphological operators, as the appropriate approach depends on the image to be segmented. A classical problem, where pre-processing is needed, is when round objects such as coffee beans or blood cells are present in the image and they are overlapping each other. In order to successfully separate them, one can use the distance transform described in [SAS02]. For grey scale images the morphological gradient of the initial image, as described in [SAS02], is suitable, but other known operators such as openings and closing could also be used. Usually, a combination of different operators is needed. Various examples from real life images and especially from traffic scenes can be found in [MBM90]. An approach for color images using the saturation-weighted cue and the brightness gradient is proposed in [AS03b]. Other approaches, which merge watershed and edge detection techniques, also exist.

The objective of image filtering is to reduce the noise and other small fluctuations in images. This noise is causing a variety of local minima to appear on the image, which need to be removed, because each one produces a region in the watershed transform, which leads further to the over-segmentation of the image. There is a variety of filters available that can be used for this aim. The problem is to find the good trade-off between smoothing and good localization of the contours: a large smoothing simplifies the detection but creates poorly localized contours whereas a reduced smoothing does not suppress enough noise [Mey04]. Linear filters tend to blur contours and smooth object boundaries [Mey04], hence are not suitable for our purpose. Non-linear filtering such as median filtering or morphological filters present a better choice for our aim. The levelings, originally presented in [Mey04], are morphological filters created specifically to be applied to an image before segmentation. They are capable of producing a simplified image without blurring or displacing the contours, so that the segmentation may be done on this simplified image. Levelings are traditionally associated with isotropic or other morphological markers, such as the Alternating Sequential Filter (ASF), which performs a series of alternating openings and closings. In [KA06] the Anisotropic Diffusion Filtering is used as marker. Moreover, in [LD05] a new class of filter is introduced, which combines a non fixed-shaped structuring element with a measure, hereby creating a class of shape-adaptive neighbourhood transforms, which can also be used for image filtering.



(a) Initial electrophoresis gel image.

(b) Watershed transformation of the gradient image resulting in over-segmented image.

(c) Markers of the blobs and of the background.

(d) Marker-controlled watershed of the gradient image.

Figure 3.6: Watershed with markers example. Initial image from [4].

Watershed with Markers

The watershed with markers approach is very similar to the original flooding algorithm, with one significant difference. Instead of having holes pierced at each minima of the surface, holes are done only in specific positions specified by markers. In this way the flooding starts only in a small set of catchment basins, instead of in every one. Again dams are built where water from two different basins would merge. For the basins that

[4] http://cmm.ensmp.fr/~beucher/wtshed.html

don't have any markers, the water simply flows in from adjacent specified basins. As each marker corresponds to one region only, the resulting segmented image will have fewer regions compared to the original approach. In this way we avoid the undesired over-segmentation. The idea of using markers in the watershed is visible through the example in Figure 3.6.

The set of markers are usually saved in the so called marker image, which is a binary image with each connected component corresponding to a marker. A marker can be either single marker points or larger marker regions. The technique used most often is the minima imposition [SAS02]. Here, we take the marker image and impose it over the initial image, eliminating in this way any undesired minima and keeping only those indicated from the marker image. The resulting image can then be segmented, with only one region built for each connected component of the marker image. For this approach, the extra filtering of the minima is not needed, as it is done by the minima imposition itself. A correct set of markers is crucial for successful watershed segmentation with markers. There are two approaches for obtaining the markers. The simplest one is by user interaction. Usually, an experienced user, according to the nature of the images, is asked to draw in the markers, which indicate the desired objects in the image. Although this method tends to produce the best results possible, it also consumes much time and resources. Therefore it is more often used in specific cases such as the medical ones and not in those, where a large number of images need to be segmented. The other approach for the marker selection is to be automatically generated. There is no standard method for creating one automatically, as it depends strongly on each application. Usually a combination of morphological operators is used, such as extended minima [SAS02], flat zones, adaptive thresholding [CYV00] and filling of closed objects. An implementation and complexity of the watershed with markers algorithm computed as a minimal cost forest is presented in [FBW01]. Additionally, the minimal set of markers problem is discussed in [LS02], where a solution based on the minimum spanning tree is presented. The importance of markers is presented in [Beu91]. Finally, an approach for interactive segmentation of objects in image sequences using propagated markers is introduced in [FR07].

Hierarchical Watershed

Unfortunately, in some cases the marker approach isn't effective. For example, some objects may be so overlapped, complex and varied, so the marker placement becomes more difficult. So we are led to further watershed segmentation methods, such as the hierarchical watershed. The main difference between this watershed and the approaches described so far is that the latter produce a single partition as output, while the former produces a hierarchy of partitions. The bottom level of the hierarchy is the same as the output of the classic watershed algorithm. As one moves up the hierarchy, regions are merged to form larger ones, until one reaches the top, which consists of a single region covering the whole image. This procedure can be seen in Figure 3.7. One can also obtain a specific number of regions by simply selecting the corresponding level of hierarchy. Alternatively, a combination of partition regions at different levels can be chosen if one

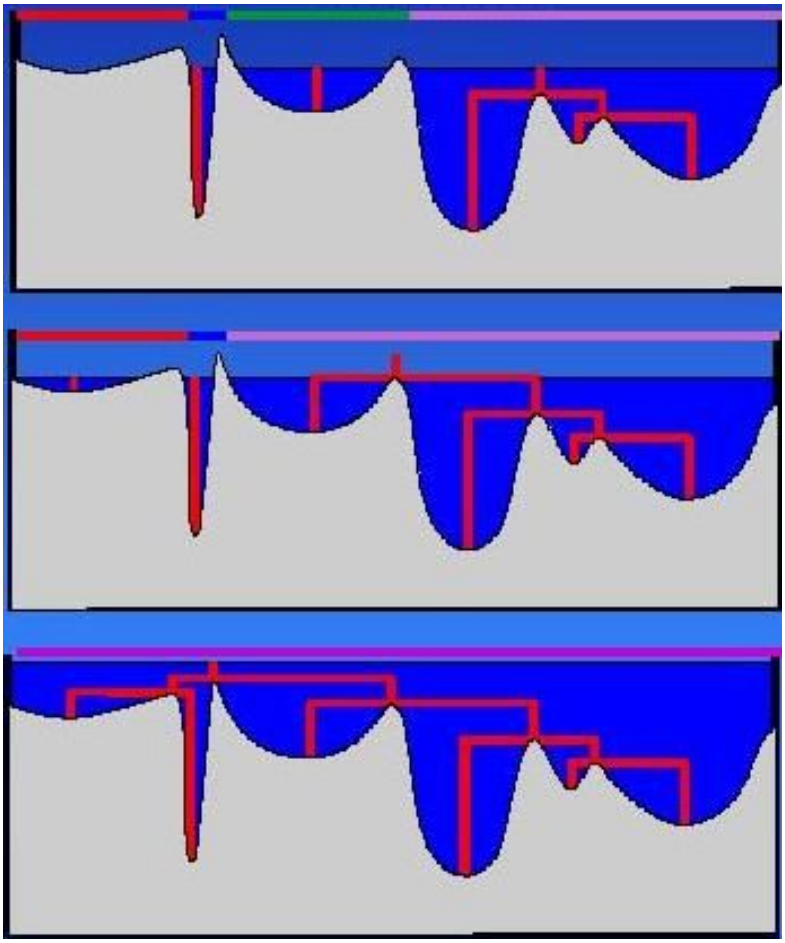


Figure 3.7: Hierarchy created from the classical watershed. The red lines correspond to the regions being merged at specific levels. From: [MBM90].

requires segmentation at a finer scale for some parts of the image and at a coarser scale for other parts [Han08].

In general, hierarchies produced by the classic watershed algorithm are not satisfactory. Therefore, new approaches have been developed. Two of them used frequently are hierarchies obtained by synchronous flooding and by the waterfall algorithm, and are discussed in the following section.

Synchronous Flooding This approach uses for the initial step the bottom level of the output of the classic watershed algorithm. Then the flooding process begins, but in this case no dams are built, when two adjacent lakes are about to merge, as in the original approach. Instead they are left to progressively merge, building in this way a hierarchy. At each step the lakes are merged pair wise, forming the new level of the hierarchy, so that each new level will have one region less. This potentially results in

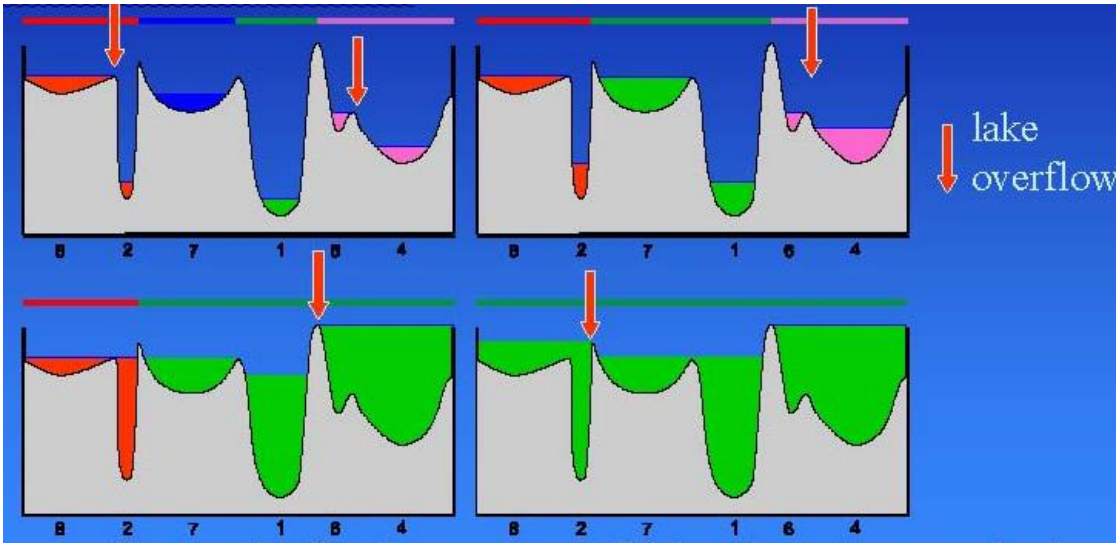


Figure 3.8: Synchronous flooding based on depth. During the flooding process, all lakes have the same depth, in contradiction to the classical watershed. From: [MBM90].

a large number of levels. At the end one can follow the hierarchy to the level with the desired number of regions and stop the splitting there.

In general, hierarchies formed using the classical watershed flooding are not perceptually satisfactory. For this reason new merging criteria have been introduced through synchronous flooding. In the standard approach, the global water level during the flooding remains constant, referred to as uniform flooding [Han08]. In the new approach the flooding is done in such a way that each lake shares, at any time, the same value for some measure. This can be depth, area, or volume. When a lake is about to overflow, it stops growing and it is absorbed by an adjacent lake [MBM90]. The meaning for each measurement can be viewed as follows. For depth, the most contrasted structures in the image are the most important, regardless of their size. Area criterion focuses on large structures in the image. The volume criterion is a trade-off between these two criteria, which gives a good approximation of visual importance and produces usually the most satisfactory results. An example of synchronous flooding based on depth is visible in Figure 3.8.

Many methods for improving the hierarchy have been proposed. Markers can be placed in regions that should be present in higher levels of hierarchy, respectively to the watershed with markers approach. The most efficient method that has been proposed is the use of graphs, and especially the so called Region Adjacency Graph (RAG). Each catchment basin is represented by a node and the edges represent the neighbourhood relations in the graph. More specifically, when two adjacent lakes are about to merge, then the edges of the corresponding vertices in the graph are weighted, where each weight is the size measurement of the full lake. It can be proved that the graph produced by this approach is a minimum spanning tree (MST) [Mey01]. Moreover, the flooding always

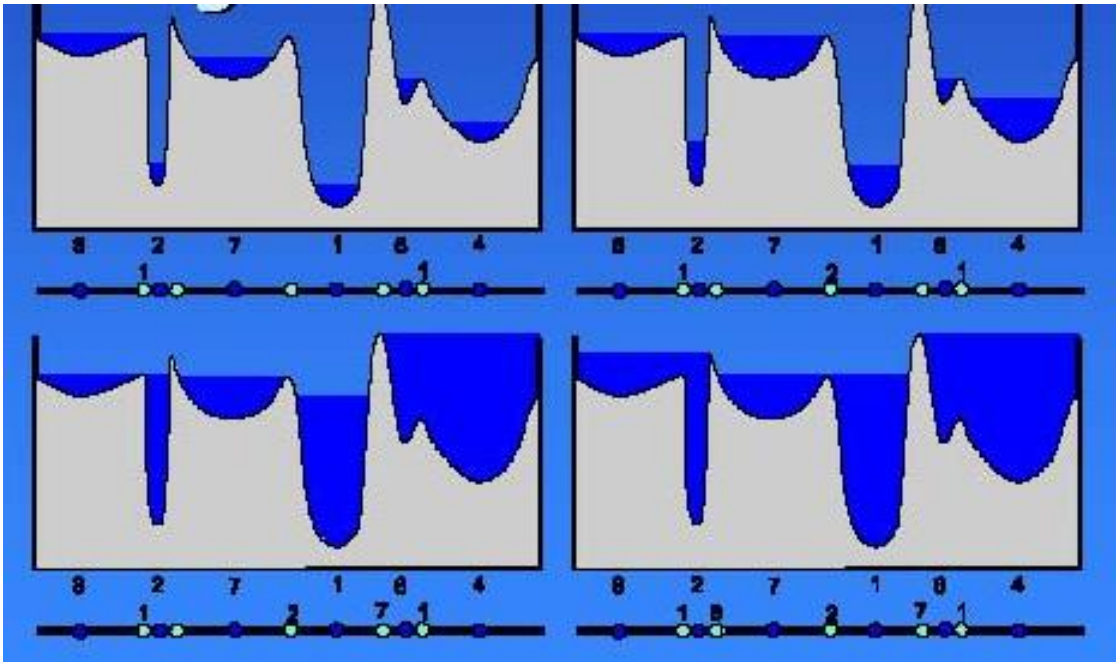


Figure 3.9: MST creation while flooding: When a lake is about to overflow, the difference between this catchment basin and its neighbour is defined as the measurement of the full lake. From: [MBM90].

follows the easiest path. But the easiest path is always included in the minimum spanning tree. Hence, the MST contains all information needed for simulating any type of flooding on the graph. Moreover, one can easily choose the number of desired regions in the segmented image by using the MST. All one has to do in order to have N regions is to cut the $N-1$ edges with the highest weights. The MST corresponding to each criterion can be created at the same time with the flooding, as illustrated in Figure 3.9. More examples can be viewed in [MBM90].

Waterfall The waterfall segmentation is another hierarchical approach, introduced in [Beu94]. This method takes into account every boundary between adjacent regions with respect to its neighbourhood. This method also takes the outcome of the classic watershed algorithm for the lowest level of the hierarchy. Originally, boundaries between adjacent catchment basins are characterised by the height of the pass-point [Han08]. If any boundary is surrounded by higher boundaries, e.g. having higher pass-points, it will disappear in the next level of hierarchy. This is demonstrated in the Figure 3.10.

This procedure is iterated to build further levels of the hierarchy. There is only one parameter in this approach, the level of hierarchy where the algorithm should stop. The number of levels needed depends on the image, but usually the levels are fewer when compared to other hierarchical approaches, as the regions tend to merge rapidly.

The initial waterfall algorithm was constructed using morphological reconstruction

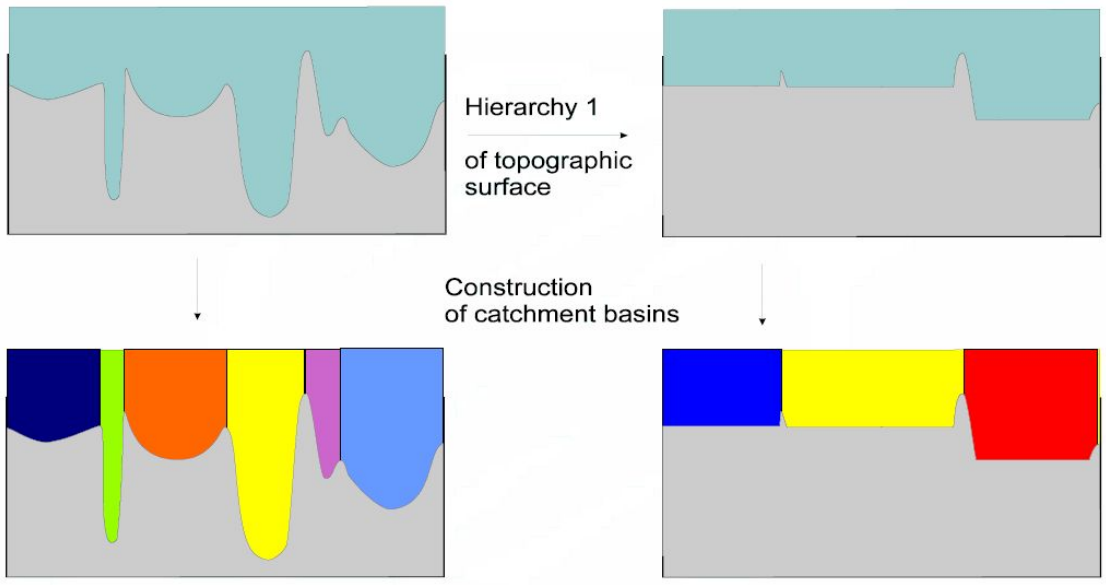


Figure 3.10: Demonstration of waterfall segmentation. In the top left picture the topographic surface is shown, and in the bottom left picture the catchment basins from the classic watershed algorithm are visible. To the top right, the next level of the hierarchy is shown, after removing the lowest pass-points, and to the bottom right the corresponding catchment basins. From: [MBM90].

operators. A more efficient algorithm, based on graphs and the Minimum Spanning Tree, is presented in [MB05]. The second implementation is faster than the original and it allows easy manipulation of the hierarchy. One can simply change the valuation of the edges or the characteristics of the catchment basins, and a new hierarchy will emerge. A thorough example of this method can be found in [MBM90].

3.5 Summary

In this chapter we described in detail the major aspect of segmentation, which plays also an important role in our approach. We also looked in detail some of the most significant segmentation methods, based on different techniques, such as edge detection or region splitting and merging. Finally, we described thoroughly the watershed transform, and saw the major algorithms for its implementation, such as watershed with markers and hierarchical watershed.

CHAPTER 4

A Segmentation-Based Object Recognition System For Images

4.1 Introduction

In this chapter we present a software prototype segmentation-based object recognition system for images. This prototype is based on the bag of keypoints algorithm, which was described in chapter 2. In the following section we describe all implementation issues in detail and also how these cope with each step of the bag of keypoints.

4.2 A Segmentation-Based Object Recognition System For Images

The prototype is based on training the classifiers with the histograms built from the occurrences of the features in every image, an idea that comes from the bag of keypoints algorithm. In addition the prototype uses over-segmentation, and color features extracted from the images. As seen in the Figure 4.1, an object recognition scenario of an unknown image consists of different tasks. But they can be divided into two main phases: training and testing.

The training phase consists of different tasks, which are seen in the figure and are described individually in the following sections. Firstly, every image in the database is over-segmented. Then, color features are extracted from every image and normalized. In the next step the extracted features are clustered using k-means clustering. Then the cluster centers, e.g. our vocabulary, from the previous phase are used to build the histograms. Finally, these histograms are used to train the SVM classifiers.

The testing phase is simpler. For an unknown image, color features are extracted, the same way as in the training phase. No clustering is required, but the histograms are

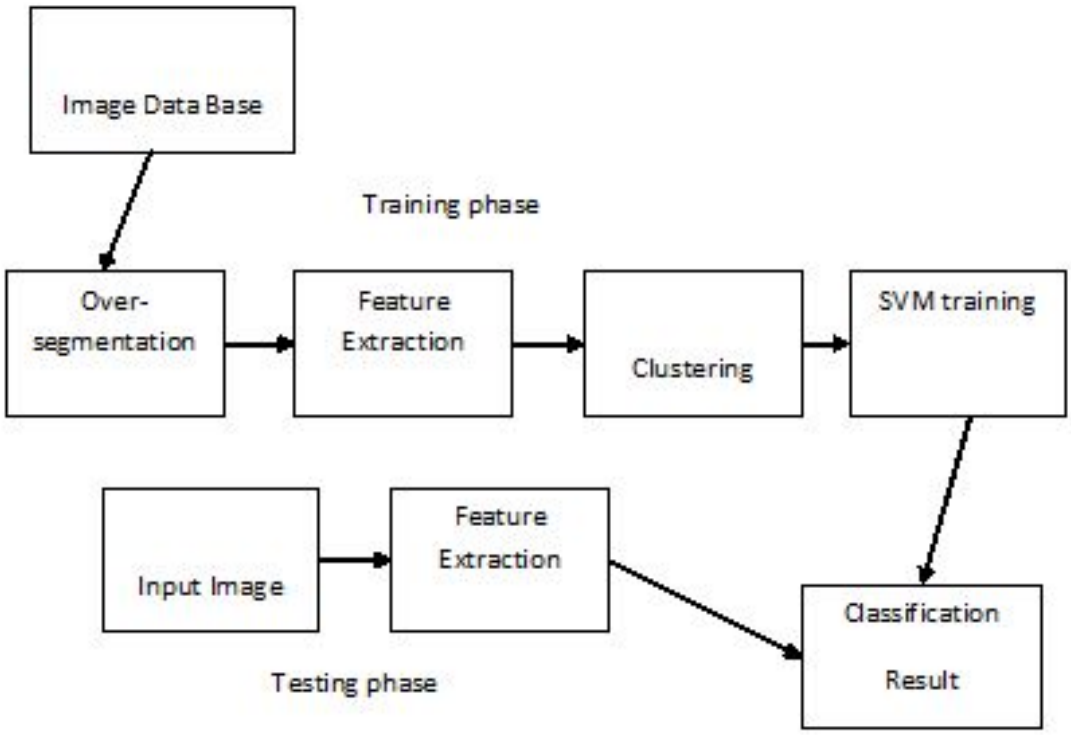


Figure 4.1: System overview.

built based on the same vocabulary as before. In the last phase, these histograms are fed to the same classifiers, which determine which category the image belongs to.

For our implementation we used the MATLAB [1] platform, as well as some free-distributed tools, which are discussed in their corresponding sections.

4.2.1 Training

The training phase provides the basic information e.g. the features, which will be used later in our approach, as input for the bag of keypoints algorithm. Each step of this phase is described in detail below.

Over-segmentation

As mentioned in the chapter about segmentation, the classic watershed transformation results to an over-segmentation of the image. This means that in the output image we have an excess of regions that make most objects in the image indistinguishable. We will try to take advantage of this in our approach.

[1] http://www.mathworks.com/products/matlab

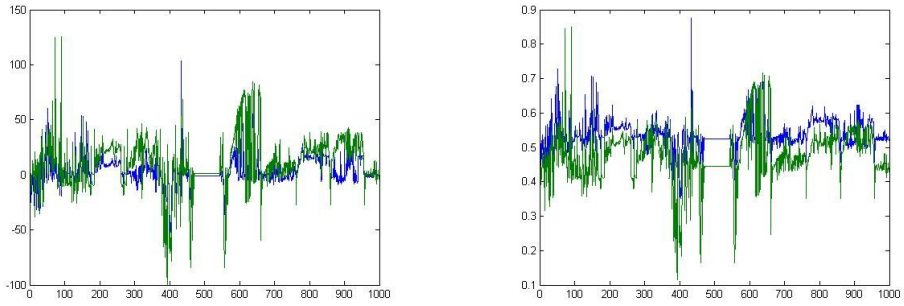
For the actual segmentation we are using the segmentator [Mar05], which is a segmentation tool developed by Beatriz Marcotegui of the Centre for Mathematical Morphology of the Mines Paris Tech. It uses two different segmentation approaches: volumic extinction values and waterfall. Both methods are based on the Minimum Spanning Tree, which offers the most efficient implementations. The tool allows also choosing the number of iterations for the waterfall approach, the number of regions for the volumic extinction value approach, the size of the filter by leveling and the color gradient used. By default no filter is used and the L1 gradient is used, which stands for the L1 norm in LHS [AS03a], improvement of the HLS color space. More detailed information about using the tool can be found in [Mar05].

In our approach we are using the aforementioned default settings for filtering and color space, and we are taking advantage of both segmentation implementations, volumic extinction value and waterfall, and over-segment our images with both of them. For the volumic extinction value we are using 100 regions and for the waterfall only 2 iterations. These values were chosen after various tests for both cases, in order to get the best results possible. It proved that the above values outperformed every other one tested.

Feature Extraction

The next step after over-segmenting the images is to extract our features from them. As mentioned in the introduction, we are extracting color information from the images. More specifically, we examine each different region that has resulted from the over-segmentation in each distinct image and we find out its color, which is homogeneous in the whole region.

The color values are read in the RGB color space, which means that we have 3 values, red, green and blue, for each region. The problem with this color space though is that it is very prone to illumination changes such as shadows, shading, specularities and object reflectance and much depended on the capturing device. This leads to correlated components of the color space. By using transformations to other color spaces, we can overcome this. Normalizing the RGB space into rgb could be a solution, but this space has the drawback of being very unstable near zero illumination. In order to overcome this weakness, we choose to use two different color spaces, the opponent (section 2.7.2) and the Hue Saturation Intensity (section 2.7.3) color spaces. These spaces have uncorrelated components, and only the photometric axes are influenced by common photometric variations. The orthogonal transformation into the opponent color space shown in section 2.7.2 provides specular variance [Stő07]. This can be explained by the following: opponent colors as yellow-blue and red-green are placed at the end points of the axes of the color space, and therefore have the greatest distance between them, which makes them more distinguishable. A polar transformation of the two axes of the opponent color space leads to the HSI space. The derivative of the hue component is both the shading and the specular quasi-invariant, which means that those light effects should not change the coefficient [J.v05]. In addition, we leave in each space the third component out of our calculations, O_3 for the opponent and I for the HSI color space as defined in section 2.7, ignoring any illumination influences on our images.



(a) Plot of unscaled data (b) Plot of scaled data

Figure 4.2: Abstract (first 1000 values) of the features plot from the opponent color space with 100 regions case.

In both cases we end up with a feature matrix from every image. In more detail, we have for each specific space a two column matrix, with each column having the values from each component. The number of values for each matrix depends on the method of segmentation and the corresponding number of regions present in the image. For example, if we segment an image with the volumic extinction values approach and set the parameter value for the number of regions equal 100, then we end up with a 100x2 size feature matrix for each of the two color spaces.

This procedure is repeated for every image in the training set, and all values for each distinct case are stored in the same feature vector. The problem that occurs is a great difference in the values of the features, ranging from big positive values to big negative values. This is visible in the Figure 4.2. In order to avoid arithmetic abnormalities due to these great differences in the values (using as above 100 regions we got values ranging from -150 to 150), we normalize each column of each vector in a range of [0, 1]. Finally, the normalized values are stored in a file, which is used for clustering. The above described procedure takes relatively long time, almost a day, as the number of features per image, but also the available images themselves, is big. For this reason, and because these values need to be accessed and processed later, we store the normalized values also in a MATLAB file.

Clustering

The next step is to perform a clustering and construct our visual vocabulary. The input is the normalized values of the feature vectors mentioned in the last section. A separate clustering is done for each color space and the corresponding cluster centers are our result.

We choose for our clustering method the k-means, which is described in section 2.3.2. We choose to use a k-means implementation, developed at Auton Labs [2]. It offers a

[2] http://www.autonlab.org/autonweb/10377.html

variety of different options about how exactly to run the method, for example using classical k-means or the accelerated version x-means [PM00a, PM00b], the value of k, the number of iterations before splitting, the number of initial clusters, the allowed number of clusters before splitting, and the maximal number of cluster centers. The number of the computed cluster centers varies, between different cases. It depends strongly on the parameters of the clustering tool, but also on the features themselves. The result is different for each color space used and also different for each different number of available regions per image after the over-segmentation.

Classifier Training

The final step of the training process is the training of the classifier(s). For this task the cluster centers computed in the last step are needed. In addition, according to the bag of keypoints algorithm, the actual bag of keypoints is built in this step. These are constructed in the following way: we count the number of occurrences of each cluster center over all images in the training set, regardless of their classes. The numbers of these occurrences form the binned histograms, e.g. the bag of keypoints. Again, in order to avoid great numerical differences, the histograms are normalized in a range of [0, 1], the same way as before. These are the input features for our SVM.

We need to train as many SVMs as the number of our classes in the training set, e.g. one SVM per class which should recognize this class. As we are handling with a multi-class problem, we take the one-against-all approach. This means that the class to be trained is trained against all other classes, which are all handled as no-class as far as the SVM training procedure is concerned. This procedure is repeated for every class in the set.

We choose to use for our SVM the LIBSVM [3], [4], which is an integrated software for support vector classification, regression and distribution estimation. Moreover, it supports multi-class classification. We scale the data, in order to avoid attributes in great numeric ranges, which usually dominate over those in smaller numeric ranges. Linear scaling of each attribute in the range [-1, +1] or [0, 1] is suggested. As mentioned above, we choose the latter one. There are four common kernels implemented in the software, from which we must choose one. The Radial Basis Function (RBF) kernel is suggested from the LIBSVM developers as the first choice, as it gives the best results in most cases. This kernel nonlinearly maps samples into a higher dimensional space so it, unlike the linear kernel, can handle the case when the relation between class labels and attributes is nonlinear [W.09]. In addition, the liear kernel is a special case of the RBF kernel [KJ03] and the sigmoid kernel behaves like RBF for certain Parameters [T.03].The kernel equation can be seen below:

$$K(x, y) = \mathrm{e}^{-|x-y|^2/2\sigma} \tag{4.1}$$

[4] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[4] http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf

While handling with the RBF kernel we must consider two parameters, C and γ, which influence strongly its function. C is referred to as the penalty parameter and γ determines the area of influence the support vector has over the data space. It is not possible to know beforehand which parameter values are suitable for each problem, so some kind of model selection, e.g. parameter search must be done. The goal is to identify good C and γ so that the classifier can accurately predict unknown data. The most common way to do that is to split the training data into two parts, of which one is considered unknown in training the classifier. Then the prediction accuracy on this set can more precisely reflect the performance on classifying unknown data. An improved version of this procedure is cross-validation. In v-fold cross-validation, we first divide the training set into v subsets of equal size. Sequentially one subset is tested using the classifier trained on the remaining $v-1$ subsets. Thus, each instance of the whole training set is predicted once so the cross-validation accuracy is the percentage of data which are correctly classified. The cross-validation has also the advantage that it can prevent the overfitting problem, thus giving better testing accuracy [W.09]. For our case, we used 5-fold cross-validation. After the best pair of C and γ is found for each distinctive case, we use this pair to train the whole data set.

Testing

The testing phase requires fewer steps than the training phase, and those steps required are very similar to the corresponding steps of the training phase. The images of this set also over-segmented, in the exact same way the training set images are. Then, the same color information is extracted from these images, just as in the training phase, and the values, after the same scaling, are again stored in the corresponding files. In this phase only one file per case is used, as the clustering is omitted. The actual testing procedure happens as follows. For every image in the set, the color feature values are extracted as in the training phase. Then, the binned histograms are constructed based on the cluster centers found in the training phase, e.g. on the occurrences of every cluster center in all images in the set. These histograms are fed to every different SVM machine trained in the first phase. As mentioned before, we have as many SVMs as classes in our data base. Then, we compare the output of every machine, and classify the unknown image according to the highest output.

4.3 Summary

In this chapter we described the bag of keypoints algorithm, which is also the algorithm we are using in our approach. Firstly we presented the main idea behind bag of keypoints, and then described in detail every step of the algorithm required. Moreover, a short description of the major issues in the background of the algorithm, such as clustering and k-means, and the the Support Vector Machine classifiers was given. Then we explained how every step of the algorithm is adapted in our object recognition system for images and we described every step and every tool used.

CHAPTER 5

Experiments and Results

5.1 Introduction

In this chapter we present our experiments on our object recognition system and the corresponding results. Firstly, in section 5.2, we describe the data base and its contents which are used in our tests. In section 5.3 we discuss our small probe – experiment on our own figures and the reasons that led to this decision. In section 5.4 we describe our evaluation technique. In section 5.5 we present the experiments and results through various tables and figures, and in section 5.6 we discuss these results.

5.2 The Data Base

We chose to use for our tests the PASCAL VOC 2006 [EGW+10] Database. This data base belongs to the PASCAL Object Recognition Database Collection, which a group of data bases put together by the EU-funded PASCAL Network of Excellence on Pattern Analysis, Statistical Modeling and Computational Learning [1]. The objectives of this collection is to put together a standardized collection of object recognition databases, to provide standardized ground truth object annotations across all databases and to provide a common set of tools for accessing and managing the database annotations [EGW+10]. Moreover, this data base was developed for the PASCAL Visual Objects Classes (VOC) Challenge 2006 [2]. The goal of this challenge is to recognize objects from a number of visual object classes in realistic scenes. By realistic we mean not already pre-segmented objects in these scenes. In other words, it is a supervised learning problem with a defined training set containing labeled images.

In this problem we are dealing with 10 classes' altogether, 9 of them can be separated in two distinctive groups: vehicles and animals. The one left out is the class "person". In

[1] http://www.pascal-network.org

[2] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2006/index.html

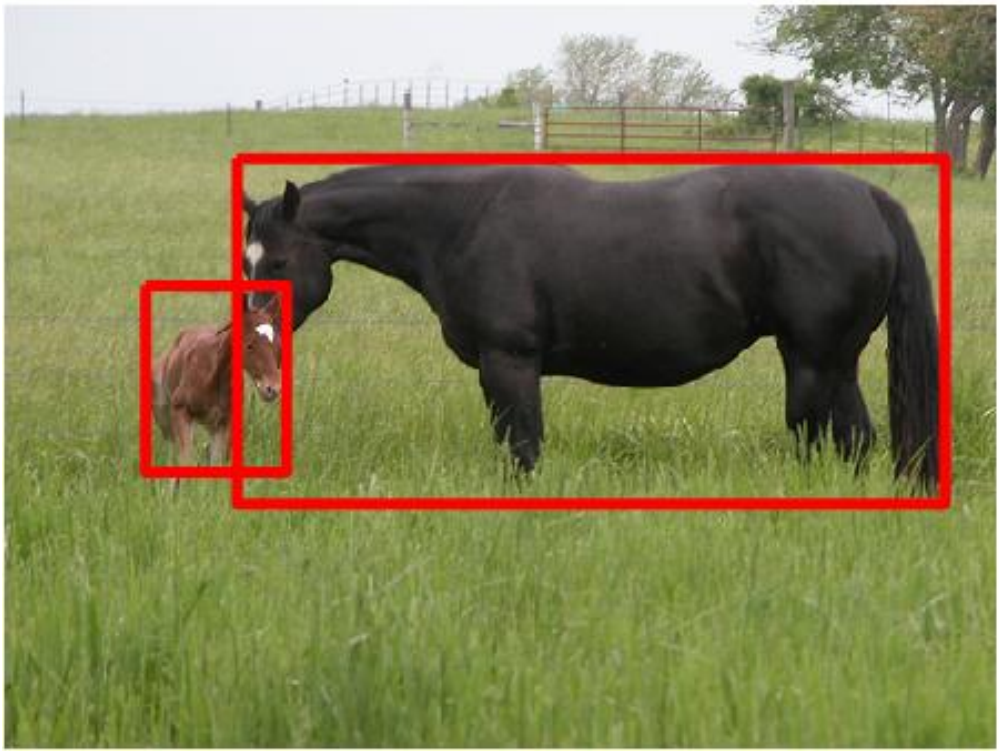


Figure 5.1: Example from the Data Base showing the annotated ground truth on the image. From [3].

the animals group we have the following classes: “cat”, “cow”, “dog”, “horse” and “sheep”. In the vehicles group we have: “bicycle”, “bus”, “car” and “motorbike”. Note that this grouping is used only for brevity’s sake, and doesn’t at all affect our implementation.

The VOC 2006 Database consists of fully annotated 5304 images, having a total number of 9507 labeled objects. This implies that in many images more than one object is present, of one or more classes, which raises also the difficulty of its correct recognition. The ground truth information in every image includes a bounding box around any object of interest, as shown in the Figure 5.1.

The images were collected from personal photographs, “flickr”, and the Microsoft Research Cambridge database for the 2006 VOC challenge. These images can be browsed online here [4], but are also available for download in png format [EGW+10]. The data has been split into two almost equally sized sets, the first for training/validating and the latter for testing. The first set has been also split into two further sets, the “train” set containing the training data and the “val” set, containing suggested validation data, which could also be used as additional training data. The union of these two sets is named as “trainval”, and is also the one used in our method for training. The images used in the challenge were manually selected to remove duplicate images, and very similar images

[3] http://pascallin.ecs.soton.ac.uk/challenges/VOC/images/voc20067a.html

Table 5.1: Summary statistics of the different data base sets. From: [EZWG06].

	train		val		trainval		test	
	Images	Objects	Images	Objects	Images	Objects	Images	Objects
Bicycle	127	161	143	162	270	323	268	326
Bus	93	118	81	117	174	235	180	233
Car	271	427	282	427	553	854	544	854
Cat	192	214	194	215	386	429	388	429
Cow	102	156	104	157	206	313	197	315
Dog	189	211	176	211	365	422	370	423
Horse	129	164	118	162	247	326	254	324
Motorbike	118	138	117	137	235	275	234	174
Person	319	577	347	579	666	1156	675	1153
Sheep	119	211	132	210	251	421	238	422
Total	1277	2377	1341	2377	2618	4754	2686	4753

Table 5.2: Figure's color description.

Figure	color 1	color 2
fig1	black	white
fig2	blue	yellow
fig3	green	yellow
fig4	blue	red
fig5	red	yellow
fig6	red	black

taken from video sequences. Subjective judgement of which objects are "recognizable" was made and images containing annotated objects which were deemed unrecognizable were discarded. The images contain objects at a variety of scales and in varying context. Many images feature the object of interest in a "dominant" position, i.e. in the centre of the image, occupying a large area of the image, and against a fairly uniform background. Table 5.1 summarizes statistics of the above described data sets.

5.3 Testing on our own Test Figures

Due to the huge amount of images present in our data base, we produced some very simple, two colored, figures, in order to test our algorithm. In more detail, we produced 6 simple figures with MS Paint, containing some basic colors, such as blue, red, green etc. Each image looks "divided" in half, which becomes clear in Figure 5.2, and contains only two of the basic colors.

[4] http://pascallin.ecs.soton.ac.uk/challenges/VOC/images/voc20067a.html

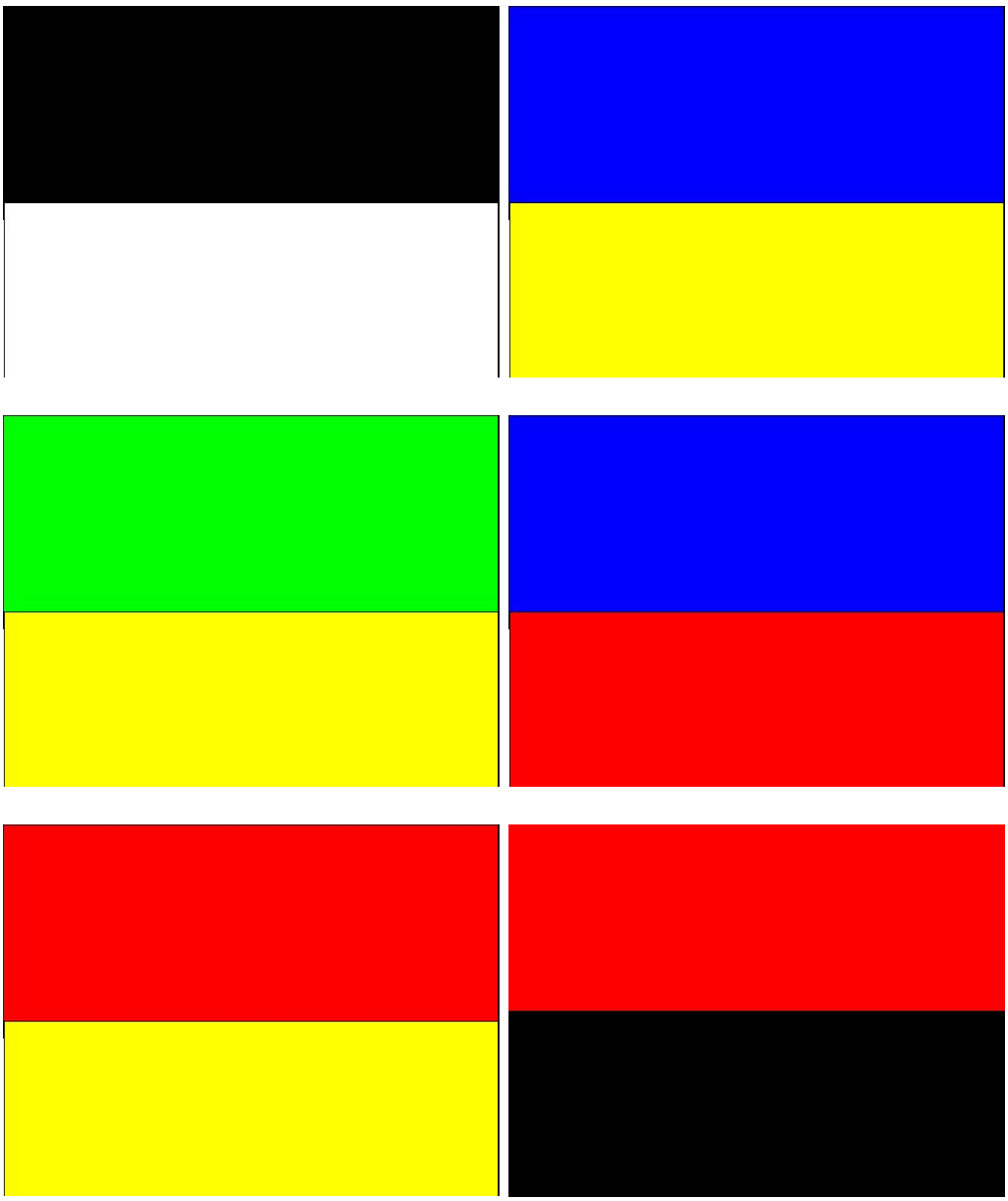


Figure 5.2: The figures mentioned in this section.

Table 5.3: Class assignment and the corresponding color.

#class	color
1	black
2	blue
3	yellow
4	green
5	red

The used colors in our example referred to as "basic" can be seen per image in Table 5.2. From these colors, white is considered as no class, and the other ones are characterized as one class each. So, for our test example, we have the 5 classes visible in Table 5.3. For our test we follow all the steps mentioned in the description of our method. In the beginning we segment each figure, which results to two distinctive regions in each figure. This makes sense, since the segmentation criterion is homogeneity and we have one region for each color present in the image. From these figures we extract the same features described in section 4.2 in the exact same way, for both the opponent and the HSI color spaces. The next step is to perform clustering, using k-means method. The resulting cluster centers were 5. Then, following the bag of keypoints, histograms of the appearances of each cluster center in every figure are built. With these histograms we train our SVMs for each of our 5 classes. Finally, we test the SVMs with the same data used for training. Each color is correctly identified for both cases and for all figures.

Judging form this small test, we can presume that our method works, at least for simple cases like the one used. Every class was correctly identified, for both the opponent and the HSI color spaces.

5.4 Our evaluation technique

The 2006 VOC challenge refers to two different problems, those of classification and detection. The results concerning classification are displayed using the Receiver Operating Characteristic (ROC), or simply ROC curve, and the area under the ROC curve, or AUC. The official results can be browsed online at [5]. A technical report summarizing the challenge and the results can be found in [EZWG06].

In order to understand the aforementioned concept we must firstly introduce the concept of Type I and Type II errors, which are used to describe possible errors made in a statistical decision process. Let us consider a two-class prediction problem, in which the results are labeled either positive (p) or negative (n) class. There are four possible results from a binary classifier. If the result of the classifier is p and the actual value is also p, then this is called a True Positive (TP). But, if the actual value is n, then this is called a False Positive (FP). In the same vein, if the classifier result is n and the actual

[5] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2006/prelimres/index.html

value is also n, this is called a True Negative (TN). Finally, a False Negative (FN) occurs if the classifier result is n but the actual value is p.

Now we can describe what a ROC curve represents. It is a graphical plot of the sensitivity or true positive rate (TPR) against the (1 −specificity) or specificity (SPC) or True Negative Rate. The sensitivity and specificity are calculated with the following equations:

$$TPR = \frac{TP}{T} = \frac{TP}{TP + FN} \tag{5.1}$$

$$SPC = \frac{TN}{N} = \frac{TN}{FP + TN} = 1 - FPR \tag{5.2}$$

TPR determines a classifier or a diagnostic test performance on classifying positive instances correctly among all positive samples available during the test. FPR, on the other hand, defines how many incorrect positive results occur among all negative samples available during the test. A ROC space is defined by FPR and TPR as x and y axes respectively, which depicts relative trade-offs between true positive (benefits) and false positive (costs). The best possible prediction method would yield a point in the upper left corner or coordinate (0, 1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). The (0, 1) point is also called a perfect classification. A completely random guess would give a point along a diagonal line (the so-called line of no-discrimination) from the left bottom to the top right corners. Finally, the AUC is, as the name yields, the area under the curve, and is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. More details, notes and practical considerations about ROC curves can be found in [Faw04, Faw06].

5.5 Results

As already mentioned in section 4.4, we use the volumic extinction and waterfall segmentation methods for our experiments. That means that we have altogether 4 different feature vectors, as for each segmentation case we are using both the opponent and the HSI color spaces. Moreover, for each distinct case we experimented with different clustering, e.g. with different amount of cluster centres. This is possible, as in the clustering tool we are using we can give as input option the desired number of resulting cluster centres. In detail, we distinguished three cases: 500, 200 and 24 centres. These values come from the clustering tool. There is an option to give as input the desired number of clusters at the beginning of the clustering procedure. If we don't give any, the tool starts with a big cluster containing all the data and stops at the optimum number of cluster centres. The outcome of this case is 24 clusters. The other two values are given as the initial number of k, which means that the clustering starts with the corresponding number of clustering centres.

Using these three different clustering cases, we came upon the following results. Firstly, in the volumic extinction case, the HSI color space produced better results. In

Table 5.4: AUC values for the volumic extinction / HSI case.

HSI	HSI 24	HSI 200	HSI 500
bicycle	**0.6396**	0.6018	0.5992
bus	0.7034	**0.7066**	0.6487
car	**0.7209**	0.6817	0.6681
motorbike	**0.6284**	0.5284	0.5285
cat	0.6049	0.5897	**0.6124**
cow	0.7572	**0.7682**	0.7578
dog	**0.6052**	0.6016	0.5830
horse	0.5755	**0.5829**	0.5130
sheep	**0.7496**	0.7389	0.7435
person	**0.6165**	0.5668	0.5786

Table 5.4 we can see the AUC values for each clustering case and every class. The best results come overall from the clustering case with the smallest amount of cluster centres. In addition, we can see the ROC curves for every class and for the first clustering case in Figure 5.3.

If we compare our two best curves (cow and sheep) with the corresponding least good curves of the official results from the 2006 VOC challenge [6] –those from MUL_1vALL or Siena –we observe that our curves are at the beginning steeper than those two, but then this steepness drops significantly, thus the little lower AUC value:

- 0.7496 in our test against 0.768 of Sienna and 0.758 of MUL_1vALL for the sheep class.
- 0.7572 in our test for the cow class against 0.774 of Siena and 0.632 of MUL_1vALL, which is significantly lower than our result.

Overall, our results are comparable to the corresponding least good ROC curves of [7].

Moving on to the opponent color space, in Table 5.5 we can see the same results for this space. In general, they are worse than the corresponding HSI values. Here we encounter also our first AUC values below the threshold of 0.5, meaning that the classifier for this case is not realistic, as the theory in [Faw04] yields. Even in the clustering case with the least cluster centres, which produces the best results overall, we encounter for the first class (bicycle) a rather small, below 0.5, AUC value. The same class has also values below 0.5 for the other two cases, but in the last one we get another three unacceptable values for the dog and horse classes.

We can see the ROC curves for all the classes for the first and best clustering case in the Figure 5.4. Overall, the curves are worse than the corresponding curves for the

[6] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2006/prelimres/index.html

[7] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2006/prelimres/index.html

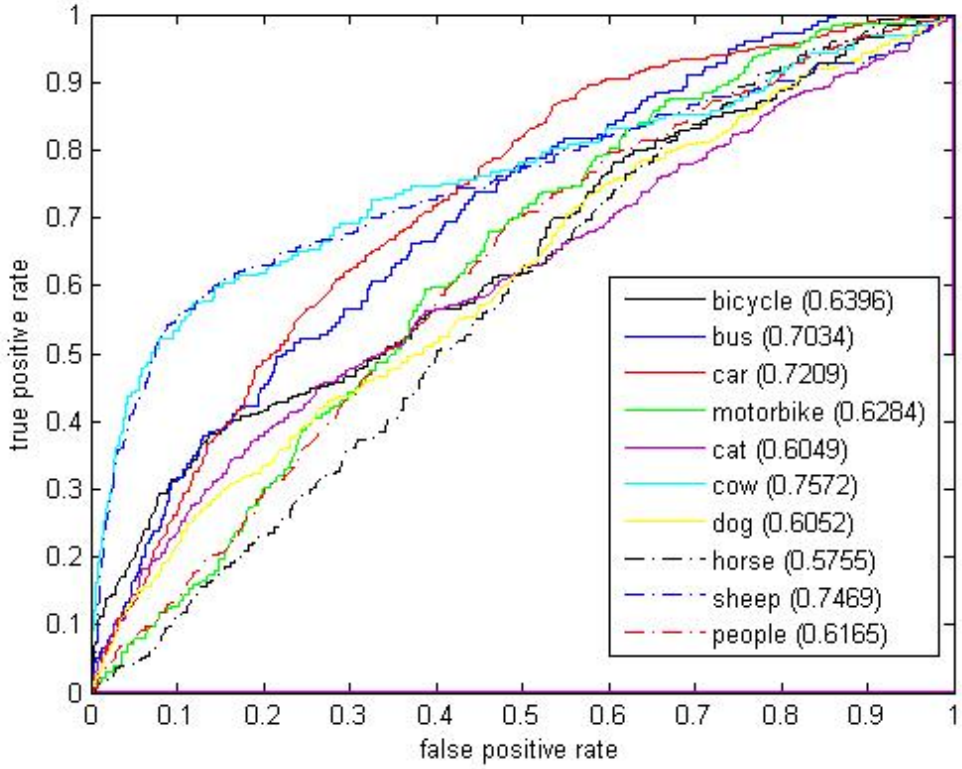


Figure 5.3: The ROC curves for the HSI case, 100 regions per image and the first clustering case.

HSI case. The unacceptable (meaning a value below 0.5) low AUC value for the bicycle class is also visible from its curve, which runs below the (0,0) –(1,1) line. We have also a couple of curves, such as those for the class cat and horse, just above this threshold. The rest run above it, and some, such as the curves for the class bus, sheep and cow, have an AUC value over 0.7. Again the cow class tops the others, with a value of 0.7332, with the bus class coming second. If we compare these two values with the corresponding values from the official results, we have:

- 0.7332 in our test for the cow class, whilst 0.756 for AP06_Batra and 0.632 for MUL_1vALL, which is significantly lower than our result.
- 0.7254 in our test for the bus class, whilst 0.749 for Siena and 0.637 for AP06_Batra, which again is lower than our value.

By moving from volumic extinction to waterfall segmentation, the results deteriorate altogether slightly. Most AUC values drop by a ~0.04 factor, especially in the 200 and 500 cases. This also means that many values are closer to the 0.5 threshold than the HSI 100 case. Still, they remain above this threshold, and again the cow and sheep classes

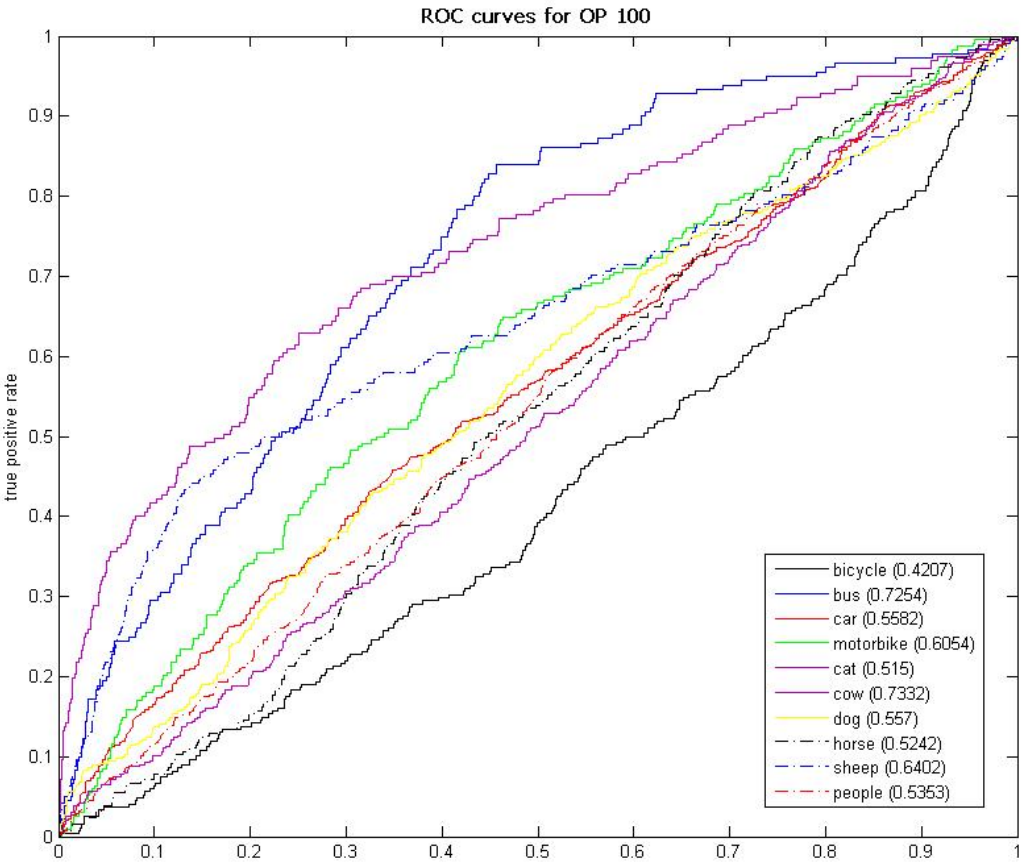


Figure 5.4: The ROC curves for the Opponent case and, 100 regions per image and the first clustering case.

Table 5.5: AUC values for the volumic extinction / Opponent case.

OP	OP 24	OP 200	OP 500
bicycle	0.4207	0.4734	**0.4916**
bus	**0.7254**	0.7166	0.7020
car	**0.5582**	0.5541	0.4885
motorbike	**0.6054**	0.5696	0.5801
cat	0.5150	0.5129	**0.5735**
cow	0.7332	**0.7406**	0.6927
dog	**0.5570**	0.5428	0.4927
horse	**0.5242**	0.5145	0.4986
sheep	0.6402	0.5974	**0.6861**
person	0.5353	**0.5667**	0.5312

Table 5.6: AUC values for the Waterfall / HSI case.

HSI WF	HSI WF 24	HSI WF 200	HSI WF 500
bicycle	**0.6206**	0.5655	0.5485
bus	**0.6761**	0.6595	0.6366
car	**0.6582**	0.6538	0.6509
motorbike	**0.6121**	0.5695	0.5865
cat	**0.6182**	0.6015	0.6108
cow	**0.7502**	0.7066	0.716
dog	**0.6168**	0.5743	0.5674
horse	**0.6044**	0.5547	0.5619
sheep	**0.7143**	0.7126	0.6987
person	**0.5817**	0.5794	0.5692

Table 5.7: AUC values for the Waterfall / Opponent case.

OP WF	OP WF 24	OP WF 200	OP WF 500
bicycle	0.4653	**0.5247**	0.4615
bus	**0.6632**	0.652	0.6402
car	**0.5504**	0.4579	0.5423
motorbike	0.5633	**0.5987**	0.5497
cat	**0.6113**	0.4886	0.5014
cow	0.6604	0.6332	**0.7455**
dog	0.5187	**0.5456**	0.489
horse	0.4427	0.5065	**0.5264**
sheep	0.3684	**0.5185**	0.5085
person	**0.5859**	0.5429	0.5701

have results close to 0.7, which are comparable, and even better in the case of cow class, with the least good official results of [Har78]. In Table 5.6 we can see the whole results for the HSI color space, and its distinctive three clustering cases.

Similar to those above are also the corresponding results for the waterfall and the opponent color space, for all three clustering cases. They can be seen in Table 5.7 and are worse than those of HSI for the waterfall segmentation, and again, there appear to be some AUC values below 0.5. There appear to be some minor differences between these values and the corresponding OP values from the volumic extinction segmentation values. In more details, some low values of the latter case are slightly better in the first case, and vice versa, some high values of the latter case are slightly worse in the first case. For another time the clustering case with the smaller amount of cluster centres produces the better total results, and again, the cow class tops the other ones. Moreover, all three values are better from the least good value of the official results [Har78], that being 0.632 of MUL_1vALL.

5.6 Discussion

A general conclusion about our experiments and results is that our AUC values, although most of them are acceptable, they are worse than those of the official results of [Har78]. Their best AUC values are up to 0.9, whilst our best results are at 0.7. The reason for this is that we haven't based our method on interest points but only on color. With interest points we have the ability to find important geometrical structures in the image, which can be used to successfully describe an object of interest and recognize it at another instance. Color cannot hold any geometry of an important object in the image. Instead we base our method on colors present in the image. After the performed over-segmentation the image is divided in many regions and from them we get our color features, both from objects of interest and from the background, without knowing which region belongs where.

Moving to specific classes, our two best classes - from the four animal classes but also in general – are the classes cow and sheep. Their AUC values are for all the tested HSI cases over 0.7, which is a very good result. For the opponent case and the volumic extinction case the AUC value for the cow class remains above 0.7 and for the sheep class is over 0.6. For the waterfall case, the cow class values remains above 0.6 and for the sheep we have two values just above 0.5 and one below. These values are explained if we firstly think that the colors usually seen on these two animals are few, e.g. cows are mostly black, white and brown or a mixture of these colors and sheep's are either white or black. Secondly, the images in the data base of these two classes are outdoor images, and the animals are mostly found on grass fields. This means that the colors for the background and our objects of interest are distinguishable. Such different colors can be easily modelled in the system and yield an adequate recognition.

The rest three animal classes have average results, between 0.5 and 0.6, and in some cases for the opponent color space below 0.5. These classes, especially the dog and cat ones, are a more difficult case for a recognition, as they have many different colors and not one or two distinctive ones, as a sheep or a cow. Moreover, the available images from the data base for these classes are mostly indoor images, which yields more complex images as far as color is concerned, with many more irrelevant objects present in the image, e.g. objects not from our ten classes, and different background. All these have many different colors that make the recognition difficult.

From the four vehicle classes, we get the best results from the bus and car classes, with values from 0.6 to 0.7 for the first one, and from 0.5 to 0.7 for the latter one. The better results are again from the HSI case. Although these objects can have many different colors, there is another helpful object in most of the images from the data base with these two objects, namely roads. This object takes up a large part of the image and has always the same color, grey. This helps the positive recognition a lot. Roads are also present in images containing the other two vehicle objects, bicycles and motorbikes. For these two classes the values are average, ranging from 0.6 to 0.5, and in the opponent color space the bicycle values go also below 0.5. In this case the problem is the actual object, and more specifically their structure. Bikes and motorbikes have many discontinuities

in their structure with many holes between the wheels. This provides us with low color information, which makes the recognition difficult.

Finally, the people class is the most complicated. The results here are mediocre, having values around 0.55 and only once above 0.6. People can have any color, due to their different clothes. The only constant color is that of skin, as long as it concerns the same race, and the portion of the object where skin is visible is very small. Moreover, in most of the images available, people are present with another object from our classes. This makes the recognition even more difficult.

CHAPTER 6

Summary and Future Work

6.1 Introduction

In this last chapter we give a summary of the thesis in section 6.2 and discuss our contribution and our results, in relation also with the methods used in computer vision. Following these thoughts, we present in section 6.3 possible future ideas for the improvement of our results and possible uses for our prototype system.

6.2 Summary and Discussion

The main topic of this thesis is object recognition. Following the burst of technology in everyday use, several applications in different science fields will benefit from an efficient image management.

We firstly presented the bag of keypoints algorithm, which is used for our approach, and described in detail its distinctive steps. We then moved to interest points, giving their definition and describing their basic characteristics and goals. Afterwards, we presented numerous interest points detectors, beginning with the corner detectors, and ending with some recent detectors. We then gave an overview of modern day interest point descriptors, followed by certain advantages and drawbacks of every method. Moreover, a small overview over specific color spaces was given.

Another major topic of computer vision is the segmentation of an image, with various useful applications. There exist numerous segmentation implementations methods, but the most common ones are edge- and region-based methods. The watershed approach combines these methods and gains from their advantages. The basic watershed methods are described in detail. Taking all this into consideration, we developed a software prototype system for object recognition in images. Our approach uses the bag of keypoints algorithm, which is based on histograms counting the occurrences of every cluster center in every image. Our method makes also use of over-segmentation and

certain color spaces, in order to achieve adequate results. Various and different results are carried out, with the use of a well known data base, in order to test our method. Our results, when compared with the official results from different organizations carried out on the same data base, are not that good but in most cases acceptable and in some cases even better.

6.3 Future Work

Just as we used only color information for our features, we could as easily use another image characteristic in order to extract our features, such as texture. As it is still a field under heavy research, there exist various texture describing algorithms. So, it would be of great interest, if we could include also texture information in our features, and to which extent that would help our experiments.

List of Figures

List of Tables

Bibliography

[AS03a] J. Angulo and J. Serra. Color segmentation by ordered mergings. In *IEEE International Conference on Image Processing, Barcelona, Spain*. September 2003.

[AS03b] J. Angulo and J. Serra. *Mathematical Morphology in Colour Spaces Applied to the Analysis of Cartographic Images*. In: Proceedings of International Workshop Semantic Processing of Spatial Data (GEOPRO'03), IPN-Mexico Ed, 2003.

[Bau00] A. Baumberg. Reliable feature matching across widely separated views. In *Proceedings of IEEE Conference in Computer Vision and Pattern Recognition*, pages 774–781. 2000.

[BB94] S. Beucher and M. Bilodeau. Road segmentation and obstacle detection by a fast watershed algorithm. In *Proceedings of IEEE Intelligent Vehicles Symposium*, pages 296–301. October 1994.

[BBY90] S. Beucher, M. Bilodeau, and X. Yu. Road segmentation by watersheds algorithms. In *Proceedings of the Pro-art vision group PROMETHEUS workshop, Sophia-Antipolis, France*. April 1990.

[Beu91] S Beucher. The watershed transform applied to image segmentation. In *Proceedings of the Pfefferkorn Conference on Signal and Image Processing in Microscopy and Microanalysis*, pages 299–314. September 1991.

[Beu94] S. Beucher. Watershed, hierarchical segmentation and waterfall algorithm. In *Mathematical Morphology and its Applications to Image Processing, Proceedings of ISMM*, pages 69–76. 1994.

[BFdW01] T. Brox, D. Farin, and P. H. N. de With. Multi-stage region merging for image segmentation. In *Proceedings of the 22nd Symposium on Information and Communication Theory in the Benelux Enschede, The Netherlands*, pages 189–196. May 2001.

[BJ85] P. Besl and R. Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, 1985.

[BL79] S. Beucher and C. Lantuejoul. Use of watersheds in contour detection. In *International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation, Rennes, France, Sept*, pages 17–21. 1979.

[BMP02] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans*, 24:509–522, 2002.

[BS02] S. Borer and S. S'usstrunk. Opponent color space motivated by retinal processing. In *Proceedings of IST Conference on Color in Graphics, Imaging and Vision (CGIV)*, volume 1, pages 187–189. 2002.

[BTvG06] H. Bay, T. Tuytelaars, and L. van Gool. Surf: Speeded up robust features. In *Proceedings of the 9th European Conference on Computer Vision*, pages 404–417. May 2006.

[Bur98] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2, 1998.

[CD95] G. W. Cook and E. J. Delp. Multiresolution sequential edge linking. In *Proceedings of the IEEE International Conference on Image Processing, Washington DC.*, pages 41–44. 1995.

[CDF+04] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *International Workshop on Statistical Learning in Computer Vision, ECCV*. 2004.

[Cri01] N. Cristianini. *Support Vector and Kernel Machines.* ICML, 2001.

[CWB00] W. Kropatsch C. Wolf, J.–M. Jolion and H. Bischof. Jolion. *W. Kropatsch and H. Bischof. Content based image retrieval using interest points and texture features. In Proceedings of the ICPR2000, Vol*, 4:234–237, September 2000.

[CYV00] S. G. Chang, B. Yu, and M. Vetterli. *: Spatially adaptive wavelet thresholding with context modeling for image denoising*, volume 9. IEEE Transactions on Image Processing, 2000.

[DBC06] S. Delest, R. Boné, and H. Cardot. *3D Watershed Transformation on Connected Vertices Structure.* CompIMAGE '06: Computational Modelling of Objects Represented in Images: fundamentals, methods and applications, October 2006.

[DEJ06] Q. Du, M. Emelianenko, and L. Ju. Convergence of the lloyd algorithm for computing centroidal voronoi tesselations. *SIAM J. Numer. Anal.*, 44:102–119, 2006.

[D'O96] S. D'O. Cotton. Colour, colour spaces and the human visual system, School of Computer Science, University of Birmingham, England, Technical Report, B15-2TT, May 1996.

[EGW+10] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. In *International Journal of Computer Vision (IJCV)*. 2010.

[EZWG06] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool. *The 2006 PASCAL Visual Object Classes Challenge (VOC2006) Results*. 2006.

[Fai98] M. D. Fairchild. *Color Appearance Models, Addison Wesley, Reading, MA*. 1998.

[Faw04] T. Fawcett. Roc graphs: Notes and practical considerations for researchers. Technical report, Palo Alto, USA, 2004.

[Faw06] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27, 2006.

[FBW01] P. Felkel, M. Bruckschwaiger, and R. Wegenkittl. Implementation and complexity of the watershed-from-markers algorithm computed as a minimal cost forest. *Computer Graphics Forum Journal*, 20(3), 2001.

[FR98] A. Ford and A. Roberts. Colour space conversion. Technical report, Westminster University, London, 1998.

[FR07] F. C. Flores and R.d. A. *Lotufo. Watershed from Propagated Markers improved by a Marker Binding Heuristic. In Proccedings of the 8th International Symposium on Mathematical Morphology*, 1:313–323, 2007.

[FSC98] G. D. Finlayson, B. Schiele, and J. L. Crowley. Comprehensive colour image normalization. In *ECCV 98: Proceedings of the 5th European Conference on Computer Vision*, volume 1, pages 475–490. 1998.

[Han08] A. Hanbury. *Image Segmentation by Region Based and Watershed Algorithms*. Wiley Encyclopedia of Computer Science and Engineering, 2008.

[Har78] F. J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. In *Proceedings of the IEEE*. 1978.

[HS88] C. Harris and M. Stevens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Visual Conference, UK*. 1988.

[J.04] J. *v.* d. Weijer, 2004.

[J.06] J. *v.* d, 2006.

[J.B01] J.B. T. *M. Roerdink and A. Meijster. The Watershed Transform: Definitions, Algorithms and Parallelization Strategies. Fundamenta Informaticae*, 41:187–228, 2001.

[Joa98] T. Joachims. *Text categorization with support vector machines: Learning with Many relevant features, ECML.* 1998.

[J.v05] J.v. d. *Weijer, T Gevers, and J. M. Geusebroek. Edge and Corner Detection by Photometric Quasi-Invariants. IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(4):625–630, 2005.

[KA06] K. Karantzalos and D. Argialas. Improving edge detection and watershed segmentation with anisotropic diffusion and morphological levellings. *In: International Journal of Remote Sensing*, 2006.

[KG08] D. Kelkar and S. Gupta. Improved quadtree method for split merge image segmentation. In *Proceedings of the First international Conference on Emerging Trends in Engineering and Technology*, pages 44–47. 2008.

[KJ03] S. S. Keerthi and C. J. Lin. *Asymptotic behaviors of support vector machines with Gaussian kernel. Neural Computation*, 15(7):1667–1689, 2003.

[KS04] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Proceedings of IEEE Conference in Computer Vision and Pattern Recognition.* 2004.

[KTM85] . K. T. Mullen. The contrast sensitivity of human color vision to red-green and blue-yellow chromatic gratings,. *J*, 359:381–400, 1985.

[KvD87] J. Koenderink and J. van Doorn. Representation of local geometry in the visual system. In *Biological Cybernetics*, volume 55, pages 367–375. 1987.

[LD05] R. Lerallut and F. Meyer É. Decencière. Image filtering using morphological amoebas. *ISMM Image and Vision Computing*, 25(4):395–404, 2005.

[Lin94] T. Lindenberg. *Scale-Space Theory in Computer Vision.* Kluwer Academic Publishers, 1994.

[Llo82] S. Lloyd. *Least squares quantization in PCM.* IEEE Transactions on Information Theory, 28(2):129-137, 1982.

[LNHT94] J. Lorenzo-Navarroa and M. Hernndez-Tejeraa. Image segmentation using a modified split and merge technique. *Cybernetics and Systems Journal*, January 1994.

[Low99] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the 7th International Joint Conference on Computer Vision.* 1999.

[Low04] D. Lowe. Distinct image features from scale-invariant keypoints. In *International Journal for Computer Vision*, volume 60, pages 91–110. 2004.

[LP05] F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 524–531. 2005.

[LS02] R. Lotufo and W. Silva. *Minimal Set of Markers for the Watershed Transform*. In: Proceedings of ISMM, 2002.

[LSP03] S. Lazebnik, C. Schmid, and J. Ponce. *Sparse texture representation using affine- Invariant neighborhoods*. 2003.

[LSTCW01] H. Lodhi, J. Shawe-Taylor, N. Christianini, and C. Watkins. *Text classification Using string kernels*. NIPS (In Advances in Neural Information Processing Systems), Vol 13, 2001.

[Mar05] B. Marcotegui. Segmentator user manual. *Centre de Morphologie Mathmatique. March*, 21, 2005.

[MB05] B. Marcotegui and S. Beucher. Fast implementation of waterfall based on graphs. In *Mathematical Morphology and its Applications to Image Processing, Proceedings of ISMM*, pages 177–186. 2005.

[MBM90] F. Meyer, S. Beucher, and B. Marcotegui. Morphological segmentation. *Journal of Visual Communication and Image Representation*, 1(1):21–46, September 1990.

[MCUP02] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*. 2002.

[Mey01] F. Meyer. An overview of morphological segmentation. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(7):1089–1118, 2001.

[Mey04] F. Meyer. Levelings. *Image Simplification Filters for Segmentation. Journal of Mathematical Imaging and Vision*, 20:59–72, 2004.

[MJ97] A. Mehnert and P. Jackwaya. An improved seeded region growing algorithm. *Pattern Recognition Letters*, 18(10):1065–1071, October 1997.

[Mor80] H. Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. Tech, 1980.

[MS01] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada*, pages 525–531. 2001.

[MS02] K. Mikolajczyk and C. Schmid. *An affine invariant interest point detector.* 2002.

[MS04] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. In *International Journal for Computer Vision*, volume 60, pages 63–86. 2004.

[MS05] K. Mikolajczyk and C. Schmid. A performance valuation of local descriptors. In *IEEE Trans*, pages 43–72. 2005.

[OPR78] R. Ohlander, K. Price, and D. R. Reddy. Picture segmentation using a recursive region splitting method. *Computer Graphics and Image Processing*, 8:313–333, 1978.

[Pav77] T. Pavlidis. *Structural Pattern Recognition.* Springer-Verlag, Berlin-Heidelberg- New York, 1977.

[PL90] T. Pavlidis and Y. T. Liow. Integrating region growing and edge detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 12. March 1990.

[PM00a] D. Pelleg and A. Moore. *Accelerating Exact k-means Algorithms with Geometric Reasoning (Technical Report CMU-CS-00-105).* Carnegie Mellon University, Pittsburgh, PA, 2000.

[PM00b] D. Pelleg and A. Moore. Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734. 2000.

[Por02] F. Porikli. Automatic threshold determination of centroid-linkage region growing by MPEG-7 dominant color descriptors. In *Proceedings of International Conference on Image Processing*, volume 1, pages 793–796. 2002.

[Poy95] C. Poynton. *A Guided Tour of Colour Space, New Foundations for Video Technology (Proceedings of the SMTPE Advanced Television and Electronic Imaging Conference), pages 167-180.* 1995.

[Poy97] C. Poynton. Frequently Asked Questions about Color. Technical report, Simon Fraser University, 1997.

[PP93] N. R. Pal and S. K. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277, 1993.

[Que07] P. Quelhas. *Scene image classification and segmentation with quantized local descriptors and latent aspect modeling.* PhD Thesis, École Polytechnique Fédérale de Lausanne, 2007.

[SAS02] P. Soille, Morphological Image Analysis, and Springer. *second edition.* 2002.

[SC00] T. Szirányi and L. Czúni. Spatio-temporal segmentation with edge relaxation and optimization using fully parallel methods. *15th International Conference on Pattern Recognition (ICPR'00)*, 4:4820, 2000.

[See02] T. Seemann. *Digital Image Processing using Local Segmentation.* PhD Thesis, Faculty of Information Technology, Monash University, Australia, 2002.

[SG86] M. J. Sabin and R. M. Gray. *Global convergence and empirical consistency of the generalized Lloyd algorithm, IEEE Transactions on Information Theory, 32(2): 148-155.* March 1986.

[SGDvdW06] N. Sebe, T. Gevers, S. Dijkstra, and J. van de Weijer. Evaluation of Intensity and Color Corner Detectors for Affine invariant Salient Regions. In *Beyond Patches Workshop, in conjunction with CVPR, New York, USA.* 2006.

[SK87] L. Sirovich and M. Kirby. A low-dimensional procedure for the characterization of human faces. In *the Journal of the Optical Society of America*, volume 4, page 519– 524. 1987.

[SS02] L. G. Shapiro and G. C. Stockman. *Computer Vision.* Prentice Hall, 2002.

[Stő07] J. Stőttinger. *Local Color Features for Image Retrieval.* Master Thesis, Technical University of Vienna, 2007.

[SW99] D. Swets and J. Weng. Hierarchical discriminant analysis for image retrieval. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(5):386–401.* 1999.

[SW06] F. W. M. Stentiford and M. D. Walker. Attention-based color correction. In *SPIE Human Vision and Electronic Imaging*, volume 6057, pages 58–167. January 2006.

[SZ02] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or how do i organize my holiday maps? In *Proceedings of 7th European Conference on Computer Vision*, volume 1, pages 414–431. 2002.

[T.03] H. T. *Lin and C.* -J. Lin, 2003.

[Th99] Th. *Leung and J.* 1999.

[TK00] S. Tong and D. Koller. *Support vector machine active learning with applications to text classification.* ICML, 2000.

[TP91] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1), 1991.

[TT03] M. Tkalcic and J. F. Tasic. Colour Spaces - perceptual historical and applicational Background. In *Proceedings of The IEEE Region 8 Conference on Computer as a Tool*, pages 304–308. 2003.

[Vap98] V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[VS91] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[VZ02] M. Varma and A. Zisserman. *Classifying materials from images: to cluster or not to cluster?* 2002.

[W.09] C. W. *Hsu.* C. -C, 2009.

[Wit83] A. P. Witkin. Scale-space filtering. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence.* 1983.

[WW00] G. W and W. S. Stiles Wyszecki. Colour science concepts and methods. In *Quantitative Data and Formulae, John Wiley and Sons, Inc.* 2000.

[ZRZ02] L. Zhu, A. Rao, and A. Zhang. *Theory of Keyblock-based image retrieval, ACM Transactions on Information Systems, 20(2): 224-257.* 2002.

[Zuc76] S. W. Zucker. Region growing: Childhood and adolescence. *Computer Graphics And Image Processing*, 5:382–399, 1976.