Technische Universität Berlin
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet CV
Prof. Dr.-Ing. Olaf Hellwich

Master Thesis

UNSUPERVISED DETECTION OF SALIENT REGIONS IN IMAGE DATABASES

Andreas Langenhagen

July 2015

Referee:
Prof. Dr.-Ing. Olaf Hellwich
Dr. Ronny Hänsch

Langenhagen
315720
Informatik M. Sc.
barn07@web.de

ERKLÄRUNG DER URHEBERSCHAFT

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Andreas Langenhagen Berlin, den 17.07.2015

ZUSAMMENFASSUNG

Die große Menge an digitalen Bilddaten, welche seit dem Beginn des Informationszeitalters stetig wächst, bringt die Herausforderung mit sich, diese große Informationsmenge auszuwerten. Aufgrund der Fülle an Informationen ist diese Aufgabe für einen Menschen mit einem hohem und zuweilen untragbaren Zeit- und Arbeitsaufwand verbunden. Im Zuge dieser Masterarbeit wird ein Rahmenwerk vorgestellt, mit dessen Hilfe Bilder aus großen Datenbanken automatisch auf interessante Bildregionen hin analysiert und nach diesen sortiert werden können. Auch das Erkennen von hervorstechenden Bildern wird ermöglicht. Das Rahmenwerk gestattet die einfache Umsetzung, Anwendung und Evaluierung von Detektoren herausstechender Bildregionen, von Bild-Beschreibungsformaten und von Clustering-Algorithmen, welche alle im Zusammenspiel große Mengen von Bildern sinnvoll gruppieren können. Beispielhaft umgesetzte Implementierungen nutzen den *SaliencyFilters* Algorithmus, einfache Bild-Beschreibungsformate für Form und Farbe sowie Clustering mit Hilfe von k-Means, OPTICS und eines einfachen Ausreißer-Erkennungs-Algorithmus. Die diversen Algorithmen und Datenstrukturen werden in mehreren Tests an der Caltech-256 Bilddatenbank evaluiert. Die Tests zeigen, dass die Qualität der hier erzeugten Partitionierungen die menschliche Klassifizierung nicht ersetzen kann, jedoch können viel versprechende Ansätze für Weiterentwicklungen erkannt werden.

ABSTRACT

The huge amount of digital images which grows constantly since the dawn of the information age causes also the challenge to evaluate those huge amounts of data. Because of the big quantities of information, such task is – for a human – connected with a high investment of time and effort. In the course of this master thesis, a framework is presented that can analyze and partition images from big databases with regard to image content that is considered interesting. The framework enables uncomplicated implementation, application and evaluation of salient region detectors, image descriptors and clustering algorithms. Those algorithms and data structures can interact in order to partition huge amounts of images according to their contents. Exemplary implementations use the *SaliencyFilters* algorithm, simple image descriptor formats for shape and color and also the clustering algorithms k-Means, OPTICS as well as a simple outlier detection algorithm. The various algorithms and data structures are evaluated in several tests on the Caltech-256 image database. The tests show that the quality of the generated partitionings cannot keep up with manual classification, but motivate for future advancements through promising results.

TABLE OF CONTENTS

1 INTRODUCTION

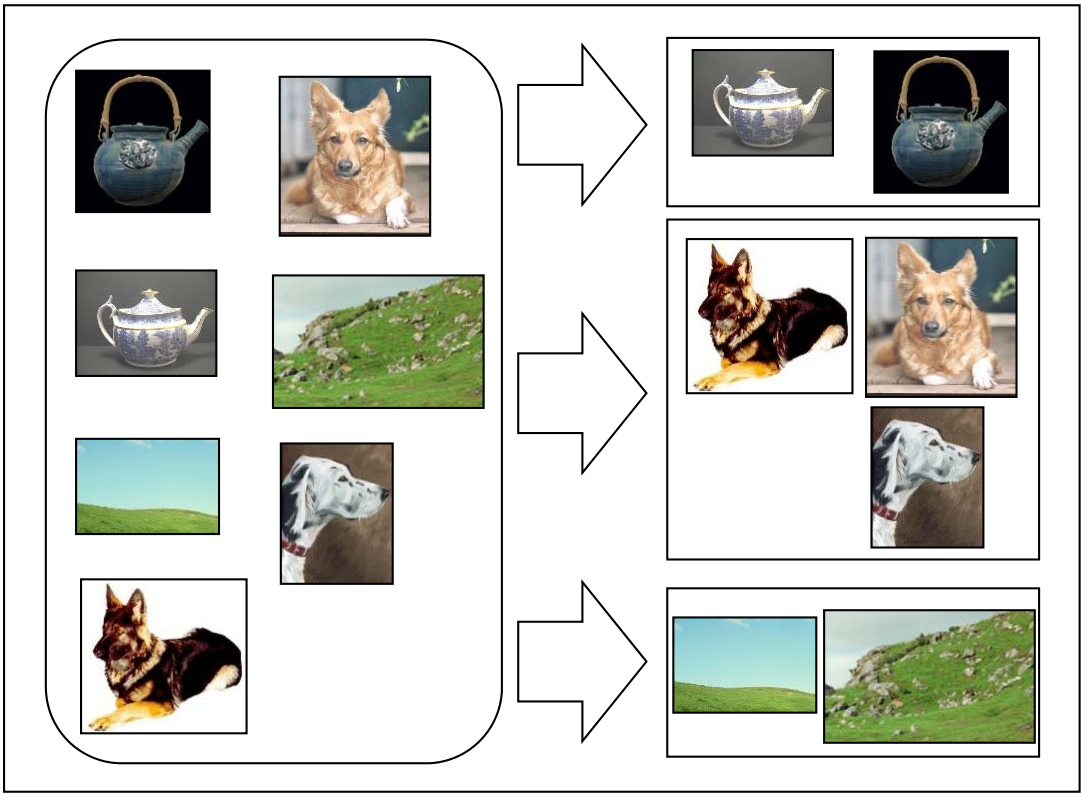
Since the beginning of the age of information, a sheer flood of information that could easily overwhelm the people is available to almost anyone. Sensors in a still widening scope of electronic devices provide data to anyone who is eager to use them, and not many years ago, the term “big data” was coined. Industry, medicine, science and military started to handle enormous amounts of data. The rigid form of digital data, as well as the usual representation modalities make it hard for humans to cognitively process bigger quantities of data. Since then people are engaged with structuring, organizing and filtering information. The flood of information also embodies in the form of photographs and other images, like for instance in series of photographs taken by an observation airplane on a search and rescue mission. Hundreds of thousands of images have to be examined eventually find what is searched for. An automatic examination could be useful in many cases. Given a set of images, a software could automatically find a subset in which the image-content of interest differs in some sense from that of the rest of the images. The task this thesis is devoted to is to automatically find conspicuous pictures in an image database. In this work, this is done by the detection, localization and classification of salient objects (Image 1). The work is unsupervised, i.e. the solution bears no learning stage. The work is further not related to any specific image content or category. The problem is highly general and can be mapped to many real world problems. Consider, as an example, the visual detection and localization of flying objects. Given a large database of images depicting the sky, some of the images may contain objects like certain clouds, airplanes or helicopters, the most may not. Searching the database for objects by hand can be tedious and time consuming. Instead, letting a machine locate objects of interest may be faster and could prove very helpful. Thinking one step further, it would also be nice to group the different objects according to their type, e.g. fighter jets to fighter jets and civil aircrafts to civil aircrafts. Such content-driven differentiation of images can really be helpful, for instance when it comes to visual failure detection in industrial component production, the extensive search of wrecked airplane parts in oceanographic satellite photos or astronomical image evaluation. Automatic detection of visual oddities can also play a big role in medical image analysis when it comes to automatic detection of damaged or unhealthy body parts, like, for instance, bones or connective tissues. Possible use cases are plenty.

This thesis presents an attempt to bring some structure in an unordered set of images by examining the image contents. This document contributes a software framework solution that translates images in a database into a comparable format and then uses a clustering algorithm to order the images according to their interesting image contents. The framework enables the batch processing of large amounts of images with a saliency detection algorithm and allows to transfer image features into a comparable descriptor format. The framework further provides a tool for clustering the images according to their generated descriptors and supplies an application to check the clustering quality. Thanks to the framework, new saliency detection algorithms, new image descriptors and image clustering algorithms can be implemented, applied and evaluated in short time, while the file handling, the bulk processing and error handling is left to the framework itself.

To identify salient regions, the *Saliency Filters* saliency detection algorithm by Perazzi et al. [1] is used. The provided solution intends to serve as a support to manual grouping and exception detection by providing a modular, dynamic framework for developers and users. The output of the work aims to be concise and informative, yet attempts to give best opportunities to be interoperable with other systems. Beyond that, the software that is implemented in the course of this thesis attempts to be reasonably fast, easy to use, maintain

and extend. It is configurable after compile-time. Additionally, two small evaluation applications which put the results of the software into evaluable metrics, are provided.

The thesis is structured as follows: Chapter 2 gives a broad breakdown of the big challenges that arise when dealing with detection, localization and clustering of interesting image parts. The following chapter also gives first thoughts about the solutions that are applied in this work to take up with the challenges. The saliency algorithm that is used is a critical part of the whole solution. Chapter 3 summarizes the state of the art concerning saliency detection algorithms and talks about unsupervised object categorization methods. The processing chain of the whole saliency detection and clustering approach is presented in Chapter 4. The *Saliency Filters* algorithm that is finally put to use for saliency detection is explained in more detail in Chapter 5. Chapter 6 and Chapter 7 delve deeper into the approaches for the other subproblems. Chapter 8 provides a rough overview of the implementation. Chapter 9 presents some results for the *SaliencyFilters* saliency algorithm. Chapter 10 states the setup for the experiments that are conducted in order to assess the quality of the approaches and Chapter 11 presents the according experiment results. A recapitulation of the work and an outlook can be found in Chapter 12 and Chapter 13.



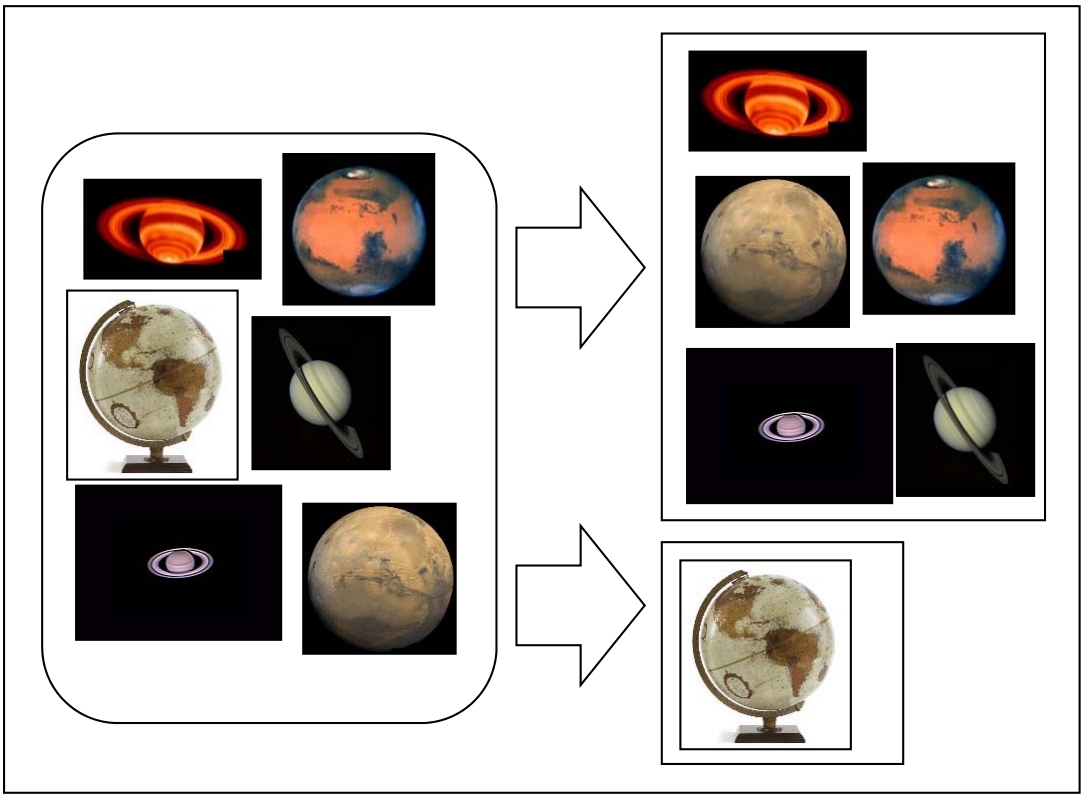


Image 1:

Upper image: The goal of this thesis is to partition an image database in order to divide the images according to the underlying classes of the depicted objects.

Lower image: This partitioning enables to find exceptional images in a database.

2 PROBLEM DISCUSSION

This Chapter gives a short discussion of the problem as a whole and then presents some thoughts about the major subproblems that occur in the course of the thesis.

The task is to implement an unsupervised detection of salient image regions in image databases and in such way to find noticeable images in these databases. Despite the fact that the actual work targets a very general problem, possible applications of such work can be the detection of unusual image material, like exceptions in image series. This includes, but isn't limited to, automatic error detection of components in quality assurance, detection of crashed plane parts in the Indian Ocean et cetera. Anyway: here, the desired outcome at the end of the process is some sort of grouping that relates to the features found in salient image regions.

In order to identify salient regions, it must be clear what is considered salient. Generally speaking, saliency implies some sort of rarity. This rarity can be found in terms of color, shape, texture or any other abstract image feature. The question of what is considered salient in the course of this thesis is answered in Section 2.1. After finding salient regions, and in order to group the images according to some measure the salient regions exhibit, the images have to be transferred into a comparable format, called *descriptor*. Many different descriptors for images and image region exist today. One task is to find a good descriptor that expresses salient region contents in a proper way. After the images have been transferred into a comparable format, they can be grouped according to a certain scheme. This grouping can be achieved either through clustering or through some linear ordering. While the former method provides the user a fully automated grouping, the second method helps the user to draw a partition line manually or enables to apply an appropriate partitioning algorithm on the linear ordering. The linear ordering also yields more information in its output, since all elements of the input set are put into a transitive relation. That is not the case with classical, rigid clustering. Exceptions to this are hierarchical clustering algorithms that put elements of the input set into a tree-representation. For the task of finding salient contents in an image database, both methods can be used. In sum, the processing chain for partitioning elements of an image database in any way can have a simple two-stage form:

1. Image descriptor generation
2. Image clustering

There are four main issues that will be addressed in the following and throughout this thesis. The first issue is the theoretical question for the definition of saliency in an unsupervised approach. The other challenges are of more technical nature, beginning with the second challenge, the reliable retrieval of the salient regions. The third challenge is the encoded representation of the salient region in a descriptor. The fourth challenge is the content-based assignment of the images to groups, which is done in a procedure called clustering. The four challenges that are the big landmarks along the processing chain and will be discussed in more detail in the following sections. The decisions that are realized in the implementation are hinted. The details of the final solution are given in Chapter 5 and the subsequent chapters.

2.1 Definition of Salient Regions

Salient regions in the course of this thesis are defined as follows:

Salient regions in images are compact areas, which exhibit large amounts of colors that are very rare in the rest of the image.

Saliency can be defined in many ways and can differ depending on many factors. Much alike, the given definition of salient regions is also rather flexible to leave room for interpretation. To living beings, saliency is completely up to the meaning of the observed things for the beings. This changes not only from species to species but also from individual to individual. Further, one individual can change its definition of saliency from situation to situation, or over time. The direction humans look and therefore what they see even depends on their feelings. In this work, however, individual aspects are left aside and focus is on two-dimensional images. Most of the attention is paid to photographs. Also, as this work focuses on salient image *regions*, saliency is understood as an area that is more or less compact and carries some sort of rarity. This rarity can be found in terms of brightness, color, saturation, shape, or texture. Not all images contain the same amount of saliency, or eye-catching content. Some images may contain more than one thing that draws attention of the viewer, making it ambiguous to find “the” salient object, and other images may even lack any salient object. Image 2 depicts four photographs that show each case.

To limit the extent of this thesis, higher-level saliency-domains like infrequencies in size or the size of object-gaps are not explicitly considered salient (see Image 3). Also, one can assume that the photographer or rendering artist put the center of attention also close to the image center. However, as this assumption is of heuristic nature, it is not of importance in the thesis. In some practical cases, such heuristics might prove helpful. The software added to this thesis is open to capture all of these aspects. Finally, salient points are not considered salient, since one point does not make a region.

2.2 Salient Region Retrieval

In order to find and extract salient regions of an image, the image must be filtered according to the definition of saliency. There exist several methods for finding salient image regions. Due to the lack of prior knowledge about the image contents or what objects to look for, a bottom-up, pre-attentive method is chosen. A desirable saliency detection algorithm that processes large image databases must be fast and must yield quality results. It is also convenient to prefer an algorithm that is simple in its implementation. That means, in order to minimize maintenance cost and make a potential extension feasible, an algorithm should be easy to understand and easy to implement. Ideally, an open implementation available is already available.

A low-level saliency detection algorithm can process the image at the pixel-level using color features and assign a saliency value to each pixel, which effectively yields a grayscale map. That map can be further converted into a black/white image mask that highlights salient regions (see Image 4). Some already existing algorithms work this way, e.g. [2], [3] or [4].

Besides such a pixel-wise approach, there exist also other approaches, like for example the approach of Perazzi et al. [1] which works in superpixel domain [5] or the graph-based method presented in [6]. Since several approaches are already implemented, this thesis’ solution does not invent a new saliency detection algorithm, but instead uses an existing approach. The algorithm used in work is the *Saliency Filters* algorithm presented by [1]. It yields subjective conformable results, is quite fast and is simple in conception. It is described in Chapter 5. Approaches of other research teams are presented in Section 3.1.

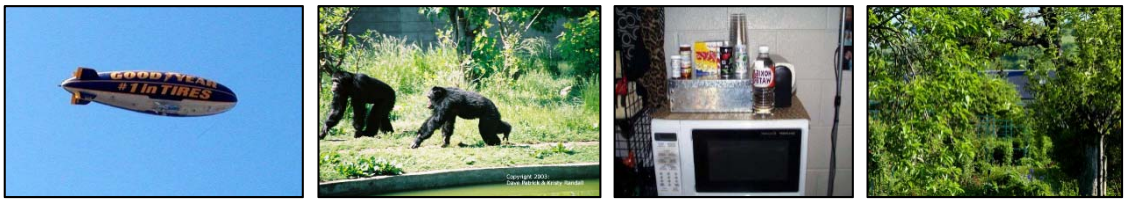


Image 2: Several random photographs. Sometimes, salient objects can easily be spotted, sometimes, it is unclear what is to be considered salient. Some images do not contain anything eye-catching at all. Images taken from the Caltech-256 benchmark set [7].

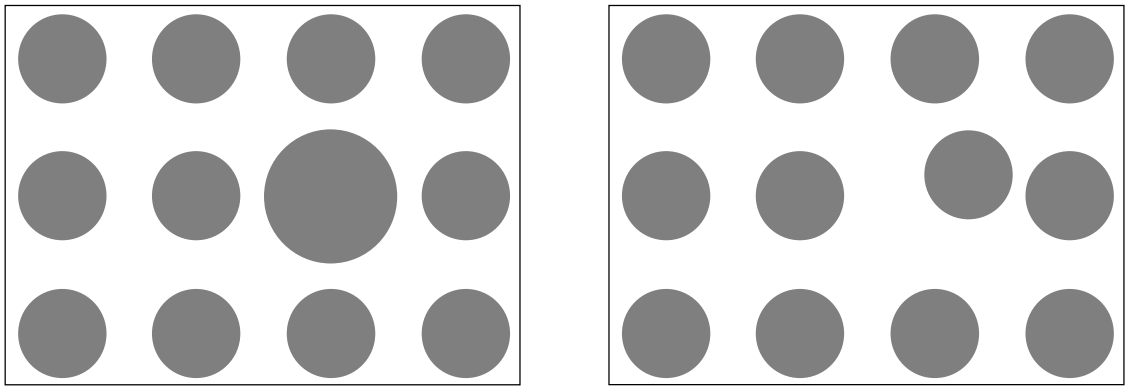


Image 3: Infrequencies in either object size or object gaps are not explicitly considered salient in this thesis.

2.3 Salient Region Descriptors

The salient region description format, called *image descriptor* or simply *descriptor*, is a data structure that captures some specified properties of the salient regions found in an image. A descriptor is typically used to query the things it represents or to compare different things with each other according to the descriptor's properties. Compared to the things they describe, descriptors are usually smaller in terms of storage space. That makes them relatively swift to load, store and to process. Descriptors can further abstract the information on which they build and can make special aspects invariant or put some sort of relativity into them. For example, two images taken under different lighting conditions depicting the same scene could result in two equal or at least very similar descriptors if these where invariant in terms of global brightness. Generally, there is no limit to what a descriptors can store. There exist already a plenty of common image descriptors. Some are concerned solely with color information, like the Color Layout Descriptor [8] or image histograms. Another approach that derived from language processing is the so-called bag-of-visual-words method [5], [9] that measures certain features on image patches and builds histograms of prototypical image patches to describe the image or image parts. Other descriptors might address shape of objects like the Fourier descriptor. It is of course possible to combine several descriptors into a new one.

Since the task is to deal with more or less compact salient image regions, the descriptor that is used in this work, incorporates the shape of the region, also the local color information is stored. The shape information should be invariant to rotation, uniform scaling and translation. The color information can be invariant to color shifts, global brightness and global contrast. For example, two cars of the same type but with different colors, the first one red and the other one green, should be put into the same group. On the other hand, a descriptor of the full moon and a descriptor of a round orange fruit should not relate. Therefore, global color transformations are only enabled at the user's wish. That being

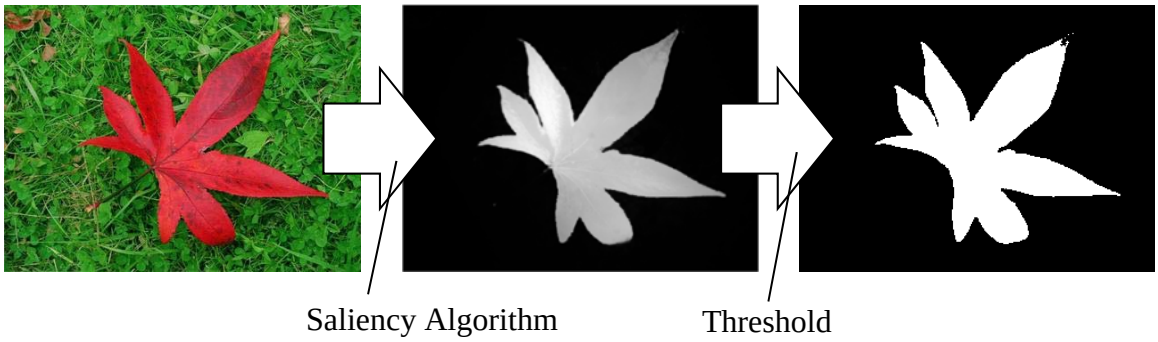


Image 4: A saliency algorithm takes an input image (left) and generates a saliency map from it (middle) that can be black/white-partitioned for identification of salient regions. Images taken from [1].

said, the used descriptor incorporates shape and color information. However, to keep the software versatile, the software only defines an interface to interchange different descriptors before compile time, but it does not specify or anticipate one beforehand.

The descriptors should be easy to compare and to work with. Therefore, they are designed as vectors of real-valued numbers of fixed length – instead of some complex data structure. The listing below recapitulates the properties of the descriptors in the solution.

Required and desired properties for a descriptor are:

- real-valued vector of arbitrary, but fixed dimension
- small with regard to memory-consumption
- invariant in terms of scaling, uniform rotation, spatial translation
- invariant in terms of global brightness, global contrast
- invariant in terms of color shifts

A question that is coupled to the retrieval of salient regions discussed in Section 2.2 is, can and will a certain method detect more than one salient region within one image? In case this happens, how should these multiple salient regions be handled by the processing chain in the software? Should (1:) only the most salient, or biggest region be treated, should (2:) all salient regions be described by one and only one description or should (3:) all regions get their own description? While option (1) would be easy to achieve it would certainly yield reduced information, since only one of n of the salient regions is considered. Option (2) would also be simple, but the results would be biased in case when multiple salient objects do not depend on each other, like, for example, a car and a ball on an image that depicts a street scene. However, there may be circumstances where multiple salient regions may be dependent on each other. Option (3) would require a more complex solution, since n descriptions could belong to one image. Their bookkeeping in the software would be more complicated, also the storage space requirements would be comparably large. The results of option (3) should be presumably stable, one could even statistically infer dependent objects even when the image quality was bad. On the other hand, a grouping of bundles of salient objects, like option (3) enables, can provoke subjective ambiguities: should the image depicting the car and the ball from the former example be put into the group “car” or into the group “ball” or should it rather be put into a group called “ball & car”? A satisfactory approach should be dynamic in the sense that a user can pick the grouping scheme at runtime.

After all, the options (1) and (2) do not seem to provide best quality, but on the implementation side, they are much simpler than option (3). The results of the options (1) and (2) might also be good, and either one may be worth an attempt. Also, option (1) and (2) can be easily implemented in the same environment under same conditions. The challenge that distinguishes option (1) and (2), which or how many salient regions may be described in the descriptors, can be handled in the module that is responsible for the generation of the descriptors itself: it only needs, besides additional parameters, the saliency maps that must be given anyway in order to generate a descriptor. Ergo, the options (1) and (2) are in fact interchangeable. This eliminates the dilemma of choosing either one or the other option but enables the implementation of both, but puts option (3) out of consideration. Because of simplicity, the comparably small effort and the uncertainty about the quality of the results, options (1) and (2) are favored over (3). Option (3) can be considered worthwhile for future investigations. However, only a subset of all saliency maps ever created may yield multiple salient regions.

2.4 Feature Clustering

There exist a variety of algorithms that can split data into different bins. These methods are called clustering algorithms, and all of them serve the same purpose: grouping, respectively partitioning data. Their behavior and results can differ strongly and one could classify them by many aspects. The most prominent types of clustering algorithms are either based on (1:) hierarchy, (2:) distribution or (3:) density of the data points. A fourth family of methods, called (4:) optimization based clustering, is usually based on some sort of centroids of pre-assumed clusters that are iteratively re-assigned until the results are satisfactory or convergence is reached. Image 5 shows example plots of representative algorithms for all four clustering families. Of course, some algorithms cannot be strictly put into one of these four families, but instead contain aspects of more than one algorithm family. It is also important to mention, that there exist no optimal clustering algorithm for the general case. Finding a good one to a given problem depends on the structure of the data.

The data that is to be clustered are the descriptors (also referred to as *feature vectors*) of the image regions. The images themselves can then be grouped accordingly. In this work. the descriptors are n-dimensional vectors of real numbers (Section 2.3), which serve as input of most typical clustering algorithms. Since, for the sake of versatility, no certain descriptor is anticipated beforehand and the problem is to cope with very general, not foreseeable image contents, no further assumptions can be made on the structure of the data. That means, on the flipside, that the structure of the data is explicitly not assumed to have any certain shape or density, thus it cannot be considered, for example, Gaussian.

Connectivity based clustering (1), also called hierarchical clustering, either starts with one big cluster that will be split into smaller ones, or starts with n one-element clusters that are merged with respect to the distance of their feature points. One well known hierarchical clustering method is the single-linkage clustering in which, starting with n one-element clusters, the two closest clusters are combined in each iteration until all elements are combined into one cluster. This way, the algorithm yields a tree structure of clusters and every complete cut through the tree describes a certain clustering. Naïve implementations of connectivity based algorithms have, due to the exhaustive comparison the of the feature point-distances, an average complexity of $O(n^3)$. This cubic time complexity is too slow when it comes to the processing of big data sets. The sophisticated method SLINK [10] reaches a time complexity of $O(n^2)$.

Distribution based clustering (2) seeks to fit a number of probabilistic distributions to the elements to be clustered, which can be seen as samples of these probability densities. Results yielded by distribution based clustering algorithms, e.g. with the Gaussian Mixture Model, do not conform to the ground truth if there is no mathematical model inherent in the data set. Since the distribution of the data in the course of this thesis is unforeseeable, distribution based clustering is not a good choice in the course of this thesis.

Density based clustering (3) defines clusters as regions of high density that are separated by regions of low density. Points of the latter can be assigned to the nearest clusters or can be managed as noise. Famous examples for density based clustering approaches are CLIQUE [11], DBSCAN [12] and its kind-of generalization OPTICS [13]. A problem occurs, when multiple clusters overlap and low-density regions that would otherwise divide them are bridged. This issue can meld two distinct clusters. On the other hand, if good care is taken when choosing the dimensions for a descriptor, the chance of overlapping clusters in feature-space can be kept small.

Optimization based clustering algorithms (4) are a family of methods that iteratively try to reconfigure the output state of the clustering result, either for a fixed number of steps or until convergence. Classic methods that use the centroids of each pre-assumed cluster are the k-Means algorithm and its siblings, k-Medians and k-Modes. K-Means is an iterative approach that consecutively first assigns the nearest points to one of k cluster centers, then, second, recalculates the cluster centers and re-iterates these two steps. K-Means is widely known and used. The biggest issue with the original k-Means clustering is the fact, that the number of clusters k must be set beforehand. Another issue is, that the algorithm will effectively partition the feature space into a Voronoi-grid and each cluster would be put into one Voronoi cell (Image 5). Cells in a Voronoi-grid have a very certain shape, e.g. they are always convex. This makes k-Means not very suitable for the problem at hand, where the shape of the real underlying clusters is not foreseeable. Nonetheless, due to its simplicity, k-Means has a good effort to outcome ratio.

Since no perfect clustering algorithm exists, data clustering is still a research topic and therefore still subject to improvements, the solution of this thesis will enable different methods. The main focus will be on a density based approach, for that algorithm-family has the biggest resemblance to manual solutions and does not rely on the shape of some probabilistic density function. The density based algorithm that is chosen is the OPTICS algorithm [13]. Besides that, the solution software also implements a k-Means clusterer and an outlier clusterer. More information on the feature space clustering solutions is presented in Chapter 7.

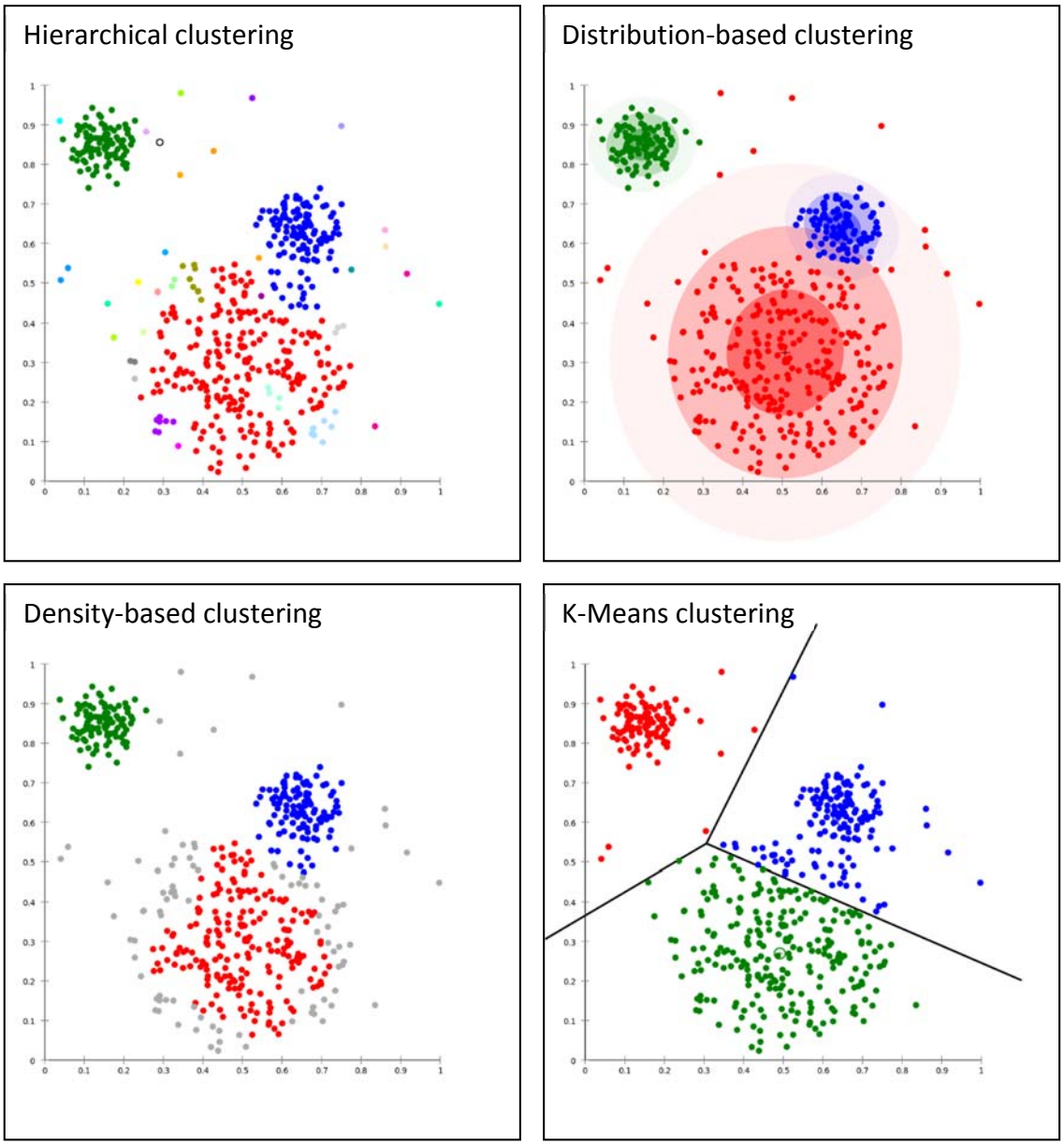


Image 5: Examples of different clustering algorithms of the different algorithm-families. In each plot, the same three Gaussian distributed point sets are drawn. Upper left: Single linkage clustering as an example of hierarchical clustering. Upper right: expectation-maximization as an example of distribution based clustering, because of the Gaussian distribution of the sample points the mappings are fairly good. Lower left: DBSCAN as an example of density-based clustering, gray pixels belong to regions of low density and are treated as noise by DBSCAN. Lower right: The optimization based approach k-Means, which partitions the data into three Voronoi cells. Images taken from Wikipedia, the free encyclopedia [14].

3 RELATED WORK

After the fundamental questions of what to achieve have been answered and the problems to solve are discussed in the previous chapters, this chapter gives an overview over work that deals with a similar problem statement or the sub-problem of extracting salient regions. The present chapter consists of two parts. The first part is dedicated to approaches that extract salient regions from images. The approaches will be briefly compared to the *Saliency Filters* algorithm of Perazzi et al. [1] that is used in the solution of this thesis. The second part of this chapter presents approaches and studies that deal with unsupervised learning of object categories. All approaches presented here are compliant with the definition of saliency as it is stated in section 2.1.

3.1 Approaches to Salient Region Detection

Saliency detection methods can be grouped into biologically inspired methods and computationally oriented ones. The former use color, contrast or edge information (e.g. [3], [15]) while latter employ more abstract ideas or means that are common in computer-vision, e.g. graph based approaches or computations in frequency domain (e.g. [16], [17]). Additionally, the algorithms can be grouped into local and global contrast measures. Local contrast measures usually calculate the contrast of one image region and compare it to its surrounding neighborhood. This measure is also called a center-surround measure. Global schemes on the other hand deal with information of the whole image and do not involve locality measures into their algorithms. All approaches presented in this section are pre-attentive, bottom-up methods, which means the approaches are low- to mid-level approaches that do not anticipate any certain feature or object type beforehand.

The visual attention system of Itti et al. [2] uses contrast information from color, intensity and edge direction in order to generate so-called feature maps. These feature maps can be combined to generate a saliency map. Varying the size of the image regions and its neighborhoods results in several feature maps with different scales that are then linearly combined to create a saliency map while being agnostic to the scale of the features. Since the feature maps have different scales the saliency maps are also of reduced size. The approach is a classic and is widely cited in literature. Image 6 depicts the general architecture of the method. The method differs much from *Saliency Filters*, since the latter does not work at different scales but abstracts fine-grained detail via superpixels [18].

Kadir et al. [19] define saliency as unpredicability in low-level attributes and spacial scale, using the shannon entropy of image patches at different scales. This approach differs strongly from approaches that are based on kernel konvolution.

Han et al. [20] extract objects of attention using a saliency map generated by Itti's method [2] and then in a second stage select only a few attention seed points from that saliency map. A Markov random field model that contains the attention values and low-level features like color, intensity and orientation contrast is employed to sequentially grow the attention objects to extract the objects sharp and correctly. The method thus elevates Itti's

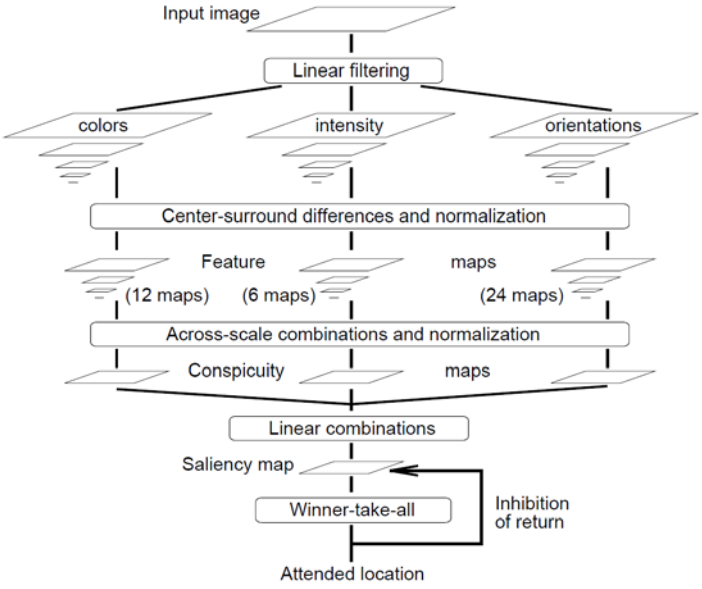


Image 6: General Architecture of Itti's saliency model [3].

method to a more stable solution but adds more complexity to it. *Saliency Filters* does not employ a statistical method like Markov fields.

Harel et al. [17] take a biologically inspired, yet graph based approach by using Markov chains. The researchers define Markov chains over various image maps and interpret the equilibrium distribution over map locations as saliency values. The weights of the Markov chain are the weighted dissimilarities of the nodes in the feature map. The researchers point out to the computational power and parallel nature of graph algorithms, due to their topographical structure, but do not come up with a parallelized implementation. Harel et al. compare their results to original human eye fixations and the results of four other studies, including the work of Itti et al [2]. For this purpose they use a benchmark set of 108 natural images. According to the test results, their method proves to be superior over the others. The approach of Harel et al. uses tools from the stochastic domain, which distinguishes from the contrast based approaches (including *Saliency Filters*) that, in most instances, add up differences of several feature maps to compute saliency maps.

Achanta et al. [3] use a local contrast method and calculate center-surround feature distances, but they do not create many feature maps with different size. Instead, they calculate the local contrast between neighboring image regions at different scales. They claim not to be biologically inspired. This method is comparable to the *Saliency Filters* approach since Achanta et al. work with local contrast measures but they do not incorporate information about uniqueness or distribution of color into their measures.

In another approach, Achanta, Hemami et al. [16] present a simple and lean method that generates a saliency map by computing the absolute difference of the mean image value with a gaussian blurred version of the image in *Lab* color space. This saliency map is then combined with a mean shift segmentation [21] of the original image to get an average saliency value for each segment. In order to retrieve salient regions, this average-saliency map can then be thresholded to filter only the most salient regions. The method is easy, fast, and as the researchers state, yet leads to better results than for example [2] or [3]. The method uses a global contrast measure, while *Saliency Filters* applies a mix of a local and global contrast measure.

Judd et al. [22] dedicate their work to finding out what people actually look at on images. Therefore they conduct an experiment with 15 users and 1003 images in which they observe the human's eye movement. Amongst other things, they discover that the centers of attention are often located in the image center. Based on the results of their experiment they derive "ground truth" saliency maps for each image. They also build a saliency-model with low-level, mid-level (horizon lines detector), and high-level (face detector) features and train a support vector machine with this model and their experimentally derived saliency maps. The model is driven by experiments and by human behavior and thus differs highly from all approaches presented here.

Cheng et al. [15] present a global contrast based approach that first blurs the image and then quantizes the pixels in RGB color space, which effectively segments the image. Next, the algorithm calculates the distance of the pixels to each other in Lab color space, weighted by the distance of the corresponding segments. The resulting saliency map has the same resolution as the input image. The approach is targeted towards natural scenes. Due to the blurring step which is necessary for guaranteeing segments of a minimum size it does not perform well on highly textured scenes. The approach differs from *Saliency Filters* in the abstraction of the image. While *Saliency Filters* abstracts by using superpixels, Cheng et al. abstract by blurring, which consequently leads to reduced result quality. Also unlike *Saliency Filters*, the approach of Cheng et al. does not implement the aspect of the distribution of color.

Xie et al. [4] operate on both low-level and mid-level cues and use the Bayesian framework. With help of the Harris point detector [23] they extract salient points and use their convex hull to generate a first, coarse salient region which they use as the prior saliency map. In addition they segment the image into superpixels which are then clustered with respect to low-level information to form a few image segments that are represented in a likelihood map. The saliency value of each pixel is determined by putting both the prior map and the likelihood map into the same Bayesian formula. This approach resembles the *Saliency Filters* approach since both methods abstract low-level cues with help of superpixels. They way salient regions are found differs nonetheless highly.

Jiang et al. [24] view the problem of finding salient regions as a facility location problem [25] and also integrate prior high-level information that they found to be relevant to humans. According to [22], humans are more attracted to warm colors like red and yellow and pixels closer to the image center and human faces. Therefore Jiang et al. use three prior maps to prioritize image parts that inhibit respective features.

3.2 Approaches to Unsupervised Object Detection

Weber et al. [26] claim to be the first team of researchers to publish a work about unsupervised object detection and categorization. It differs from the thesis at hand in the way that it is not made for partitioning image datasets but to recognize an object in an image in real time. Their method includes an unsupervised training step in which object features are learned. Weber et al. treat objects as probabilistic constellations of rigid parts, or features. That means, objects consist of an unspecified number of salient image patches, i.e. the object parts, and a shape, which is the positional constellation of the parts. Parts can be, for example, eyes, nose, ears and mouth which are together forming an object called face. For Weber et al., object detection consists of three stages. First, in an image, suspected parts are found with help of interest point operators that look for highly textured image regions, and an image clustering algorithm that favors large regions. The second stage forms likely object hypotheses, i.e. constellations of appropriate parts, considering both complete as well as partial visible objects; not all detected parts have to belong to an object, and some parts of an object may not be visible in an image due to perspective or occlusion.

The objects are modelled as a probability density function depending on the parts that are found, the parts that are missing (which is modeled as a hidden variable) and a set of indices depicting the parts that belong to the foreground object (which is also a hidden variable). The third stage uses the object's joint probability density function for either calculating the likelihood that any hypothesis arises from an object, which is in other words object detection or that one specific hypothesis arises from an object, which Weber et al. call object localization. Training of different object hypotheses is done via Expectation Maximization. The researchers trained their model with images of heads, leaves and cars and put also images depicting clutter into their test-sets. The method proved stable against clutter and is able to represent subsets of different object classes, e.g. in form of partial occlusion. The researchers where thus able to distinguish different classes of leaves or to categorize human heads from different viewing-angles.

Fulkerson et al. [27] promote to use superpixels in order to find and classify objects. In their supervised approach, the scientists segment an image using the quick shift superpixel algorithm [28] on the high-dimensional feature vector composed of the x and y pixel coordinates and the pixel's color in Luv color space. The classification is done using a bag-of-features classifier. Fulkerson et al. extract SIFT descriptors [29] for each point at a fixed scale and orientation. The extracted descriptors are first quantized using a k-Means dictionary and afterwards they are aggregated into a histogram for each superpixel. In order to find bigger and connected objects (wheels and a body that form a car), histograms of adjacent superpixels are aggregated. A support vector machine is trained on the superpixel level with the most frequent class label the corresponding superpixel contains. Despite Fulkerson et al. work with a superpixel approach, they do not aim to find salient regions as the *Saliency Filters* algorithm itself does. Instead they classify on their data. This the thesis at hand, this step is not done with help of a training phase but, after finding examining all images, by clustering the data, e.g. with a k-Means clustering. In the thesis at hand, superpixel clustering is merely (but nonetheless importantly) used to preprocess the images for increased robustness and processing speed.

Andreeto et al. [30] aim to unify unsupervised image segmentation and object recognition. They use a parametric probabilistic model for shape- and color-representation of the segments. For data description they use the bag-of-visual-words [5] approach. Their algorithm discovers correspondence between segments in different images which are being processed simultaneously. The researchers create a generative segmentation model that consists to one half of a parametric Gaussian distribution and to another half of a kernel density estimator, i.e. a Parzen Window [31]. They call their model a semi-parametric mixture model (SPMM). Image 7 gives an overview over their model. Different segments are modelled by their own density functions. Data is sampled and assigned to the segments given the density functions. The approach of Andreeto et al. differs highly from the approach of this thesis, because Andreeto et al. are working with a generative probabilistic model and use a different approach to describe image patches, namely via the bag-of-visual-words approach.

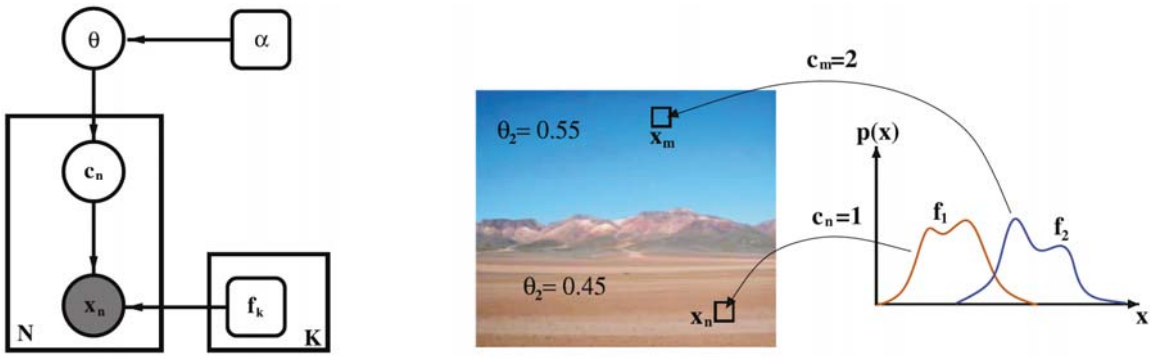


Image 7: The approach of Andreeto et al. [30]. Left: The generative model for image segmentation. The gray node x_n represents observations, i.e. pixel features. The node c_n represents the segment assignment for the observation x_n. The node θ represents the mixing coefficients for each image segment. The two rounded boxes α and f_k represent the hyperparameters for Dirichlet distributions over θ and the density function for each segment k. Finally, N is the total number of pixels in the image. Right: Image formation process as described on the left side. An image is composed of two segments: ground (45 percent of the image) and sky (55 percent of the image). An observation x_n is obtained by first sampling the assignment variable c_n. Assuming $c_n = 1$, the corresponding density f_1 is used to sample x_n as a member of the ground segment. Similarly, a second observation x_m in the sky is sampled from the corresponding density function f_2 when $c_m = 2$. Image taken from [30].

The work of Tuytelaars et al. [32] has not the aim to present a novel approach to unsupervised object recognition but their publication comes as a comparison of different state of the art methods. They evaluate and compare methods for learning to recognize objects in images. They experiment with probabilistic methods based on latent variable methods and clustering methods. The researchers compared the methods on benchmark image databases like the Caltech-256 [7] and MSRC2 [33]. Tuytelaars et al. also propose a framework for evaluation of unsupervised object discovery methods. Tuytelaars et al. realize, that since the ending of the last century the computer vision research community has drawn great attention to the category-level object recognition. They find that many approaches are only poorly tested on small test data sets that allowed the parameter to be fine-tuned, which in turn results in over-optimistic outcomes. Tuytelaars et al. propose an evaluation scheme that allows multiple categories per image. Their comparison involves baseline methods such as random assignment, k-Means clustering, principal component analysis and more advanced methods like latent variable methods and spectral clustering methods. In their testing framework, the image descriptions are all based on the same representation technique, namely the bag-of-visual-words [5] approach that describes the image with a set of quantized image patch descriptors. The scientists discover, that clustering methods give best results, when it comes to one category per image. They conclude that fully unsupervised methods for object discovery, under realistic circumstances, "is still a largely unsolved problem" [32].

4 THE PROCESSING CHAIN

The last chapters introduced and analyzed the problem of extracting salient image regions from images in large databases and grouping the salient objects into disjoint sets. Chapter 2 provided a first look onto the whole problem and the major subproblems. The state of the art concerning salient region detection and unsupervised object detection and categorization was presented in Chapter 3. This chapter provides information about the solution in general. Details on the solutions to the subproblems are given in the following chapters.

First, to recap the general design decisions, the solution aspires to be simple and flexible, therefore a modular approach is chosen. The intent behind the flexibility is to make changing different algorithms and techniques simple to be able to tackle the subproblems in different ways if one way does not prove to be satisfactory. This way, also multiple solutions to the subproblems can be implemented. In order to allow external systems to be used, the input- and output files are plain image files and text files that list certain values.

The processing chain is a simple two-stage process of first extracting and transferring features of each image into a descriptor, i.e. a feature vector, and then clustering the images according to the descriptors (Image 8). Both stages can be processed separately and the results of each stage may be valuable for the user. For this reason, both stages are split into separate applications that can be executed one after another. The three subtasks, namely the salient region retrieval, the generation of feature vectors and the clustering can be fulfilled by different approaches, i.e. by different modules.

In the first stage, for every image, the salient region maps are retrieved with help of the *Saliency Filters* algorithm [1]. In order to derive binary saliency map (a saliency mask) this thesis' solution provides both a simple application of a threshold value as well as a masking with help of *GrabCut* [34]. *GrabCut* is an image foreground/background segmentation algorithm based on minimal graph cuts. A saliency map can be used to define a prior map for the *GrabCut* algorithm. *GrabCut* is slightly slower than a classic threshold method but the resulting saliency masks are closer the ground truth [1]. The usage of a min-cut segmentation is recommended by Perazzi et al. [1]. The next step is to determine the shape of the salient region with the help of the saliency mask. Based on the information about the shape of the biggest salient region and the image color, a descriptor for the image is being created (Chapter 6). If no salient region can be found, or the biggest salient region size is below a given threshold, the image is being reported and considered garbage, which means it contains no objects of interest. The same holds for every case if an error occurs during the calculations. The descriptors of all non-garbage images are stored in a file, along with another file that stores the paths to the corresponding images. Garbage-images do not yield descriptors and are not further processed by the second stage. Intermediate results of the first stage, like saliency maps, can be saved, if specified. A simplified flow diagram of the processing chain that contains the main program flow is depicted in Image 9.

The second stage takes the main output of the first stage as input and clusters the descriptors (the feature vectors) with a not further specified clustering algorithm. Each feature vector is assigned to a cluster. The clustering stage creates a file with the clustering assignments that corresponds line-wise to the input files. Also, if specified, a set of directories that relates to the clustering structure and contains symbolic links of the clustered images can be created. By this means, a simple and concise way to intuitively check the quality of the clustering is introduced. A simplified flow diagram of the processing chain is depicted in Image 10. More on the realized clustering algorithms can be found in Chapter 7. An overview over the final software implementations can be found in Chapter 8.

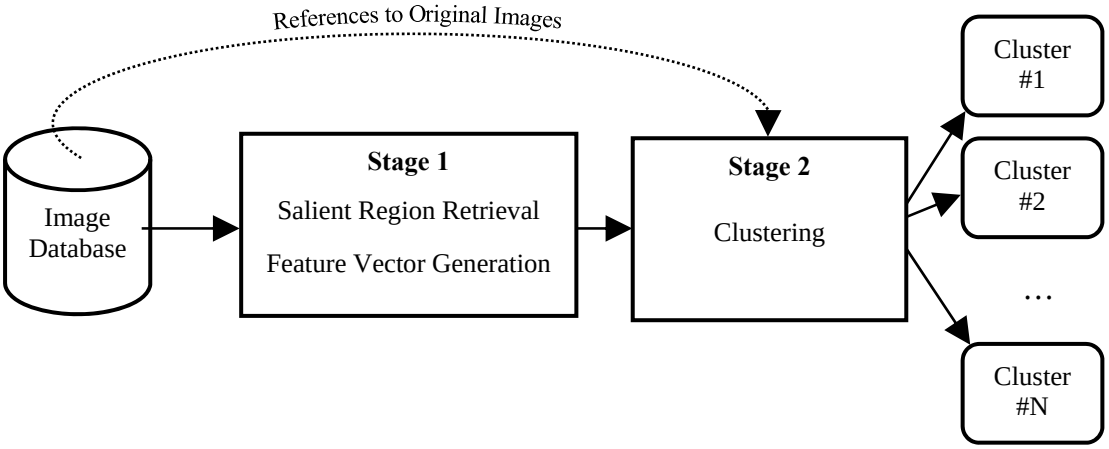


Image 8: The general processing chain of the solution. Feature vectors of stage one are passed to the clustering stage that partitions the original images according to the distribution of the feature vectors in their feature space.

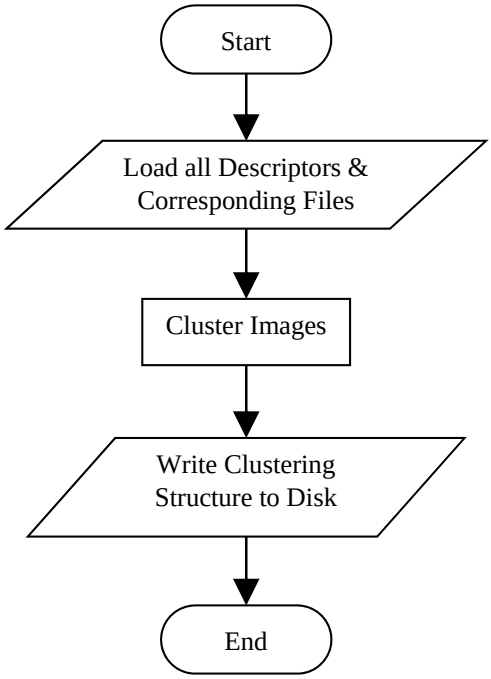


Image 9: Simplified flow diagram of the solution's second stage, the clustering stage.

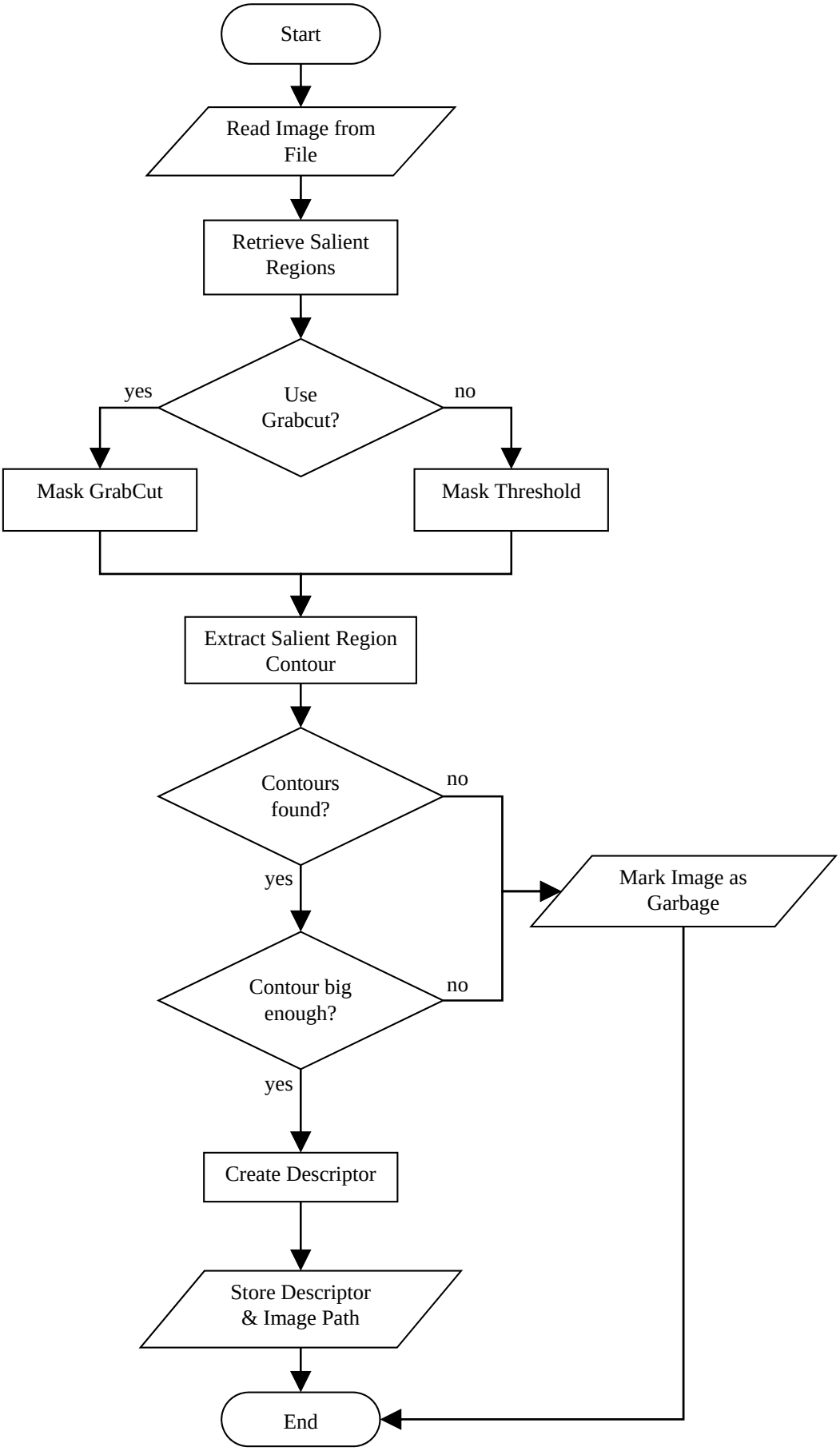


Image 10: The simplified flow diagram of the solution's first stage for one image. This processing chain will be repeated for each image.

5 SALIENCY FILTERS ALGORITHM

The previous sections described the problem of finding salient regions in images. Chapter 3 summarizes the state of the art and shows some approaches of other researchers. Chapter 4 shows the general solution pipeline of this thesis. The current chapter deals with the used saliency detection algorithm, *Saliency Filters* [1], in more detail.

The saliency algorithm that is used in the scope of the thesis is the *Saliency Filters* algorithm developed by Perazzi et al. [1]. *Saliency Filters* reflects on a variety of other preceding approaches, reconsiders some of the design choices of these methods (see Chapter 3) and proposes a conceptually clear algorithm for contrast-based saliency estimation. Still, *Saliency Filters* does not simply improve older algorithms but uses some of their main ideas and puts them into a new framework. The work was published in 2012 and is therefore one of the more recent attempts. It follows the simple concept of local and global contrast measures, which was, for example, also used by [15], [3] and [16]. *Saliency Filters* combines local and global contrast measures by utilizing a Gaussian filtering framework, that uses the *Permutohedral Lattice* [35] [36] in order to calculate Gaussian filters with linear time complexity. The Permutohedral Lattice is a geometrical model that interpolates Gaussian distributed values with help of barycentric coordinates in any given dimension.

According to the publication [1], *Saliency Filters* outperforms the methods of [37], [38], [39], [2], [17], [3], [40], [16] and the methods presented in [15] in a precision/recall test on a database of 1000 human segmented natural images [41]. Precision measures the ratio of correctly labeled salient pixels to all as salient labeled pixels, recall measures the ratio of correctly labeled salient pixels to the number of ground truth salient pixels. Precision and recall are explained in further detail in Subsection 10.1.1 in another context.

Philipp Krähenbühl provides an unofficial re-implementation of the source code of the *Saliency Filters* algorithm on his web page [42]. The sources are written in C++. Since the *Saliency Filters* algorithm yields high-resolution results at fast speed it is the only saliency algorithm that is used in the course of the thesis. However, the solution application is kept modular to easily enable other approaches.

5.1 Algorithm Overview

The algorithm consists of four steps, two of them perform contrast measures. The first step is the abstraction of the image by means of superpixel clustering with SLIC superpixels [18]. Superpixel clustering constructs small clusters, or image elements, from pixels of similar color and location, so that only the mean color of each superpixel may be used for further processing (Image 12). This abstraction step reduces the amount of calculations that must be executed and reduces the impact of noise in the image while, converse to image blurring, it preserves edges and mid-level detail.

The second step is the calculation of the uniqueness of each image element with respect to color and with help of a weighted Gaussian distance function. This step leads to a uniqueness-map at the superpixel-level.

The third step of the algorithm consists of the calculation of the spatial distribution of the color of each image element. Colors that are distributed over the whole image yield a small value while colors that can only be found within some small region yield a large value. The contrast measures are weighted by a Gaussian distance function, just like the contrast measurement in step two.

The fourth step combines both the uniqueness-map and the distribution-map and interpolates the values at pixel level with help of a high-dimensional bilateral Gaussian distance function. That Gaussian distance function is evaluated by the same Gaussian filtering framework that is applied in step two and three [35]. The calculation yields a full-resolution saliency-map. Image 11 depicts the results of each stage of the algorithm on an example image.

The following four subsections provide a detailed description of the four major steps of the *Saliency Filters* algorithm.

(a) Source image (b) Abstraction (c) Uniqueness (d) Distribution (e) Saliency (f) Ground truth

Image 11: Different stages of the *Saliency Filters* algorithm. Images taken from [1].

5.1.1 Abstraction

The image is decomposed into small and compact elements that preserve structure and edges but extinguish small detail. Hereby pixels of similar color and location are associated with the same element. Strong edges or contours are preserved. Such an abstraction reduces not only the impact of noise and unnecessary detail but also helps to speed up the calculations since the number of elements to deal with is only a small fraction of the original number of image pixels. The researchers use an adaption of SLIC superpixels [18] to segment the image into perceptually uniform regions. The SLIC superpixels algorithm segments an image using k-Means clustering in $RGBXY$ space. The usage of Euclidean distances in $RGBXY$ space yields local, compact and edge aware superpixels, but does not guarantee connectivity. An adaption of SLIC that works with geodesic image distance [43] however guarantees connectivity, while it retains locality, compactness and edge awareness of SLIC superpixels [26]. Image 12 shows the effect of the adapted superpixel clustering to an image.

5.1.2 Element Uniqueness

This first of two contrast measures implements the commonly employed assumption that image regions which stand out from other regions catch the human attention and should be labeled more salient. *Saliency Filters* evaluates how different each respective element is from all other image elements, essentially by measuring the rarity of each element's color. The measurement of rarity is typical for most contrast-based saliency detection algorithms, like, for example, [2], [3], [16] and [15]. Due to the abstraction step of the image data to the element level the number of calculations that is to be performed is comparably small and thus a uniqueness-map can be computed within a small amount of time.

The element uniqueness is defined as the rarity of a segment i given its position p_i and color c_i in Lab color space compared to all other segments j:

$$U_i = \sum_{j=1}^{N} \left\| c_i - c_j \right\|^2 * \underbrace{w(p_i, p_j)}_{w_{ij}^{(p)}}. \tag{1}$$

Image 12: An image and a SLIC superpixel segmentation with geodesic image distance and 800 clusters after four iterations. Clusters are compact, connected and preserve the most obvious edges.

The weight measure by ${w_{ij}}^{(p)}$ combines global and local contrast estimation and controls the influence radius of the uniqueness operator. Therefore ${w_{ij}}^{(p)}$ can be used as a local weight. A function ${w_{ij}}^{(p)}$ that is local yields a local contrast term, whereas a function ${w_{ij}}^{(p)} \approx 1$ results in a global uniqueness operator which is insensitive to local contrast variation. For ${w_{ij}}^{(p)}$ *Saliency Filters* uses the Gaussian weight:

$$ {w_{ij}}^{(p)} = \frac{1}{Z_i} e^{\left(-\frac{1}{2\sigma_p^2}\|p_i - p_j\|^2\right)}. \tag{2} $$

The term σ_p controls the range of the uniqueness operator and Z_i is a normalization factor ensuring $\sum_{j=1}^{N} {w_{ij}}^{(p)} = 1$. The range controlling variable σ_p is set to 0.25 by Perazzi et al., which allows for a balance between local and global effects [1]. It is not changed in the solution of this thesis but can be altered in a configuration file. The Gaussian equation (1) can be evaluated in linear time $O(N)$ using the Permutohedral Lattice [1], [35]. The uniqueness map for some images is depicted in Image 13.

5.1.3 Element Distribution

While saliency implies uniqueness, this might not always be the only thing to consider. Ideally, colors that belong to the background are distributed over the whole image and thus exhibit a high spatial variance, whereas foreground objects are generally more compact. The compactness and locality of the image elements created in the abstraction step allows to define a corresponding second measure that renders unique elements of similar color more salient when they are found in a particular image region rather than being distributed over the entire image.

For a segment i the distribution measure is defined as the spatial variance D_i of its color c_i. Low variance indicates a spatially compact object, high variance indicates a less compact object, e.g. the image background. Elements that yield low variance should thus be considered more salient than elements of higher variance:

$$D_i = \sum_{j=1}^{N} \|p_j - \mu_i\|^2 * \underbrace{w(c_i, c_j)}_{w_{ij}{}^{(c)}}. \tag{3}$$

The term $w_{ij}{}^{(c)}$ describes the similarity of color c_i and c_j of two segments i and j, p_j is again the position of segment j and

$$\mu_i = \sum_{j=1}^{N} w_{ij}{}^{(c)} * p_j \tag{4}$$

defines the weighted mean position of color c_i. The similarity measure of color $w_{ij}{}^{(c)}$ can again be defined using a Gaussian equation that is similar to the weight $w_{ij}{}^{(p)}$ defined in Equation (2):

$$w_{ij}{}^{(c)} = \frac{1}{Z_i} e^{\left(-\frac{1}{2\sigma_c^2}\|c_i - c_j\|^2\right)}. \tag{5}$$

The parameter σ_c controls the color sensitivity of the element distribution. The variable σ_c is set to 20 by Perazzi et al. and is not changed in the work at hand. Again, like the Equation (1), the Equation (3) can be evaluated with linear runtime complexity $O(N)$ using the Permutohedral Lattice [1], [35]. Image 13 shows the distribution maps of some images.

5.1.4 Saliency Assignment

The two above contrast measures are defined on a per-element level. The final step is to assign actual saliency values to the input image on the pixel-level in order to get a sharp and pixel-accurate saliency map. First, the values for uniqueness U_i and distribution D_i are normalized to the range $[0..1]$. Both measures are assumed to be independent. The saliency value for each element is given by:

$$S_i = U_i * e^{(-k * D_i)}. \tag{6}$$

According to the Perazzi et al. the distribution measure D_i is found to be of higher significance and discriminative power, therefore they emphasized it with an exponential function. The researchers used a scaling factor of $k = 6$ throughout their experiments. This value is also used within the solution of this thesis.

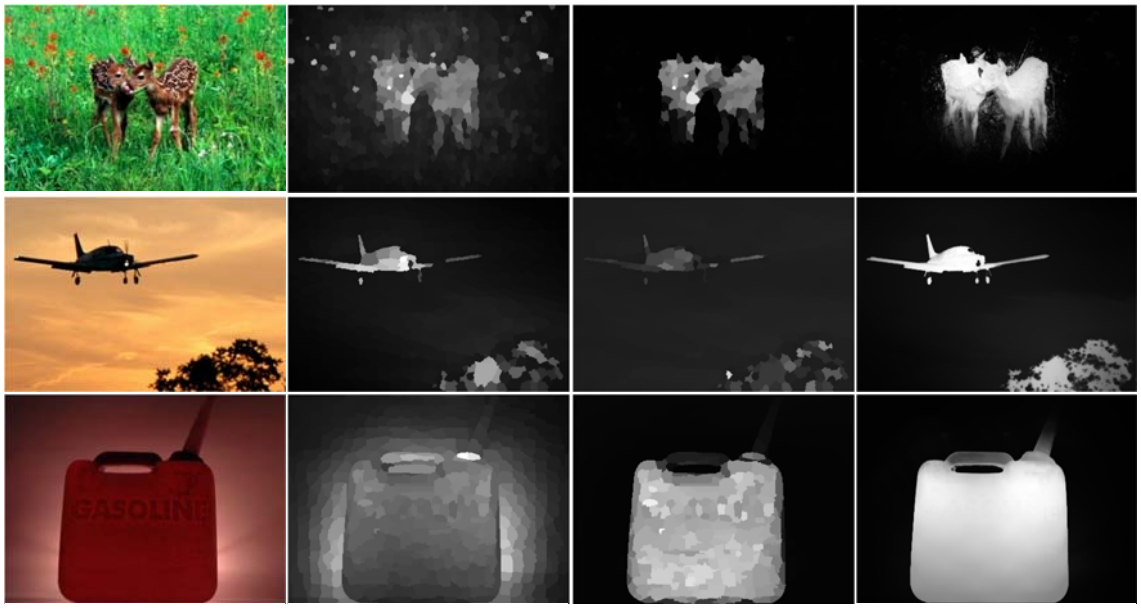
Finally, the saliency values are to be up-sampled from the element-level to the pixel-level. Simple assignment of the value S_i to every pixel in the superpixel element i would, however, carry all segmentation errors of the abstraction step. Therefore, *Saliency Filters* adopts an idea proposed in [44]. The saliency $\tilde{S}_i$ of a pixel is defined as a weighted linear combination of the saliency values S_j of its surrounding elements:

$$\tilde{S}_i = \sum_{j=1}^{N} w_{ij} * S_j. \tag{7}$$

The weight w_{ij} is defined as a high-dimensional bilateral Gaussian weight:

$$w_{ij} = \frac{1}{Z_i} e^{\left(-\frac{1}{2}\left(\alpha\|c_i - c_j\|^2 + \beta\|p_i - p_j\|^2\right)\right)}. \tag{8}$$

The up-sampling process is hence color and local-sensitive. The parameters α and β control the sensitivity for color and position. Perazzi et al. found $\alpha = \frac{1}{30}$ and $\beta = \frac{1}{30}$ to work well in practice. These values are also not changed throughout the solution of this thesis. As in the two preceding steps the evaluation of the Gaussian filter can be done with linear time complexity $O(N)$ with help of the Permutohedral Lattice framework presented by [35]. At last, the algorithm scales the saliency map to the range $[0..1]$ or to contain at least 10% of salient pixels. Image 13 depicts the uniqueness, distribution and combined saliency map for three example images.



(a) Source Image (b) Uniqueness (c) Distribution (d) Saliency

Image 13: From left to right: Source images, their uniqueness maps, distribution maps and combined saliency maps. Image taken from [1].

6 SALIENT REGION DESCRIPTORS

This chapter provides further information about the three salient region descriptors, the Fourier Shape Descriptor (Section 6.1), the Color Histogram Descriptor (Section 6.2) and the Combined Fourier Shape and Color Histogram Descriptor (Section 6.3).

There are many ways describe the salient region in form of a real-valued feature vector. Yet, as a descriptor, the feature vector must be rather small. That means, as a consequence, it is unlikely that a descriptor can include every information of the salient region it describes. As stated in section 2.3 the descriptors should store information about the shape of the salient region as well as the low-level pixel information about color, brightness and saturation. For this reason, this work contains three different descriptors: one solely for shape information, one solely for color information and one descriptor that combines the first two to a shape-color descriptor.

6.1 Fourier Shape Descriptor

The Fourier Shape Descriptor represents the shape of a salient region as the normalized magnitude-part of the Fourier transformation of the clockwise (or counterclockwise) ordered boundary points of the salient region mask. The first component of the Fourier transform, the one that expresses the translation of the shape, is discarded. Scale normalization is done by dividing each component by the value of the second component. Since, after the normalization, the second component has always a constant value of one, it can also be discarded. From the components left, only keeping the first n components of the Fourier transform and thus only keeping the n most meaningful frequencies assures fixed length of the description vector. Image 14 visualizes the process of creating a Fourier descriptor from a saliency mask. The number of frequencies to be kept into account and thus the size of the descriptor can be set in a configuration file.

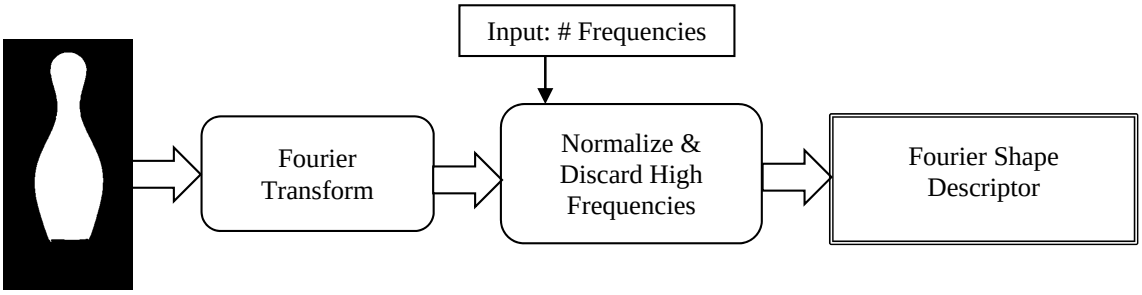


Image 14: Processing chain for creation of a Fourier Shape Descriptor. The boundary points of the saliency mask and the number of frequencies are used as the input.

6.2 Color Histogram Descriptor

The Color Histogram Descriptor counts the HSV color values in the image, normalizes them and stores these values within a histogram of fixed size. The color information of the whole image can be examined or optionally only the colors found in the salient region. When examining color information of the whole image, the saliency mask has no impact on the resulting descriptor.

The HSV color space is chosen over other color spaces, because it separates the concepts of color value, brightness and saturation, which is good when it comes to the comparison of alike objects under different lighting conditions. The Lab color space, which employs similar concepts like the HSV space and which models the human color perception better than other color spaces, is not considered best here, because the calculations of from and to Lab color space are more complex and a qualitative gain over HSV color space is not expected.

The size and thus the granularity of the histogram can be set in a configuration file prior to running the descriptor generating program. Also, one can specify to use pixel information of the whole image or just to use the pixels that are found within the salient region. The descriptor can be configured to use channel-wise auto-correlated histogram values. If the histogram channels undergo an auto-correlation, the histogram is made invariant to global color transformations like changes in brightness and color shifts, which in turn enables comparison of similar objects under different lighting conditions or objects with different colors, e.g. two cars of same type but different color. Since auto-correlation reduces the quality of the histogram information and may not be desirable in any case (Image 16), the decision whether to apply auto-correlation or not is handed to the user. Image 15 visualizes the creation process for a Color Histogram Descriptor.

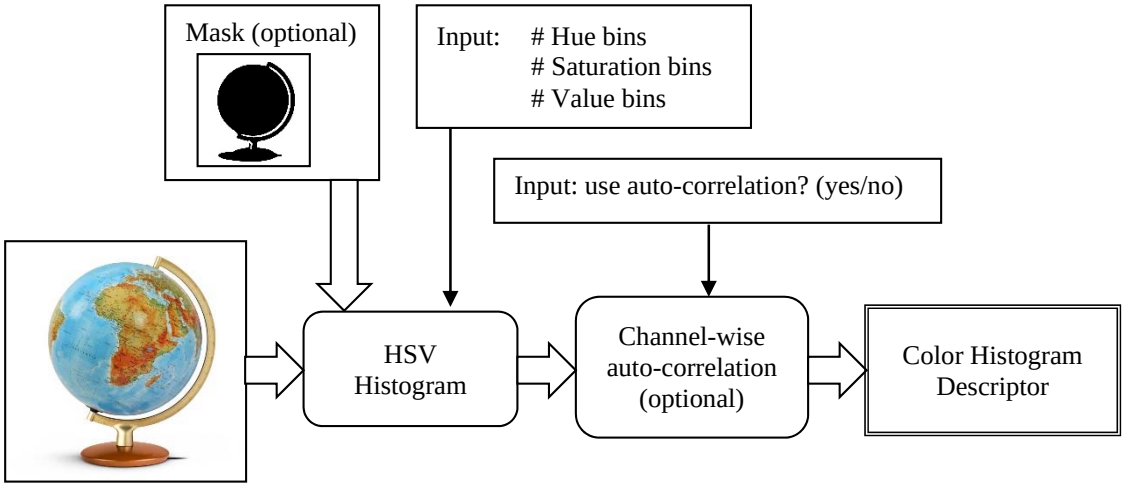


Image 15: Processing chain for creation of a Color Histogram Descriptor. If desired, the image is masked by its saliency mask. Also, if requested, the auto-correlation of the histogram can be turned on and off.

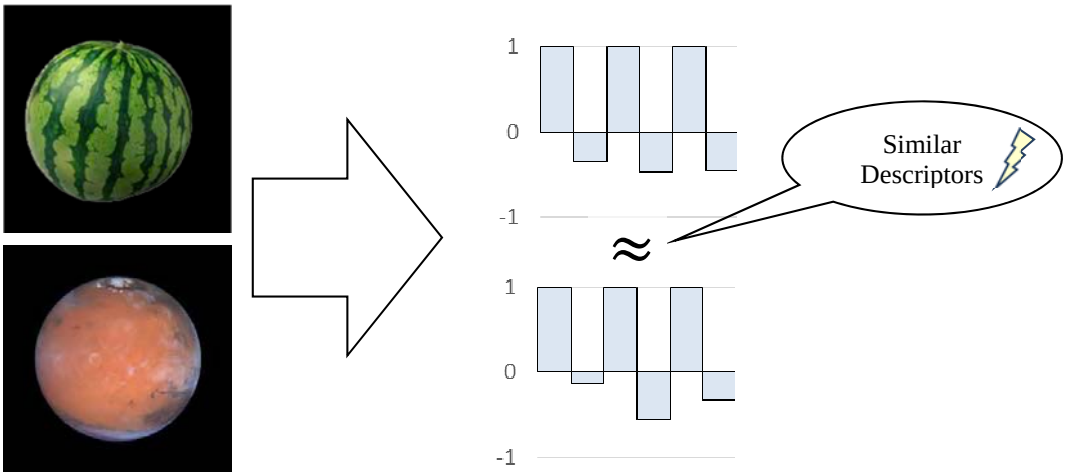


Image 16: Drawbacks of auto-correlation: Images can yield similar auto-correlated *HSV* color histograms although, on the semantic level, the depicted objects have not much in common.

6.3 Combined Fourier Shape and Histogram Color Descriptor

Since descriptors in the solution of this thesis are expressed as simple vectors of real-valued numbers, they can be easily combined to form more complex descriptors. The third descriptor which stores information about shape and color is in fact a concatenation of the first two descriptors. The Combined Fourier Shape and Histogram Color Descriptor unifies the Fourier Shape Descriptor and the Color Histogram Descriptor. All parameters that can be set for the single descriptors are also changeable in the implementation of this descriptor. Image 17 illustrates the relation of the three descriptors.

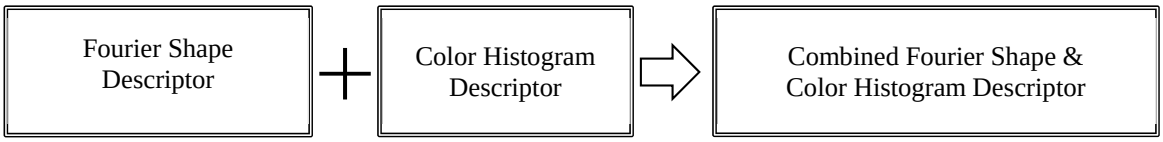


Image 17: The Combined Fourier Shape and Histogram Color Descriptor is simply a concatenation of the two separate descriptors.

7 FEATURE CLUSTERING

This chapter describes the implemented methods for feature space clustering, that is the partitioning of the descriptors. There are three different solutions for clustering the feature vectors provided in this thesis. The first solution partitions the feature space using a k-Means algorithm. The second solution finds points with the biggest distance to their nearest neighbors, which effectively finds outliers in the sense of density. These outliers can be seen as a kind of salient feature vectors. The third solution uses the OPTICS ordering [13] and divides the data set by the peaks in the OPTICS ordered reachability-histogram to identify clusters.

7.1 K-Means Clustering

Application of the k-Means algorithm [45] on the feature vectors yields k putative cluster centers. Each feature point is addressed to the cluster with the nearest center point. This method divides the feature space into k Voronoi cells, each cell representing one cluster, see Image 5. K-Means clustering is not a good solution if the clusters do not exhibit a blob-like shape, but it is a simple and widely used method. It is conceptually simple and, due to its iterative, optimization based approach, flexible in terms of runtime.

K-Means partitions the feature space by seeding k cluster centers and iteratively assigns each feature point to its nearest cluster center and then recalculates the cluster center positions until a number of steps have progressed or the algorithm converges. In this thesis, the number of iterations is not fixed, i.e. the algorithm runs until convergence. The distance measure that is used in the thesis' solution is the Euclidean distance. K-Means attempts to minimize the within-cluster sum-of-squares of k clusters $C = \{C_1, C_2, \dots, C_k\}$:

$$\arg\min_{c} \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2, \tag{9}$$

where μ_i is the mean of all points assigned to cluster C_i. There are several ways to calculate the k-Means. The one that is used in the thesis is a hierarchical implementation built into *OpenCV* inside the package `cv::flann`. More details on the implementation can be found in Chapter 8. The most common approach, in literature also referred to as Lloyd's algorithm, however, consists two steps that are, after an initialization step, repeatedly performed until the desired number of iterations is performed or convergence is reached. The initialization of the k cluster centers can be done in different forms. The naïve implementation is simply to assign random values to the k mean values. After that initialization comes the assignment step, also called expectation step, or E-step. The assignment step assigns each feature point to the cluster whose mean is closest to the feature point. This step partitions the data according to the Voronoi diagram that can be built by the cluster means. Consequently, in each step t all clusters $C_i^{(t)}$ are created as follows:

$$C_i^{(t)} = \left\{ x \middle| \left\| x - \mu_i^{(t)} \right\| \leq \left\| x - \mu_j^{(t)} \right\| \forall j, 1 \leq j \leq k \right\}, \tag{10}$$

where x is an observation and μ_i is the mean of the cluster i. One point can only belong to one and only one cluster at a time. The second step of the algorithm is the update step, likewise known as maximization step, or M-step. In this update step, since points may have changed their cluster membership, new means of each cluster are calculated:

$$\mu_i^{(t+1)} = \frac{1}{\left|C_i^{(t)}\right|} * \sum_{x \in C_i} x. \tag{11}$$

The assignment step and the update step can be applied until convergence is reached. The algorithm is proven to converge in a local optimum, but there is no guarantee that k-Means finds the global optimum of Equation (9) [45].

K-Means performs well when it comes to blob-like clusters which have a convex shape. The algorithm is likely to deliver bad results if the ground-truth clusters have some out-reaching branches in feature space, and it definitely fails when it comes to detection of nested clusters. When the number of clusters k and the number of feature space dimensions d is fixed, the algorithm can be solved in big-O time complexity $O(n^{d*k+1} * \log n)$, where n is the number of feature space samples [46].

7.2 Outlier Clustering

The Outlier clustering method presented hereby is not a clustering method by the classic definition but an outlier detection algorithm. Given a positive integer n it finds the n points which have the biggest distance to their nearest neighbor. Hence the method yields just two disjoint sets of samples: normal samples and outliers. Outliers in the sense of Outlier clustering are samples that fulfill the following condition:

$$C_n = \{x_i | \|x_i - y\| \geq \|y - z\|, x_i \neq y, x_i \neq z, \forall i, 1 \leq i \leq n\}, \tag{12}$$

Where n is the number of outliers to look for and x_i, y and z are points in feature space. The Outlier clustering algorithm does not necessarily find points around the convex hull of the feature space but maybe also inner points within regions of low density. Under certain circumstances, such points could even be located nearby cluster centers when applying other clustering methods, such as k-Means. However, the most remote elements and their corresponding images can be seen as salient elements within the image data base, because they stand out in the sense of density. Image 18 depicts an example for Outlier clustering.

The distance measure used in the solution is the Euclidean distance. In order to find the most remote samples in an unstructured feature space, one has to calculate the distance of every sample to every other sample which leads to a big-O time complexity of $O(d * n \log n)$ where n is the amount of samples in the feature space and d is the number of dimensions.

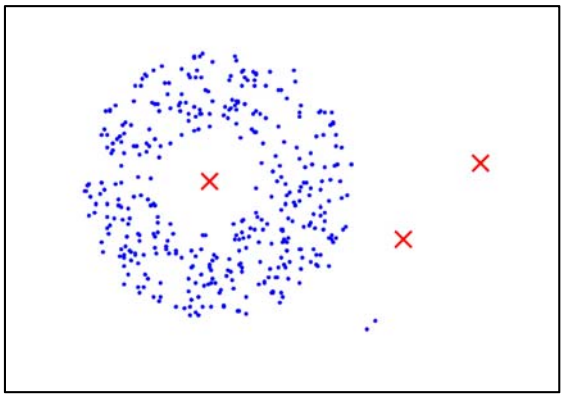


Image 18: Example for Outlier clustering. The three most remote points are drawn as red crosses. The two remote blue points in the lower right are not considered as most remote, because, despite their remoteness to the main cluster, their nearest-neighbor distance is still small. In order to eliminate this issue, one can also use OPTICs to find outliers.

7.3 OPTICS Clustering

Clustering using the density-based OPTICS algorithm [13] makes use of the augmented OPTICS ordering of the feature vectors. OPTICS orders the points in a mixture of depth-first and breadth-first search and assigns a so a so-called reachability-value to each point, which is an indicator for the density of the region around the points. The reachability-distance is a measure of the distance to a cluster's core points. Core points are inner feature vectors in regions of high density. Peaks in the OPTICS ordered reachability-plot divide to regions of low density, which are supposed to be cluster borders, or outliers. OPTICS allows to find clusters of arbitrary shape, even nested clusters can be found. This property makes OPTICS clustering favorable in situations where the shape of a cluster cannot anticipated beforehand. Depending on the input parameters, it is further possible to use OPTICS for noise detection. OPTICS is structural equivalent to the density based clustering algorithm DBSCAN [12].

Given an OPTICS ordering, there are several simple ways to detect clusters. It is possible to (1:) manually look for the n most eye-catching clusters, (2:) to partition the feature vectors at a certain reachability threshold or (3:) to find the most persistent maxima in the reachability plot, i.e. the most outstanding points with the sparsest neighborhood. Other, more sophisticated approaches (e.g. [47] exist, but are not considered in the scope of this thesis. They can be used in future research. The solution of this thesis uses method (3). Image 20 depicts two clustering results of OPTICS ordered data sets, both partitioned using the most persistent reachability-maxima.

In order to explain OPTICS, let's introduce the variables that are further used for algorithm explanation. Let p be a sample from the feature vectors, let ε be a distance value, let $N_\varepsilon(p)$ be the ε-neighborhood of p (including p), let $MinPts$ be a natural number and let $MinPts\_dist(p)$ be the distance of p to its $MinPts$' neighbor. The variables ε and $MinPts$ are the two tuning-parameters of the OPTICS algorithm. The role of ε is to describe the search-radius for finding neighboring feature points. The higher the value of ε, the more points may be found in the local neighborhood. This has impact on the speed of the algorithm: if a large neighborhood is to be processed, the algorithm is slower as if the neighborhood is small. Nonetheless, ε can be constantly set to ∞ in order to take all samples into account when looking at the neighborhood. The variable $MinPts$ describes the minimum count of points that are to be found in the ε-neighborhood in order to appoint a sample to be core point. The higher the $MinPts$, the denser an ε-neighborhood must be to appoint

core points. Furthermore, two distance values, the core-distance and the reachability-distance, must be defined. The core-distance of a sample p is the smallest distance so that at least $MinPts$ samples lie within the radius of a hyper-sphere around p, but it must be smaller than ε. Hence, the core-distance of a sample p is defined as $MinPts\_dist(p)$, if there are at least $MinPts$ samples in the ε-neighborhood of p, otherwise the core-distance of p is undefined:

$$core\_distance_{\varepsilon,MinPts}(p) = \begin{cases} UNDEFINED & ,if\ |N_\varepsilon(p)| < MinPts \\ MinPts\_dist(p) & ,otherwise \end{cases}. \quad (13)$$

If p's core distance is not undefined, then p is called a core point.

The reachability-distance of a sample p with respect to another sample o is defined as follows:

$$reachability\_distance_{\varepsilon,MinPts}(p,o) = \begin{cases} UNDEFINED & ,if\ |N_\varepsilon(o)| < MinPts \\ \max\left(core_{distance_{\varepsilon,MinPts}}(o), distance(p,o)\right) & ,otherwise \end{cases}. \quad (14)$$

In other words, the reachability-distance of a sample p with respect to another sample o is the minimum distance such that p is directly density-reachable from o if o is a core point. If o is not a core point, then p's reachability distance is undefined. Image 19 visualizes core-distance and reachability-distance on a small example.

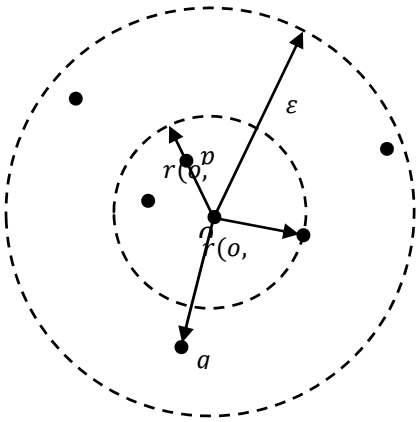


Image 19: Core-distance of sample o and reachability-distances $r(p,o)$ and $r(q,o)$ for $MinPts = 4$.

7.3.1 OPTICS Pseudo Code

For the sake of clarity, the pseudo code that can be found in the original OPTICS paper [13] is repeated in this subsection. OPTICS consists in its original implementation of two functions and an implementation of an ordered list, in which point samples are ordered

according to their reachability-distances. The first function calls the OPTICS implementation, the second one is an iterative process that adds new points to the cluster order. The first function is defined along the following lines:

```
OPTICS( SetOfObjects, ε, MinPts, OrderedFile)
    OrderedFile.open();
    FOR i FROM 1 TO SetOfObjects.size() DO
        Object := SetOfObjects.get(i);
        IF NOT Object.Processed THEN
            ExpandClusterOrder(SetOfObjects, Object, ε, MinPts,
                               OrderedFile);
    OrderedFile.close();
END;
```

The function `OPTICS` opens the output file, and calls the second function, `ExpandClusterOrder`, for every point sample that is not processed yet. `ExpandClusterOrder` processes and writes the current point and some neighboring points into the output file. `ExpandClusterOrder` is defined as follows:

```
ExpandClusterOrder(SetOfObjects, Object, ε, MinPts, OrderedFile)
    neighbors := SetOfObjects.neighbors(Object, ε);
    Object.Processed := TRUE;
    Object.reachability_distance := UNDEFINED;
    Object.setCoreDistance(neighbors, ε, MinPts);
    OrderedFile.write(Object);
    IF Object.core_distance <> UNDEFINED THEN
        OrderSeeds.update(neighbors, Object);
        WHILE NOT OrderSeeds.empty() DO
            currentObject := OrderSeeds.next();
            neighbors := SetOfObjects.neighbors(currentObject, ε);
            currentObject.Processed := TRUE;
            currentObject.setCoreDistance(neighbors, ε, MinPts);
            OrderedFile.write(currentObject);
            IF currentObject.core_distance <> UNDEFINED THEN
                OrderSeeds.update(neighbors, currentObject);
END;
```

`ExpandClusterOrder` takes one point as an argument and retrieves all points within its ε-neighborhood. Prior to calculations, it asserts the reachability-distance of each point to be undefined simply by setting it to this value. It further calculates the core distance and writes the point to the ordered file. Note that the reachability-distance of every first point that `ExpandClusterOrder` processes is undefined by design. Points with undefined reachability-distance either indicate outliers or cluster-borders. They can, however, be assigned to a cluster in a post processing step. If the core-distance of the first point is not undefined, i.e. the point is within some region of high density, its neighbors are inserted into a list `OrderSeeds` whose elements are stored in ascending order with respect to their reachability-distances. These reachability-distances are calculated in the course of the `update`-method of the `OrderSeeds` object. All points in the ordered list are processed like the first point until the list is empty and the control flow is handed back to the loop in the `OPTICS` function. Intuitively, the execution on the ordered list can be seen as a mixture

between breadth-first-search and depth-first-search; this part of the algorithm expands the cluster ordering in a density based way. The next piece of pseudo code depicts a possible implementation of `OrderSeeds::update`:

```
OrderSeeds::update(neighbors, CenterObject);
    c_dist := CenterObject.core_distance;
    FORALL Object FROM neighbors DO
        IF NOT Object.Processed THEN
            new_r_dist := max(c_dist, CenterObject.dist(Object));
            IF Object.reachability_distance = UNDEFINED THEN
                Object.reachability_distance := new_r_dist;
                insert(Object, new_r_dist);
            ELSE // Object already in OrderSeeds
                IF new_r_dist < Object.reachability_distance THEN
                    Object.reachability_distance := new_r_dist;
                    decrease(Object, new_r_dist);
END;
```

The method `OrderSeeds::update` calculates new reachability-distances for all unprocessed neighbors of the given sample point. If the points are not yet in the ordered list, they are inserted with their given reachability-distance. If they are already in the list, their newly calculated reachability-distance is compared to their already assigned reachability-distance. If the new reachability-distance is smaller, the sample point's value is updated and their position in the ordered list is corrected. By this means, OPTICS yields a density-based ordering of the points.

The cluster-ordering of the points that is generated by OPTICS can be illustrated in a reachability-plot like the ones found in Image 20. Neighboring points in feature space are likely to be associated to neighboring bars in the plot. The height of the bars depicts the reachability-values of the corresponding points. Points in a sparse region possess a big reachability-distance, therefore produce high bars and vice versa. Some points have an undefined reachability-value. Such points are marked as green bars in the image, which represents a reachability-value of ∞. In the lower plot, however, there is only the first point marked as unreachable, although it cannot be seen due to the scaling of the huge data set. The reason why there are no more unreachable points, is because ε is set to ∞ and thus the ε-neighborhood of every sample consists of every point in the feature space. Since, in the depicted data set, there are more 5435 feature vectors and thus more samples than $MinPts$ (which is two, in this example), every point has a $MinPts$'s neighbor, which is the nearest neighbor, in this case. Consequently, every point has a reachability-distance. This example shows that ε can always be set to ∞ and still provides good results. Concerning $MinPts$, one can see that the lower plot, in which $MinPts = 2$ is more jagged than the plot in the middle, for which $MinPts = 30$. The higher the $MinPts$ value the weaker impact of the depth-first-search aspect, which causes a single-link effect, thus the number of points in the ordered list is smaller and the ordering in the whole output list weaker.

In order to identify clusters, one can look for coherent valleys in the reachability-plot, as these identify regions of high density. Since there is no unique clustering for an arbitrary set of feature vectors, one can also visually identify nested valleys up to any depth. To cluster the data automatically, there are several ways to ways to identify clusters. One simple method is, to draw a horizontal threshold line through the reachability-plot at a certain height. Every reachability-value above that height would count as an outlier, which would in turn confine the valleys that would now represent the clusters. Another method is to

automatically find the n highest and most persistent peaks, i.e. the "sparsest" and most prominent points within the OPTICS ordered reachability-list, and to use these peaks as a cluster border. Both methods are implemented in this thesis' solution. The method that is to be used can be specified in a configuration file. Sander et al. [47] present a more sophisticated method for cluster extraction, but this method is not implemented in the solution. It is worthwhile to investigate it for future research. The naïve implementation of OPTICS grants a runtime complexity of $O(n^2)$ in the worst case. It is dependent on ε's actual value, with a smaller ε resulting in an algorithm speedup due to fewer points that are expected to be found in a local neighborhood. Also, the usage of spatial indices like k-d trees or binary space partitions can speed up the neighborhood queries. Since speed is not of big matter for the clustering stage of this thesis' solution such spatial indices are not implemented in the clustering application.

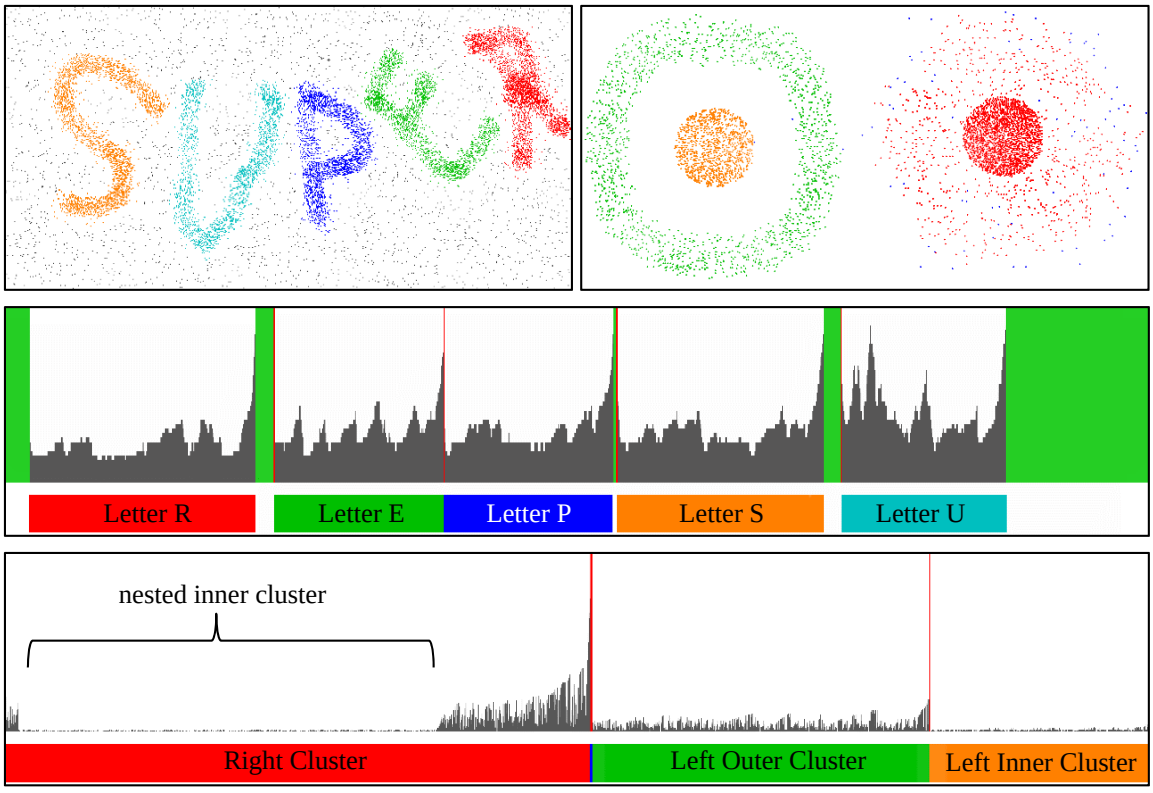


Image 20: Two data sets that are automatically clustered by the n most persistent histogram peaks of the OPTICS ordering. Thin vertical red bars in the graph plots mark theses peaks and hence indicate cluster borders.

Upper left: Five clusters of 11735 points, of arbitrary shape and surrounding noise. The graph in the middle depicts the reachability-plot of the OPTICS ordered points. Green bars mark points that have an undefined reachability-distance, these points are considered noise. One can see that density-connected points of the original image are ordered into the same range in the histogram. The parameters are set to $\varepsilon = 10$ and $MinPts = 30$.

Upper Right: Three clusters of 5435 points with different density. The right cluster contains a nested cluster of high density. A very small cluster of low density that surrounds the red cluster is marked in blue. The reachability-plot is in the bottommost image. The input parameters are set to $\varepsilon = \infty$ and $MinPts = 2$. Although the simple partitioning algorithm that seeks the most persistent histogram peaks cannot divide the nested red cluster, one can visually divide the cluster in the histogram plot by the reachability-distances: the region that corresponds to the right cluster has a big gap of small reachability-values. These values correspond to the dense points of the inner cluster.

8 IMPLEMENTATION

This chapter gives overview over the implementation of the application. Section 8.1 describes the design motives and the general architecture of the solution, the subsequent sections describe the implementation details of the solutions to the individual problems. Information about the specific usage of the applications can be found in the README file of the solution and in the applications.

8.1 Architecture Overview

This section gives a brief overview over technologies that are used for the practical part of the thesis. It states the programming language, used programming environment used libraries and tools for documentation purposes.

The solution to the actual problem is divided into two applications, The *FeatureGenerator* and the *Clusterer*. Two additional applications are implemented for evaluation purposes. The first one is simply called *Evaluation* and the fourth application is the *OPTICSAnalyzer*. *Evaluation* performs some routines assess the correctness of the clustering step. *OPTICSAnalyzer* serves as a tool to check the outcome of the OPTICS clustering manually. Several Batch scripts and Python scripts serve as automation tools for building and cleaning the solution. They are not further reported within this document.

8.1.1 Architecture Principles

The applications must be stable even in the case of errors, e.g. errors regarding access to image files and its data. The software must be further interoperable with other systems, therefore its input/output shall have a simple data format, preferably one that is well-known.

The solution separates the creation of feature vectors and their clustering in two different applications. This allows the user to apply other methods for each task, e.g. to use some third party application. The communication interface is a simple file driven realization. While the user can specify a set of program arguments through a configuration-file for each application, the applications create text-files, folder structures, image-files or symbolic links as output. The output of the *FeatureGenerator* application is further used as the input for the clustering application. The same holds for the *Evaluation* application that evaluates the clustering results. The communication on file-level is chosen because it is easy to implement and yet yields, through its openness, possibilities to check and change the results manually or with a third-party tool.

The applications are written in a mix of procedural and object-oriented style. When possible, code is written in form of classic functions. Only when object-oriented programming is beneficial, object-oriented constructs like classes are introduced (i.e. when object-orientation results in improved readability of source code or provides worthwhile stages of abstraction). The code is thus modular at some key points. Modules in the context of this work are classes or cohesive compounds of code that are part of the solution but do not depend on the rest of the solution except for the common types and functions.

The applications are not interactive, they provide live information via console output and log file.

8.1.2 Used Technologies

The programming language of choice is C++ x11, because it provides great freedom to programming style and makes modular software possible. It supports well known, well regarded and stable libraries for any task that comes up in the thesis. This includes computer vision tasks as well as more generic functions like file handling operations. C++ allows fine grained optimizations and allows to compile fast programs. Speed is of matter since the task of salient region detection and feature vector creation can be computationally expensive. In addition, the image database to handle is very large, so a fast implementation is can spare much time. Further on, C++ is one of the most widely used programming languages in the world and thus can be understood by many people and has a big palette of libraries to offer, including the most widespread and mature computer vision library, *OpenCV* [48].

The development environment that is used for the practical part of the thesis is the *Microsoft Visual Studio 2010*. Assembly configurations for both 32 bit and 64 bit assemblies are provided for the result applications.

The source code is documented using the documentation tool *Doxygen* [49].

Besides the C++ standard template library, different programming libraries, are used. The *boost* library [50], version 1.56, namely its sub-packages *chrono*, *filesystem*, *log* and *program_options* are put to use. For coloring and formatting of console output the small and platform-independent library *rlutil* [51] is used.

For image processing purposes *OpenCV*, version 2.49 [48] is used.

Small utility scripts to compile or clean the Microsoft Visual Studio solution or the output files are written as Batch or Python scripts.

Additionally, the *OPTICSAnalyzer*, a small GUI-driven utility application for evaluation of the OPTICS clustering method is written in C#. It targets the *.NET Framework 3.5 Client Profile* and uses Windows Forms.

All applications have been tested under Window 7. The core applications, however, are designed to run on all popular platforms.

8.2 The FeatureGenerator Application

The *FeatureGenerator* crawls an image database and creates a descriptor (Section 2.3, p. 6, Chapter 6, pp. 24) for each image. As input, the application takes a configuration file that specifies several parameters and paths to all needed files and directories. The *FeatureGenerator* also creates several output files. For a complete listing of input parameters and output refer to the README file or directly to the *FeatureGenerator* implementation.

The *FeatureGenerator* consists of a core part that parses input parameters, crawls the image database and writes the output to files. There are two modules that can vary the behavior of the *FeatureGenerator*. One module is responsible for the saliency detection, the other one controls the creation of the descriptors for each image. Each module contains an abstract class that specifies the interface for both the saliency detectors and descriptor extractors, respectively. The solution provides one `SaliencyFilters` saliency detector class and three feature extractor classes: `ContourExtractor`, `HistogramExtractor` and `ContourHistogramExtractor`. The latter one combines the two former extractors. The saliency detector and each feature extractor is registered in the applications core system, so the implementations can be changed via configuration file after compile time. Image 21 sketches the coarse relationships of the modules in the application.

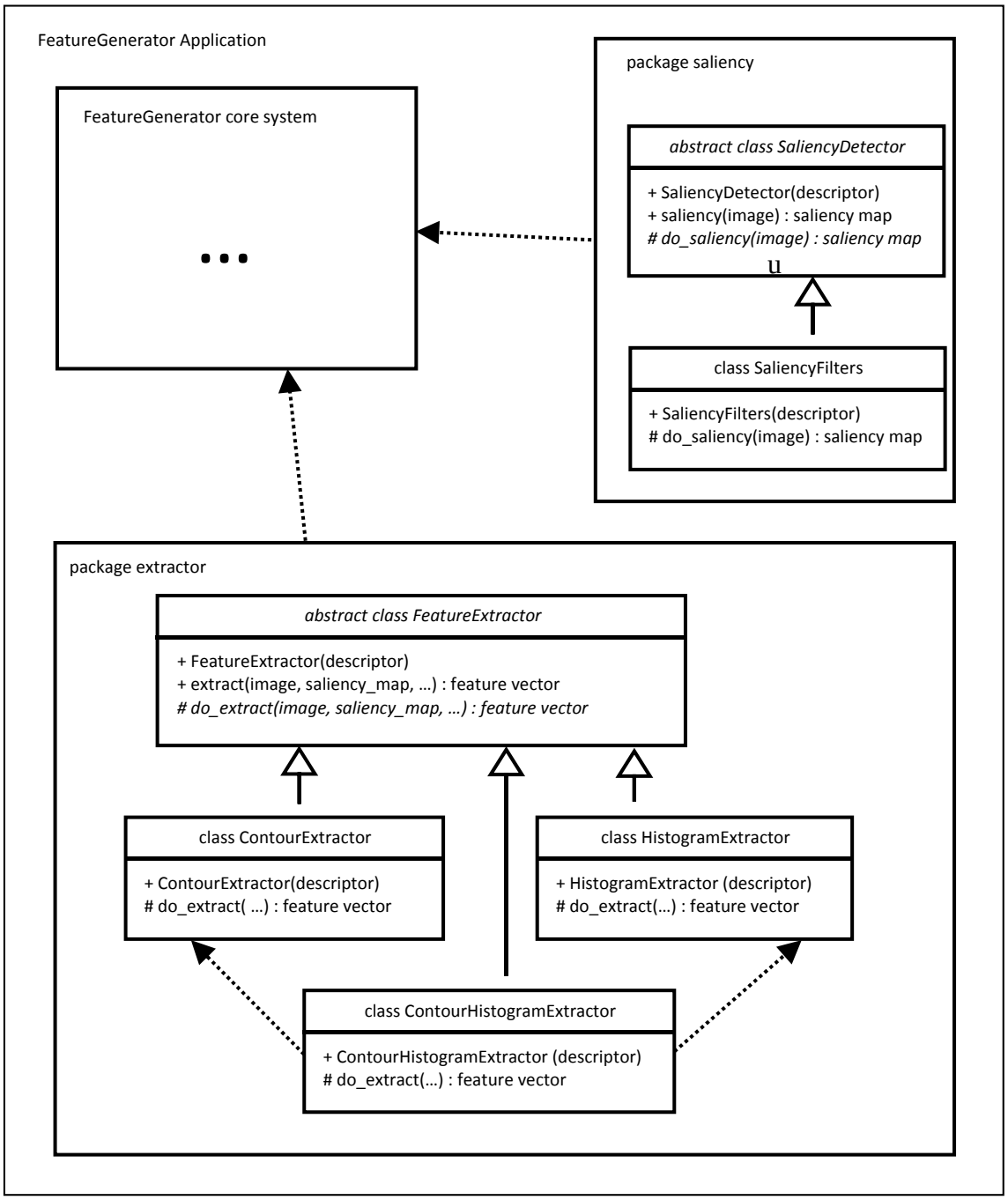


Image 21: The main parts of the *FeatureGenerator* application and modules. The Saliency Detector is implemented in the form of the *Saliency Filters* implementation. The feature extractor is implemented by three interchangeable classes: the `ContourExtractor`, the `HistogramExtractor` and the `ContourHistogramExtractor`. The sketch differs slightly from the real implementation for the sake of readability.

8.3 The Clusterer Application

The *Clusterer* takes a list of image-paths and their corresponding descriptors and clusters the images according to their descriptors. As additional input, it receives a configuration file in which several parameters can be specified. Several files are created by the *Clusterer*. For a specified listing of input parameters and output refer to the README file or directly to the *Clusterer* implementation.

The *Clusterer* consists of a very small core system, containing boiler plate code, helper functions and the main routine. Despite this core system, there is a package named `clusterer` that stores the abstract class `Clusterer`. This class provides the interface for algorithms that can be used within the application. Similar to the modularization in the *FeatureGenerator*, subclasses of `Clusterer` can be registered in the application in order to use them. Three clustering algorithms are implemented within the classes `KMeansClusterer`, `OutlierClusterer` and `OPTICSClusterer`. Image 22 sketches the coarse relationships of the modules in the application.

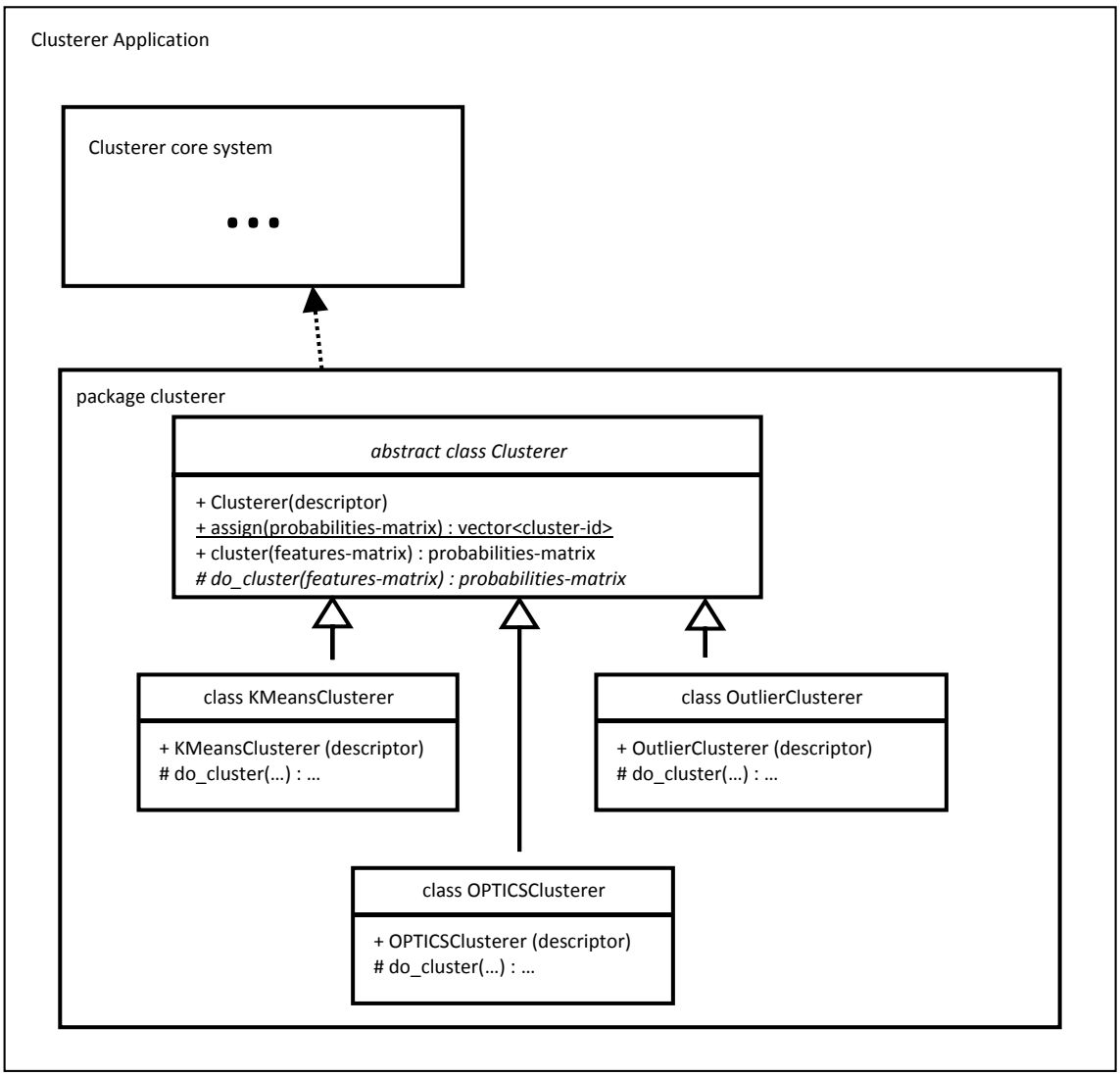


Image 22: Main parts of the *Clusterer* application and the `clusterer` package. Three `Clusterer`-classes are implemented: `KMeansClusterer`, the `OutlierClusterer` and the `OPTICSClusterer`. The sketch differs slightly from the real implementation for the sake of readability.

8.4 The Evaluation Application

The *Evaluation* application is a small program that takes some of the files that have been created by the *Clusterer* as well as the paths to the original test database. It is used in conjunction with the images of the Caltech-256 image database [7], which consists of 257 (sic!) image classes. Each image file is assigned to exactly one class. The correct class membership of every image can be inferred from the folder structure of the image database. Thus, by evaluating the cluster-assignments with help of the true classes that are originally assigned to each image within the Caltech-256 the correctness of the clustering procedure can be assessed. Any image database that is structural equivalent to the Caltech-256 image database (e.g. the Caltech-101 image database [52]) can be used instead of the Caltech-256. The results of the *Evaluation* application are logged and written to a separate file. The experiments are explained in detail in Chapter 10.

8.5 The OPTICSAnalyzer Application

The *OPTICSAnalzyer* is a small, self-explanatory GUI-tool for manual evaluation of OPTICS clustered images. It shows the reachability plot for the clustering. One can select parts of the graph and inspect the images that lie within the selection. In this way, one can inspect the OPTICS clustered results and try a manual clustering on the OPTICS ordered structure. One can further define an outlier threshold to identify outliers in the sense of OPTICS, i.e. feature space points that lie in sparse regions. *OPTICSAnalyzer* is written in C# and is not intended to be platform-independent. Image 23 depicts the OPTICS analyzer UI window.

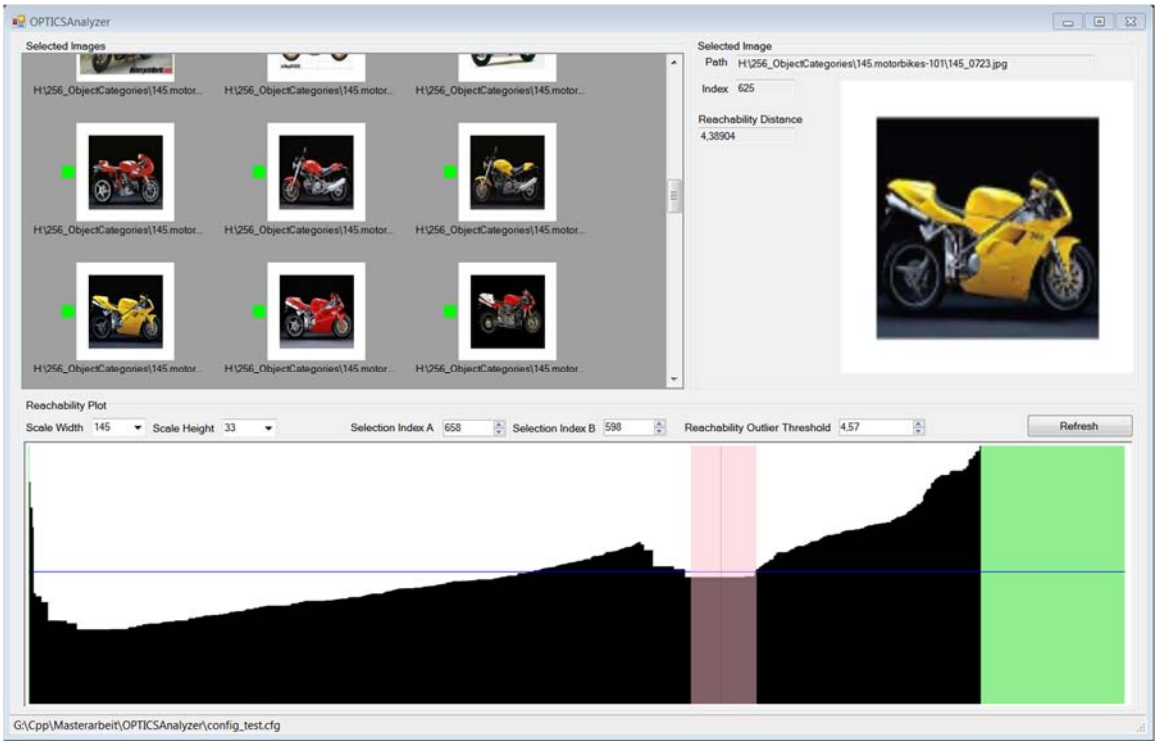


Image 23: The *OPTICSAnalyzer* UI. The lower part depicts the reachability plot of an OPTICS ordered set of samples. The upper left shows the images that correspond to the selected region, the upper right part gives detailed information about one selected image.

9 SALIENCY FILTERS RESULTS

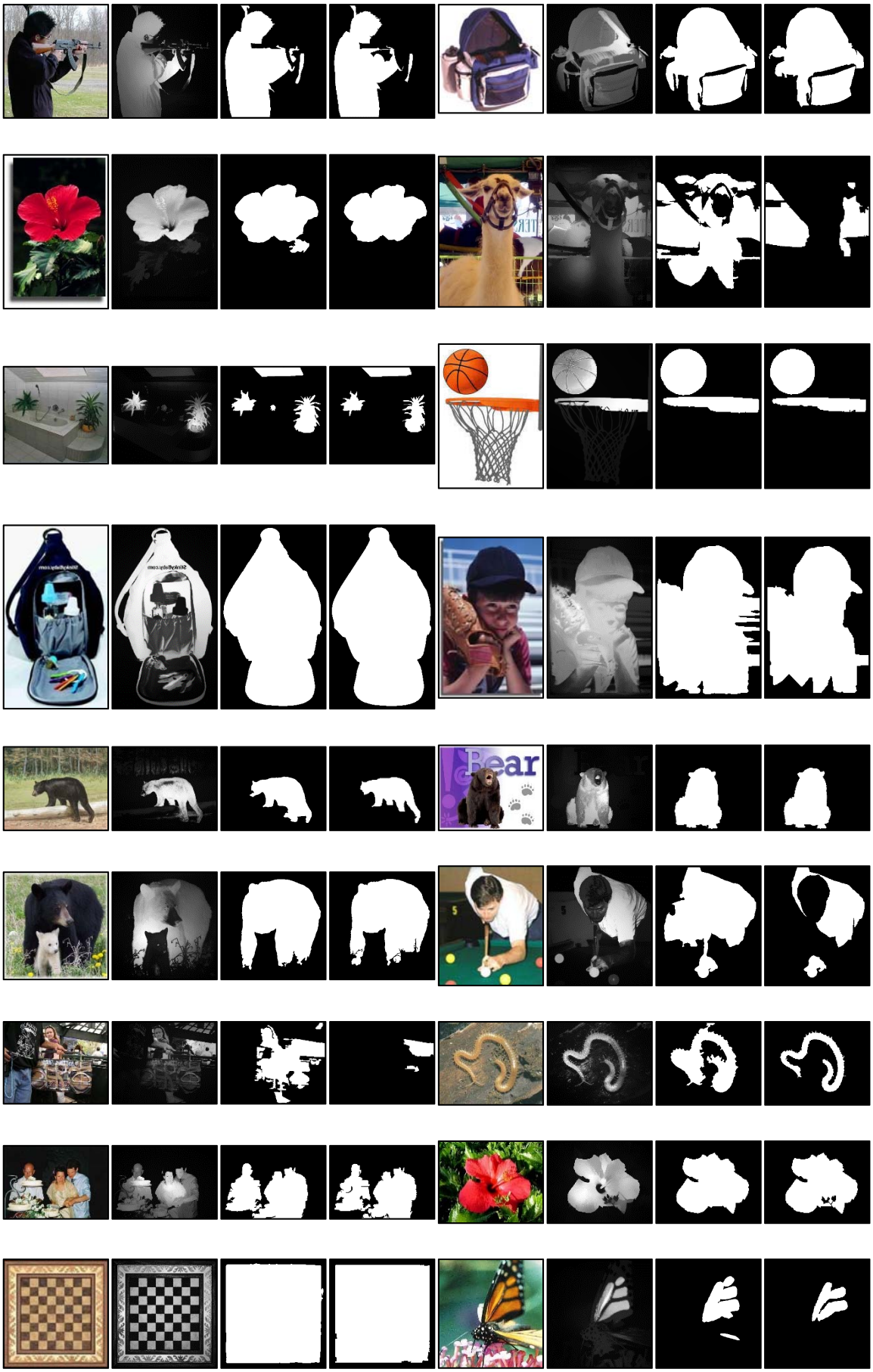
This chapter presents some exemplary results of the *Saliency Filters* algorithm. The following images (Image 24) below depict several images from the Caltech-256 image database [7] and their corresponding saliency maps created by the *Saliency Filters* algorithm. Also, binary masks which are created from the saliency maps by application of a simple threshold and also by application of the *GrabCut* algorithm are shown. Prior to the masking with simple threshold, the saliency maps are blurred with a 7×7 Gaussian blur in order to erase irritating artifacts at the borders of the masks and to create big and coherent segments. Due to this blurring step, the binary masks created by threshold-application inhibit rounder mask borders than the binary masks created with *GrabCut*. In the case of threshold-application, saliency map pixels with 15% brightness or above are set to white (which means these pixels are interpreted as foreground). The foreground-probability parameter for the *GrabCut* segmentation is also set to 15%. The prior maps for *GrabCut* are the saliency maps. The values have been experimentally determined. The various variable settings for the *Saliency Filters* (explained in Chapter 5) can be found in Table 1 below.

***Saliency Filters* settings:**

Parameter Name	Value
Number of Superpixels	400
Number of k-Means Iterations for SLIC	5
Superpixel Color Weight	1
Positional Sigma σ_p	0.25
Color Sigma σ_c	20.0
Up-sampling Parameter k	3
Minimum Saliency Percentage	10%
Color Sensitivity Parameter α	$\frac{1}{3}$
Position Sensitivity Parameter β	$\frac{1}{3}$

Table 1: The values of the parameters of *Saliency Filters* algorithm used for salient region extraction.

Saliency Filters can yield unwanted results when the foreground object has multiple colors, the lighting conditions are bad (e.g. in case of reflections on metallic surfaces) or the background consists of many different colors and is highly textured. In order to reduce the effect of difficult lighting conditions, improvements can possibly be made when the brightness-channel is treated separately from the other two color space channels. Another additional approach for improvement arises, when it comes to photographs or images that exhibit depth of field. Since the object of interest in most cases is focused, one could also use the blurriness of the background to distinct image foreground object from the rest of the image.



Original | Saliency Map | Threshold | GrabCut | Original | Saliency Map | Threshold | GrabCut

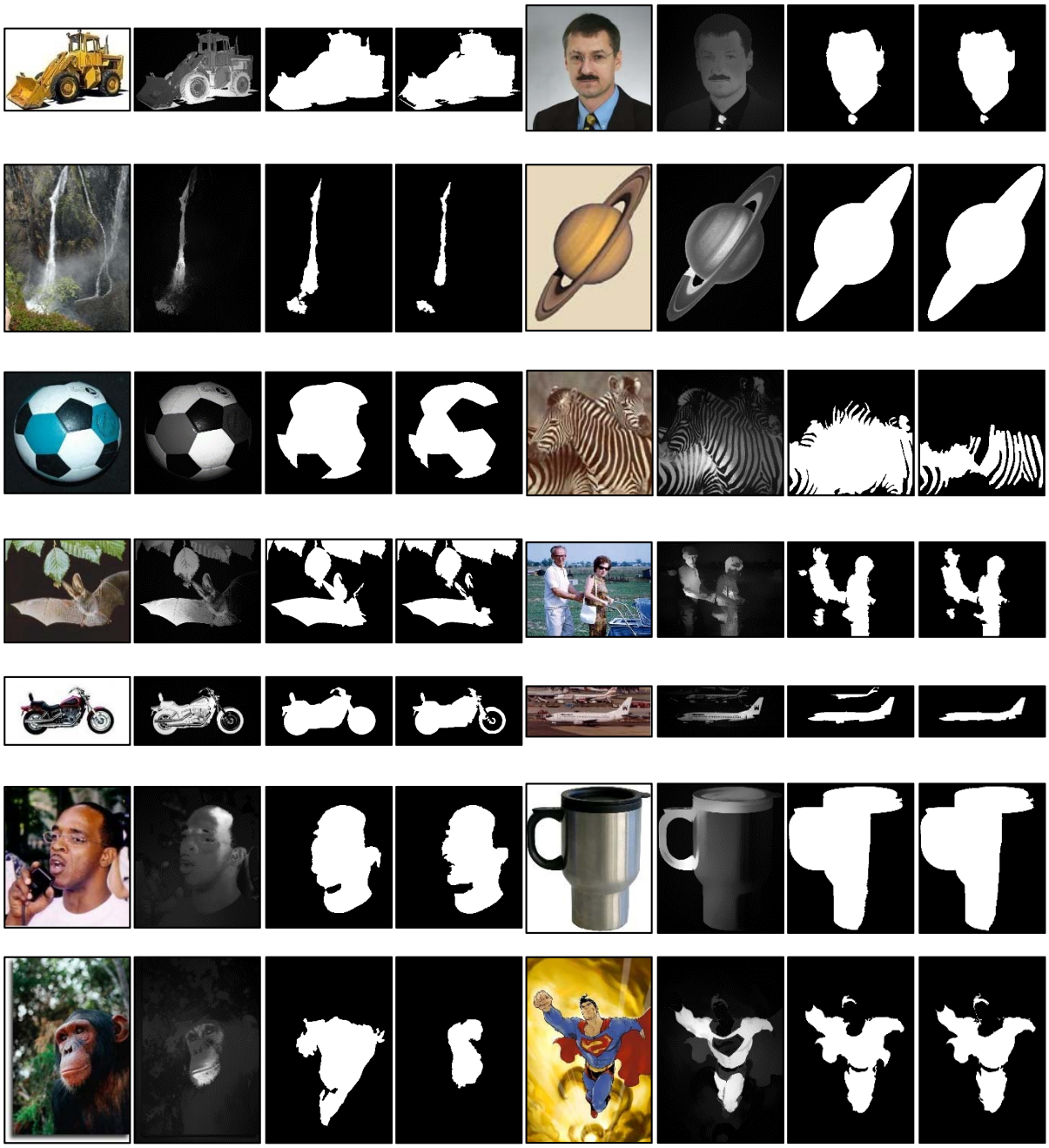


Image 24: From left to right: Several images from the Caltech-256 image database, corresponding saliency maps created with the *Saliency Filters* algorithm and extracted binary masks, created by simple threshold-application and by application of *GrabCut*. In both cases, if pixels of the input the saliency map have a 15% brightness, they are considered foreground. The binary masks created with simple threshold inhibit rounder segment-borders, because, prior to the threshold-application, the saliency maps are blurred by a 7×7 Gaussian filter in order to reduce the impact of small artifacts and to create big, coherent segments. The *Saliency Filters* algorithm yields good results, if the background contains evenly distributed colors. *Saliency Filters* fails, if the object of interest cannot be clearly determined, i.e. when more than one foreground object exists, or the foreground objects have multiple colors or when the foreground objects are only partially lit, e.g. with one side lit and the other side shadowed. One can see that, due to the various viewing angles and forms an object can have, the form of the binary mask can vary widely within one category and may thus be unsuitable to recognize objects in the very general case.

10 EVALUATION STUDIES

After the first chapters provided an overview over the problem, the related work and finally the proposition of the solution, this chapter introduces the experiments that are conducted to assess the quality of the clustering results. The first section provides general information about the scoring measures that are used to assess the quality of the descriptors and the clusterings. These measures are done by means of precision and recall and F-measure, which is a combination of the former two. Section 10.2 presents the Caltech-256 image database [7], the image database that serves as the benchmark database in the experiments. The sections three to five provide information about the settings of the *Saliency Filters* algorithm, the settings of the descriptors and the settings of the clusterers. Section 10.4 provides information about the experiments that involve the descriptors and the k-Means and OPTICS clustering. Section 10.5 provides information about the experiments that involve the Outlier clusterer, which, since it does not create a classic cluster-partitioning of the images, is evaluated separately.

10.1 Quality Measurement Approach

In order to assess the quality of the clustering that is created by the descriptors and clusterers, the clustering results are compared to the original image-categories that are given by a benchmark image set. In this thesis, the benchmark image set is the Caltech-256 image database [7]. The Caltech-256 image database consists of 257 sets of images, each with each image in the same set depicting an object of the same category. The Caltech-256 image database is covered in more detail in Section 10.2.

To enable comparisons of the clusters to ground truth categories, the clusters are mapped to the categories. Each cluster that is created is assigned to exactly one category of the benchmark set, specifically the category whose images occur most often within the cluster. For example, a cluster that contains six images of dogs and three images of cats will be compared to the ground truth category “dog”. The clustering performance can be measured on the resulting binary classification problem – “dog”, “no dog” by means of precision, recall and F-measure (Section 10.1.1). Each cluster that is created undergoes a precision and recall measure. The average of the values for all clusters is used to assert the quality of one experiment. Image 25 sketches a schematic binary classification.

With the problem-transformation to a binary classification, several clusters can be assigned to the same category. Concerning the “dog”-example, there may be another cluster that contains four images of dogs and two images of manatees. This cluster is also compared to the ground truth category “dog”. The clusters that are compared to the same category (e.g. "dog") are kept distinct (that means they are not melded together). It is considered faulty when different clusters relate to the same category. When two or more clusters are relate to the same category, the recall values for all these clusters will be comparably small, i.e. none of these cluster-category mappings can ever yield a recall value of 100%. This aspect of precision and recall measures is thus beneficial in order to penalize an incorrect division of the ground truth categories in the clustering results. This means, on the other hand, that a smaller number clusters compared to the actual number of categories can result in high recall values. In order to identify incorrect clusterings despite high recall values, the number of automatically identified clusters is stated in the results. Precision, recall and F-measure are explained in the next subsection.

In addition to precision, recall and F-measure, the distributions of the number of images among the clusters, i.e. how much images are found within each cluster, are evaluated.

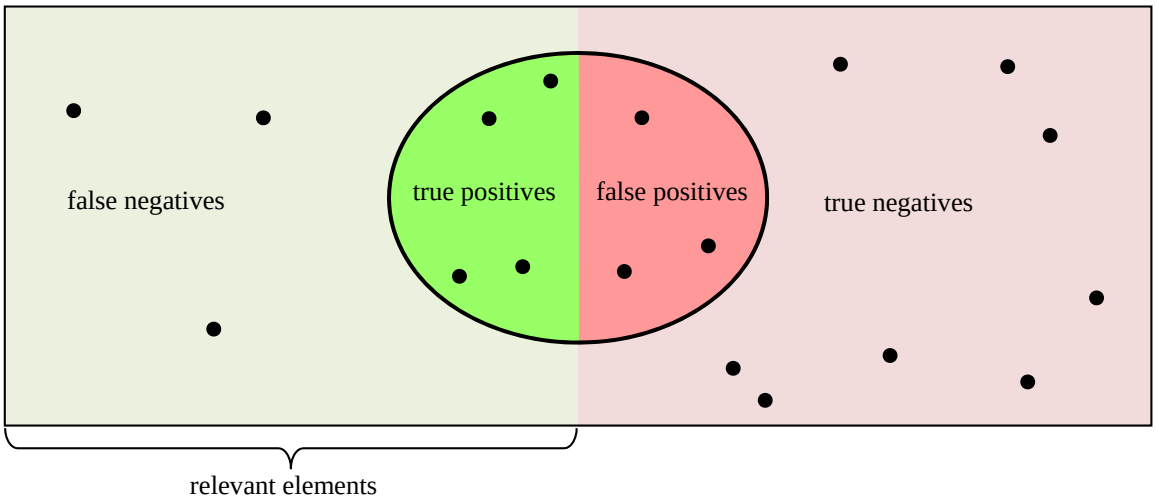


Image 25: An example binary classification. The ellipse sketches the retrieved elements, elements on green background represent all elements that are relevant.

10.1.1 Precision/Recall Measures and F-Measure

Precision (also called positive predictive value) is the fraction of retrieved instances that are relevant, i.e. the fraction of true positives to all retrieved elements. Recall (also known as sensitivity) is the fraction of correctly retrieved elements to all elements that are truly relevant:

$$precision = \frac{|true\ positives|}{|true\ positives| + |false\ positives|} \tag{15}$$

$$recall = \frac{|true\ positives|}{|true\ positives| + |false\ negatives|} \tag{16}$$

Values for precision and recall can thus lie within a range of [0..1]. If several clusters are assigned to the same category, the recall rates won't reach a value of 1 for each of these clusters. This characteristic is used to penalize the division of ground truth classes.

The F-measure (also called $F_1 - Score$) is the harmonic mean of precision and recall:

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{17}$$

10.2 The Caltech-256 Image Database

The Caltech-256 image database [7] is used as a benchmark data base in all experiments and consists of 30607 images in 257 different sets of images, with each image of one set portraying an object of the same category. An exception is the 257th category "clutter" that contains images of random small cutouts from big photographs and no foreground objects.

Each category of the Caltech-256 contains 80 to 800 images. The images represent a wide variety of natural and artificial objects and diverse lighting conditions, poses, backgrounds, image sizes and camera metrics [7], not only among the categories but also within the categories. Image 26 depicts some example images from three categories. One can see that the images are not only photographs but also drawings or animated images.

The since the category “clutter” contains unspecified image contents with no foreground object, it is not included in the test scenarios.

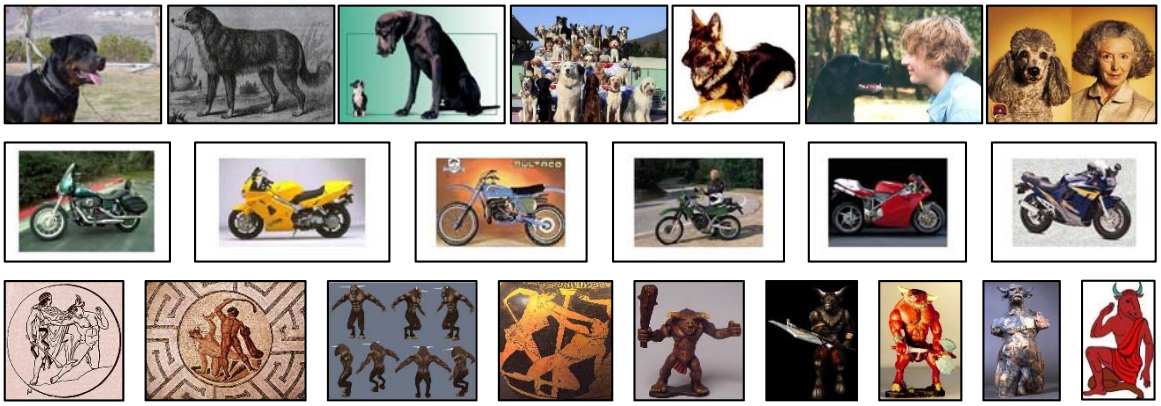


Image 26: Some images of the Caltech-256-categories "dog" "motorbikes-101" and "minotaur".

Because the Caltech-256 image dataset consists of a wide variety of categories, and also within-category differences between the images can be big, different series of experiments are conducted with different subsets of the Caltech-256 categories. One series runs tests on the whole Caltech-256 dataset (except for the category “clutter”), one series runs tests only on a small subset of categories with comparably small within-category-differences. The big subset is furthermore abbreviated BIG, the small subset is furthermore abbreviated $SMALL$.

In order to test the Outlier clusterer, categories are filled with a small number of images that do not belong to the categories. The following listing states the 14 categories that are used in the $SMALL$ subset:

- 015.bonsai-101
- 022.buddha-101
- 091.grand-piano-101
- 114.ibis-101
- 118.iris
- 121.kangaroo-101
- 123.ketch-101
- 129.leopards-101
- 137.mars
- 145.motorbikes-101
- 172.revolver-101
- 201.starfish-101
- 204.sunflower-101
- 253.faces-easy-101

The BIG data set contains 29780 images in total, the $SMALL$ data set contains 2574 images in total.

10.3 Experiment Settings

The following subsections describe the settings of the three configurable modules for every conducted experiment: the saliency detection, the descriptor generation and the clustering.

10.3.1 Saliency Filters Settings

The settings for the saliency detection with help of *Saliency Filters* are specified in Table 1 on Page 39. In order to extract binary saliency masks, *GrabCut* is used with a foreground probability of 15%. The minimum size of a valid salient region is set to 400 pixels. If no salient region within in image has a size of 400 pixels or more, the image is considered garbage, which means no descriptor will be created for this image and it is, consequently, not further processed by the clustering algorithms.

10.3.2 Descriptor Generator Settings

The solution provides three separate descriptors: *Fourier Shape Descriptor* (abbreviated $CONTOUR$), *Color Histogram Descriptor* (abbreviated $HISTOGRAM$) and the *Combined Fourier Shape and Color Histogram Descriptor* (abbreviated $CONTOUR - HISTOGRAM$). Because the $HISTOGRAM$ descriptor can store either color information of the whole image or only of the salient region, experiment-combinations for both variants are conducted (the variants are abbreviated $HISTOGRAM_{PARTIAL}$ and $HISTOGRAM_{COMPLETE}$, respectively). The experiments with the $CONTOUR - HISTOGRAM$ descriptor are also conducted for both variants ($CONTOUR - HISTOGRAM_{PARTIAL}$ and $CONTOUR - HISTOGRAM_{COMPLETE}$). The option for auto-correlation of the histogram-based descriptors is deactivated in all experiments, because the images of the Caltech-256 are too general to allow invariance in terms of global color changes like color shifts. This option can prove helpful in case of a special database consisting only of artificial objects, because artificial objects of same category are more likely to vary in color, e.g. cars of the same type but with different color [53].

The $CONTOUR$ descriptor takes just one argument, which is the number of Fourier components used to describe the shape of the biggest salient region. The number of Fourier components is set to 40, which includes the first two components that determine translation and scaling. Because these first two Fourier components are erased in the normalization process, the number of dimensions of the $CONTOUR$ descriptor is 38.

The descriptors $HISTOGRAM_{PARTIAL}$ and $HISTOGRAM_{COMPLETE}$ contain 10 bins for each hue-, saturation- and color-value channel, which sums up to 30 dimensions in total. As stated above, the auto-correlation is deactivated within all experiments.

The $CONTOUR - HISTOGRAM$ descriptor variants combine the $CONTOUR$ descriptor with the $HISTOGRAM$ descriptor variants. The number of Fourier components is set to 40 (the first two components will be erased in the normalization process). The number of histogram bins for each hue-, saturation- and value-channel are each set to 10, making 30 histogram dimensions in total. The total number of dimensions of the $CONTOUR - HISTOGRAM$ descriptors is thus 68.

The values for the settings of the five descriptor variants can be found in Table 2.

Descriptor variants and settings:

Descriptor	Parameter Name	Value
$CONTOUR$	# Fourier components	40 (effectively 38)
$HISTOGRAM_{PARTIAL}$	# hue bins	10
	# saturation bins	10
	# value bins	10
	use channel-wise autocor-relation	false
	use color histogram of whole image	false
$HISTOGRAM_{COMPLETE}$	# hue bins	10
	# saturation bins	10
	# value bins	10
	use channel-wise autocor-relation	false
	use color histogram of whole image	true
$CONTOUR - HISTOGRAM_{PARTIAL}$	# Fourier components	40 (effectively 38)
	# hue bins	10
	# saturation bins	10
	# value bins	10
	use channel-wise autocor-relation	false
	use color histogram of whole image	false
$CONTOUR - HISTOGRAM_{COMPLETE}$	# Fourier components	40 (effectively 38)
	# hue bins	10
	# saturation bins	10
	# value bins	10
	use channel-wise autocor-relation	false
	use color histogram of whole image	true

Table 2: The specific settings for the different descriptor-variants that are used in the experiments.

10.3.3 Clusterer Settings

The k-Means clusterer and the OPTICS clusterer are set up with different settings for experiments with the BIG and the $SMALL$ data sets. The implementation of the k-Means clusterer takes one integer as input argument that specifies the anticipated number of clusters. The k-Means algorithm runs until convergence.

The OPTICS clusterer takes five arguments. The first argument specifies whether to look for the most outstanding clusters, i.e. the clusters that are partitioned by the most persistent peaks in the reachability-plot or for all clusters that are divided by values in the reachability-plot that exceed a given threshold. The second argument specifies the value for the OPTICS parameter ε. The third parameter is the OPICS argument $MinPts$. The number of clusters is specified by the fourth parameter. The fifth parameter is the outlier threshold. If the reachability-distance of a sample point exceeds this threshold, the sample is marked as an outlier. Set to ∞, this parameter effectively specifies that no samples are marked as outliers and all samples are grouped into a certain cluster. The parameter values that are used in the k-Means experiments are specified in Table 3, the parameter values for the OPTICS clustering experiments can be found in Table 4. In order to apply an as much as possible universal parameter setting, $\varepsilon = \infty$ and $min_{pts} = 50$ in all experiments. Tweaking the values of ε and $MinPts$ with help of knowledge from prior experiments can result in better clustering performance, but would impair the generality of the experiment settings.

The Outlier clusterer (abbreviated $OUTLIER$) is tested with ten different settings. The only parameter that can be set for the Outlier clusterer is the number of outliers to be found. It will be set to an integral number within $\{1,..,10\}$ in the experiments. The naming scheme of the different variants is $OUTLIER_i$, for instance $OUTLIER_3$ for the variant that finds the three strongest outliers.

k-Means clusterer settings:

Dataset	Parameter Name	Value
BIG	# clusters	270
$SMALL$	# clusters	20

Table 3: Parameter settings for the k-Means clusterer for experiments on the *BIG* dataset and the *SMALL* dataset.

OPTICS clusterer settings:

Dataset	Parameter Name	Value
BIG	mode	find n clusters
	ε	∞
	$MinPts$	50
	# clusters	270
	outlier threshold	∞
$SMALL$	mode	find n clusters
	ε	∞
	$MinPts$	50
	# clusters	20
	outlier threshold	∞

Table 4: Parameter settings for the OPTICS clusterer for experiments on the *BIG* dataset and the *SMALL* dataset.

Outlier clusterer settings:

Dataset	Parameter Name	Value
all	# outliers	$\{1,..,10\}$

Table 5: Parameter settings for the Outlier clusterer for experiments.

10.4 Experiments on Image Clustering

In order to test and evaluate the quality of the five different salient region descriptors (respectively the five variants introduced in Section 10.3.2) and the three clustering algorithms, twenty experiments with different settings are conducted. The experiments permute over the combinations of the image data sets BIG and $SMALL$, the five descriptor variants and the two clustering algorithms k-Means clustering and OPTICS clustering. The Outlier clustering method will be examined separately, because, unlike the other clustering algorithms, Outlier clustering does not seek to produce a category-grouping and thus the results cannot compared the results of the other clustering algorithms. Information about separate experiments for the Outlier clusterer are presented in Section 10.5.

10.4.1 Experiment Combinations

Since there exist five descriptors, two clusterers and also two image test sets, several experiments with different combinations are conducted:

$$\binom{BIG}{SMALL} \times \begin{pmatrix} CONTOUR \\ HISTOGRAM_{PARTIAL} \\ HISTOGRAM_{COMPLETE} \\ CONTOUR - HISTOGRAM_{PARTIAL} \\ CONTOUR - HISTOGRAM_{COMPLETE} \end{pmatrix} \times \binom{k - Means}{OPTICS}.$$

The combination of all variants yields 20 combinations. They are listed and numbered in Table 7.

10.5 Experiments on Outlier Clustering

In order to test and evaluate the Outlier clusterer, five series of respectively 100 experiments are conducted. In each experiment, one random category of the Caltech-256 image set is augmented by $o \in \{1,..10\}$ random images of other categories. These o images serve as outliers that shall be detected by the Outlier clusterer. For every combination of descriptor variant and category, ten experiments are conducted, each time the number of outliers is incremented by one. The experiments count the number of correctly detected outliers and calculate their mean in order to show the hit rate of the Outlier clusterer for each descriptor variant and for each number of outliers. The number of outliers is known by the beforehand.

In the following, the augmented image categories are denoted as $CAT_{n,o}$. The subscript n denotes the index of the Caltech-256 category that is used, the subscript o denotes the number of random outlier images. For example, $CAT_{212,7}$ depicts the 212th category "teapot" and is augmented by seven random outlier images that do not correspond to that category. This means, outlier images do not belong to the original category which they augment. Each series uses another descriptor variant, which makes 500 experiments in total:

$$\begin{pmatrix} CONTOUR \\ HISTOGRAM_{PARTIAL} \\ HISTOGRAM_{COMPLETE} \\ CONTOUR - HISTOGRAM_{PARTIAL} \\ CONTOUR - HISTOGRAM_{COMPLETE} \end{pmatrix} \times \begin{pmatrix} CAT_{a,1} \\ \vdots \\ CAT_{a,10} \\ CAT_{j,1} \\ \vdots \\ CAT_{j,10} \end{pmatrix} * \begin{pmatrix} OUTLIER_1 \\ \vdots \\ OUTLIER_{10} \end{pmatrix}.$$

Note that, for instance, $CAT_{a,10}$ does not necessarily contain the outlier that is put into $CAT_{a,1}$. The outlier images are fixed for each $CAT_{n,o}$, but do not relate to the outliers in another augmented category $CAT_{n,p}$.

The categories that are used for evaluation with the outlier clusterer are chosen randomly. The complete augmented categories are not listed here for the sake of brevity. The complete descriptors files can be found in the *Experiments* folder of the practical solution. The following Table 6 lists the categories that are used:

Categories used for Outlier clustering tests:

Category index	Category name	# images per category
22	buddha-101	97
37	chess-board	120
48	conch	103
109	hot-tub	156
168	raccoon	140
183	sextant	100
196	spaghetti	104
212	teapot	136
216	tennis-ball	98
232	t-shirt	358

Table 6: The 10 categories on which Outlier clustering tests are conducted.

The 20 experiments on image clustering:

№	Image Set	Descriptor	Clustering algorithm
1	BIG	$CONTOUR$	$k-Means$
2	BIG	$HISTOGRAM_{PARTIAL}$	$k-Means$
3	BIG	$HISTOGRAM_{COMPLETE}$	$k-Means$
4	BIG	$CONTOUR - HISTOGRAM_{PARTIAL}$	$k-Means$
5	BIG	$CONTOUR - HISTOGRAM_{COMPLETE}$	$k-Means$
6	BIG	$CONTOUR$	$OPTICS$
7	BIG	$HISTOGRAM_{PARTIAL}$	$OPTICS$
8	BIG	$HISTOGRAM_{COMPLETE}$	$OPTICS$
9	BIG	$CONTOUR - HISTOGRAM_{PARTIAL}$	$OPTICS$
10	BIG	$CONTOUR - HISTOGRAM_{COMPLETE}$	$OPTICS$
11	$SMALL$	$CONTOUR$	$k-Means$
12	$SMALL$	$HISTOGRAM_{PARTIAL}$	$k-Means$
13	$SMALL$	$HISTOGRAM_{COMPLETE}$	$k-Means$
14	$SMALL$	$CONTOUR - HISTOGRAM_{PARTIAL}$	$k-Means$
15	$SMALL$	$CONTOUR - HISTOGRAM_{COMPLETE}$	$k-Means$
16	$SMALL$	$CONTOUR$	$OPTICS$
17	$SMALL$	$HISTOGRAM_{PARTIAL}$	$OPTICS$
18	$SMALL$	$HISTOGRAM_{COMPLETE}$	$OPTICS$
19	$SMALL$	$CONTOUR - HISTOGRAM_{PARTIAL}$	$OPTICS$
20	$SMALL$	$CONTOUR - HISTOGRAM_{COMPLETE}$	$OPTICS$

Table 7: The 20 experiments: Combinations of descriptors and clustering algorithms and the two image sets. These are the combinations that are tested in the course the clustering experiments. The numbers in the left column serve as aliases for the individual experiments. Experiments 1 to 10 are conducted on the BIG image set, experiments 11 to 20 are carried out on the *SMALL* image set.

11 EVALUATION STUDIES RESULTS

This chapter provides the results of the experiments that are introduced in Chapter 10. The first two sections state and discuss the results for images clustering, the sections three and four give insights into the results of the outlier clustering experiments.

11.1 Results for Image Clustering

Table 8 on Page 54 depicts the precision, recall rates and the F-measures for the 20 experiments on image clustering. Image 27 on Page 55 depicts the results of Table 8 in form of a bar plot. The complete configuration files and log-files of the evaluation applications can be found in the folder *Experiments* in the solution folder.

One can see, that working with histogram values yields better results than working solely with contour values, but the values are generally small in all cases (F-measure below 0.5 in all cases). The experiments that work on the $SMALL$ image database subset (experiments 11-20) achieve generally higher precision, recall and F-measure values. This can be due to the comparably small amount of images and the low within-class variances of the image categories used in the $SMALL$ image set.

The experiments 11 to 15 show a very high recall rate with values ~0.9, which can be explained by the fact that only one cluster was found by the hierarchical k-Means algorithm. The recall values are still <1, since a small number of the processed images is rated as garbage, which means no salient region could be found within them. These images have not been considered in the clustering step and could there not be found in a cluster.

The number of clusters is generally small compared to the number of underlying categories, especially in the hierarchical k-Means experiments 11 to 15 and the OPTICS related experiments 6 to 10. Since the number of automatically retrieved clusters is, except in the cases of the experiments 1 to 5 and 16 to 19, much lower than the number of original categories, the clustering results are expected to be of poor quality in most cases. Also, the values for precision, recall and F-measure are expected to be biased due to the small number of retrieved clusters and the high rate of clusters that contain only one or a few images. A small amount of images in a cluster (e.g. one image) results in high precision value. A very big cluster is likely to result in a high recall value. Generally, the experiments on the $SMALL$ image set yield higher precision and recall values than the experiments on the BIG image set.

In all experiments, the images are distributed among the clusters according to the Power-Law Distribution [54], which means that a fraction of clusters contain the majority of all samples, while majority of clusters contains only a very small number of samples. The distribution of the images among the clusters is depicted in Image 30 and Image 31. In the images, the clusters are ordered according to the number of contained elements. The ordered amount of elements per category in the BIG and $SMALL$ data sets is shown in Image 28. Again, the categories are ordered according to their sizes. One can see that the images in the original categories are more evenly distributed than within the clusterings and most categories contain a similar amount and also a minimum amount of images. The plots in Image 30 and Image 31 show the number of images within one cluster and do not display, which images can be found within one certain cluster.

In the experiment series related to k-Means on the BIG image set (experiments 1 to 5), the two variants of the $CONTOUR - HISTOGRAM$ descriptor (experiments 4 and 5) tend to yield better results regarding precision than the other three descriptor-types. However,

since the precision, recall and F-measure values are strongly influenced by the sizes of the retrieved clusters, no reliable comparison of the descriptors can be made. The plots of the cluster-size distributions related to the $CONTOUR - HISTOGRAM$ descriptors (Image 30, experiments 4, 5) show a more extreme Power-Law distribution than the plots that are related to the other descriptors. That means, compared to the experiments 1, 2 and 3, the samples in the experiments 4 and 5 are less evenly distributed. For example, 93 of the 249 retrieved clusters of experiment 4 yield a precision value of 1, but all these 93 clusters contain just one element. A similar observation can be made on the OPTICS-related experiments 6 to 10, and also on the OPTICS-related experiments on the $SMALL$ image set (experiments 16 to 20).

Image 32 and Image 33 show image samples from clusters created by hierarchical k-Means in experiment 4. In Image 32, one can see that the clustering is seemingly dominated by color, even though more dimensions contain shape information (38 dimensions shape information and 30 dimensions color information). Image 33 shows some images that are part of a huge cluster that contains 4010 images, which is roughly five times the size of the biggest category. The images within that cluster contain, for the most part, some grey, blue or silver color regions, but there exist no more obvious unifying features, notably no semantic correlation.

The images Image 34, Image 35 and Image 36 show samples of clusters created in the OPTICS-related experiment 18. The clusters are small and yield high precision values, but low recall values. One can see that the OPTICS clustering can be stable to outliers in the small clusters. In the big clusters, however, where most of the samples reside, precision values are also small. By using the *OPTICSAnalyzer*, one can see in the reachability-plot, that the OPTICS ordered images contain some reasonable clustering structure where the reachability-distances are low. That means, in a small region where the reachability-distances are low, neighboring samples are likely to belong to the same category. The higher the reachability-distances within a range of the reachability-histogram, the more likely it is that the images within that range are not related to each other. However, even for the ranges in which the reachability-distances are low, the automatic cluster partitioning which makes use of the n most persistent peaks in the OPTICS reachability plot is not always capable to identify structure and correctly ordered image groups which form putative sub-clusters.

Deeper and clearer insights into the clustering results can be obtained by directly viewing the clustering-ordered symbolic links that can be created by the *Clusterer* application, by using the *Evaluation* application and by using the *OPTICSAnalyzer* application. Image 29 depicts a screenshot of the OPTICSAnalyzer on experiment 18 with a range of images that are correctly grouped, but are not automatically partitioned by the n most persistent peaks method that is used to partition the OPTICS ordering. One can also see in that image, that the smoothness of the reachability-plot does not allow to find many clusters. This problem could be alleviated by usage of different values for ε and $MinPts$.

The 20 experiment results:

№	# Clusters	Precision	Recall	F-Measure
1	249	0.389412	0.0202346	0.038470213
2	249	0.110678	0.043276	0.062222497
3	249	0.126699	0.038729	0.059324003
4	249	0.509861	0.0254215	0.048428377
5	248	0.406715	0.0275591	0.051620391
6	28	0.237949	0.0747805	0.113797676
7	99	0.186866	0.0454774	0.073151894
8	34	0.255981	0.10041	0.144240748
9	47	0.21685	0.0474346	0.077841789
10	23	0.316179	0.14895	0.202502368
11	1	0.311047	0.938596	0.467249398
12	1	0.320261	0.982456	0.483055554
13	1	0.320261	0.982456	0.483055554
14	1	0.310917	0.938596	0.467102707
15	1	0.310917	0.938596	0.467102707
16	14	0.57471	0.092803	0.159801568
17	18	0.636117	0.112506	0.191196314
18	19	0.760829	0.141948	0.239257657
19	19	0.763597	0.081005	0.14647177
20	5	0.703172	0.277362	0.39781016

Table 8: Precision, recall rates and F-measure for the 20 experiments on image clustering. Recall values of experiments 11 to 15 are high because the hierarchical k-Means clustering only returned one cluster. The values are however not 100%, because in some images no salient region of sufficient size could be found with the given settings.

The values for precision, recall and thus also for F-measure are considered biased, because the distribution of the images among the clusters resembles a Power-Law distribution (see Image 30 and Image 31) which does not correspond to the original distribution of the images among the categories.

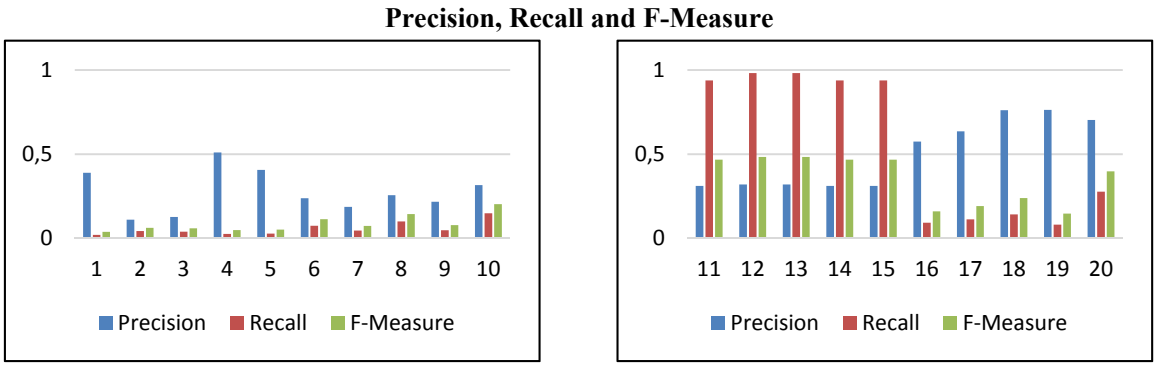


Image 27: Precision, recall and F-measure for the experiments in the *BIG* image set (left) and the experiments on the *SMALL* image set (right). Vertical axes: Precision, recall and F-measure values. Horizontal axes: Experiment number.

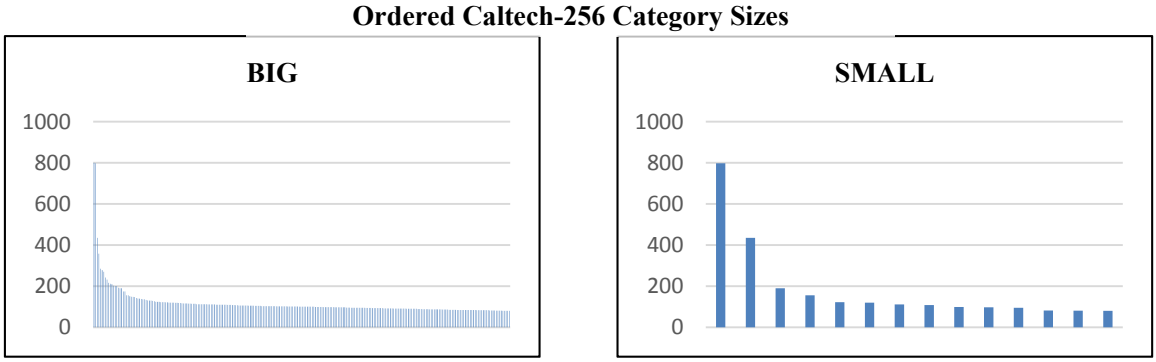


Image 28: The ordered sizes of the categories of the *BIG* and *SMALL* image sets. Vertical axes: Number of images per category. Horizontal axes: Category, ordered according to size.

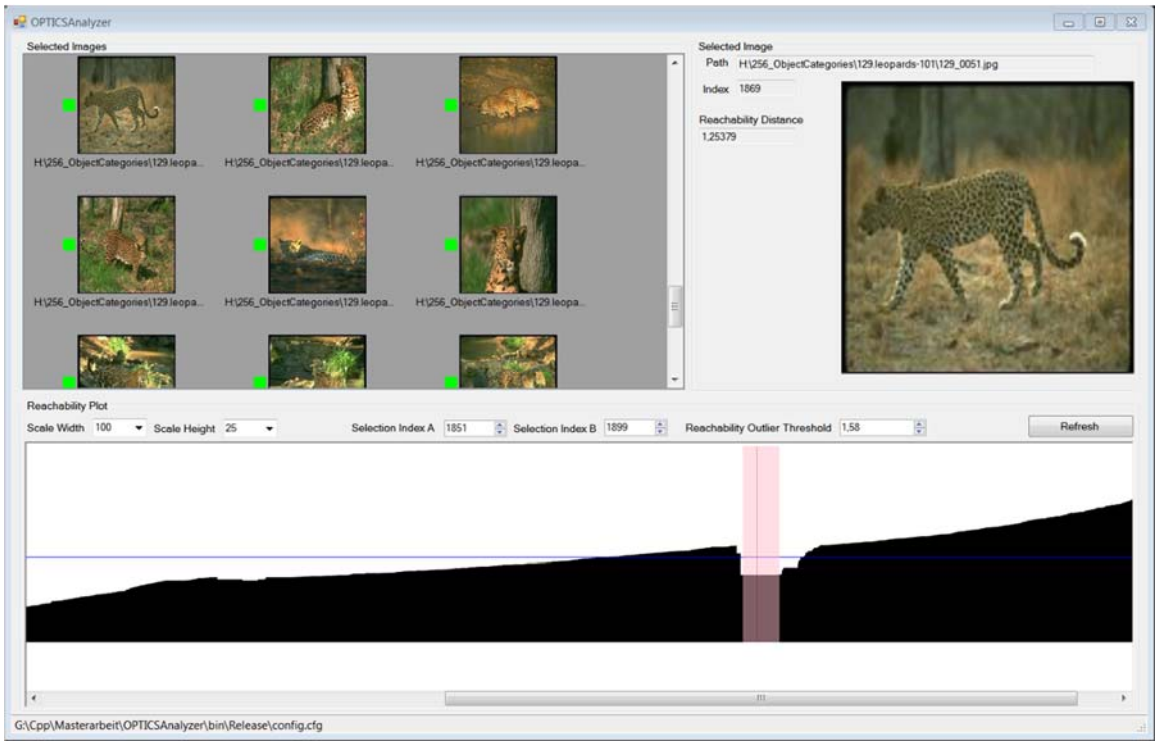


Image 29: Correctly ordered images of leopards found in the *OPTICSAnalyzer* in the reachability-plot of experiment 18. The used automatic partitioning method does not identify this sub-cluster.

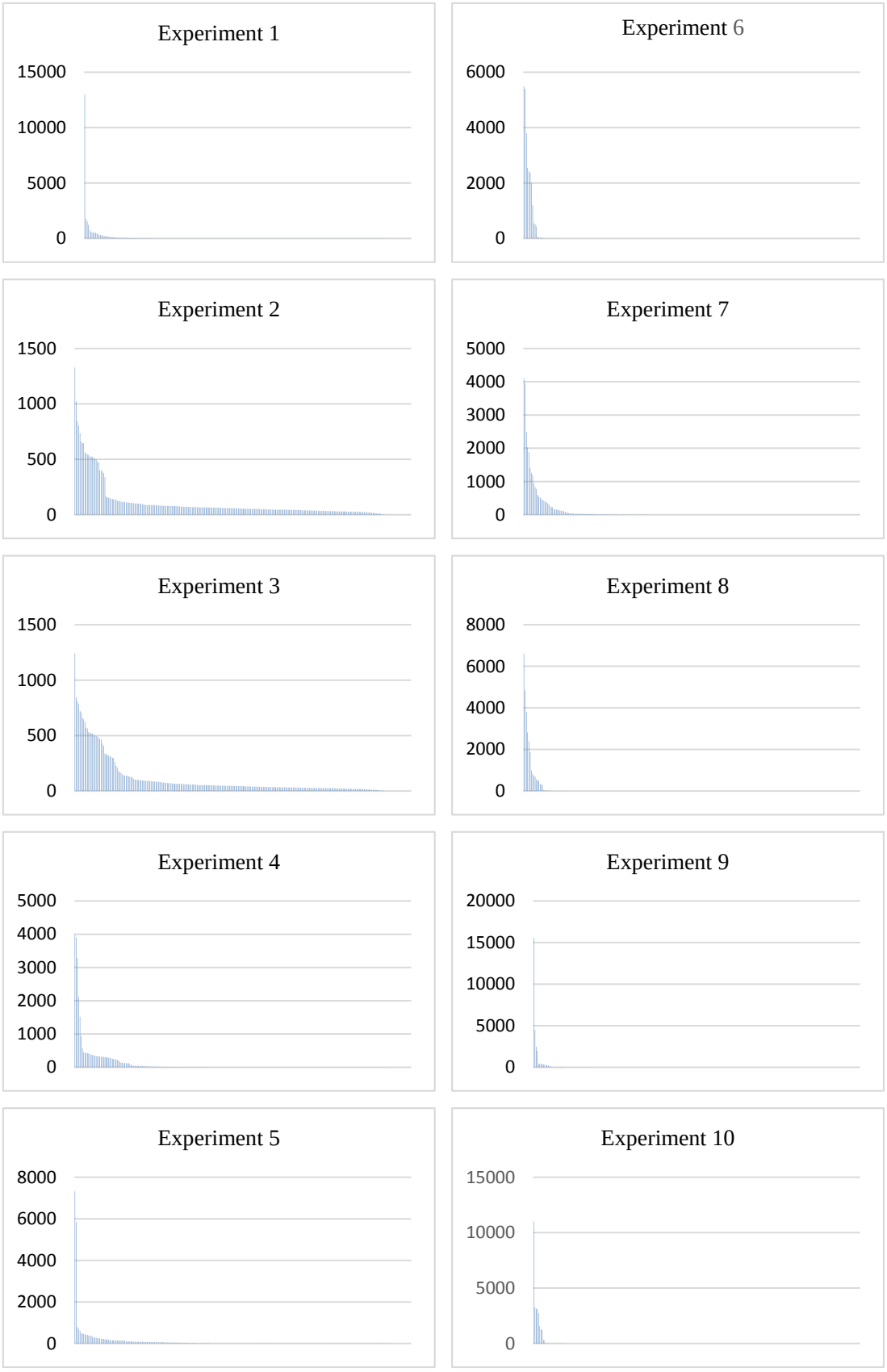


Image 30: The ordered sizes of the clusters of the experiments 1-10. Compare to Image 28 (left).

Vertical axis: number of images per cluster.

Horizontal axis: cluster.

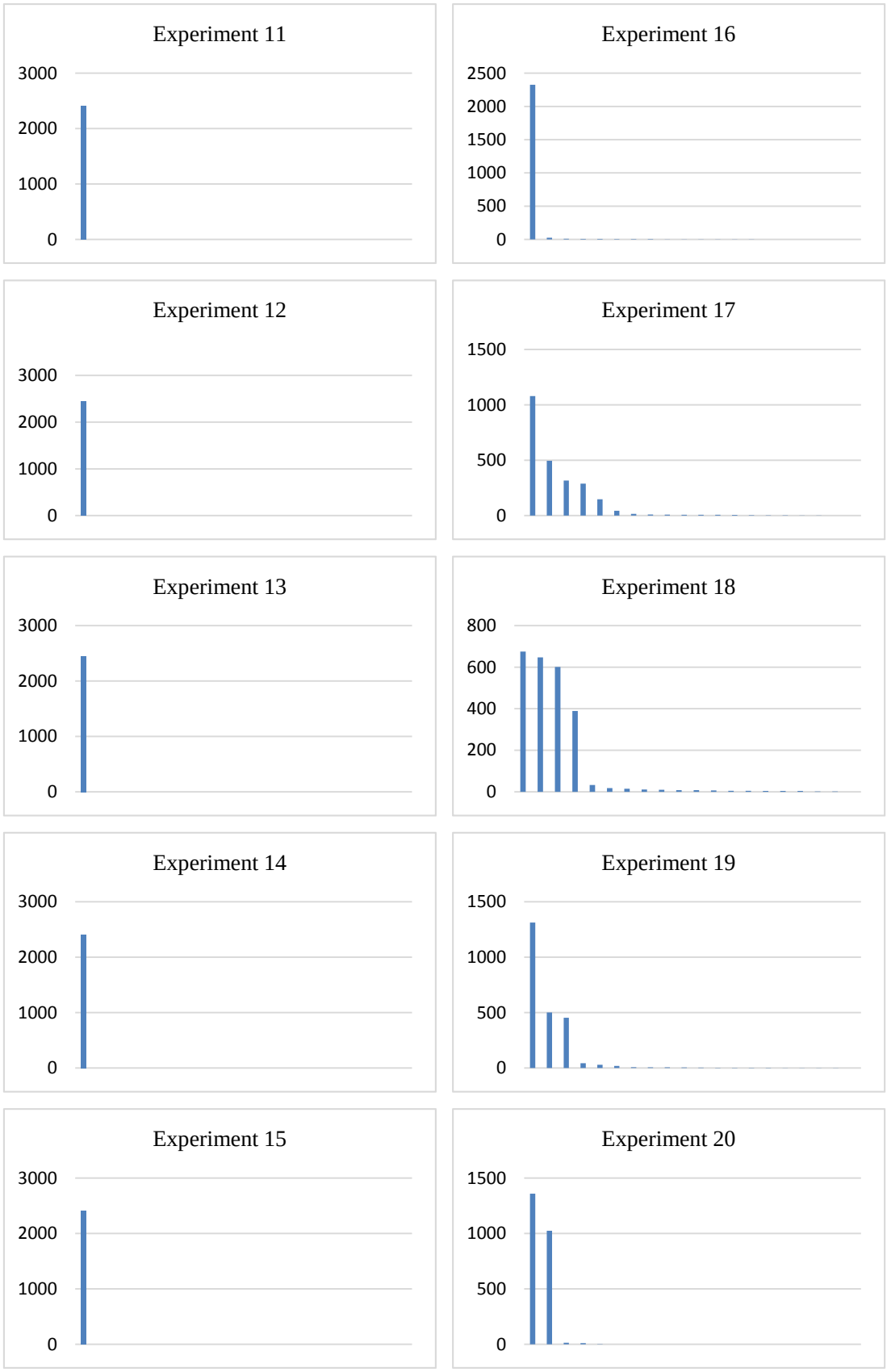


Image 31: The ordered sizes of the clusters of the experiments 11-20. Vertical axes: number of images per cluster. Horizontal axes: cluster. Compare to Image 28 (right).

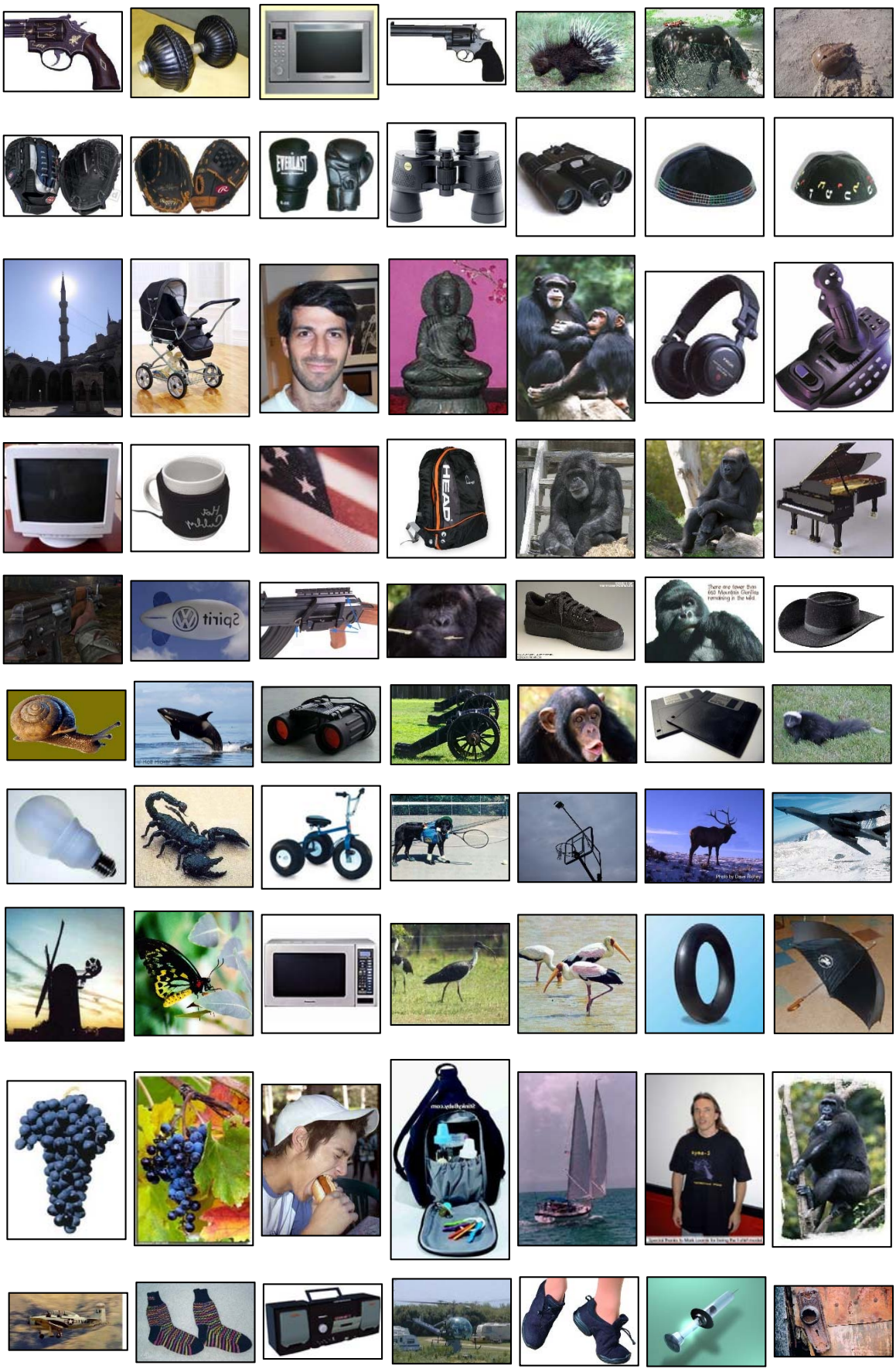


Image 32: Some images of a cluster created in experiment 4. In most cases, the salient objects contain the same dark color, but the shape and the semantic meaning of the objects varies extensively. The original cluster contains 339 images.

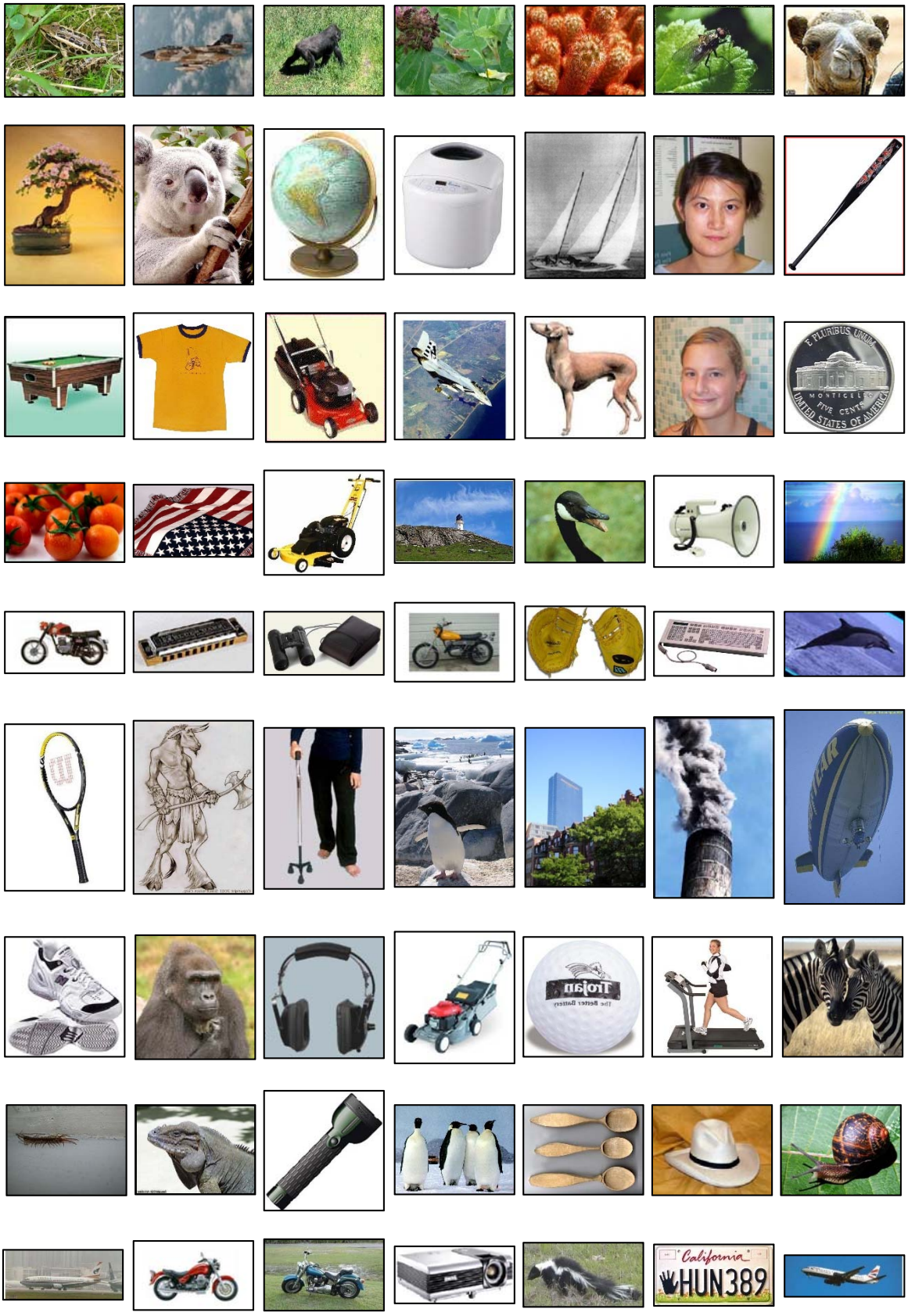


Image 33: Example for bad clustering results. Some images of a cluster created in experiment 4. The whole cluster contains 4010 images. Many images contain a grey or blue color, but no further pattern in terms of shape or color is observable.

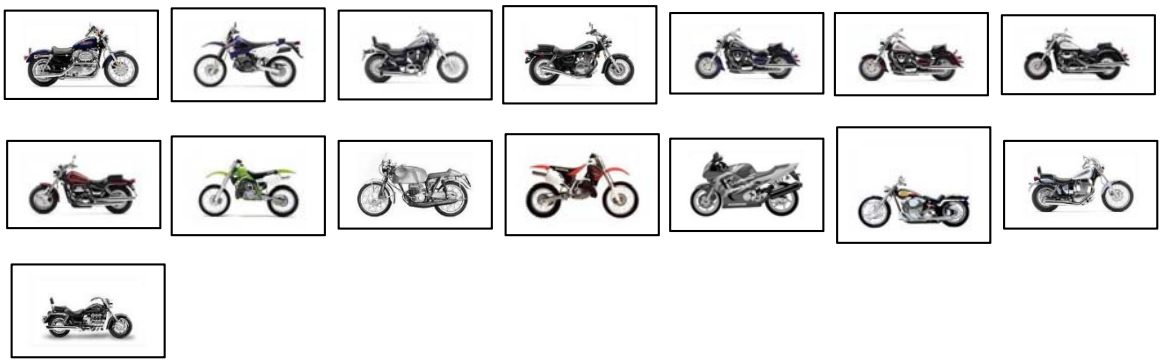


Image 34: A complete cluster created in experiment 18. The cluster contains 15 images and yields a precision of 1.0 and a recall value of 0.018797. The corresponding category "motorbikes.101" contains 798 images.

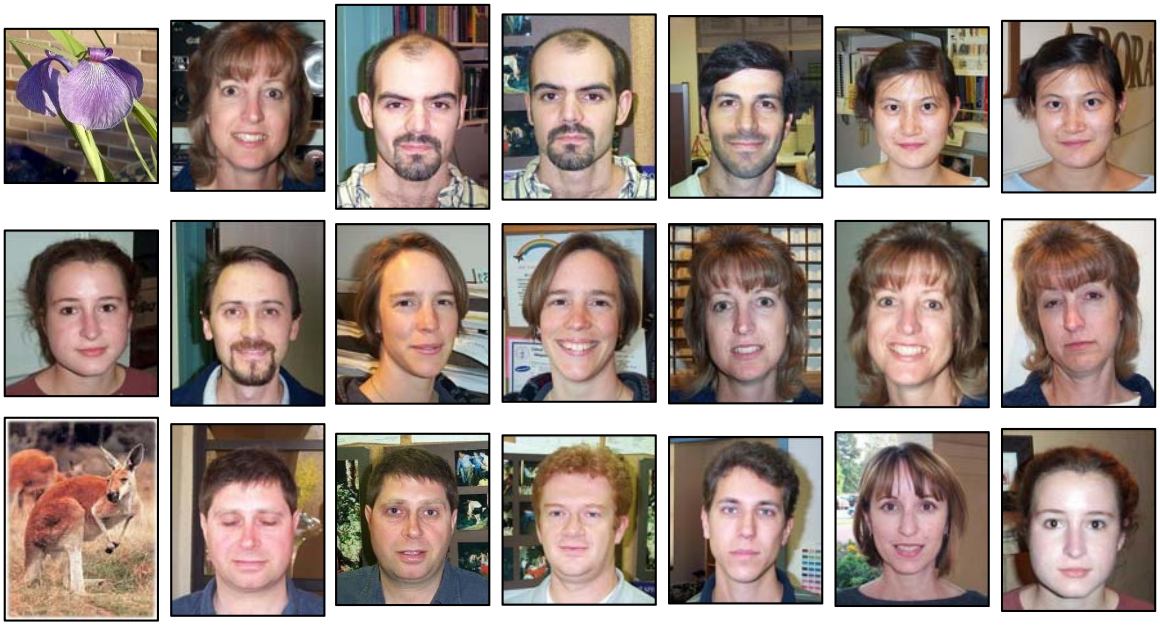


Image 35: Part of a cluster created in experiment 18. The cluster yields a precision of 0.94 and a recall value of 0.071. The original cluster contains 33 images. The corresponding category "faces-easy-101" contains 435 images.

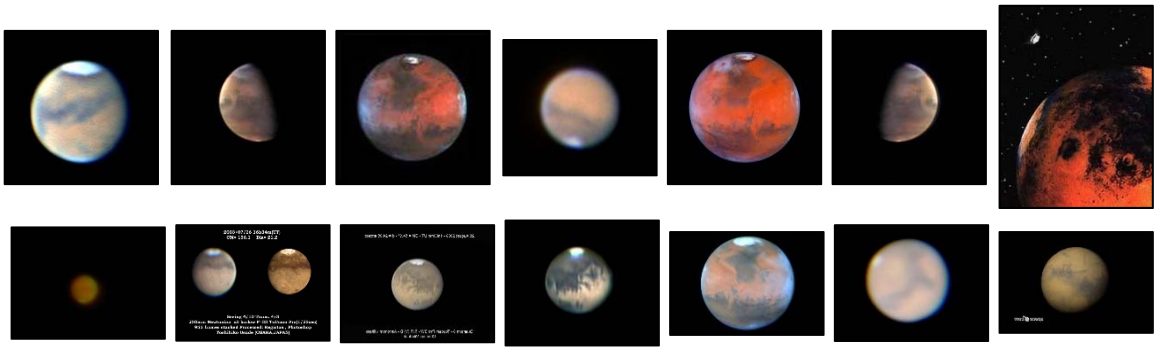


Image 36: Part of a cluster created in experiment 18. The cluster yields a precision of 1.0 and a recall value of 0.12. The original cluster contains 18 images. The corresponding category "mars" contains 156 images.

11.2 Interpretation of the Image Clustering Experiment Results

As the distribution of the images within the clusters (Image 30, Image 31) implies, the results on image clustering are overall not good enough to cluster the data in a correct and reliable fashion. Most of the images are put into a few clusters while most of the clusters contain only a few images. Since the clustering algorithms are working standard approaches, the distribution of the data in feature space seems to be problematic. This can also be confirmed by regarding the amount of created clusters. Since the number of clusters, especially for the OPTICS related experiments 6 to 10 and the k-Means related experiments 10 to 15 differs greatly from the number of underlying real categories, the structure of the samples in feature space is likely to be inadequate partitioned. Such a problematic distribution of samples indicates the usage of unsuitable descriptors. The use of more sophisticated, discriminative low- and mid-level descriptors could be of help here. Such descriptors could employ information about texture and about sharpness or blurriness.

In Image 32, one can see that the impact of color on the clustering is comparably high in experiment 4, although the Fourier components that carry for shape information yield more dimensions than the dimensions that are used for color information. This could be due to the fact that the shapes of the salient regions are in many cases unforeseeable because of varying viewing angles on the depicted scene and the various poses a foreground object can exhibit. The variety of possibilities of how an object of a certain category can be depicted can be huge. This makes it infeasible to just work on low-level information to build descriptors that solely build on shape information.

In order to eliminate the problem of reduced number of clusters for k-Means, a classical k-Means approach, instead of a hierarchical approach could be used.

Usage of the *OPTICSAnalyzer* shows that, on the BIG image set, the OPTICS clustering fails to find correct clusters for the most part, because the feature space is very densely populated. However, the OPTICS related experiments 16 to 20 create small clusters that contain only a few images, but these few images tend to belong to the same category. So, while the automatic partitioning algorithm seems to fail at finding the big clusters, it is able to find at least some small sub-clusters. As the pictures in Image 34, Image 35 and Image 36 indicate, the OPTICS clustering can bring promising structure into the data when the image sets are small and the feature space is not too cluttered (i.e. when better descriptors are used). Tuning of the OPTICS parameters could produce better results. Also, OPTICS-driven structuring on the sub-cluster level can provide a foundation for further aggregation of the images in order to create a stable clustering on the whole-category level. Another helpful improvement on OPTICS clustering can be, to use a more sophisticated partitioning approach, like the one proposed by Sander et al. [47] instead of partitioning by the n most persistent elements.

All in all, the obtained results of the experiments leave much room for improvement. In the first place, future studies can improve the quality of the clusterings by usage of better and more sophisticated descriptors, and secondly by application of other or improved clustering methods.

11.3 Results for Outlier Clustering

Table 9 below depicts the averaged hitting rates of the experiments. Image 37 depicts the results of Table 9 in form of a bar plot. The hitting rates are averaged over each $CAT_{n,o}$ with fixed number of outliers o. The table-headlines “Series 1” to “Series 5” map to the five descriptor variants $CONTOUR$, $HISTOGRAM_{PARTIAL}$, $HISTOGRAM_{COMPLETE}$, $CONTOUR - HISTOGRAM_{PARTIAL}$ and $CONTOUR - HISTOGRAM_{COMPLETE}$. The last

row of the table shows the averaged hitting rates for all experiments on a certain descriptor variant.

In the cases where just one outlier exists, the outlier detection yields a hitting rate of 100% in all cases. This means, on each Caltech-256 category that is used, one outlier can be retrieved by any descriptor in each of the 10 respective experiments.

The table shows, that the $HISTOGRAM$ descriptor variants have a minimum hitting rate of 49% and perform always better than the $CONTOUR$ related histograms. The $HISTOGRAM_{PARTIAL}$ descriptor that does not incorporate color information of the background yields the best results in the experiments with five to ten outliers. The $HISTOGRAM_{COMPLETE}$ descriptor performs better on the augmented categories with one to three outliers. On the experiments with four outliers, both $HISTOGRAM$ descriptor variants perform with the same averaged hitting rate. The performance of both $CONTOUR - HISTOGRAM$ descriptor variants is generally low, in most cases even lower than the performance of the $CONTOUR$ descriptor.

The total average hitting rate over all experiments is 49.62%. Table 10 reveals, that the hitting performance reduces with the number of outliers. In cases with multiple outliers, non-outlier images are falsely detected as be outliers.

Averaged positive hitting rates of the outlier detection experiments:

# Outliers	Series 1	Series 2	Series 3	Series 4	Series 5
1	100,00%	100,00%	100,00%	100,00%	100,00%
2	55,00%	75,00%	85,00%	55,00%	50,00%
3	40,00%	66,67%	70,00%	40,00%	36,67%
4	35,00%	62,50%	62,50%	35,00%	32,50%
5	28,00%	72,00%	58,00%	26,00%	24,00%
6	26,67%	65,00%	56,67%	25,00%	26,67%
7	25,71%	71,43%	51,43%	27,14%	25,71%
8	27,50%	62,50%	51,25%	25,00%	20,00%
9	21,11%	56,67%	56,67%	23,33%	25,56%
10	32,00%	65,00%	49,00%	30,00%	25,00%
Average	**39,10%**	**69,68%**	**64,05%**	**38,65%**	**36,61%**

Table 9: Averaged hitting rates for the outlier detection algorithms on the augmented image categories, ordered by descriptor type. Series 1 conforms to $CONTOUR$, Series 2 to $HISTOGRAM_{PARTIAL}$ Series 3 to $HISTOGRAM_{COMPLETE}$, Series 4 to $CONTOUR - HISTOGRAM_{PARTIAL}$ and Series 5 to the $CONTOUR - HISTOGRAM_{COMPLETE}$ variant.

Averaged hitting rates over all experiment Series w.r.t the number of outliers:

1	2	3	4	5	6	7	8	9	10
100,00%	64,00%	50,67%	45,50%	41,60%	40,00%	40,29%	37,25%	36,67%	40,20%

Table 10: Averaged summed hitting rates for the Outlier detection algorithms on the augmented image categories, structured by number of outliers.

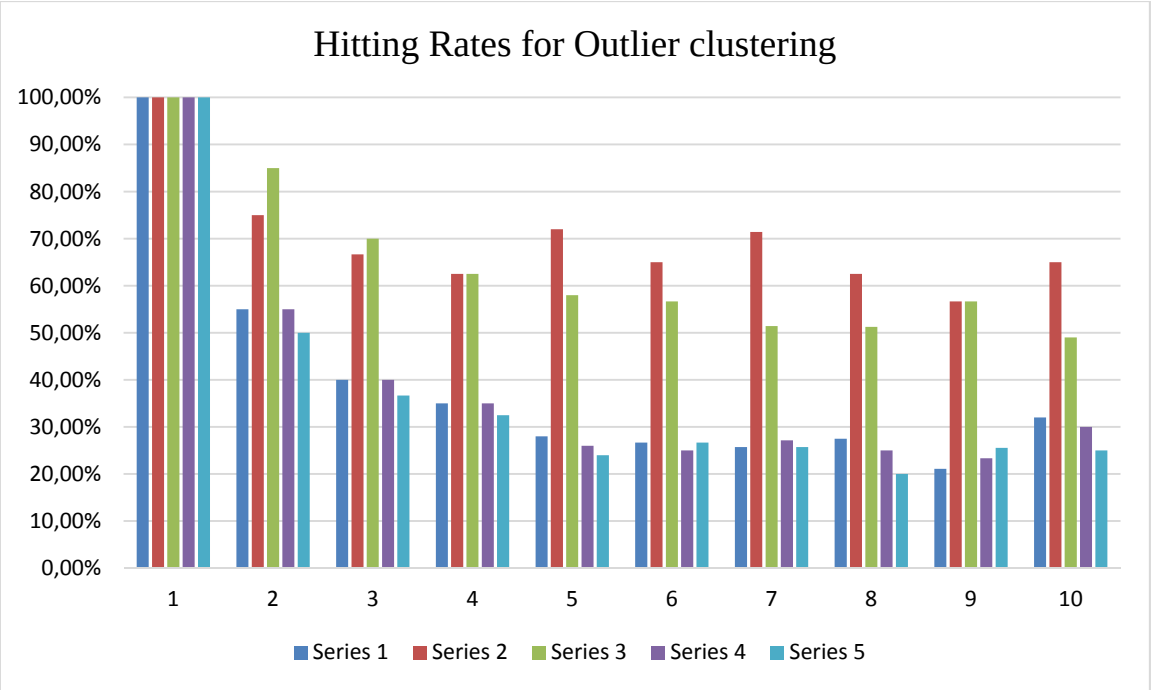


Image 37: Bar plot of the averaged hitting rates for the Outlier detection algorithms on the augmented image categories, structured by descriptor type.

Vertical axis: The average hitting rate.

Horizontal axis: The number of outliers that were to be found.

11.4 Interpretation of the Outlier Clustering Experiment Results

The results of the outlier detection imply, that category-specific outliers are best to be found by means of color information. The bad performance of descriptors that use shape information could be due to the fact of the unforeseeable shape of the salient objects within the images, which makes it even impractical to find a common shape of object depictions within one category. It could however yield good results in categories in which the shape of an object is predictable. The poor performance of the $CONTOUR - HISTOGRAM$ descriptor variants could be due to a poor structuring of the samples in feature space.

The possible reason, why the hitting rate reduces with increased number of outliers can be the fact that the categories are augmented with outliers that are more similar to the original images of the categories. Also, two outliers that are similar to each other could, because of their small distance feature space, not be identified as outliers. An OPTICS ordered clustering with small ε and reasonable large $MinPts$ value could be used to identify such small clusters and highlight them as outliers. The experiments at hand, the hitting rate for outliers tends to converge at ~60% and ~50% for the $HISTORGAM$ descriptors.

All in all, the results imply, that such outlier detection with help of color histogram descriptors can be supportive to manual outlier detection.

12 RECAPITULATION & CONCLUSION

The task was to formulate an unsupervised approach that can discover images in a database that stand out in some way from the other images in the database. The multi-staged approach was to first locate and then identify the foreground objects in the images. With help of an intermediate step which translates the information in an image into a descriptor (in this thesis a real-valued vector) a set of images can then be partitioned by a specified clustering algorithm. In order to realize the approach, a rigorous framework has been implemented. The framework provides a concise environment to create salient region detectors, image descriptors and clustering algorithms and allows their combined application on big image databases. The framework can be used for rapid prototyping of such algorithms and data structures. Furthermore, the framework provides simple yet effective means to quickly assess the quality of the clustering methods. With help of directories that mirror the clustering structure and symbolic links of the original images, the partitioning can be directly applied to on the file level.

In the course of the work, several saliency detection algorithms have been examined and one of them, the *Saliency Filters* algorithm by Perazzi et al. [1] was eventually used for saliency detection. Furthermore, three types of salient region descriptors have been implemented: one descriptor for the shape, one for color information and a third one for shape and color information. The clustering algorithms that are used in the practical part of this thesis are k-Means and an OPTICS ordered clustering. A third clustering approach that was implemented is able to find outliers in the sense of density. It is called the *OutlierClusterer*. All those algorithms and descriptor types allow for parameterization after compile time.

Several experiments on the Caltech-256 image database have been conducted to assess the quality of the combinations of descriptors and clustering algorithms. The clustering-quality was measured by first transferring the clustering problem to many binary classification problems which were then be evaluated by means of precision, recall and the F-measure. Due to the varying number of clusters, and due to the faulty clustering results, results of the experiments were strongly biased and yielded comparably high values when the results were apparently bad, e.g. in cases where the clustering algorithm only created one cluster. On small subsets of the Caltech-256 image database, the tests yielded higher-rated and partly promising results. It could be shown that OPTICS ordered clustering is able correctly extract small fractions of bigger categories, but no experiment yielded the correct extraction of the majority of all images in one category. Separate series of experiments were conducted on the *OutlierClusterer*. The results of those experiments showed that outliers in image sets which depict single object categories can be found with a reliability of about 50% with aid of color histogram information.

13 OUTLOOK

In future, more studies can be conducted to improve the automatic content-specific partitioning of image databases. Besides the investigation of different settings for the plethora of parameters that can be set, new saliency detection algorithms, more sophisticated descriptor types and better suited clustering algorithms can be implemented. Saliency detection algorithms can also use information about blurriness in image regions in order to distinct foreground objects from the image background. New descriptors can employ not only information about shape and color, but could also incorporate information about texture. When dealing with a big image database, descriptors can also be improved if an intermediate probabilistic reasoning model is used to refine all descriptors by investigating and altering similar descriptors. In order to improve the results on OPTICS clustering, a better partitioning method like the one presented in [47] can be used. In order to detect outlier images, a more sophisticated outlier detector can be created. Also, OPTICS ordered clustering can be adapted to find outliers in a configurable and thus versatile fashion. Concerning the clustering of image sets, other, more suitable forms of experiments can be conducted. These forms of experiments should evaluate multi-class clustering in a stable way. Additionally, the experiments can be conducted on other image databases. These image databases may be specific to one category which allows for a better fine tuning of the parameters. Some databases contain only standardized depictions of objects, e.g. different cars that were photographed from restricted viewing angles, like back, front and sides. Such standardization of the image contents can lead to improved clustering quality and make this work more applicable to practical problems. All in all, the practical part of this thesis provides a concise tool for rapid development of new algorithms and image descriptors which makes it easy to pursue further studies on the topic.

14 LIST OF IMAGES

15 LIST OF TABLES

16 ACKNOWLEDGEMENTS

I want to thank my adviser Dr. Hänsch for giving me support, not only on the scientific level but also for encouragement and his good faith. I also want to thank Professor Dr. Hellwich and the whole team of the Computer Vision Department at Technische Universität Berlin. I want to give thanks to Philipp Krähenbühl for his un-official implementation of the *SaliencyFilters* algorithm, Yeara Kozlov and Tino Weinkauf for providing the *Persistence1D* API and Tapio Vierros for providing the *rlutil* library. I also want to thank my many, many friends for the lively discussions and ideas they brought into this thesis, especially Andreas Salzmann who helped me with all kinds of issues regarding feature space clustering. Last but not least I want to thank my parents for supporting me throughout my time at the university.

17 REFERENCES

[1] F. Perazzi, P. Krähenbühl, Y. Pritch and A. Hornung, „Saliency Filters: Contrast Based Filtering for Salient Region Detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, RI, USA, 2012.

[2] L. Itti, C. Koch and E. Niebur, „A Model of Saliency-based Visual Attention for Rapid Scene Analysis," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume:20, Issue: 11)*, Passadena, CA, USA, 2002.

[3] R. Achanta, F. Estrada, P. Wils and S. Süsstrunk, „Salient Region Detection and Segmentation," in *International Conference on Computer Vision Systems (ICVS '08), Vol. 5008*, Lausanne, CH, 2008.

[4] Y. Xie, H. Lu and M.-H. Yang, „Bayesian Saliency via Low and Mid Level Cues," *IEEE Transactions on Image Processing Vol. 22, No. 5*, pp. 1689-1698, May 2013.

[5] T. Leung and J. Malik, „Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons," *International Journal of Computer Vision, Volume 43, Issue 1*, pp. 29-44, June 2001.

[6] V. Gopalakrishnan, Y. Hu and D. Rajan, „Random Walks on Graphs for Salient Object Detection in Images," in *IEEE Transactions on Image Processing (Volume 19, Issue 12)*, Singapore, SG, 2010.

[7] G. Griffin, A. Holub and P. Perona, „Caltech-256 Object Category Data Set," California Institute of Technology, Pasadena, CA, USA, 2007.

[8] S. L. Verdaguer, „Color Based Image Classification and Description," Universitat Politècnica de Catalunya, Catalunya, Spain, 2009.

[9] G. Csurka, C. R. Dance, L. Fan and J. B. C. Willamowsky, „Visual Categorization with Bags of Keypoints," in *ECCV International Workshop on Statistical Learning in Computer Vision*, Prague, Czech Republic, 2004.

[10] R. Sibson, „SLINK: An optimally efficient algorithm for the single-link cluster method," *The Computer Journal, Volume 16, Issue 1*, pp. 30-34, 1973.

[11] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, „Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," in *SIGMOD '98 Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, New York, NY, USA, 1998.

[12] M. Ester, H.-P. Kriegel and J. X. X. Sander, „A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Portland, OR, USA, 1996.

[13] M. Ankerst, M. M. Breuning, H.-P. Kriegel and S. Jörg, „OPTICS: Ordering Points To Identify the Clustering Structure," in *SIGMOD '99 Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 1999.

[14] t. F. E. Wikipedia, „Cluster analysis - Wikipedia, the free encyclopedia,“ 16 July 2015. [Online]. Available: https://en.wikipedia.org/wiki/Cluster_analysis.

[15] M.-M. Cheng, G.-X. Zhang, N. J. Mitra, X. Huang and S.-M. Hu, „Global Contrast based Salient Region Detection,“ in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, RI, USA, 2011.

[16] R. Achanta, S. Hemami, F. Estrada and S. Süsstrunk, „Frequency-tuned Salient Region Detection,“ in *IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2009*, Miami, FL, USA, 2009.

[17] J. Harel, C. Koch and P. Perona, „Graph-Based Visual Saliency,“ in *Proceedings of Neural Information Processing Systems (NIPS)*, Pasadena, USA, NIPS, 2006, pp. 545-552.

[18] R. Achanta, A. Shaji, K. Smith, A. Luchi, P. Fua and S. Süsstrunk, „SLIC Superpixels,“ *EPFL Technical Report No. 149300*, June 2010.

[19] T. Kadir, A. Zisserman and M. Brady, „An affine invariant salient region detector,“ in *Computer Vision - ECCV 2004*, Oxford, UK, Springer Berlin Heidelberg, 2004, pp. 228-241.

[20] J. Han, K. N. Ngan, M. Li and Z. Hong-Jiang, „Unsupervised Extraction of Visual Attention Objects in Color Images,“ in *IEEE Transactions on Circuits and Systems for Video Technology (Volume:16 , Issue: 1)*, Hong Kong, CN, 2005.

[21] D. Comaniciu and P. Meer, „Mean Shift: A Robust Approach Towards Feature Space Analysis,“ in *IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume:24, Issue: 5)*, Princeton, NJ, USA, 2002.

[22] T. Judd and K. D. F. T. A. Ehinger, „Learning to Predict Where Humans Look,“ in *IEEE 12th International Conference on Computer Vision (ICCV)*, Kyoto, JP, 2009.

[23] C. Harris and M. Stephens, „A Combined Corner and Edge Detector,“ in *Proc. of Fourth Alvey Vision Conference*, Plessey Research Roke Manor, UK, 1988.

[24] Z. Jiang and L. S. Davis, „Submodular Salient Region Detection,“ in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Portland, OR, USA, 2013.

[25] P. Kling, „Facility Location,“ Paderborn, DE, 2008.

[26] M. Weber, M. Welling and P. Perona, „Towards Automatic Discovery of Object Categories,“ in *Conference on Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE (Volume 2)*, Hilton Head Island, SC, USA, 2000.

[27] B. Fulkerson, A. Vedaldi and S. Soatto, „Class Segmentation and Object Localization with Superpixel Neighborhoods,“ in *12th International Conference on Computer Vision*, Kyoto, 2009.

[28] A. Vedaldi and S. Soatto, „Quick Shift and Kernel Methods for Mode Seeking,“ in *Computer Vision – ECCV 2008, 10th European Conference on Computer Vision, Proceedings, Part IV*, Marseille, France, 2008.

[29] D. G. Lowe, „Distinctive Image Features from Scale-Invariant Keypoints,“ *International Journal of Computer Vision, Volume 60, Issue 2,* pp. 91-110, November 2004.

[30] M. Andreeto, L. Zelnik-Manor and P. Perona, „Unsupervised Learning of Categorical Segments in Image Collections,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 1842-1855, 23 July 2012.

[31] R. O. Duda, P. E. Hart and D. G. Stork, „Parzen Windows,“ in *Pattern Classification,* Wiley-Interscience, 2000, pp. 164-174.

[32] T. Tuytelaars, C. H. Lampert, M. B. Blaschko and W. Buntine, „Unsupervised Object Discovery: A Comparison,“ *International Journal on Computer Vision, Volume 88, Issue 2,* pp. 284-302, June 2010.

[33] Microsoft Research, „Object Class Recognition - Microsoft Research,“ 24 6 2015. [Online]. Available: http://research.microsoft.com/en-us/projects/objectclassrecognition/.

[34] C. Rother, V. Kolmogorov and A. Blake, „"GrabCut" - Interactive Foreground Extraction using Iterated Graph Cuts,“ in *Proceeding SIGGRAPH '04 ACM SIGGRAPH 2004 Papers*, New York, NY, USA, 2004.

[35] A. Adams, J. Baek and M. Abraham Davis, „Fast High-Dimensional Filtering Using the Permutohedral Lattice,“ in *Computer Graphics Forum (EG 2010) Proceedings*, Stanford, CA, USA, 2010.

[36] J. Baek and A. Adams, „Some Useful Properties of the Permutohedral Lattice for Gaussian Filtering,“ Stanford, CA, USA.

[37] Hou and Zhang, „Saliency Detection: A Spectral Residual Approach.,“ 2007.

[38] Ma and Zhang, „Contrast-Based Image Attention Analysis by Using Fuzzy Growing,“ 2003.

[39] Zhai and Shah, „Visual Attention Detection in Video Sequences Using Spatiotemporal Cues,“ 2006.

[40] Goferman, L. Zelnik-Manor and Tal, „Context-Aware Saliency Detection,“ 2010.

[41] Martin, Fowlkes, Tal and Malik, „A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics,“ 2001, pp. 416-423.

[42] P. Krähenbühl, „Philipp Krähenbühl,“ 2015. [Online]. Available: http://www.philkr.net. [Visited on 16 April 2015].

[43] A. Criminisi, T. Sharp, C. Rother and P. P'erez, „Geodesic image and video editing,“ *ACM Transactions on Graphics (TOG), No. 5,* pp. 134:1-134:4, October 2010.

[44] J. Dolson, J. Baek, C. Plagemann and S. Thrun, „Upsampling Range Data in Dynamic Environments,“ in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, USA, 2010.

[45] C. M. Bishop, „K-means Clustering,“ in *Pattern Recognition and Machine Learning*, Cambridge, Springer, 2006, pp. 424-430.

[46] M. Inaba, N. Katoh and H. Imai, „AppApplications of Weighted Voronoi Diagrams and Randomization to Variance-Based k-Clustering,“ in *Proceedings of the tenth annual symposium on Computational geometry (SCG '94)*, New York, NY, USA, 1994.

[47] J. Sander, X. Qin, L. Zhiyong, N. Niu and A. Kovarsky, „Automatic Extraction of Clusters from Hierarchical Clustering Representations,“ in *Proceedings of the 7th Pacific-Asia conference on Advances in knowledge discovery and data mining (PAKDD '03)*, Seoul, Korea, 2003.

[48] Itseez, „OpenCV,“ 3 June 2015. [Online]. Available: http://opencv.org/.

[49] „Doxygen,“ 3 June 2015. [Online]. Available: http://www.doxygen.org/.

[50] B. Dawes and D. Abrahams, „Boost C++ Libraries,“ 3 June 2015. [Online]. Available: http://www.boost.org/.

[51] T. Verros, „rlutil.h,“ 3 June 2015. [Online]. Available: http://tapiov.net/rlutil/docs/HTML/files/rlutil-h.html.

[52] F.-F. Li, M. Andreetto and M. '. Ranzato, 5 April 2006. [Online]. Available: http://www.vision.caltech.edu/Image_Datasets/Caltech101/.

[53] F. S. v. d. W. J. Khan and M. Vanrell, „Modulating Shape Features by Color Attention for Object Recognition,“ in *International Journal of Computer Vision*, Springer US, 2012, pp. 49-64.

[54] A. Clauset, C. R. Shalizi and M. E. J. Newman, „Power-Law Distributions in Empirical Data,“ *SIAM Review 51*, pp. 661-703, 2009.

[55] L. Fei-Fei and P. Perona, „A Bayesian Hierarchical Model for Learning Natural Scene Categories,“ in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005. (Volume: 2)*, Pasadena, 2005.