computer graphics & vision

Inffeldgasse 16/2, A-8010 Graz

www.icg.tu-graz.ac.at

TUG

Sketch Recognition

bag-of-visual-words for classifying sketches of generic object categories

Michael Scheer

Institute for Computer Graphics and Vision
Graz University of Technology, Austria

Master's Thesis
Graz, Austria, April 2014

Supervisor/Assessor
Dipl.-Ing. Dr.techn. Michael Donoser

contact: Michael Scheer, m.scheer@student.tugraz.at

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ....
(Unterschrift)

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....
date (signature)

Abstract

In this thesis we address the problem of recognizing hand-drawn sketches of versatile types of object categories. This has diverse application fields in computer vision as well as computer graphics. Sketch recognition can e.g. be integrated in search engines, where the query is an image instead of text. Further, the 3D modeling of a complex scene can e.g. be supported with this technique. However, the task is difficult, as the appearance of the sketch depends on the drawer and his/her drawing skills. Further, the imagination of what a typical sketch of a certain object looks like, varies a lot. Another challenge is the visual likeness of specific categories. Therefore, most previous works in this field constrained themselves to a specific sketch domain such as e.g. recognizing primitives or symbols. In this thesis we introduce a general sketch recognition algorithm without this constraint. We investigate the relation between conventional image and sketch recognition. Further, we discuss our assumption that well studied techniques of the image recognition domain can be successfully transferred into the sketch domain. In order to do so, we perform extensive evaluations for every step involved in a Bag-of-Visual-Words based image recognition pipeline. We show that it is indeed possible to apply techniques designed for the image domain to sketch recognition, however the individual steps have to be carefully tuned in order to obtain a high performance. Our sketch recognition algorithm is designed to be both accurate as well as efficient to be able to provide immediate recognition feedback. In experiments we show that our approach achieves state-of-the-art performance on a large-scale sketch dataset with 250 categories, predicting the correct category for 63% of the sketches. Further, we show an augmented reality application, where a 3D model of the category of an online drawn sketch is augmented in a live video stream.

Keywords: *sketch recognition, image representation, bag-of-visual-words, visual dictionary learning, sparse coding, spatial pyramid*

Kurzfassung

In dieser Abschlussarbeit beschäftigen wir uns mit dem Problem der Erkennung von handgezeichneten Skizzen von vielseitigen Arten von Objektkategorien. Dies hat vielfältige Anwendungsfelder im Bereich des Maschinellen Sehens sowie der Computergraphik. Skizzenerkennung kann z.B. in Suchmaschinen eingebaut werden, sodass als Suchanfrage ein Bild anstatt eines Textes verwendet wird. Weiters kann z.B. auch das Modellieren einer komplexen 3D Szene mit diesem Verfahren unterstützt werden. Allerdings ist diese Aufgabe schwierig, da das Aussehen einer Skizze von dem/der ZeichnerIn abhängt und seiner/ihrer Zeichenfähigkeit. Zusätzlich variiert die Vorstellung wie eine typische Skizze eines bestimmten Objektes aussieht, stark. Eine zusätzliche Herausforderung ist die visuelle Ähnlichkeit konkreter Kategorien. Aufgrund dessen hat sich der Großteil der früheren Arbeiten auf diesem Gebiet auf ein spezielles Gebiet von Skizzen eingeschränkt wie z.B. die Erkennung von Standardformen oder Symbolen. In dieser Abschlussarbeit stellen wir einen allgemeinen Algorithmus zur Erkennung von Skizzen ohne diese Einschränkung vor. Wir untersuchen die Beziehung zwischen konventioneller Bild- und Skizzenerkennung. Weiters diskutieren wir unsere Annahme, dass ausführlich untersuchte Verfahren aus dem Bereich der Bilderkennung erfolgreich in die Skizzendomäne überführt werden können. Um dies umzusetzen führen wir umfangreiche Auswertungen für jeden benötigten Schritt einer Bag-of-Visual-Words basierten Bilderkennungspipeline durch. Wir zeigen, dass das Anwenden, der für den Bildbereich konzipierten Methoden, auf Skizzenerkennung tatsächlich möglich ist, allerdings müssen die einzelnen Schritte sorgfältig abgestimmt werden um eine hohe Leistung zu erreichen. Unser Skizzenerkennungsalgorithmus ist so konstruiert, dass er sowohl präzise als auch effizient ist um eine unmittelbare Rückmeldung zur Erkennung bereitzustellen. In Experimenten zeigen wir, dass unser Ansatz State of the Art Ergebnisse auf einen umfangreichen Skizzen-Datensatz, bestehend aus 250 Kategorien, erreicht, welcher die richtige Kategorie von 63% der Skizzen voraussagt. Weiters führen wir eine Augmented Reality Applikation vor, in der ein 3D Modell von der Kategorie einer online gezeichneten Skizze in einen live Video-Stream dargestellt wird.

Stichwörter: *Skizzenerkennung, Bilddarstellung, bag-of-visual-words, visual dictionary learning, sparse coding, spatial pyramid*

Acknowledgments

First of all I would like to thank my thesis supervisor Dr. Michael Donoser for his superb support and productive discussions. He always provided valuable input which helped me to accomplish the challenges involved in this thesis. Further, I would like to thank Univ.-Prof. Dr. Horst Bischof for his introduction into the topic of computer vision during basic lectures which awoke my curiosity for this field.

Finally, I would like to thank Irene, my family, and my friends for supporting me during my whole student days.

Contents

1 Introduction

Sketches give humans the ability to describe their world or certain objects of it without using verbal communication. Drawing sketches can therefore be seen as an alternative way of communication. Interpreting and understanding sketches is not constrained by country borders or language regions. If someone e.g. draws a sketch of a simple stickman, it would be recognizable by most of the people around the world. As a result of that, in some scenarios it can be easier to exchange information by sketches rather than by verbal communication or written text.

We frequently encounter sketches in our everyday life, as one typical example let us consider the safety instruction card available in every airliner. This card is essential because not all travelers have the ability of understanding the language spoken on board or to hear the instructions of the crew e.g. due to certain confounding factors. An example of such a safety card can be seen in Figure 1. Although not a single word is used, we recognize and understand the provided information. We learn e.g. which preparations are needed for the take-off and landing procedures: to bring our seat in a vertical position and to store all of our handluggage in the overhead locker. Further the safety card depicts the steps necessary in case of an emergency landing. Therefore we see that understanding sketches is not only essential but can even be life-saving. Pictograms can also be seen as a form of sketches and are used for information brokerage. They are e.g. often placed on airports, rail stations or on motorways to provide information about the location of certain facilities, such as a call box, a lavatory or a filling station. Again this happens without using any text. Examples of pictograms frequently seen in road traffic or on motorway service areas are shown in Figure 2.

Sketches are also a popular tool for certain professional fields, such as engineering, textile design or architecture. Often the first step in the development of a new project is to bring down initial ideas on paper through sketches. One popular example is car development where initial design ideas are often depicted and stranded with sketches. Sometimes this step is supported by 3D-rendering software.

The history of sketches even starts in prehistoric times. Cave paintings can be seen here as an early type of sketches. These paintings were usually found on cave walls or ceilings all over the world (e.g. Europe, Africa, North and South America, India, Australia and Southeast Asia) [3]. The oldest ones were found in northern Spain (Cave of El Castillo) and were created approximately 40 000 years ago [3]. In Figure 3 we can clearly see the relation between cave art and what we would nowadays define as sketches. Although this painting was drawn 40 000 years ago, we can easily name the object

Figure 1: A safety card taken from a Boeing 737-600 of the SAS Scandinavian Airline [1].

Figure 2: Generic pictograms often found in road traffic [2].

category shown – "rhinoceros". This is also a desirable property for sketches.

Figure 3: A cave painting showing a rhinoceros in the Chauvet Cave in France [4].

We already see that the task of sketch recognition accompanies us in our everyday lives. In the following subsections we provide an overview, define the task of sketch recognition in a more formal way and the contributions of this master thesis.

1.1 Overview and Definition

In this master thesis we want to investigate the ability of computer vision approaches to perform sketch recognition. Such an algorithm can be applied to many fields of both computer vision and computer graphics. A search engine can benefit due to the fact that a user has the possibility of drawing a sketch rather than providing a keyword. Further, a carefully designed sketch representation can be used to obtain similar sketches or real-world images given a user drawing. Also the 3D modeling of complex scenes can be supported. Instead of a textual search a more suitable search query, a drawing of the desired model, is provided. An example for that would be a sketch of e.g. a specific chair model from a certain viewpoint. In that way the desired 3D model can be found more easily and faster. In further consequence such an accelerated search can also be interesting for the computer game and film industry. In general, many fields can benefit from a successful sketch recognition approach.

However, most prior research in this area is limited to one specific domain such as primitives or mathematical formulas. Although this is sufficient for individual problems, we introduce an approach which is capable of recognizing versatile types of object categories without this constraint. Only few researchers have addressed this problem so far and in general the proposed

algorithms are computationally expensive or only achieve a limited recognition accuracy. Instead, we propose an algorithm with a high recognition rate which at the same time is efficient enough to provide immediate feedback. In order to do so, we first reviewed the state-of-the-art in the related field of image recognition. The reason for this is that we compared the two recognition domains and identified shared properties. The comparison is discussed in detail in Section 1.2. As a result of that observation we decided to integrate well studied image recognition techniques in our sketch domain approach. In order to select methods which are also suitable in the sketch domain we perform extensive evaluations whose results are incorporated in our final sketch recognition algorithm.

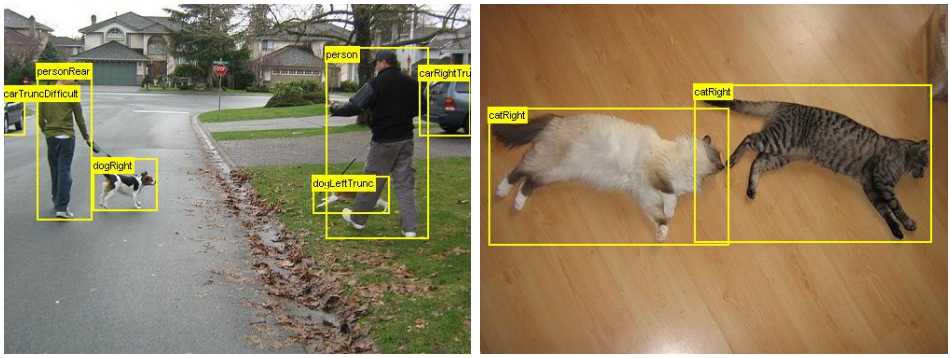
We define a sketch as a human drawing which depicts exactly one object of a certain category. Further, we declare the goal of sketch recognition as the interpretation of the sketch content as one category out of a predefined set (see Figure 4 for example sketches of different objects). Sketch recognition does not include a localization or detection step but aims at classifying the sketch image. In contrast, in image understanding or object detection the goal is to localize and classify all objects available in an image. See Figure 5 for two results of an object detection approach. However, the task of image recognition is obviously related to sketch recognition.

(a) airplane (b) fish (c) pizza (d) monkey (e) bicycle (f) revolver

Figure 4: Example sketches of six (out of 250) categories from the TU Berlin sketch dataset [5].

1.2 Image and Sketch Recognition Analysis

Image recognition is performed on natural images (e.g. a real-life image of an object, a landmark or an animal). Therefore multiple objects of one or even more categories can be present in one single image (problem 1). See e.g. Figure 6(a) which contains a cropped elephant, two sitting people and a bench. Still, the ground truth category of this image is "elephant". Further, background clutter can be included in natural images (problem 2). This means that background objects or things are available. In Figure 6(a) background



(a) dog, person, car (b) cat

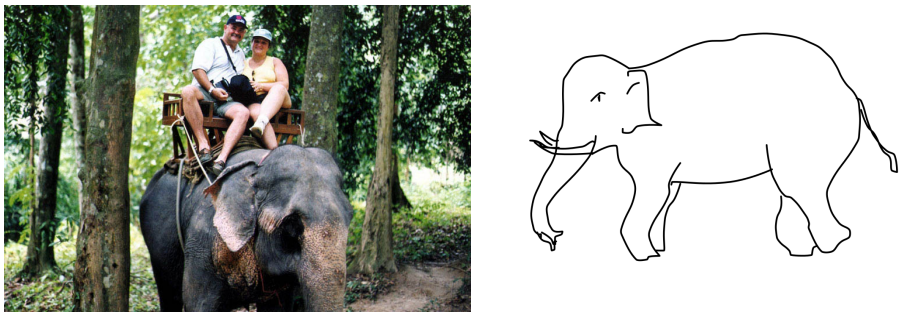
Figure 5: Results of an object detection approach taken from the Pascal VOC 2007 dataset [6]. The captions of the two figures state the object categories found in the two images.

clutter in the form of grass, boles and leaves is present. However, additional background can contain useful information for the recognition task. Ice floes e.g. can help to recognize images of the category "ice bear" or trees/grass can provide evidence for the "brown bear" label. The next challenging problem is the intra-class variability (problem 3). Objects of one and the same category can have a large amount of visual representations. The general category "dog" e.g. consists of all dog breeds or the class "car" includes all car types and models. The pose of certain objects inside one category can vary (e.g. a cat can sit on a scratcher or lies on the floor). A related problem is the inter-class variability (problem 4). Although objects belong to different categories, their visual appearance can look similar (e.g. for images containing bicycles or motorbikes). The problem of various scales and translations also exists for image recognition (problem 5). In other words, the size and position of the object inside the image are not constant. Additionally, objects can be photographed from different viewpoints (problem 6). The resulting images can e.g. show a cat in a frontal view, side view or even back view. When using natural images the problem of illumination can arise (problem 7). Two images showing an identical scene can differ from each other due to different weather conditions (e.g. sunny and rainy) or recording time (e.g. noon and midnight). As a final problem in the image recognition domain we identify the possibility of occlusions (problem 8). For an image taken e.g. inside a forest, a tree bole can mask parts of a brown bear.

In the sketch domain usually binary images, which show the presence or absence of sketch lines for each pixel, are used. See Figure 6(b) for an ex-

ample sketch of an elephant. As only one object is depicted per sketch the problem of multiple objects (problem 1 in the image domain) does not occur. Further, no background clutter (2) is present in sketch images as only the object itself is drawn. However, the issue of intra-class variability (3) is ubiquitous for sketches. We claim that this problem is exacerbated compared to the image domain due to the fact that the visual representation of an object solely depends on the drawer. Sketches containing the same object can have very different characteristics and appearances depending on the drawer. This factor is influenced by the drawing skills as well as how the person thinks a typical instance of the given category looks like. A drawer with the task to draw a sketch of e.g. a "cat", could think of a cat sitting on a windowsill or lying on the floor. See Figure 7 for four sketches containing a cat in various poses. Further, this can also be related to the viewpoint problem (6) in the image domain. However, we argue that this is not true for all categories. A sketch of a cigarette e.g. is likely to be drawn in a meaningful side view. We claim that for sketches there is a set of "typical" views for each category. Therefore, at a smaller scale the viewpoint problem is present in the sketch domain but is indeed less difficult than for natural images. The inter-class variability (4) is also effected due to the already mentioned drawer-dependent sketch characteristics. For natural images showing e.g. a seagull, a pigeon or a duck, the classification process is not a big problem for most humans. However, in the sketch domain this can be a challenging task as we see in Figure 8. Due to poor drawing skills and the lack of additional (background/context) information (e.g. color or water) it is hard to distinguish between these three bird categories. We argue that the problems with varying scale and translation (5) do not exist in the sketch domain. As only the sketch object is present, the extent can be determined. With this information one can manipulate the object to achieve a constant sketch scale and position. In contrast to natural images the problems of illumination (7) and occlusions (8) do not occur. Unlike the image domain there exists a lack of datasets and learning data for sketch recognition (problem 9). To the best of our knowledge there only exists the large-scale TU Berlin sketch dataset [5] suitable for our work in computer vision literature.

The "big picture" of the investigated problems both for image and sketch recognition can be found in Figure 9. From the previous analysis and the summary of it given in Table 1, we see that there is indeed a close relation between both fields. Although the images to be processed greatly differ from each other (see Figure 6), similar problems occur. With the intra and inter-class variability and the viewpoint issue, all of the shared problems can be consolidated under the term visual appearance. However, we argue that the



(a) natural image (b) sketch image

Figure 6: Comparison of a natural (from the Pascal VOC 2007 dataset [6]) and a sketch image (from the TU Berlin sketch dataset [5]) of the category "elephant".

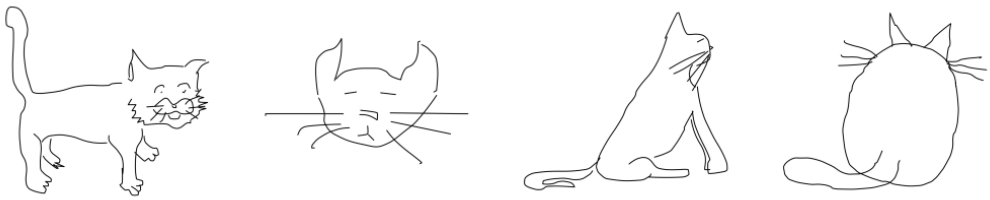
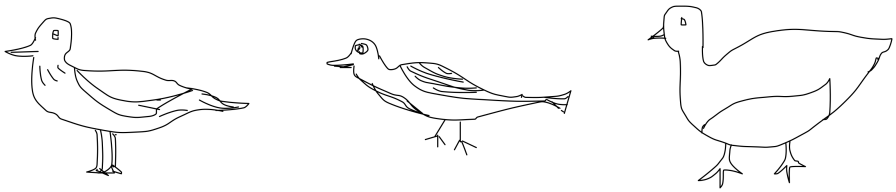


Figure 7: Four example sketches of the category "cat" with different representations taken from the TU Berlin sketch dataset [5].



(a) seagull (b) pigeon (c) duck

Figure 8: Example for similar looking sketches from different but related bird categories (taken from the TU Berlin sketch dataset [5]).

intra-class as well as the inter-class variability are more distinctive in the sketch domain. This comes from the dependence on the sketch drawer. On the other hand, more "typical" viewpoints need to be considered for natural images.

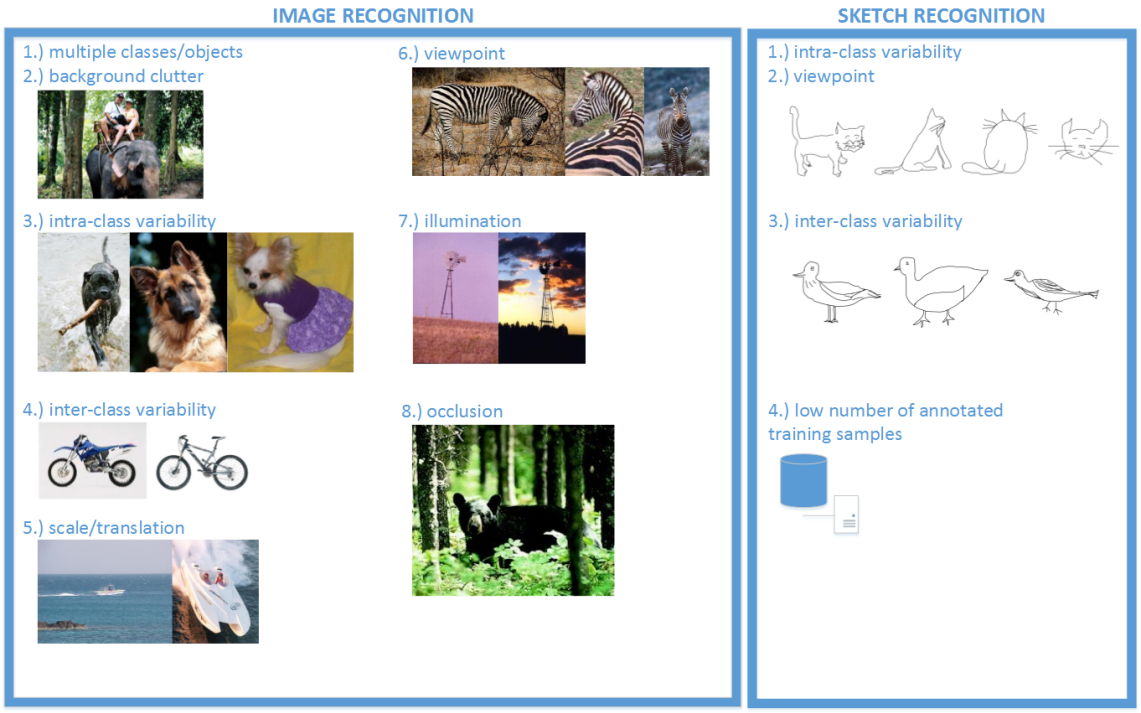


Figure 9: "Big picture" of the analyzed problems regarding image and sketch recognition. The images used are taken from the Caltech-256 dataset [7].

In general we can say that humans are superb in handling these difficult problems. Still, as mentioned and shown in Figure 8, the problem of related and similar-looking categories can be hard even for humans. From a computer vision point of view all problems are challenging. However, for the image recognition domain, methods were developed which tackle these problems and achieve high recognition rates. Due to the relation discussed we are encouraged that such methods can also overcome the difficulties in the sketch domain.

1.3 Conclusion and Contributions

In this master thesis we propose a novel algorithm for domain-independent sketch recognition. We start with a detailed analysis of both image and sketch recognition as well as their relations. Based on our insights we decided to

	Image Recognition	**Sketch Recognition**
Task	Recognize image category	Recognize image category
Images	Natural color images	Binary sketch images
Multiple Objects	Yes	No
Background	Yes	No
Intra-class Variability	High	Very High
Inter-class Variability	High	Very High
Fixed Scale/Position	No	Yes
Amount of Viewpoints	Medium	Low
Varying Illumination	Yes	No
Occlusions	Yes	No
Dataset Amount	High	Low

Table 1: A tabular conclusion of the relation between the sketch and image recognition.

transfer well studied image recognition methods to the sketch domain. A comprehensive list of techniques from relevant related work in the image domain is provided. In order to select suitable and compatible methods for sketch recognition, an extensive evaluation is performed. To the best of our knowledge we are the first to examine this topic in such extent. The result of that evaluation phase is a sketch recognition algorithm which achieves state-of-the-art performance on a large-scale sketch dataset. Beside recognition performance we designed our pipeline to be efficient in the aspect of runtime. This makes our algorithm suitable for interactive applications. Further, our introduced sketch representation can be used in various retrieval domains. We also show that we can achieve a reasonable performance for the image recognition task without modifications of the algorithm.

The rest of this thesis is organized as follows. In Section 2 we investigate relevant related work in general. This includes the original Bag-of-Visual-Words method as well as other approaches for image and sketch recognition. Additionally, we discuss sketch-based retrieval methods. In Section 3 we discuss how we use the Bag-of-Visual-Words method for sketch recognition. Further, this section provides information for the investigated image recognition methods. In Section 4 the extensive evaluation phase is covered and our final sketch recognition algorithm is explained. We deal with final experiments, statistics, recognition results and a developed augmented reality application using our novel sketch recognition algorithm in Section 5. In Section 6 we conclude and discuss directions for future work.

2 Related Work

We first discuss the fundamental Bag-of-Visual-Words (BoVW) approach in Section 2.1 as we focus on related work based on this method. Further, our introduced sketch recognition algorithm uses this BoVW pipeline. An overview of the state-of-the-art in the field of image recognition is given in Section 2.2. This comes from the fact that we want to incorporate techniques of this field in our sketch recognition algorithm as explained in Section 1.1 and 1.2. We finally investigate related sketch recognition and sketch-based retrieval approaches in Section 2.3.

2.1 Bag-of-Visual-Words Overview

The BoVW method was introduced in the "Video Google" approach of Sivic and Zisserman [8] in the context of object retrieval in video sequences. An image region containing the desired object is selected in a reference frame. In the following we call this region the query region. Whenever this region reoccurs in any frame of the video sequence it should be identified and localized. The approach is inspired by the related Bag-of-Words (BoW) model used for text retrieval [9].

In the initial step of the approach, interest regions are detected for all frames of the video sequence[1]. This is done by using Maximally Stable Extremal Regions (MSER) [10] which describe blob like structures by ellipses. The Scale-Invariant Feature Transform (SIFT) descriptor [11] is used to describe these regions in a robust way. In this context robust means that viewpoint changes up to 60 degrees as well as illumination changes can be handled [11]. The obtained regions are tracked in the video sequence. Regions which do not occur in more than four consecutive frames are rejected to reduce noise. For the remaining regions, the average descriptors regarding the tracking results, are used as a region representation.

In the next step a certain amount of the obtained descriptors is used to create the so called "visual vocabulary" or "visual dictionary" using the k-means clustering algorithm [12]. The individual elements of the visual vocabulary are called "visual words" or "cluster centers" in the context of the k-means algorithm.

This visual vocabulary is applied for vector quantization or "descriptor encoding". This basically means that a descriptor (represented by a vector) is expressed by its "most similar" visual word. Thus, each descriptor is assigned to its "closest" cluster center in the descriptor space. We denote

[1] actually Sivic and Zisserman restrict themselves to one keyframe per second

this method as "hard encoding". This is done for all descriptors of all frames in the video sequence.

As also the descriptors inside the query region are encoded, baseline matches are already obtained. This comes from the fact that multiple descriptors are assigned to the same visual words. Still, the matches are noisy and contain outliers. Therefore, a so called "stop list" procedure is used, which rejects the most common and the most uncommon visual words and the corresponding matches. Another integrated mechanism uses the idea of spatial consistency. This means that descriptors inside the query region should be matched to descriptors that are situated in a certain neighborhood in the investigated frame. If this requirement is not fulfilled, the matches are rejected. The area containing the remaining matches forms the retrieved region of interest. The number of obtained matches is taken as the query ranking measure. Sivic and Zisserman achieved promising retrieval results on two full length feature films with this method.

Another task investigated in [8] is (full) image retrieval. To characterize a full scene, additional steps to the already discussed method have to be performed. With all descriptors of an image assigned to one visual word, a histogram of visual word occurrences is constructed. This histogram or frequency vector is used as an image representation which is further weighted by the term frequency-inverse document frequency (tf-idf) to encode the significance of certain visual words. A similarity measure like the cosine distance between the resulting frequency vector of the query image and the histograms of the images of a database is used to obtain the retrieval results.

To sum up describing an image using the "traditional" BoVW approach consists of the following steps: First, interest regions are extracted and a descriptor per region is obtained. After that, the visual vocabulary, previously learned by applying a clustering algorithm on a set of descriptors from a training data set, is used to encode the descriptors. The resulting histogram of visual word occurrences is weighted to form the final image representation.

In the following we take a look at related work in the field of image recognition which is based on this BoVW approach.

2.2 Image Recognition

The BoVW method was introduced to the image recognition domain by Csurka et al. in [13]. To obtain interest regions, the Harris affine detector [14] is used. As in the original Video Google approach [8], SIFT descriptors

[11] are calculated to represent the detected regions. Further, k-means is used to learn the visual vocabulary and the descriptors are assigned using the hard encoding method. The resulting histogram of visual word occurrences forms the image representation. One-versus-all kernel Support Vector Machines (SVMs) are trained using annotated training images. The one-versus-all strategy means that for each category a binary SVM is learned to distinguish this category from the remaining classes. Given a novel image and the corresponding image representation, the responses of all SVMs are obtained. The category of the SVM with the highest classification response is used as the predicted image category. The approach showed promising results on two standard image recognition datasets.

As a drawback of the conventional BoVW based methods, Lazebnik et al. [15] identified the lack of usage of spatial information, since the information about the spatial position of the individual descriptors is completely lost (keyword "Bag" in Bag-of-Visual-Words). The idea of introducing sub-regions or blocks to describe local regions (e.g. [16]) as well as the full image (e.g. [17, 18]) is a common method in the field of image and object recognition. Therefore, Lazebnik et al. [15] proposed the usage of a pyramid structure to include spatial information. The image is sequentially split in a pyramid-like way[2]. For each arising sub-region a histogram of visual word occurences is obtained, considering only descriptors that are located inside the sub-region. The image is therefore represented by a set of histograms which are further weighted according to their level in the spatial pyramid. The final image representation is the concatenation of the individual histograms. This mechanism is called the spatial pyramid framework. Further, Lazebnik et al. [15] did not use a region detector but sampled image patches on a regular grid. The approach was evaluated on three scene categorization datasets and it was shown that the introduction of spatial information led to an increased performance compared to the baseline BoVW approach. Beside the increased descriptor dimensionality and, as a consequence, increased runtime, the spatial pyramid framework is still considered as state-of-the-art in this field [19].

However, Yang et al. [20] argued that the spatial pyramid framework has to be used with nonlinear kernels to obtain such performances. Although in [21] it was shown that histogram intersection kernel SVMs can be trained and evaluated efficiently, in [20] the even more efficient linear SVMs are used. Further, instead of hard encoding, the sparse coding method was introduced.

[2] a 3-level pyramid (1×1, 2×2, 4×4)

The basic idea of sparse coding is that one descriptor can be assigned to multiple visual words. By using a linear combination of visual words the descriptor can be modeled in a more accurate way. At the same time, the amount of selected visual words should be small, hence the resulting assignment vector should be sparse. This encoding method was already introduced to diverse fields of computer vision. For example in [22, 23] for image denoising, in [24] for the task of image super-resolution and in [25] for texture segmentation. Further, histograms of sparse codes obtained from raw image patches were recently successfully incorporated for object detection in [26]. In the image recognition domain it was used on raw image patches in [27, 28]. Additionally, as many related approaches (e.g. [29, 30, 31]) dealt with the topic of using more sophisticated discriminative dictionary learning methods rather than k-means, Yang et al. introduced a novel algorithm to learn a visual vocabulary optimized for sparse coding. In contrast, the optimization criterion of k-means is independent of the used encoding method. For the obtained sparse codes, the spatial pyramid framework is applied. In contrast to [15] a strategy called max-pooling is applied. This means, that for all codes of a sub-region, the maximum value per visual word entry is obtained. Yang et al. claimed that this strategy is more robust to local transformations [20]. Despite the usage of a more efficient classifier – linear SVMs – it was shown that the performance is at least competitive or even outperforms the approach of Lazebnik et al. [15] on certain datasets.

In [32], Wang et al. argued that the codes of [20] are sparse, but on the other hand the selected visual words may not be visually similar to the given descriptor. Therefore, the novel Locality-constrained Linear Coding (LLC) method was introduced to solve this issue. Another advantage of LLC compared to sparse coding is that the resulting optimization problem has an analytical solution which makes computation more efficient. The LLC computation can further be accelerated by only considering the k nearest-neighbor visual words. This comes from the fact that this strategy already covers the locality constraint. The nearest-neighbor search is performed efficiently by an approximate method. Additionally, a novel dictionary learning algorithm which is optimized for LLC was introduced in [32]. Three image recognition datasets were used for performance evaluation. It was shown that this algorithm outperforms the approach of Yang et al. [20] and experiments pointed out that this approach achieves state-of-the-art performance.

An extension to the BoVW method was introduced in [33] by Perronnin et al. Instead of only considering the visual word assignments for the encoding step, additional information about the distribution of the descriptors is

incorporated. Therefore, Fisher Vector (FV) encoding was suggested. The basic idea is to fit a parametric generative model to the descriptors and finally encode the derivatives of the log-likelihood of the model with respect to its parameters [34]. In [33], a Gaussian Mixture Model (GMM) was used as such a model which further represents the visual vocabulary. In contrast to related BoVW approach, it was shown in [33] that a smaller number of visual words is sufficient for this more sophisticated FV encoding method. Perronnin et al. further improved the FV encoding scheme in [35]. Among others, the usage of FV encoding with the spatial pyramid framework [15] was suggested. The resulting improved version of FV encoding (IFV) achieved state-of-the-art performance on two challenging image recognition datasets and can still be seen as the state-of-the-art in the field of image recognition.

2.3 Sketch Recognition

In the beginning of this section we discuss sketch recognition approaches which focus on a specific domain (e.g. low-level primitives, chemical drawings, electric circuit diagrams etc.). After that, we cover works done in the general sketch classification domain. Finally, we deal with approaches in the related field of sketch-based retrieval.

The task of sketch recognition is simplified by the restriction to one specific domain. We denote this as "constrained sketch recognition". In that case usually only a small amount of categories has to be recognized. Due to the shared domain the problems of intra- and inter-class variability can but not necessarily must be less difficult. A line e.g. can be easily distinguished from a circle in the low-level primitive domain. Related and similar symbols (e.g. for the military diagrams) however are still hard to recognize. Nevertheless, a more specialized sketch representation can be developed. There exists a lot of related work in the field of constrained sketch recognition. We now briefly discuss some selected approaches of that domain.

Constrained Sketch Recognition
Paulson and Hammond [36] proposed a method to recognize eight low-level primitives: lines, polylines, circles, ellipses, arcs, curves, spirals and helices. A sketch is classified as a single primitive or as a combination of multiple primitives, and a beautified version of the sketch is returned. This means that e.g. after a line is recognized, it is replaced by a straight line which connects the two obtained endpoints to create a perfect straight version. Related algorithms are used for the rest of the primitives. An example for this sketch recognition and beautification approach can be seen in Figure 10. The

actual recognition of the individual primitives is performed using geometric constraints. A hierarchy of low-level primitives with an additional ranking algorithm in order to obtain the most likely combination of primitives is introduced. In an experiment almost perfect recognition results were achieved.

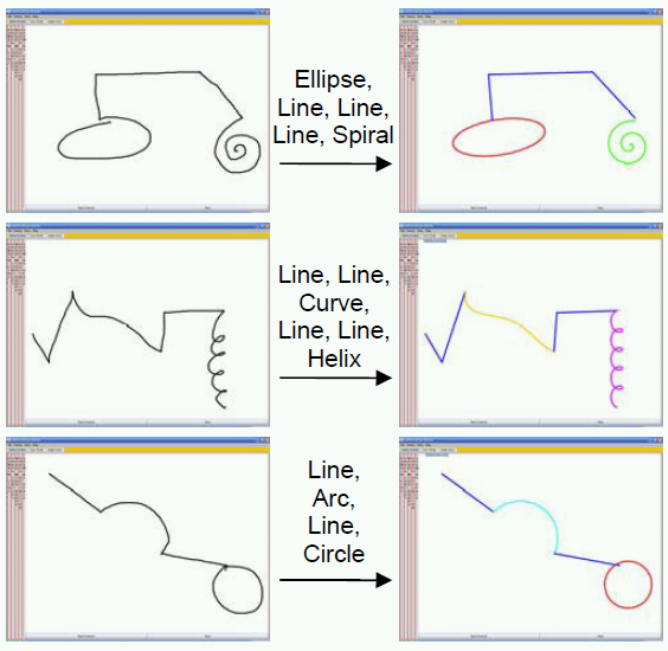


Figure 10: Three example sketches (combinations of primitives) and their corresponding beautified version according to the recognized primitives (captions of the arrows) [36]. Each primitive type is illustrated by a different color.

An application called MathPad was developed by LaViola Jr. and Zeleznik [37] to solve and visualize mathematic problems. It is e.g. possible to recognize a sketch of an equation system which is further solved and the result displayed. An additional feature is the automatic creation of plots for the given mathematical expressions. An example sketch used to visualize the damped harmonic oscillation can be seen in Figure 11. To interact with the system, markers can be drawn inside the sketch (e.g. a dot, line or box) or the corresponding icon in the graphical user interface (gui) can be selected. A three stage algorithm was introduced to perform the actual recognition. In the first step, the sketch is normalized and denoising techniques are applied. Additionally, dominant points inside the sketch are obtained and certain statistics for them are computed. After that, the first recognition step eliminates the most unlikely mathematical symbols. This step produces fast but

coarse recognition results. In the last step, the leftover symbols are considered. The dominant points are used to get the fine recognition result. After the symbols are classified, a post-processing parsing step is used to detect e.g. exponents and fractions. The approach worked fine in most of the cases, if proper drawer-specific training data was used. However, in [37] it was claimed that the recognition stage needs some extensions to work as expected.

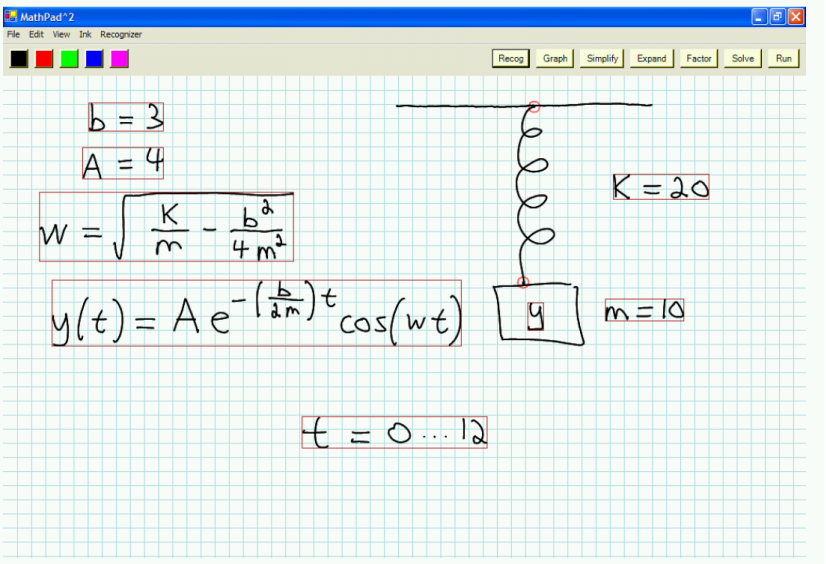


Figure 11: Mathpad example to visualize the damped harmonic oscillation [37].

In [38] Ouyang and Davis introduced an approach to recognize and interpret chemical drawings or formulas. The output of this system is a complete molecular structure [38]. Visual features for different levels of detail are used in the recognition framework. A conditional random field model is applied as a joint graphical model classifier [38]. The approach works in real-time and achieved an almost perfect recognition rate in experiments.

A system for recognizing electric circuit diagrams was developed by Sezgin and Davis in [39]. This approach makes use of the stroke drawing order and handles interspersing drawings [39]. An algorithm based on dynamic Bayesian networks is incorporated for the recognition. Remarkable recognition rates were achieved, however the system was not able to obtain a recognition result in real-time.

Hammond et al. [40] suggested an approach for the recognition of battle scenarios in terms of military course-of-action (coa) diagrams. A large amount of symbols can be classified. In this domain, symbols can be complex and e.g. contain primitives ordered in a hierarchical order as well as handwritten text. In the first step of the algorithm boundary lines are detected and handwriting as well as sub-unit detection procedures are performed. As a next step, corners are obtained before primitive classification using [36] takes place. After that, so called mid-level shapes are determined. Finally, the high-level shape recognition procedure is applied using all of the previously obtained results. As a post-processing step, sub-units which were not recognized, are used as an input for the arrow recognizer. Further, all recognized symbols are tagged with a unique symbol identification code to label their semantic meaning. The approach achieved a high accuracy in experiments when considering the top recognition results per symbol.

Generic Sketch Recognition
Sun et al. [41] introduced a general sketch recognition approach. This means that no domain restriction was given. A dataset consisting of one million clipart images was collected from the internet and used as a knowledge base. Sun et al. claimed that clipart images have similar characteristics as sketches [41]. However, clipart images can additionally contain textural information such as e.g. titles or surrounding text. A sketch-to-clipart image search engine, based on [42], was used to retrieve visually similar cliparts given a sketch. For the obtained results (the retrieved clipart collection), matching scores are available which are used in the subsequent recognition method. For this proposed algorithm, a hierarchical model, consisting of an object topic layer (the category) and a shape topic layer (the shape), is learned. Sketches of different object categories and therefore object topics as e.g. "sun" and "bulk", can be assigned to the same shape topic in the corresponding layer. The model is built with the Expectation-Maximization (EM) algorithm and uses both visual and textural information of the retrieved clipart collection. The recognition result is obtained as the category with the highest model response. In experiments it was shown that this Query-adapted Shape Topic (QST) model works well on a shape dataset as well as on the introduced sketch dataset [41]. Although the dataset consists of a large number of categories, only one sketch per class is available. Further we argue that the additional clipart search engine is not necessary for sketch recognition and that with this approach it is not possible to get immediate recognition results.

A more relevant algorithm was developed by Eitz et al. [5], where for the first time the general sketch recognition task was examined in a large scale setting

[5]. For that reason a sketch dataset consisting of 250 categories and overall 20 000 sketches was introduced. More details about this dataset, which we also use throughout this thesis, are given in Section 4.1. The developed algorithm is based on the BoVW model. For large image patches extracted on a regular and overlapping grid, the introduced Local Histograms of Oriented Gradients (SHOG) [43] descriptors are obtained. SHOG is closely related to SIFT [11]. However no magnitude information is taken into account. In the encoding step, soft or kernel-codebook encoding [44, 45] is used. The basic idea of soft encoding is to assign one descriptor to a set of similar visual words. As in [13], one-versus-all kernel SVMs are incorporated as a classifier. A remarkable recognition accuracy was achieved on the introduced dataset when the majority of the sketches were used for classifier learning. Further, a human sketch recognition experiment on this dataset was performed and the results were discussed in [5]. We argue that although high recognition scores are achieved for this BoVW approach, more sophisticated pipeline elements can lead to an improved performance.

Li et al. [46] proposed a recognition algorithm based on the ensemble matching strategy. In contrast to [5], the descriptors are obtained for patches located on sketch lines. A star graph is created with the patch positions used as nodes. The edges are links between these positions and the graph center. The descriptors are introduced as edge weights. The ensemble matching problem can be seen as a graph matching problem: given a query graph, find the most similar graph in the database. Further, to limit the amount of graphs to be compared, category filtering is applied. This means, that the most promising categories are obtained by a default BoVW based procedure similar to [5]. Afterwards, the graph matching problem is applied to the graphs of the remaining categories. The final recognition result is obtained as the category which occurs most often in the k nearest-neighbor graphs of the database. When using the sketch dataset introduced in [5], this ensemble matching based approach outperformed the algorithm of Eitz et al. [5] with a remarkable gap. Without the usage of category filtering still a remarkable recognition rate was achieved. However, we argue that due to the additional category filtering procedure and the computational expensive ensemble matching problem, immediate recognition results are not possible.

Sketch-based Retrieval

In [47] a sketch-based approach for image retrieval was proposed by Riemenschneider et al. Given a sketch of an object, corresponding natural images of a database depicting that object should be found. Both sketches and natural images in the dataset are described by their shape. In more detail the

shape is represented by contours which are defined as a connected sequence of points [47]. This contours come from sketch strokes or from edges obtained from natural images. The contours are described analyzing fragments which are connected subsets of a contour represented by an ordered list of L points [47]. A novel descriptor is introduced, which consists of angles between lines that connect points on the fragment. Each point is used as a reference point and compared to the remaining points. This results in a $L \times L$ descriptor matrix for each contour fragment. A vocabulary tree [48] is used to build the visual vocabulary from the obtained descriptor matrices of the database images. A vocabulary tree uses k-means in a hierarchical way and is efficient both in learning and for the nearest-neighbor search which is in further consequence used for descriptor assignment. The retrieval scores are obtained by using assignment statistics from the vocabulary tree. The database images with highest scores are provided as the retrieval result. Experiments revealed that the introduced shape descriptor performs on average 25% better than related shape descriptors. In experiments a remarkable performance was shown which was further increased when hand-drawn prototype models were used for the database.

A sketch-based 3D model retrieval approach was developed in [49]. The goal is to return 3D models which look similar to a given sketch of an object. The algorithm is based on the BoVW sketch recognition pipeline used in [5]. However, adaptions as well as additional steps are necessary to match 2D sketch images to 3D models. First, the amount of uniformly distributed views per 3D model is fixed. Further, for the selected viewpoints, line rendering techniques are applied in order to get 2D sketch-like drawings. The Gabor Local Line-Based Feature (GALIF) representation based on a Gabor filter bank is used to obtain filter response images. These filter response images are used to obtain the final descriptor in a similar way as done in [5]. The visual dictionary is created from GALIF descriptors of all models and views. In contrast to [5], hard encoding is applied to obtain the histogram of visual word occurrences which forms the final sketch/model representation. Given the query sketch, the 3D models and the corresponding views with the highest matching scores for the representations are obtained as the retrieval result. A remarkable retrieval score was achieved when a 3D model dataset and one sketch per model (as a query) were used. Experiments pointed out that the developed retrieval algorithm outperformed leading sketch-based retrieval approaches [49].

Chao et al. [50] introduced an algorithm for sketch retrieval. Given a query sketch, similar looking sketches of a database should be obtained. For each

sketch image, sampling positions that are situated on sketch lines are used. For these positions image patches are extracted and descriptors are calculated using Poisson-based Histogram of Oriented Gradients (PHOG) [50]. As in [47] a vocabulary tree [48] is used to learn the visual vocabulary. When assigning the descriptors of a sketch to the vocabulary tree, the histogram of all investigated paths is used as a sketch signature [50]. A spatial pyramid is incorporated and the concatenation of the individual histograms forms the final sketch representation. The retrieval results are obtained by comparing the representations of the query sketch and the dataset sketches. The database sketches with the highest similarity scores are defined as the retrieval result. For experiments the sketch dataset introduced in [5], an office icon library and a shape dataset were used. On all three datasets the algorithm outperformed alternative retrieval methods in the sketch retrieval domain. However, these alternative methods were developed for different computer vision tasks, such as sketch-based image retrieval, shape matching and image recognition.

3 Bag-of-Visual-Words for Sketch Recognition

For classifying sketches, we first have to train a classifier using a sketch image representation suited to deal with the versatile properties of sketches (see Section 1.2). The same representation is then analyzed for a previously unseen test sample to uniquely assign a class label. We adapt the Bag-of-Visual-Words (BoVW) method discussed in Section 2.1 to obtain such a sketch representation. The general recognition pipeline, is discussed in detail in Section 3.1. We explain each pipeline element and outline possible variants in Sections 3.2.1-3.2.6. Section 3.2.1 introduces necessary pre-processing steps. Sampling strategies as well as the evaluated descriptor types are explained in Section 3.2.2. Further, methods to learn the visual vocabulary are discussed in Section 3.2.3. The encoding step is handled in Section 3.2.4. In Section 3.2.5 we outline the subsequent pipeline element of code pooling. Finally, the investigated classifier types are discussed in Section 3.2.6. We therefore have multiple variants for each pipeline step which we use in our extensive evaluation phase in order to obtain an algorithm which is (1) powerful regarding recognition performance and (2) is efficient.

3.1 Sketch Recognition Pipeline

Our sketch recognition pipeline follows the BoVW method [8]. As for every machine learning approach the process consists of a learning and a classification phase. First, a classifier has to be trained using a suitable sketch representation. This sketch representation is also used in the classification phase to recognize the category of novel sketches. A visual depiction of the general pipeline (including both phases) is illustrated in Figure 12.

For the learning procedure a set of annotated sketch images has to be provided as training data. To obtain a reasonable sketch representation, it is important to pre-process the sketch images first, in order to achieve a recognition approach of high quality. This step can include e.g. image scaling, type conversion or smoothing. In general, normalization issues are tackled with this processing. Further, for each pre-processed sketch image, descriptors are extracted based on the defined sampling strategy (i.e. the locations for local patches). These descriptors are used to describe local areas of the sketch images or the whole sketch images themselves (depending on the descriptor type) in a compact but still robust way. Optionally, a certain (e.g. random) subset of descriptors can be chosen for further processing. In the next step

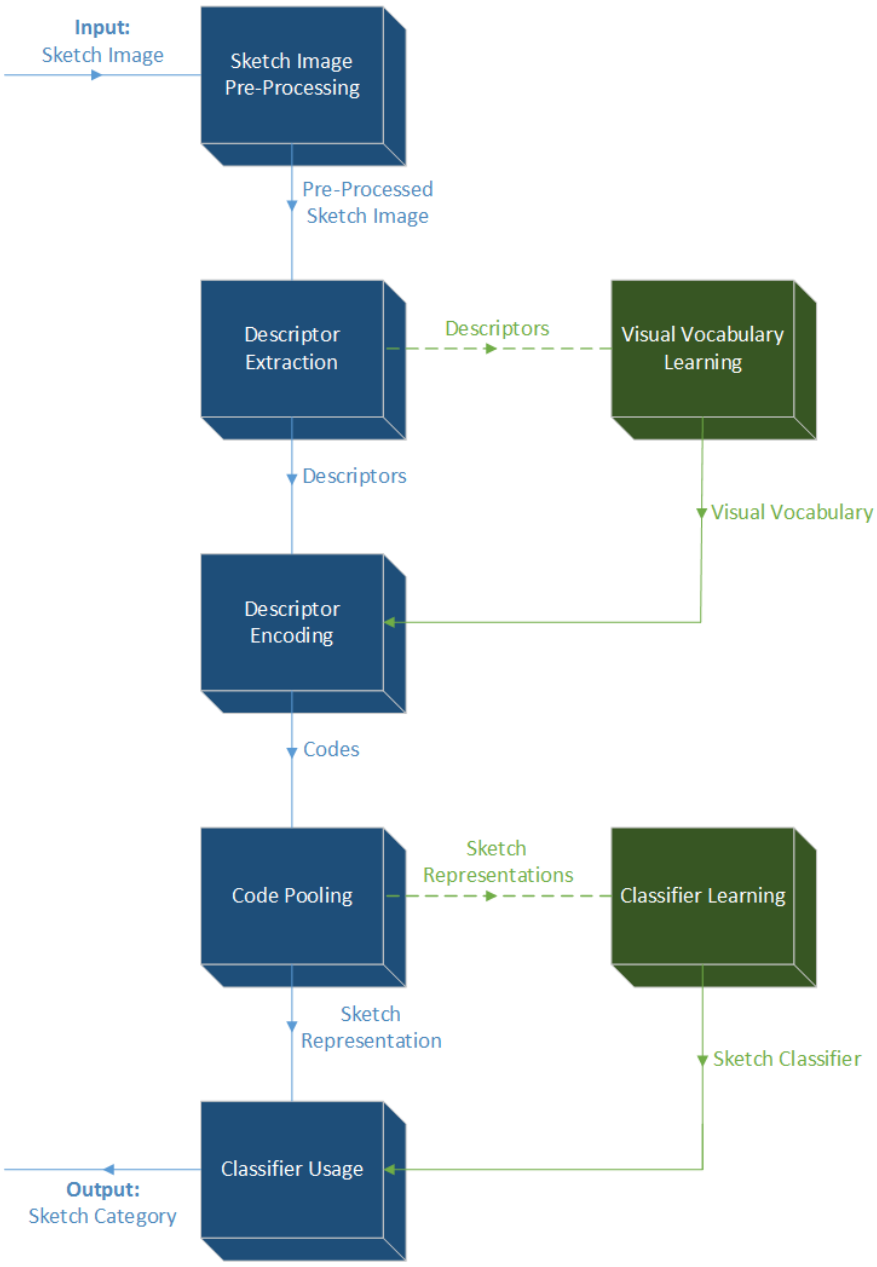


Figure 12: Our general sketch recognition pipeline. The elements are represented by rectangles/boxes whereas arrows depict the in- and output for these steps. Elements with a green color are the parts of the algorithm which are only involved in the learning process. The outputs of these elements (marked with a green solid line with arrows) are further used in the classification process as an input.

the visual dictionary is learned. The resulting visual words can be seen as prototypical examples obtained by descriptor clustering. In the encoding step the visual dictionary is used to describe each extracted descriptor in a highly compact way. In order to do so, descriptors are assigned to one or more visual words. Further, this assignment can be weighted to express the importance of certain visual words. The resulting codes are then grouped in the pooling step. The pooling procedure aggregates the information available over a defined spatial area. Therefore, localization information (i.e. the positions where the descriptors are extracted) is thrown away. This yields a final descriptor for each sketch which forms our sketch representation. Finally, these individual sketch representations are used with their known ground truth categories to learn the sketch classifier.

To recognize a novel image, the same sketch representation is built using the previously learned visual dictionary. Further, this representation is passed to the trained classifier, which predicts the sketch category.

We now discuss each step in more detail and explain possible techniques from both the sketch and image recognition domain.

3.2 Pipeline Elements

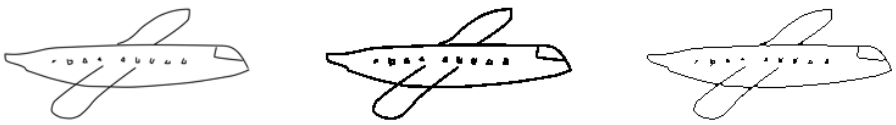
In this subsection we take a closer look at all pipeline elements and investigate variants evaluated during our extensive evaluations (Section 4). For techniques which are developed for image recognition approaches we point out their origin.

3.2.1 Sketch Image Pre-Processing

Image pre-processing is an essential mechanism to obtain a high performance recognition approach as it sets the basis for the rest of the algorithm. In our approach this step consists of the following steps. First, the sketch image is scaled to a fixed size of 256×256 pixels. Further, the bounding box of the sketch itself is calculated and scaled such that the longer side has a fixed size of 202 pixels. Additionally, the sketch is centered inside the 256×256 pixel image.

Next, we convert the sketch image into one of the following three sketch formats. The *grayscale* format is created by applying anti-aliasing methods on the drawn binary sketch. As illustrated in Figure 13(a) the sketch lines look smooth and natural. Note that the TU Berlin sketch dataset [5], which is used throughout this thesis, is available in this format. As an alternative

sketch format we introduce a binary representation of the sketch by applying a thresholding operation. Only image pixels with a grayscale value below 255 are set to one. This format results in a sketch which looks angular as shown in Figure 13(b). Further, the sketch lines look thicker than in the *grayscale* format. As a result of that observation we denote this format as $binary-thick$. In order to obtain a more natural binary representation of the sketch we introduce the $binary-thin$ format, where we apply morphological operations to thin out the binary sketch. An example sketch of this format is given in Figure 13(c). From a visual point of view this format looks more similar to the *grayscale* format than to the $binary-thick$. However, from a technical view the *grayscale* and the $binary-thick$ format only differ in the way the sketch borders are represented. In the $binary-thick$ format only binary information is available, whereas in the *grayscale* format an integer value $(0\ldots255)$ is used. When using gradient-based descriptor approaches such as Scale-Invariant Feature Transform (SIFT) [11], the two binary formats lead to constant gradient magnitudes. For the *grayscale* format these values are variable. The two binary formats only differ in the amount of pixels set. An evaluation regarding the three sketch formats is performed in Section 4.4.



(a) Grayscale (b) Binary-Thick (c) Binary-Thin

Figure 13: The three defined sketch formats.

As a final image pre-processing step we apply image smoothing. This can e.g. improve the accuracy when calculating the gradient orientation as well as the gradient magnitude of an image. As a result of that, we argue that the sketch can also be described in a more robust way, which leads to a better overall recognition performance.

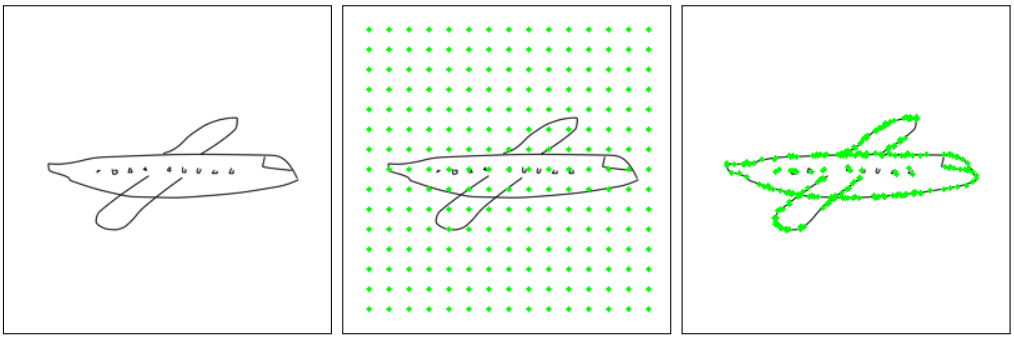
3.2.2 Descriptor Extraction

In the context of the descriptor extraction step, we discuss two aspects. First, the strategy of how descriptors are extracted is defined. This strategy includes the amount of features as well as the positions where the extraction process takes place. However, this strategy is only relevant for local descriptor types. As a second aspect we investigate variants of the actual descriptors.

Sampling Strategy

In this thesis we investigate two sampling strategies, *grid* and *point* sampling. When *grid* sampling is used, local descriptors are extracted on a regular grid, i.e. every n pixels. *Point* sampling means that the locations at which local features are extracted, lie on sketch line pixels. For the *grayscale* sketch format we define these pixels as pixels which are set to one in the corresponding $binary - thick$ format. *Point* sampling is related to how the locations are obtained when using the Shape Context descriptor [51]. Both methods were already successfully integrated in the sketch recognition domain. In [5], grid sampling is used whereas point sampling is integrated in [46]. However, to the best of our knowledge no direct comparison has been performed for both strategies in the sketch domain. Therefore, we discuss our evaluation regarding this topic in Section 4.5.1.

In the image recognition domain *grid* sampling is often the sampling method of choice as already discussed in Section 2.2. However, for sketch recognition it is possible that certain locations do not include any sketch content. In this case we reject these empty descriptors. On the other hand, more kinds of representations of the sketch are available with *grid* sampling. In contrast, with *point* sampling the restriction that the center pixel has to lie on a sketch line is present. As a consequence, no empty descriptors are extracted with *point* sampling. A visual illustration of the two sampling strategies can be found in Figure 14.



(a) Sketch Image (b) Grid Sampling (c) Point Sampling

Figure 14: Illustration of the two sampling strategies investigated in this thesis. An equal amount of 225 locations is used for both strategies.

Additionally, we introduce the possibility to choose a maximum amount of descriptors used per sketch for further processing, e.g. for reducing the memory profile. However, in general this also leads to a less accurate sketch

description. If applied, we choose the defined amount of descriptors at random for each sketch. The impact of this optional strategy is discussed in our evaluations in Section 4.5.1.

Descriptor Type

The second important aspect regarding descriptor extraction is the actual choice of a suitable descriptor type. As briefly mentioned before we distinguish between local and global descriptors. Local descriptors are used to describe small parts/areas of images. Therefore, the full image is described by a set of local descriptors. In contrast, with a global descriptor the full image is represented by one single descriptor. To cover a large range of descriptor types we choose methods which are based on different ideas, e.g. gradient-based, similarity-based etc. We selected methods which are widely used in the field of image and sketch recognition. In the following we briefly discuss the selected descriptor types:

1. Dense Scale-Invariant Feature Transform (SIFT) [52, 53]
   Note that Dense SIFT actually combines a sampling strategy and a descriptor type, where the latter part is the famous SIFT descriptor [11]. However, in contrast to the "traditional" SIFT approach, the native preparation steps are bypassed. This means that neither a keypoint detection at different scales, nor an orientation assignment for the found keypoints are performed. Instead, the keypoint locations are previously defined at a fixed scale and rotation. These locations are distributed in a dense manner over the image as for grid sampling (Figure 14(b)). In other words, the descriptors are computed at locations on a regular grid every n pixels [53]. Then, the actual SIFT descriptor calculation is performed for all obtained locations as follows: the patch around each location is sampled. For that patch, the gradient magnitudes and orientations for each pixel are obtained, where the corresponding gradient magnitude is ranged in one of the eight bins according to the pixel's gradient orientation. Additionally, the magnitude values are weighted by a Gaussian weighting function according to their distance to the patch center. Further, the patch is split into 4×4 sub-regions. By only considering the gradient magnitudes and orientations within each sub-region, one histogram per sub-region is obtained. The gradient magnitudes from neighboring sub-regions are incorporated by trilinear interpolation. Finally by combining these 16 histograms with each of them having eight bins, a 128 dimensional descriptor is formed. As a final step, the descriptor is normalized to unit length.

   The Dense SIFT method is widely used in image recognition approaches

such as e.g. [53, 19, 15, 20, 32]. Additionally, in [46] it was introduced to the sketch recognition domain.

2. Histogram of Oriented Gradients (HOG) [52]
   In contrast to SIFT, HOG is a global descriptor. However, HOG is built on the same basic ideas and therefore calculates gradient-based histograms. The image is organized into so called cells which are situated densely in the image (grid sampling). For each of these cells a gradient-based histogram is calculated. The histogram creation follows the SIFT descriptor procedure as already discussed. However, nine instead of eight orientation bins are used. Multiple cells are grouped into larger and overlapping spatial blocks [52]. The cell histograms of all blocks are normalized and concatenated to build the final global descriptor.

   To the best of our knowledge this global descriptor has not been used for sketch recognition up to this point. However, it is interesting how HOG with its single global descriptor performs compared to the local Dense SIFT method in this task.

3. Local Histograms of Oriented Gradients (SHOG) [5]
   This descriptor type again is closely related to the SIFT descriptor. However, this descriptor as introduced in [54], is designed specifically for the task of sketch recognition. The calculation follows [11] but the information about the gradient magnitude is ignored. This comes from the fact that when using binary sketches, the gradient magnitude does not lead to additional useful information. Therefore, one sub-region histogram does only represent orientation rather than magnitude responses. In [5] four orientation bins are used and bilinear interpolation is performed with a grid sampling strategy. This results in a final descriptor vector with a length of 64.

   The SHOG descriptor is incorporated in the successful sketch recognition approach by Eitz et al. [5].

4. Shape Context[51]
   This descriptor type was introduced by Belongie and Malik [51] to perform shape matching or to determine shape correspondences. As we can see in Figure 4 one could argue that a sketch is a shape or contour of an object. Therefore, the task of shape matching can be seen as related to matching sketches. To calculate the descriptor n sample points are obtained from contours of the shape. Alternatively, edges can be used for real-life images. This corresponds to our point sampling strategy as already defined in this section. For a selected reference point from this point set, a log-polar

coordinate system is set up. In further consequence this system is used to create a histogram which represents the relative coordinates of the remaining $n-1$ points. Therefore, the relative distribution of points is captured [51]. In [51], this histogram consists of 12 orientation bins and five distance bins, yielding a 60-dimensional descriptor. To describe the complete shape, all n points of the set are used as a reference once. This strategy results in n descriptors to represent the whole shape.

As already discussed, shape and sketch representations are obviously related. Hence, the performance of this descriptor was already evaluated for sketch-based image retrieval in [54]. Although this descriptor was outperformed by the SHOG representation, we include the Shape Context into our evaluations to further investigate the relation between shapes and sketches.

5. Self Similarity [55]
   In [55] the Self Similarity descriptor is introduced to describe an object in an image. In further consequence, matches within other images depicting this object should be found. The method is based on the geometric layout of local self similarities rather than image properties [55]. To obtain the descriptor, a reference pixel q is selected. A small reference patch (e.g. 5×5 pixels) centered at q is compared to a larger surrounding image region (e.g. with a radius of 40) which is also centered at q. The sum of square differences (SSD) is used as a similarity measure. As for the Shape Context descriptor, a log-polar coordinate system is set up for the resulting correlation surface. The consequential histogram is called the Self Similarity descriptor and in [55] 20 orientation and four distance bins are used. This leads to an 80-dimensional descriptor vector. To describe the full image, the descriptor is calculated every five pixels in a grid sampling manner.

   An interesting relation to sketch recognition is also investigated in [55]. It is shown that this descriptor works well for sketch-based image retrieval, although pictogram-like hand-sketched templates are used as sketches. However, these templates are closely related to what we defined as a sketch. Further, a descriptor comparison in the context of sketch recognition was performed in [46]. Again a gradient-based method (Dense SIFT) outperformed this descriptor type. However, we use the Self Similarity descriptor in our evaluations due to the observations in [55].

As an optional procedure we apply the RootSIFT strategy, as introduced in [56] by Arandjelovic and Zisserman, for the SIFT-based descriptor types. In contrast to the initial descriptor, the square root of it is taken. By using the

Hellinger kernel instead of the Euclidean distance metric, a more accurate similarity measurement can be achieved as shown in [56]. Although introduced for object retrieval, Arandjelovic and Zisserman concluded that this strategy can further improve the performance in various computer vision domains such as object detection or image recognition. Therefore, we include this procedure in our evaluation, as the additional computation can be performed at low cost due to highly optimized operations.

The evaluations regarding the descriptor type can be found in Section 4.5.2.

3.2.3 Visual Dictionary Learning

The visual dictionary can be seen as the core part of the BoVW method as it is used in the encoding step to represent the individual descriptors. In further consequence, these codes form the basis of the final sketch representation. To obtain a visual dictionary of high quality it is essential to select a suitable learning procedure. Given a set of n descriptors $X = [x_1, \ldots, x_n]^T \in \mathbb{R}^{n \times d}$ with a dimensionality of d, the goal is to learn a visual dictionary $B = [b_1, \ldots, b_k]^T \in \mathbb{R}^{k \times d}$ consisting of k visual words. To the best of our knowledge, so far sketch recognition approaches based on the BoVW method only used k-means to learn the visual dictionary regardless of the incorporated encoding method. However, in the image recognition domain and as discussed in Section 2.2 these two pipeline steps can be coordinated to achieve an increased performance. Therefore, we introduce successful visual dictionary learning approaches in our evaluation. A list of investigated visual dictionary learning algorithms is given in the following enumeration. Further, we briefly discuss each learning procedure and, if applicable, also explain the relation to the coupled encoding method.

1. Random
   This method defines k randomly selected descriptors of X as the visual dictionary B. This strategy is e.g. suggested in [57]. It is further shown in [57] that the obtained dictionaries can be competitive to dictionaries created by more sophisticated learning algorithms when a suitable encoding procedure (e.g. sparse coding) is used.

   Due to this fact and as it is a primitive and efficient way to obtain the visual dictionary, we add this approach to our evaluation.

2. K-Means [12]
   The famous k-means algorithm is widely used in many successful BoVW-based approaches such as e.g. [8, 15, 5]. Given X, the goal is to find

the k cluster centers/visual words B as well as the descriptor-to-cluster assignments $u_1, \ldots, u_n \in \{1, \ldots, k\}$ such that the distances between the descriptors and the assigned cluster centers are minimal. More formal, the cost function

$$E = \sum_{i=1}^{n} \|x_i - b_{u_i}\|^2 \tag{1}$$

should be minimized, where b_{u_i} is the assigned cluster center of descriptor x_i (indexed by u_i).

The first step of the algorithm is the initialization of the cluster centers $b_1, \ldots, b_k \in \mathbb{R}^d$. This can be done in two ways. The first version randomly selects the cluster centers as it is done when using the random dictionary strategy discussed above. The second initialization method is called k-means++ [58] and greedily selects k descriptors such that their distance is maximized. Next, the descriptors are assigned to their closest initial cluster centers by a nearest-neighbor search. In a mathematical way the assignment is performed such that

$$u_i = \underset{j}{argmin} \|x_i - b_j\|^2 \tag{2}$$

is satisfied. The resulting assignment vector is used in the subsequent step to update the initial cluster centers as

$$b_j = avg\{x_i : u_i = j\}. \tag{3}$$

This means that the cluster center b_j is set to the average of all descriptors previously assigned this cluster. The assignment and update steps are repeated until convergence of the cluster centers.

Due to its wide application and the integration in the successful sketch recognition approach of Eitz et al. [5], we use this algorithm as well as both initialization strategies for our evaluation.

3. Approximated K-Means [59, 60]
   As the assignment step of k-means can be time-consuming, the idea is to replace the exact nearest-neighbor search by an approximated one. We apply the idea of [61] to include a randomized kd-tree forest for this task. A kd-tree is a data structure based on binary trees which is able to handle k-dimensional data. It can further be used to perform an approximated nearest-neighbor search highly efficient. In a traditional kd-tree, at each non-leaf node the data is split into two parts. This is done by choosing the descriptor dimension with the highest variance and by using the median

value of that dimension as a split threshold. This splitting procedure is iteratively performed for both resulting paths until no further split is possible. For a randomized kd-tree the splitting dimension as well as the splitting value are obtained in an approximated way. By creating not one but multiple kd-trees a forest is built. For the actual nearest-neighbor search all kd-trees are used simultaneously. After a certain number of examined nodes, the most promising candidates are returned.

In [61] it is shown that this approximated version achieves similar performance as exact k-means in an object retrieval setup. As these two aspects are in accord with the two constraints for our algorithm, we include this k-means version in our evaluation.

4. K-SVD [62]
   The K-SVD dictionary learning algorithm can be seen as a generalization of k-means. Instead of assigning one descriptor to exactly one visual word, with K-SVD it is possible to represent a descriptor by a linear combination of multiple visual words. The algorithm aims at solving the optimization problem

$$\min_{U,B} \|X - BU\|_F^2 \;\; s.t. \; \forall i, \|u_i\|_0 \leq T, \tag{4}$$

   where $U \in \mathbb{R}^{n \times k}$ is the encoding matrix for all descriptors in X. Further $\|\cdot\|_0$ means the zero-norm which counts the non-zero elements of one code vector $u_i \in U$ and T is a sparsity parameter which limits the amount of visual words used per descriptor. Therefore, this problem statement is related to sparse coding which we explain in full detail in Section 3.2.4. In following we denote this encoding formulation as l_0 sparse coding. Due to the joint problem formulation, a visual dictionary optimized for the l_0 sparse encoding scheme is learned. The algorithm solves this problem in an iterative way. At the first step, the dictionary B is fixed and the sparse code matrix U is obtained using the Orthogonal Matching Pursuit (OMP) [63] algorithm which is explained in Section 3.2.4. Next, the sparse code matrix U is fixed in order to update the dictionary B using Singular Value Decomposition (SVD). Instead of exact SVD, numerical methods, as proposed in [64], can be used to make the algorithm highly efficient.

   Therefore, it can be seen as an alternative to the time-consuming methods introduced in [20, 32]. It is further shown (e.g. in [20, 32]) that encoding specialized dictionaries leads to remarkable performance boosts in the domain of image recognition. Hence, we incorporate it as a possible dictionary learning algorithm.

5. Online Dictionary Learning for Sparse Coding (ODL) [65]
   Similar to K-SVD, the ODL algorithm is developed for the usage with sparse coding. Therefore, again the dictionary is explicitly optimized for the specific encoding method. However, the l_1-norm sparse encoding formulation is considered. The joint optimization problem

$$\min_{U,B} \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{2} \|x_i - Bu_i\|_2^2 + \lambda \|u_i\|_1 \right) \tag{5}$$

   is solved by alternately fixing the dictionary B and the sparse code matrix U. As before λ is a sparsity parameter. For each iteration of the algorithm, one descriptor is drawn at random. For iteration t we denote this descriptor as x_t. With the fixed dictionary obtained in the last iteration B_{t-1}, the sparse code u_t for x_t can be calculated as

$$u_t = \min_{u} \frac{1}{2} \|x_t - B_{t-1}u\|_2^2 + \lambda \|u\|_1 . \tag{6}$$

   Note that in this sparse coding formulation the l_1-norm is included. However, as discussed in [65] this also leads to sparse solution in terms of the zero-norm. To solve this problem the Least Angle Regression (LARS) algorithm [66] is applied, which is discussed in Section 3.2.4. The encoding u_t for the sample x_t is used to update the dictionary B_{t-1} by the optimization problem

$$B_t = \min_{B} \frac{1}{t} \sum_{i=1}^{t} \frac{1}{2} \|x_i - Bu_i\|_2^2 + \lambda \|u_i\|_1 . \tag{7}$$

   It is solved by processing one dictionary column/visual word at a time using algebraic operations. After a number of T iterations, the final visual dictionary $B = B_T$ is obtained.

   Whereas K-SVD uses OMP, which can be seen as an approximation for the encoding step with a fixed maximum number of non-zero coefficients, ODL integrates the LARS algorithm which produces more accurate reconstructions [65]. Both LARS as well as the algebraic dictionary update step can be performed in a fast way. However, the K-SVD algorithm with OMP is more efficient. Nevertheless, we want to compare these two learning procedures in our evaluation phase.

6. Gaussian Mixture Model (GMM)
   A more complex type of a visual dictionary can be obtained by the usage

of a GMM learned from the descriptors X. A GMM consists of a set of K Gaussian distributions and is defined mathematically by

$$p(x|\phi) = \sum_{k=1}^{K} p(x|\mu_k, \Sigma_k)\pi_k \tag{8}$$

and

$$p(x|\mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^d det\Sigma_k}} \exp(-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)) \tag{9}$$

where x is one descriptor and ϕ is the collection of all parameters needed to specify the GMM. This includes $\{\pi_1, \mu_1, \Sigma_1, \ldots, \pi_K, \mu_K, \Sigma_K\}$, with the prior probabilities $\pi_i \in \mathbb{R}$. The means as well as the covariances of the Gaussian distribution with index i are represented by $\mu_i \in \mathbb{R}^d$ and $\Sigma_i \in \mathbb{R}^{d\times d}$ respectively. The resulting visual dictionary can therefore be seen to be probabilistic.

With the descriptors X, the GMM parameters ϕ are learned by the Expectation Maximization (EM) algorithm [67]. The goal of this algorithm is to find the parameter set ϕ such that the likelihood of the GMM given X is maximized. ϕ is initialized by using the k-means algorithm as discussed before. The obtained cluster centers are used as the initial means μ_i. Further, the covariance matrices Σ_i can be obtained by using the data assignments, and the initial prior probabilities π_i are set to the mass of the k-means clusters. The EM algorithm consists of two steps which iteratively update the Gaussian distributions: the data assignment step (expectation) and the update step (maximization) which uses the assignments. As in sparse coding one descriptor can be assigned to multiple Gaussian distributions. The GMM defines the assignment of descriptor x_i by

$$q_i(k) = \frac{p(x_i|\mu_k, \Sigma_k)\pi_k}{\sum_{j=1}^{K} p(x_i|\mu_j, \Sigma_j)\pi_j} \tag{10}$$

for $k = 1, \ldots, K$. This dictionary type was suggested as an improvement to the BoVW method in [68] as it covers more information (covariance and prior probability) than traditional dictionaries. Further, this dictionary type forms the basis for more complex encoding schemes such as e.g. Fisher Vector (FV) encoding [33, 35] which we discuss later on in Section 3.2.4. In recent image recognition approaches this combination often achieved state-of-the-art performance. To the best of our knowledge, this dictionary type has not been introduced in the sketch domain so far. Therefore, we include GMM learning as one option in our evaluation.

The results of our evaluation regarding the dictionary learning algorithm are discussed in Section 4.6.

3.2.4 Descriptor Encoding

The set of descriptors $X = [x_1, \ldots, x_n]^T \in \mathbb{R}^{n \times d}$ can now be represented by the elements of the learned visual dictionary $B = [b_1, \ldots, b_k]^T \in \mathbb{R}^{k \times d}$. This step is called descriptor encoding as one descriptor x_i is represented by its resulting code c_i produced by the applied encoding method. The representations of all descriptors form the code matrix $C = [c_1, \ldots, c_n]^T \in \mathbb{R}^{n \times k}$. Therefore, the encoding step produces an efficient reconstruction for the descriptors using a compact set of representatives – the visual words. However, the dimensions of the code matrix C depend on the encoding method used. When the encoding scheme includes an additional pooling strategy, as e.g. for the FV encoding, the code matrix becomes a vector which forms the final sketch or image representation rather than the individual descriptor codes. However, we define the default code matrix dimensions as $n \times k$. We now briefly discuss each evaluated encoding method and, if available, point out included procedures, e.g. a pooling strategy. Further, we discuss relations to encoding-specific dictionary learning methods already explained in Section 3.2.3.

1. Hard Encoding (vector quantization) [8]
   The simplest method is to choose the visual word b_j which is most similar to the given descriptor x_i. In the descriptor space one can also say that b_j is the nearest-neighbor of x_i with

   $$j = \underset{j \in \{1,\ldots,k\}}{argmin} \|x_i - b_j\|^2 . \tag{11}$$

   This is equivalent to the assignment step of the k-means algorithm already discussed in Section 3.2.3. The resulting code c_i is a zero vector of size k with only one non-zero element at position j – the index of the selected visual word.

   Despite its simplicity, this encoding method is part of successful BoVW-based approaches in the image recognition domain as e.g. in [8, 15]. Therefore, we include this encoding method as a baseline approach.

2. Approximated Hard Encoding [69]
   As all descriptors in X have to be compared to all visual words of the dictionary B, hard encoding can be a time-consuming method. Therefore, a kd-tree is introduced to perform the nearest-neighbor search is an

approximated and efficient way. This strategy was already discussed for the approximated version of k-means in Section 3.2.3.

A drawback of this method is that by limiting the number of comparisons, it is not guaranteed anymore that the exact nearest-neighbor is found. Therefore, the descriptor representation can become less accurate. However, in [61] it is shown that the overall performance of a BoVW-based object retrieval approach using this approximated version is similar. As we also aim at designing an efficient recognition algorithm we introduce this fast encoding scheme into our evaluation.

3. Soft Encoding [44, 45]
In contrast to the previous two encoding methods, soft encoding represents one single descriptor x_i as a linear combination of multiple visual words $B_L = [b_1, \ldots, b_l]^T \in B$ with $l << k$. Therefore, the corresponding code c_i now also consists of l non-zero elements. The l visual words are selected as the l nearest-neighbors of x_i. Additionally, the l individual code entries are weighted as

$$w_j = exp(-\frac{dist_j^2}{2\sigma^2}), \tag{12}$$

where $dist_j$ is the distance/similarity between the visual word $b_j \in B_L$ and x_i. The σ-parameter is used to adapt the spatial scale for high responses. Therefore, closer visual words within the l nearest-neighbors achieve a higher code response.

In contrast to hard encoding, the descriptor x_i can be represented in a more accurate way as multiple visual words are involved. If e.g. a descriptor is almost equally close to two visual words, hard encoding only chooses the nearest-neighbor visual word. With soft encoding both visual words are considered and weighted according to their similarity. Further, a kd-tree is used to make this encoding method as efficient as approximated hard encoding. As this encoding method is both efficient and successfully used in object retrieval [44], image recognition [45] and sketch recognition [5], we introduce this method to our evaluation.

4. Sparse Coding [20]
As already discussed in Section 2.2, the two goals of sparse coding are (1) an accurate representation by using multiple but (2) few visual words to make the code vector c_i sparse. The two criteria lead to two variances of problem formulations with the first defined as

$$\underset{c_i}{argmin} \ \frac{1}{2} \left\| x_i - c_i B \right\|^2 + \lambda \left\| c_i \right\|_1 . \tag{13}$$

We denote this as the "original" sparse coding problem. In literature it is also often called the Least Absolute Shrinkage and Selection Operator (LASSO) problem. λ is again a sparsity regularization parameter. Further $\|\cdot\|_1$ denotes the l_1-norm. Note that, as already mentioned, this norm does not count the amount of non-zero elements in c_i. However, as discussed in [65], this l_1 penalty term also produces sparse results regarding the zero-norm. The problem can be solved by using the LARS algorithm [66].

The basic idea of this regression algorithm is that given a target vector y, the sample $z_0 \in Z$ with the highest correlation to y should be selected. To reconstruct y, the vector z_0 is followed until the sample does not have the highest correlation with y anymore. A new sample $z_1 \in Z$ now has the highest similarity. At this point the new direction of the approximation is calculated as the angle bisection of the two samples z_0 and z_1. The two steps of direction and length computation are repeated until l vectors are used. Further, the subsequent vector directions are calculated by using the angles of all utilized samples. For this algorithm it is guaranteed that a global optimum is found [70, 71]. A graphical illustration of the LARS algorithm for a toy example can be seen in Figure 15. In the context of our sparse coding problem, the target vector is the descriptor x_i and the samples correspond to visual words in B.

We already discussed how to obtain an optimized dictionary for this encoding scheme with the ODL algorithm in Section 3.2.3.

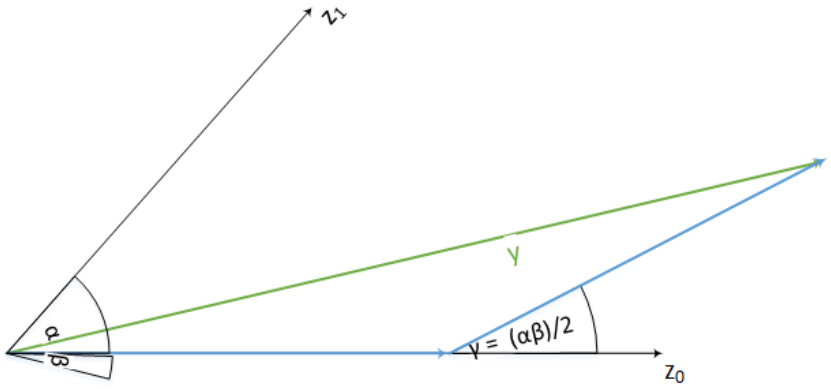


Figure 15: Illustration of the LARS algorithm for a toy example. The green vector y is the target vector. z_0 and z_1 are the sample vectors. The obtained reconstruction is depicted in blue.

An alternative formulation for this encoding is the l_0 sparse coding defined

as

$$\underset{c_i}{argmin} \ \|x_i - c_i B\|^2 \ \ s.t. \ \|c_i\|_0 \leq T, \tag{14}$$

which represents a least-squares problem [72]. With T, an upper bound for the amount of non-zero elements of the code c_i is defined. Further $\|\cdot\|_0$ means the zero-norm which counts this amount. This stays in direct contrast to the previously defined sparse coding problem formulation. As mentioned in Section 3.2.3, the OMP algorithm [63] is used to obtain a solution for this problem.

First, the current target vector y_0 is set to the actual target vector y. The sample z_0 with the highest correlation with y_0 is selected. Further, y is projected orthogonally to z_0 to obtain p. A new current target vector y_1 is obtained by $y_1 = y - z_0 p$. Next, the new sample with highest correlation is selected and these steps are repeated. Therefore OMP again is a greedy algorithm.

The K-SVD dictionary learning algorithm specialized for this problem formulation was already explained in Section 3.2.3.

In contrast to LARS, only an approximate solution rather than a global optimum is found [70]. However, OMP obtains this approximation highly efficiently in contrast to LARS where the computation can be runtime consuming. According to [65], the l_1 formulation (sparse coding) leads to more stable and smooth reconstructions. With the zero-norm formulation (l_0 sparse coding), a sparser representation, which however is not robust to noise, is obtained [65]. This means that small variations of the descriptors lead to completely different codes with different visual words selected.

An advantage of sparse coding compared to both hard and soft encoding is, that the exact restriction of the amount of visual words to be considered is abolished. Therefore, different descriptors can be represented by various amounts of visual words. However, the downside of sparse coding compared to the previous methods is that it is not guaranteed that the chosen visual words are close in the descriptor space [32]. This can happen to favor the sparsity constraint. In further consequence this can weaken the descriptor reconstruction accuracy. Nevertheless, sparse coding was already integrated in state-of-the-art image recognition approaches (e.g. in [20]). In [57], the power of this encoding method was shown even for poorly designed visual dictionaries. However, to the best of our knowledge, sparse coding has not been introduced in the sketch recognition domain so far. Therefore, we investigate the impact of both formulations in the sketch domain.

5. Locality-constrained Linear Coding (LLC) [32]
   This encoding scheme is based on the observation in [73] that sparsity follows locality but not vice versa. This means that sparse codes can represent a descriptor by visual words which are not similar. Therefore, the sparsity term in the sparse coding formation is replaced by a locality term. With that adaption, the optimization problem can be formulated as

$$\min_{c_i} \ \|x_i - c_i B\|^2 + \lambda \|dist_i \otimes c_i\|^2 . \tag{15}$$

   In this formation $\otimes$ means an element-wise multiplication and $dist_i \in \mathbb{R}^k$ is a vector containing the distances between the descriptor x_i and all visual words in B. With the additional term the selected visual words are similar to x_i. Therefore, a better reconstruction of x_i can be achieved [32]. However, the obtained LLC codes are not sparse in terms of the zero-norm. By using a thresholding operation this issue can be fixed. Further, an efficient approximation to LLC is also introduced in [32]. First, given the descriptor x_i, the l nearest-neighbor visual words $B_L = [b_1, \ldots, b_l]^T \in B$ with $l << k$ are obtained. Then the optimization problem becomes

$$\min_{c_i} \ \|x_i - c_i B_L\|^2 . \tag{16}$$

   Therefore, the optimization problem is solved in a smaller linear subspace B_L. By using a kd-tree, the nearest-neighbors can be obtained efficiently. In consequence, the code computation is also fast.

   This encoding method achieved state-of-the-art image recognition performance at the time of publication and outperformed the sparse coding method. Again to the best of our knowledge this method has not been applied to the sketch domain. Therefore, we include it into our evaluation phase.

6. Vector of Locally Aggregated Descriptors (VLAD) Encoding [74, 75]
   The main idea behind VLAD encoding is to represent the distribution of the descriptors X with respect to the visual dictionary B [74]. Therefore, for each descriptor $x_i \in X$, the corresponding nearest-neighbor visual word is obtained. Next, VLAD captures the differences between each descriptor and its obtained closest visual word for each descriptor dimension. A kd vector C is used for all descriptors and visual words. The information is aggregated for the set of descriptors X as

$$C_{i,j} = \sum_{i=1}^{n} x_{i,j} - b_{i,j}, \tag{17}$$

where b_i represents the nearest-neighbor visual word of x_i and j is the index of the descriptor dimension. Therefore, this encoding scheme further includes a pooling strategy, as a varying set of descriptors X is represented by a fixed size vector C. Additionally the VLAD representation uses an additional normalization step for vector C.

Similar to hard encoding, VLAD only considers the nearest-neighbor visual word. However, hard encoding only uses the assignment information whereas for VLAD the differences between the descriptors and visual words for each descriptor dimension are used. Compared to all previously discussed encoding schemes, VLAD includes an additional pooling strategy. Therefore no individual codes but the final sketch representation is obtained. In [74] it was shown that fewer visual words are needed when using VLAD compared to other encoding schemes to achieve a similar performance.

This representation method was successfully integrated in large-scale image search [74] as well as object retrieval [75] approaches. However, its suitability for sketch recognition has not been investigated to the best of our knowledge. Therefore, we include this combined encoding and pooling mechanism in our evaluation.

7. Fisher Vector (FV) Encoding [33, 35]
The basic idea of this encoding scheme as suggested in [33] is to fit a parametric generative model (e.g. a GMM) to the descriptors X and finally encode the derivatives of the log-likelihood of the model with respect to its parameters [34]. We use a GMM as a visual dictionary with $\phi = \{\pi_1, \mu_1, \Sigma_1, \ldots, \pi_K, \mu_K, \Sigma_K\}$, which represent the prior probabilities, means and covariances of the individual Gaussian distributions respectively. Therefore, the visual words are Gaussian distributions. The GMM is learned by the EM algorithm as previously discussed in Section 3.2.3. The covariance matrices are assumed to be diagonal as in [35]. The corresponding covariance vectors are denoted by σ_i^2. The d-dimensional (descriptor size) gradients with respect to the mean (μ_k) and the standard deviation (σ_k) of the Gaussian distribution k are given by

$$C_k^{\mu} = \frac{1}{N\sqrt{\pi_k}} \sum_{i=1}^{N} q_i(k) \left(\frac{x_i - \mu_k}{\sigma_k} \right) \tag{18}$$

and

$$C_k^{\sigma} = \frac{1}{N\sqrt{2\pi_k}} \sum_{i=1}^{N} q_i(k) \left[\frac{(x_i - \mu_k)^2}{\sigma_k^2} - 1 \right], \tag{19}$$

[35]. Further, $q_i(k)$ is the soft assignment of the descriptor x_i to this distribution. Therefore, again it is possible to assign a descriptor to multiple visual words. The assignment of descriptor x_i to the visual word k is defined as

$$q_i(k) = \frac{p(x_i|\mu_k, \Sigma_k)\pi_k}{\sum_{j=1}^{K} p(x_i|\mu_j, \Sigma_j)\pi_j}. \tag{20}$$

The final FV is created by the concatenation of the gradient vectors for all K Gaussian distributions. In that way a pooling strategy is already included. Therefore, the final sketch representation with a vector dimension of $2Kd$ is defined as

$$C = [C_1^\mu, C_1^\sigma, \ldots, C_K^\mu, C_K^\sigma]\,. \tag{21}$$

The Improved Fisher Vector (IFV) [35] further processes this FV representation in two ways. First, the Hellinger kernel is applied by performing a signed square root operation on every element of the FV representation. This comes from the observation that the higher the number Gaussian distributions in the GMM, the lower the assignment values $q_i(k)$ are. In further consequence, the FV becomes sparse which leads to poor measurements for the euclidean distance metric [35]. By applying the square root the values increase, which provide better measurements. This observation is similar to the one already discussed in context of RootSIFT [56] in Section 3.2.2. Further, the resulting vector is l_2-normalized to obtain the final representation. As this vector already forms the final sketch representation no further pooling strategy has to be applied. Due to the fact that high order statistics are encoded, less visual words are needed (as for VLAD) compared to previous encoding approaches.

This representation often achieved state-of-the-art results in image recognition approaches and is still an active research field. Again, to the best of our knowledge, this representation has not been applied to the task of sketch recognition. Therefore, we include this combined encoding/pooling strategy in its improved version in our evaluation.

The evaluation for the previously discussed encoding schemes is discussed in Section 4.7.

3.2.5 Code Pooling

Up to this point, a sketch is represented by an unordered list of codes $C = [c_1, \ldots, c_n]^T \in \mathbb{R}^{n \times k}$, where n is the amount of extracted descriptors and k is the length of one code c_i. Note that this is not true if the VLAD or FV

combined encoding/pooling method is used. In that case the information available in the set of codes is already merged to form a sketch representation. However, for encoding methods without this strategy, a pooling mechanism $f(C)$ is used to aggregate the information available in the individual codes in a defined spatial area to obtain the final sketch representation $S \in \mathbb{R}^m$. In the following we present and explain our three pooling mechanisms investigated.

1. Sum
   A simple choice for the pooling mechanism is the addition operator. For all individual codes in the matrix C, the column- or visual word-wise sum is used to obtain the final sketch representation S as

$$S_j = f(C) = \sum_{i=1}^{n} c_{i,j} \tag{22}$$

   for $j = 1, \ldots, k$. This pooling strategy is e.g. used in [15] for image recognition in combination with a spatial pyramid, which we discuss later on in this section.

2. Average
   This pooling mechanism is e.g. used in the traditional BoVW approach [8] and extends the sum pooling method by an additional normalization by the amount of codes used. More formally it is defined as

$$S_j = f(C) = \frac{1}{n} \sum_{i=1}^{n} c_{i,j} \tag{23}$$

   for $j = 1, \ldots, k$. In [5] this pooling mechanism is applied to get the final sketch representation.

3. Max
   In contrast to the previous methods, max pooling does not process all individual code values. In [20] this strategy is defined as

$$S = f(C) = [max(c_{1,1}, \ldots, c_{n,1}), \ldots, max(c_{1,k}, \ldots, c_{n,k})]. \tag{24}$$

   Therefore only the maximal code response of all codes and for each visual word is used for the final sketch representation. Yang et al. argued that max pooling is also applied in the human visual cortex and that the resulting representation is more robust to local transformations than sum or average pooling [20]. This observation was also made in [76]. However, we want to investigate if we can simply transfer this knowledge to the sketch recognition domain. Further, no previous work introduced this pooling mechanism to sketch images, although it has been successfully integrated in image recognition approaches such as e.g. [20, 32].

For traditional BoVW-based approaches the spatial area considered by the pooling mechanism is often the whole image. However, in that case the available localization information (i.e. the positions where the local descriptors have been extracted) is discarded. As already discussed in Section 2.2, a more sophisticated method is to use a pyramid-like structure – the spatial pyramid [15]. For each of the J sub-regions R_i of the pyramid, a part of the sketch representation S_i is obtained by applying the pooling method for the codes which are located in that sub-region. The final sketch representation is the concatenation of the J individual parts as $S = [S_1, \ldots, S_J]^T \in \mathbb{R}^{mJ}$. Therefore, the size of the obtained sketch representation grows by a multiplication factor of involved pyramid sub-regions. All of the previously explained pooling mechanisms can easily be integrated in the spatial pyramid framework by covering each sub-region individually. Note that although VLAD and FV encoding already produce a final sketch representation, one can create multiple of these representations for certain image parts and use them in the spatial pyramid framework with a certain pooling scheme (e.g. avg as it is used in [35]).

In the image recognition domain it was shown that the recognition performance can be improved by using a spatial pyramid [15] and it is frequently used in state-of-the-art approaches. In [5], initial experiments showed that using this framework does not increase the sketch recognition performance. However, we introduce the variant of using a spatial pyramid in our evaluation as we want to check this observation. Therefore, we compare the pooling strategies at a single level and with the spatial pyramid framework used in Section 4.8.

3.2.6 Classifier Learning

With the final sketch representations available, the final step of our algorithm is the actual sketch classification. Therefore, a suitable classifier has to be learned, as the best sketch representation is worthless without a proper classifier. Our goal is to learn a classifier F from the training data $\{(S_1, y_1), \ldots, (S_M, y_M)\}$ in a supervised manner. This means that the ground truth sketch categories $y_i \in \{1, \ldots, Cat\_N\}$ are known for each sketch and its corresponding representation S_i. There are Cat_N sketch categories to distinguish and M denotes the amount of training sketches available. We now discuss our two choices of investigated classifiers types.

1. One-Versus-All Support Vector Machines (SVMs)
   The SVM is a powerful and well studied method for binary classification. In the following we briefly discuss this binary task and explain how it can be used for our multi-class classification problem.

The main idea of a SVM is to select a separator (e.g. a hyperplane) that divides the training data $[x_1, \ldots, x_M]$ according to their class membership $[y_1, \ldots, y_M] \in \{+1, -1\}$. Further, as a constraint, the distance of the hyperplane to the nearest data points of both classes should be maximized. This distance is called the margin. In Figure 16 we present a toy example for the 2D case with a maximum margin line A and another line B with a smaller and therefore not optimal margin. The separating hyperplane is defined as $w \cdot x - b = 0$, where x is a set of points that are located on the hyperplane and w is the normal vector to this hyperplane. The offset of the vector to the coordinate origin is defined as $\frac{b}{\|w\|}$. The two support vectors are given by $w \cdot x - b = 1$ and $w \cdot x - b = -1$ respectively. As we want to maximize the margin and do not allow data points to fall into this area we can define the SVM learning problem by the following optimization formulation

$$\underset{w,b}{argmin}\ \frac{1}{2}\|w\|^2\ \ s.t.\ \forall\ i \in \{1, \ldots, M\} :\ y_i(w \cdot x_i + b) \geq 1. \tag{25}$$

Additionally, if the data points are not linearly separable, slack variables ζ_i can be introduced for each data point i to allow misclassifications. In that way a so called soft margin is learned. However, the margin between the correctly classified data samples is still maximized. The optimization problem for the soft margin case is defined as

$$\begin{aligned}\underset{w,b}{argmin}\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\zeta_i \\ s.t.\ \forall\ i \in \{1, \ldots, M\} :\ y_i(w \cdot x_i + b) \geq 1 - \zeta_i \wedge \zeta_i \geq 0,\end{aligned} \tag{26}$$

with C being a tradeoff parameter between the training error and the margin size.

By using the dual form of this problem, the SVM decision function can be obtained as

$$F(\hat{x}) = \sum_{i=1}^{M}\alpha_i\kappa(\hat{x}, x_i) + b, \tag{27}$$

where α_i are the Lagrange multipliers with $\alpha_i > 0$ only for data points which are located on one of the two support vectors. Further, $\hat{x}$ is the novel data point (or as in our case the sketch representation) which should be classified and $\kappa(\cdot, \cdot)$ means the used kernel. For linear SVMs $\kappa(\hat{x}, x_i) = \hat{x}^T x_i$.

To use this binary classifier for a multi-class problem we integrate the one-versus-all strategy. We learn one binary SVM per category to distinguish

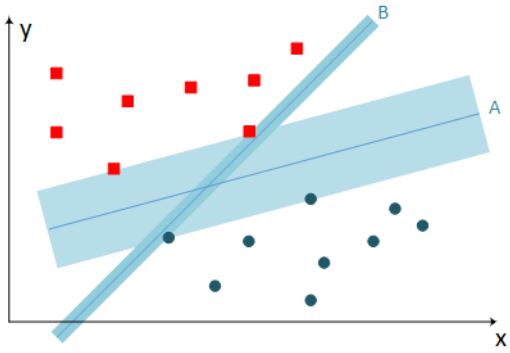


Figure 16: Illustration of the basic idea of a SVM.

it from the remaining categories. For example for the sketch category "cat", we use all of the training sketch representations with that ground truth label as the category "+1". The rest of the sketch representations is labeled with "−1". For predicting a novel sketch, we obtain the classifier responses for all SVMs learned in that way. The category with the highest "+1" response SVM is selected as the predicted label. Using linear SVMs leads to a training complexity of $\mathcal{O}(Cat\_NM)$ and is therefore linear with the amount of training data. Further, linear SVMs have a constant classification complexity of $\mathcal{O}(1)$. A drawback of this classifier method is that not all data is linearly separable, therefore the non-linear SVM was introduced.

The intention behind non-linear or kernel-based SVMs is that although the data is not linearly separable, one can use the hyperplane as discussed above to divide the data with the so called kernel trick. The idea is to transfer the data to a different feature space where the separation using a hyperplane is possible. Any kernel function can be used but we restrict ourselves to the Gaussian radial basic function defined as $\kappa(x_i, x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2)$. To use kernel-based SVMs for multiclass problems, we again apply the one-versus-all strategy discussed above. Therefore, the drawback of linear SVMs is eliminated because of the kernel trick. In general, a smaller classification error can be achieved with kernel based SVMs. However, the training complexity is increased to $\mathcal{O}(M^2 \sim M^3)$. Further, also the testing complexity raises to $\mathcal{O}(M)$. Therefore, the computations of a non-linear SVM are more expensive than for a linear SVM.

As both variants were already successfully used in both the image and

sketch recognition domain, we include these two SVM types in our evaluation phase.

2. Random Forest (RF) [77]
   A classifier which is capable of handling multi-class problems directly is the RF [77]. The RF is an ensemble of decision trees, where a tree consists of split- and leaf-nodes. The split-nodes are used to split the data in two parts according to a certain criteria – the split function f_{split}. Leaf-nodes contain class-specific information which is discussed later in this paragraph. In further consequence, each tree forms a hierarchical structure. n_{trees} trees are built for the RF using a random subset of the training data for each tree. This strategy is denoted as bagging. At each split a certain set of split functions $f_{split,i}$ is used to divide the data. The split result which leads to the highest information gain is kept for further tree creation. When a specified tree depth is reached or no further split is possible, the tree creation process is finished. Each obtained leaf node stores a histogram with the amount of training samples that reached this node for every category available. In such a way the histogram represents probabilities that a sample reaching this leaf node is of a specific category. A novel data sample is passed down each tree of the forest. This means that depending on the previously obtained split functions, a leaf-node is reached for each tree. To predict the category of the data sample, the probabilities of the reached leaf nodes are summed up and averaged to obtain the final probabilities. The class with the highest probability is defined as the predicted category.

   An advantage over the SVM-based classifiers previously discussed is that only one classifier has to be learned and evaluated – although it consists of multiple trees. As for linear SVMs, the training procedure can be performed efficiently with an upper bound complexity of $\mathcal{O}(n_{trees}(mM \log M))$ where m is the sketch representation dimensionality. Further, in general the classification accuracy is comparable to SVMs-based classifiers and RFs are known to have good generalization properties [77]. RFs were already integrated in successful image detection (e.g. [78]) and image recognition approaches (e.g. [19]). Therefore, we include this classifier type in our evaluation.

The adaptability of the discussed classifier types in the domain of sketch recognition is discussed in Section 4.9.

In the following section we perform our extensive pipeline evaluation for the

obtained list of element variants and additionally finalize our sketch recognition algorithm.

4 Pipeline Evaluation and Final Algorithm

This section covers our extensive evaluations performed in order to obtain our optimized sketch recognition algorithm. As already mentioned in Section 3 we want to design an algorithm according to two aspects. First, the pipeline should achieve a high recognition accuracy. However, at the same time the algorithm should be efficient. The second constraint is important for real-time applications which e.g. are used in Augmented Reality (AR) systems (see Section 5.7 for such a demo application). In such applications it is necessary to provide immediate feedback to the user request, e.g. for the drawn sketch. Additionally, we also want our learning procedure to be fast. As this is the offline part of the approach, fast means that the visual dictionary as well as the classifier should be learned within a couple of hours. In further consequence, we relax the first constraint in order to favor more efficient methods. However, we still select variants of a high recognition quality. Since we figure out that both factors interfere with each other, we finally propose two algorithm variants. The first one is designed to achieve the best recognition performance at a reasonable runtime. Further, a more efficient but less accurate approach is introduced.

First of all we discuss the sketch dataset used in all of our pipeline element evaluations as well as final experiments in Section 4.1. Next, we discuss implementation issues in Section 4.2. Further, we explain the strategy used for the evaluation phase and define our evaluation baseline pipeline in Section 4.3. The actual evaluations of the individual pipeline elements are covered in detail in Sections 4.4-4.9. The evaluation regarding sketch image pre-processing is discussed in Section 4.4. All evaluations in the context of descriptor extraction (sampling strategy/descriptor type) are explained in Section 4.5. The dictionary learning method is investigated in Section 4.6. The evaluation of the encoding step can be found in Section 4.7. The pooling mechanism and its corresponding evaluation phase is covered in Section 4.8. The final pipeline element, the classifier, with its learning algorithm is investigated in Section 4.9. A conclusion of the evaluation process is provided in Section 4.10. Our two final algorithms with their element choices are declared and explained in Section 4.11.

4.1 Dataset

In this thesis we use the TU Berlin sketch dataset introduced in [5]. To the best of our knowledge, this is the only large-scale sketch dataset publicly available. In this context, large-scale means that both a large amount

of sketch categories as well as a large number of sketches per category are present. The dataset contains 20 000 human sketches of 250 object categories. For each category an equal amount of 80 sketches exists. A large range of everyday life objects is covered by the dataset according to [5]. Note however that humans are able to distinguish among 30 000 visual categories and that an adult English speaker uses 20 000 words [79]. According to [5], the 250 categories were defined in the following way: first the 1 000 most common labels from LabelMe [80] were obtained and duplicates were removed. Additionally, categories were sorted out that do not follow one of the two following criteria: first, the categories should be recognizable alone by their shape without additional context. As a second criteria each category should be specific enough that relatively few visual representations are available. The category "building" e.g. would not be specific enough. Note however that also categories were included in the dataset which, in our opinion, do not follow these two criteria as e.g. the categories "seagull", "pigeon" or "standing bird" as seen in Figure 8. This inter-class variability problem was already discussed in Section 1.2. However, the category set which satisfied these criteria was extended with categories of the Princeton Shape Benchmark [81] as well as the Caltech-256 dataset [7]. Finally, additional categories were introduced based on suggestions obtained from the members of the scientific group of the authors of [5].

The sketch images were obtained by using Amazon Mechanical Turk (AMT) by 1 350 ordinary people. This means that no artists were involved in the sketch creation process. For the created sketches both the problems of intra- and inter-class variability as well as the viewpoint issue exist (see Section 1.2 for the discussion of these problems). However, the following rules were set up for drawing the sketches: only outlines of the object without additional context should be drawn. It was also not allowed to add text or large black areas to the sketches. As a final rule the authors of [5] defined that the category of the sketch should be easily recognizable. The rules were also graphically depicted to the drawers as shown in Figure 17. The obtained sketches were manually examined according to those rules. Finally, the remaining sketches were truncated to get exactly 80 sketches for each category. Eitz et al. claimed that this amount is enough to capture most of the variance within a sketch category [5]. This is also shown in experiments in [5] as the performance gain gets smaller as more data are used in the training phase.

The provided dataset is organized in 10 folds. So one can e.g. train with increasingly training set sizes of 8, 16, etc. sketches per category. Each sketch image in this dataset is available as a grayscale image probably due to anti-aliasing methods used during the creation process. Further, each sketch has a constant size of $1\,111 \times 1\,111$ pixels and is normalized in such a way that

the longer side of the bounding box of the sketch has a uniform size and is centered in the image. This mechanism solves the problem of non-uniform scale and position of the sketches as already discussed in Section 1.2.

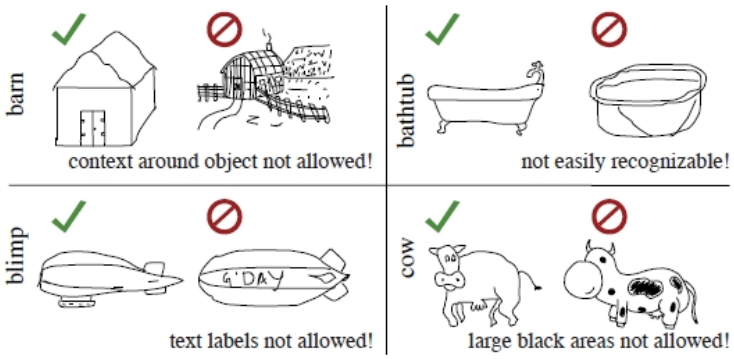


Figure 17: The graphical representation of rules for creating the sketches of the TU Berlin sketch dataset (taken from [5]). This hint was shown to the drawers who participated in the AMT creation task.

4.2 Implementation

The implementations of the evaluation framework as well as the final experiment framework, discussed later on in Section 5, were done in Matlab R2012b. As it would have been infeasible to implement all evaluated methods on our own, we integrated external libraries and source codes. We used the open source VLFeat toolbox [82] which contained most of the functionality needed for the frameworks. However, we also integrated the publicly available code of [83] for the Self Similarity descriptor. For computing the Shape Context descriptor we used provided code of [51]. The implementation of sparse coding as well as of the Online Dictionary Learning for Sparse Coding (ODL) algorithm were integrated using the SPArse Modeling Software (SPAMS) toolbox [65]. For the K-SVD algorithm, source code of [64] was used. The available Locality-constrained Linear Coding (LLC) source code provided by [32] was further included in the implementations. Soft encoding was computed with source code taken from [84]. For the classifier type evaluation we utilized the LIBLINEAR [85] and the LIBSVM toolbox [86] for both Support Vector Machine (SVM) types (linear and kernel based). Further, a Matlab Random Forest (RF) implementation of [87] was used. For the evaluation of the Local Histograms of Oriented Gradients (SHOG) descriptor we used our own implementation. All evaluations were performed

on a desktop computer with a 3.2Ghz Quad Core CPU and a memory of 8GB.

4.3 Evaluation Strategy and Baseline Pipeline

In this section we first of all discuss strategic considerations for obtaining an optimized recognition algorithm according to the two constraints already defined. As a reminder we refer to our general recognition pipeline in Figure 12 and to Section 3.1 for an explanation of the pipeline. Then, we define our evaluation baseline pipeline which is used for our evaluations.

Our strategy is to investigate each pipeline element successively in an independent manner. For example, we consider adapting the sketch image pre-processing and try to optimize it according to our two criteria. In other words, we evaluate the variants (sketch formats) introduced in Section 3.2.1 (for the pre-processing step) and choose the most suitable one. Once the evaluation is finished, we declare this variant to be used in the image pre-processing element of the final recognition algorithm. Further, we update the current evaluation pipeline with this element and begin our evaluation for the next pipeline step (descriptor extraction). Therefore, we systematically exchange the pipeline element variants according to the obtained evaluation results. Further note that all of the pipeline elements are based on each other. Therefore, the evaluation results of one step are directly integrated in all remaining pipeline elements. This strategy is performed until the evaluation of the last pipeline element, the sketch classifier, is finished. We apply this evaluation strategy, as a complete one (i.e. investigate all variants of all steps with all variants of the remaining steps) is infeasible.

As mentioned before we use the TU Berlin sketch dataset outlined in Section 4.1. Further, we decide to use the first three (of 10) folds of the dataset. Two folds are utilized for the training procedure and the recognition performance is analyzed using the sketches of the remaining fold. Therefore we have 4 000 sketches for training and 2 000 sketches for testing from all 250 categories. We perform 3-fold cross-validation and repeat each iteration per fold five times to suppress the impact of randomness which can occur for individual pipeline elements (e.g. in the original k-means algorithm or the RF learning algorithm). In contrast to [5], we do not fix our visual dictionary learned from descriptors of the whole dataset. Instead, we only use descriptors from the current training set and therefore learn a dictionary for each iteration. As a result of that, we also have to calculate the codes and in further consequence the sketch representations for each of the 15 iterations (three folds $\times$

five repetitions). If not specified otherwise, we use this data setup in all of our evaluations in this section.

We evaluate the actual performance of an individual setup, in a classification scenario. Thus, given previously unseen test sketches, we infer the corresponding category using our pipeline and simply count the number of correctly recognized test samples. Therefore, the recognition score/performance is the percentage of correctly classified test sketches. More formally we define it as

$$score = \frac{|\{sketch_i : class_{pred}(sketch_i) = class(sketch_i), i \in testset\_i\}|}{|testset\_i|}, \tag{28}$$

where $class_{pred}$ is the predicted category and $class$ is the ground truth sketch label. Further, $testset\_i$ contains the indices of the test sketches. All provided scores in the subsequent evaluation sections are average values over all three folds and respective iterations (overall $3 \times 5 = 15$ scores).

Evaluation Baseline Pipeline
As a pre-processing step the sketch images are scaled to a fixed size of 256×256 pixels. As we use the TU Berlin sketch dataset, we do not have to scale and translate the sketch content as discussed in Section 4.1. After that, the sketches are converted into the $binary - thin$ format. We further smooth the images using a default Gaussian kernel. In the next step we extract Scale-Invariant Feature Transform (SIFT) descriptors on a dense grid every four pixels using a patch size of 52. Descriptors which contain no information are rejected. From the remaining descriptors we select a maximum of 1 000 for further processing per sketch and apply l_2 normalization. Further, we randomly sample 100 000 descriptors from sketches of the training set to learn our k-means dictionary consisting of 500 visual words. We use the k-means++ initialization method and an accelerated algorithm introduced in [88]. In further consequence, the visual words are also l_2-normalized. To encode our 1 000 descriptors per sketch we use the (exact) hard/histogram encoding method. Sum pooling with successive l_2-normalization is used to obtain the final sketch representations. As a classifier we define a Random Forest consisting of 500 trees, which uses 10 randomly sampled variables for each split.

We defined our evaluation baseline pipeline after initial experiments. Therefore, pipeline element variants were chosen which showed a good performance in our sketch recognition framework. A summarized illustration of the evaluation baseline pipeline is shown in Figure 18.

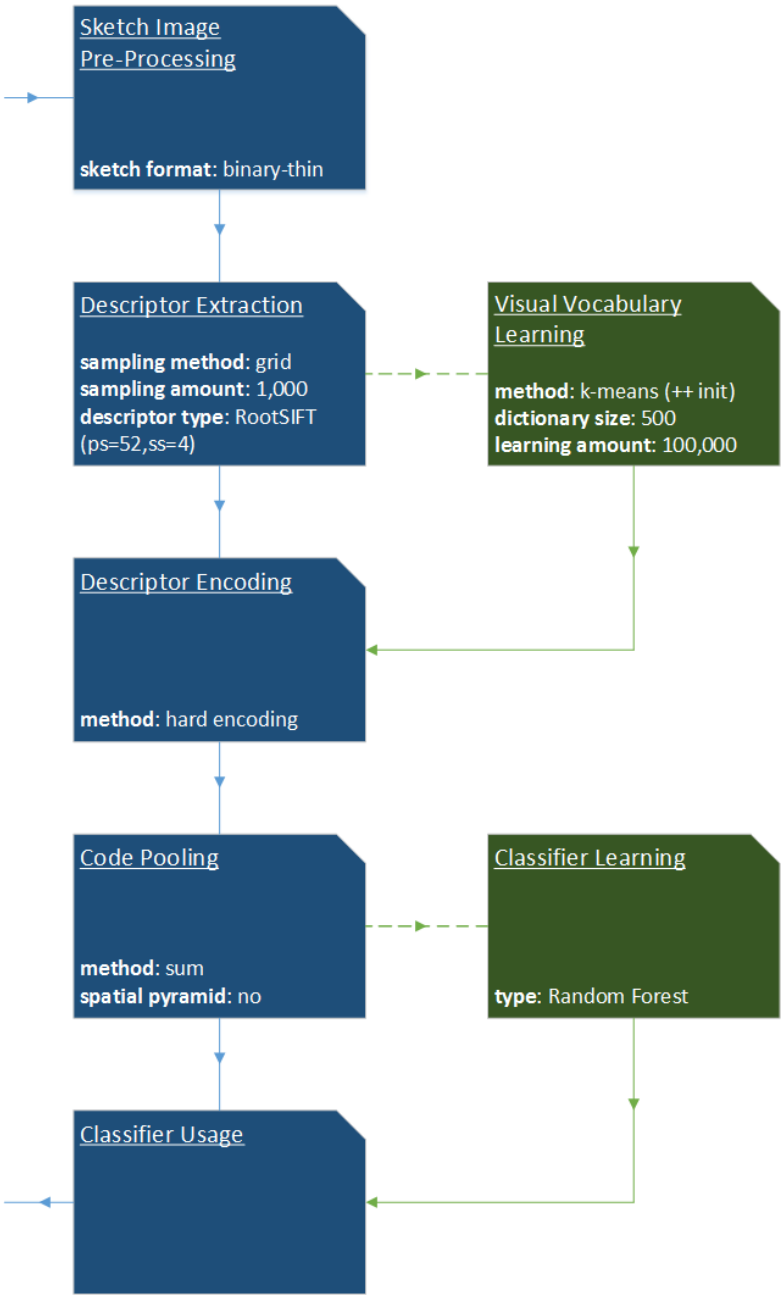


Figure 18: Graphical representation of our evaluation baseline pipeline. Elements with a green color are only involved in the training procedure. The outputs of these elements (marked with a green solid line with arrows) are further used in the classification phase. Information about each pipeline element is given in the corresponding rectangle.

4.4 Image Pre-Processing

As we already fixed all but the sketch format for this pipeline step, as discussed in Section 3.2.1, the goal of this evaluation is to obtain the most suitable sketch format. We compare the three sketch formats *grayscale*, $binary - thick$ and $binary - thin$. In Section 3.2.1 we already defined how these formats are obtained, and provided a visual example of the three sketch formats in Figure 13. For this experiment we use the evaluation baseline pipeline as defined in Section 4.3 and only modify the sketch format used in the image pre-processing step.

The impact of the different sketch formats is shown in the boxplot visualizations in Figure 19. First of all, we notice that for each format, the recognition performance varies by approximately $\pm 3.5\%$ over all 15 runs due to the involved randomness. Remember that the descriptors used per sketch as well as for the dictionary learning step are chosen at random. Further, the Random Forest is learned in a non-deterministic way. Therefore, we argue that such variations are reasonable. We also observe that the average scores of all three formats differ by about 1% ($binary-thin$: 31.56%, $binary-thick$: 32.16% and *grayscale*: 32.53%). We argue that one would also expect this small range, because the *grayscale* format basically only provides additional magnitude information when the gradient is calculated (e.g. for SIFT) in contrast to the $binary - thick$ format. Although this additional information can be useful in the domain of natural images, for sketch images this information does not lead to much benefit due to the lack of noise and background. For example Eitz et al. [5] and Chao et al. [50] completely ignored the gradient magnitude when computing their SIFT-based descriptors for the corresponding sketch recognition/retrieval approaches. We further argue that by using the morphological operation to thin out the sketch images, information is lost for the $binary - thin$ format. As the diversity issue is a major problem of sketch recognition this additional information can be useful.

Hence, in the remaining evaluations we stick to use the *grayscale* format, as the additional gradient information includes useful information for the descriptor calculation (when a gradient-based method is used), although the impact is only minor.

4.5 Descriptor Extraction

In this section we investigate two important aspects regarding the descriptor extraction step. First, the sampling strategy is discussed in Section 4.5.1. Further, the different descriptor types are evaluated in Section 4.5.2.

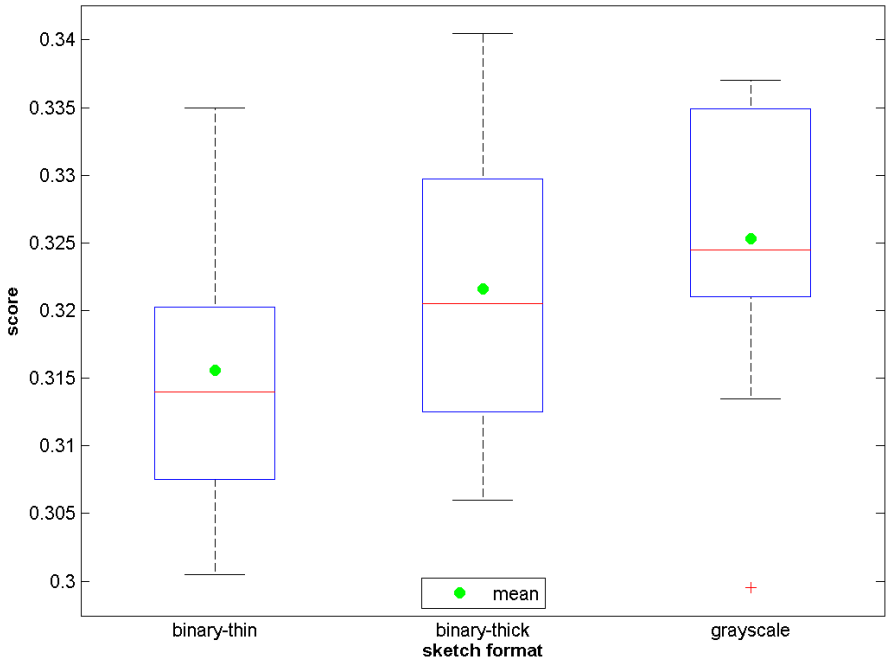


Figure 19: Evaluation of the three sketch formats: (a) grayscale, (b) binary-thick and (c) binary-thin. The boxplot visualization of the 15 scores for each format is shown together with the corresponding means scores (green markers).

4.5.1 Sampling Strategy

Two topics are covered in this part of our evaluations. First, we investigate the question of which sampling method is more suitable for sketch recognition. Therefore, we compare the *grid* and the *point* sampling method as defined in Section 3.2.2. The second aspect of this topic is to identify an optimal number of points to sample per sketch. Obviously, this is an important factor regarding the overall runtime. The fewer points are sampled, the lower is the computational effort. On the other hand, a smaller number might result in a less accurate sketch representation.

Sampling Method
We first test the two sampling methods (*grid* and *point*). When using *grid* sampling with our default parameters (a patch size of 52 and a step size of 4), 3 025 SIFT descriptors are extracted for each sketch image with a size of 256×256 pixels. Further, as empty descriptors are rejected, an average descriptor amount of roughly 2 600 per sketch is obtained in this evaluation. For *point* sampling, a patch is extracted for every pixel of the sketch lines.

This average pixel amount in this evaluation is approximately 2 800, which further represents the amount of extracted descriptors. Note however that for this evaluation, as discussed in the evaluation baseline pipeline (Section 4.3), a descriptor amount of 1 000 is selected at random for each sketch.

The results of this evaluation can be seen in Figure 20. We clearly see that the *grid* sampling strategy outperforms *point* sampling. The performance with *point* sampling is on average 5.5% worse (26.99%) than with *grid* sampling (32.53%).

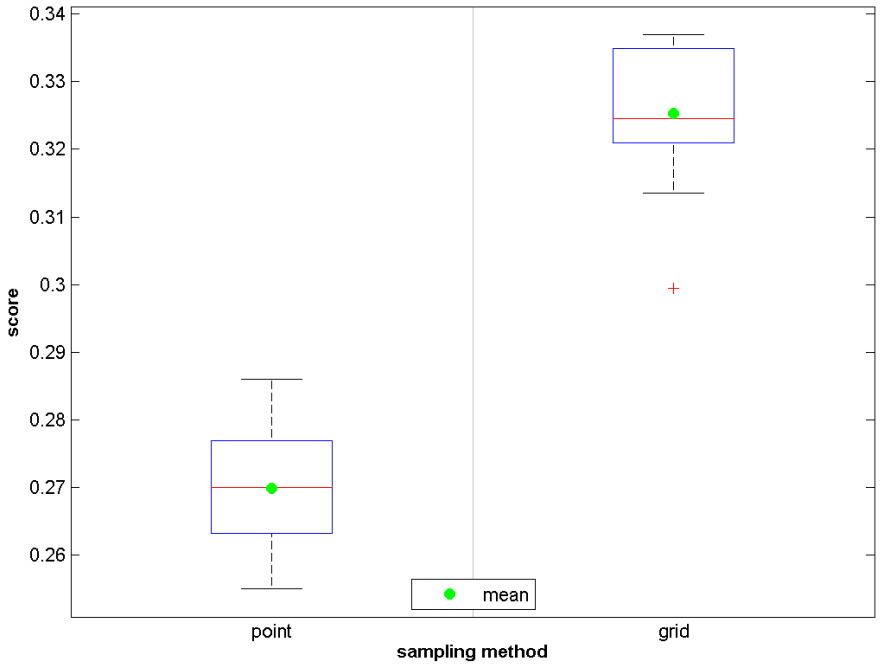


Figure 20: Evaluation of the two sampling methods: (a) grid and (b) point. The boxplot visualization of the 15 iterations for each method is shown together with the corresponding means scores.

We perform a second evaluation regarding *point* sampling and adapt the amount of descriptors used. This comes from the fact that both sampling strategies lead to a different amount of extracted descriptors. Further, we want to debilitate the argument that the bad performance of the *point* sampling iterations is a result of the fixed descriptor amount. Indeed, the performance of the *point* sampling method increases with additional descriptors. However, the results are not competitive to the performance for *grid* sampling. We argue that *grid* sampling outperforms *point* sampling because with *grid* sampling the extracted patches are not constrained. This means,

that with *point* sampling the patch center pixel always contains a sketch line, whereas this is not the case when *grid* sampling is used. So in general we get more diverse, but fewer dense patches with *grid* sampling than with *point* sampling. However, we further argue that variability is more important than density as variability helps us when facing the challenges of sketch recognition as discussed in Section 1.2. We therefore conclude that *grid* sampling is more suitable for our task than *point* sampling. Note however, that in [50], *point* sampling is used for sketch-based sketch retrieval. As a consequence we stick to the *grid* sampling method in the remaining evaluations.

Sampling Amount

In this evaluation we consider the amount of descriptors used per sketch for *grid* sampling. The boxplot visualization of the results can be found in Figure 21. We observe that a higher descriptor amount leads on average to better scores as already investigated for *point* sampling. The highest score of 35.67% is achieved when all available descriptors are used. Compared to our baseline with 1 000 descriptors, the average performance increases by roughly 3%. This is an interesting outcome because the amount of descriptors per sketch is highly diverse (the range lies between 1 400 and 3 000). This comes from the fact that although all sketches are centered and scaled to a fixed longest bounding box length, the amount of empty patches and therefore descriptors varies. Two example sketches with such varying descriptor amounts can be found in Figure 22. As a result of that observation, we expected that by using all descriptors and therefore a varying amount for each sketch image, normalization issues would occur (e.g. categories with a higher amount would be favored). However, our evaluation contradicts this theory. We argue that the more descriptors are used, the more sketch diversity is covered which results in better overall recognition results. Hence, we use all available descriptors for each sketch in the remaining evaluations. This comes with a higher but still reasonable computational effort.

Conclusion and Findings

In the previous evaluations we investigated issues regarding the sampling strategy. We compared two sampling methods and observed that *grid* sampling outperforms *point* sampling. This is even true if more descriptors are used for *point* sampling. Therefore, we defined *grid* sampling as our method of choice. Further, we investigated the impact of the amount of descriptors used per sketch image. Finally, the observation was made that despite normalization issues (a different amount of descriptors per sketch), using all available descriptors with *grid* sampling is the most suitable sampling strategy for sketch recognition.

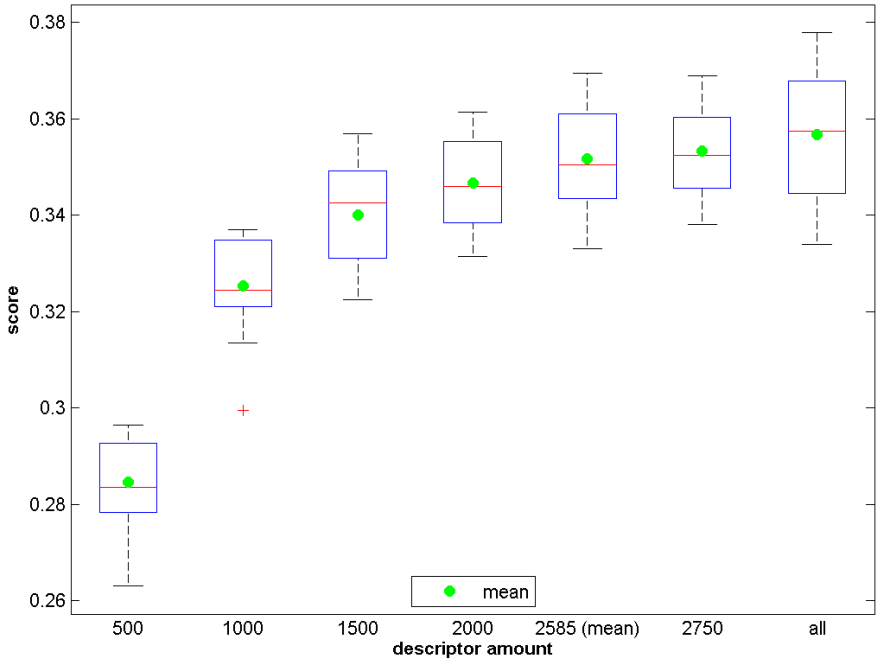
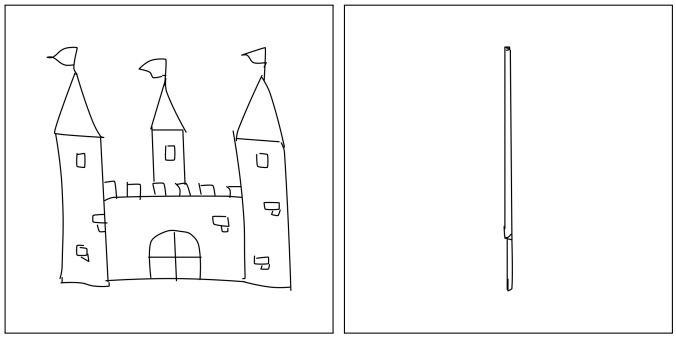


Figure 21: Evaluation of the descriptor amount used per sketch when grid sampling is applied. The boxplot visualization of the 15 iterations for each amount is shown together with the corresponding means scores.



(a) high ("castle") (b) low ("baseball bat")

Figure 22: Two example sketches which result in a high and a low amount of extracted descriptors when grid sampling is used.

4.5.2 Descriptor Type

In this section we want to determine the descriptor type which leads to the most accurate recognition results. Different descriptor types (local, global, gradient-based, shape-based etc.) are investigated in that context. Further, the insights are used to optimize parameters for the selected feature representation, Dense SIFT, in an extensive evaluation. Finally, we investigate if the RootSIFT strategy suggested in [56] does increase the recognition performance.

We evaluate the performance of five descriptor types: Dense SIFT [53], Histogram of Oriented Gradients (HOG) [52], Local Histograms of Oriented Gradients (SHOG) [5], Shape Context [51] and Self Similarity [55]. We already explained each descriptor type in detail in Section 3.2.2.

For the Shape Context as well as the Self Similarity descriptor we do not perform image smoothing as a pre-processing step as we argue that no gradient calculation is performed for these methods. Further, we choose this setting because in our opinion this is a more natural way of using these two descriptors. For Dense SIFT, we fix the patch size to be 52 and set the step size to four as defined in our evaluation baseline pipeline in Section 4.3. For SHOG, we use our own implementation and use the parameter values as discussed in [5]. Therefore a patch size of 90 and a step size of six are applied to compute the SHOG descriptors. In the context of the Self Similarity descriptor, a patch size of 52 means that the reference 5×5 patch is compared to patches inside a circular region with a radius of 26. We use default values for the remaining parameters as defined in the implementation of [83]. For the Shape Context descriptor we again only adapt the parameters in such a way that a circular region with a radius of 26 is considered. Further, we use this descriptor type with *point* sampling as discussed in [51].

The results of this evaluation can be found in Figure 23. First of all, we see that the Dense SIFT patch representation outperforms the remaining descriptor types with an average score of 35.67%. Although SHOG is designed to be used with sketches the performance is roughly 4% worse with a score of 31.95%. All remaining descriptor types are not competitive at all with an average score which is at least 16% worse than for Dense SIFT descriptors. This observation is also supported by the findings in [54, 46] which also perform a similar evaluation.

However, we like to point out the noticeable performance of the global HOG descriptor with a mean score of 17.23%. This is interesting as only one descriptor is used per sketch image. It is obvious that such a global representation does not lead to superior results for sketch recognition due to the

diversity problems. Nevertheless, its average performance is almost similar to the evaluation which uses the Shape Context descriptor (19.03%). Further, it is more successful than the Self Similarity descriptor (15.15%). We argue that this already points out the strength of gradient-based descriptors for sketch recognition. Even a global gradient-based representation is competitive to a local shape-based method. The Shape Context descriptor which is specialized for describing shapes, such as sketches usually are, performs worse than we expected. We argue that a reason for this may be the large diversity of sketches. However, gradient-based methods can handle this problem in a convenient way. Note that this statement is also true for the image domain as gradient-based methods are usually the local or global descriptor method of choice. Additionally, we perform an evaluation for the SHOG descriptor using parameter values as for Dense SIFT ($patch\ size = 52,\ step\ size = 4$). A better recognition performance is achieved for SHOG with this parameter setting which is however significantly lower than with Dense SIFT. Therefore, we argue that the SHOG descriptor is a suitable but not optimal way to describe local sketch image patches. Further, we argue that the additional gradient-magnitude information comprised in the SIFT descriptor does help to increase the recognition accuracy. This is also supported by Li et al. [46] which also used SIFT descriptors.
Therefore, we select the local gradient-based Dense SIFT representation for our final algorithm as also the runtime needed is the lowest compared to the remaining methods in this evaluation. Hence, we use the Dense SIFT descriptor for all subsequent evaluations.

We further investigate the image smoothing parameter σ in combination with Dense SIFT descriptors. After the evaluation we fix this value to be 2.0 for all remaining evaluations.

Dense SIFT Parameter Optimization
Next, we want to optimize the recognition performance by adapting the parameters patch and step size of Dense SIFT. The amount of overlap depends on the step size. For step size values smaller than the patch size an overlap occurs. We perform evaluations regarding non-overlapping and overlapping patches.

Due to performance reasons we change our evaluation setup for these two evaluations. We use two (instead of three) folds from the sketch dataset and do not perform cross-validation. Note that we come back to the performance topic once we explain the set of investigated parameter pairs for the overlapping case. We use one fold for the training procedure and the second fold to obtain the recognition performance. Another modification to the current evaluation pipeline is, that instead of (exact) hard encoding, the approxi-

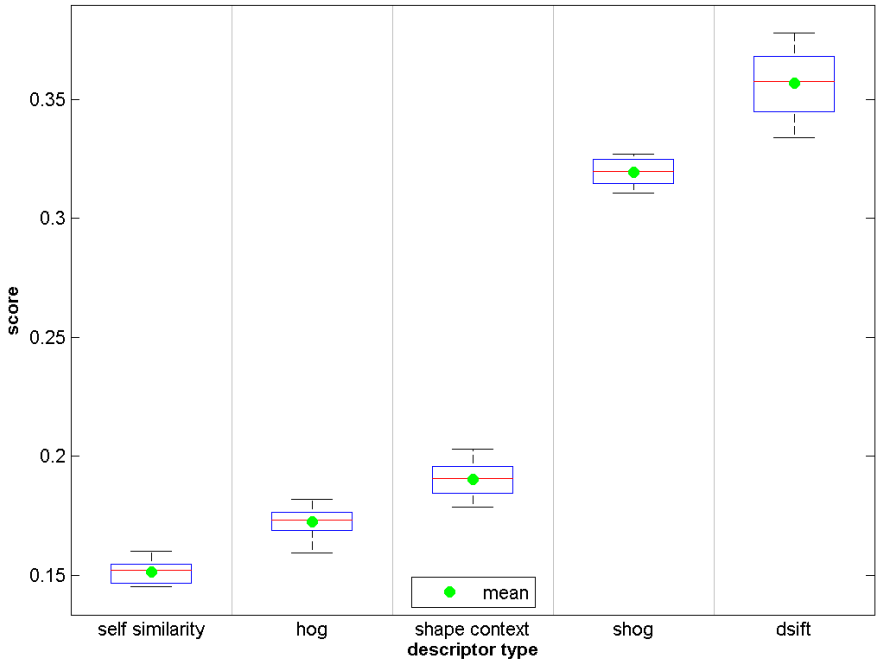


Figure 23: Evaluation regarding the descriptor type. The boxplot visualizations show the recognition performances of five different descriptors (Dense SIFT, HOG, SHOG, Shape Context and Self Similarity). Additionally, mean scores are provided.

mated version, using a kd-tree, is used for faster descriptor assignment. Note however, that we repeat this experiment with our proper evaluation setup and pipeline for a small set of best performing parameter pairs to get representative scores later on in this section.

In the first evaluation we only consider non-overlapping patches (*step size* = *patch size*). This means that the patches are directly concatenated to each other. However, we obtain poor recognition scores with this setting for all investigated patch size values with a best performance of 9.30%. Therefore, we already recognize that overlapping patches are necessary.

The improvements, if we allowed patches to overlap each other, are further analyzed. As mentioned before an overlap occurs if the step size is smaller than the patch size. As patch sizes we use the values 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 52, 64, 72, 80, 88, 96, 104, 112, 120 and 128. We investigate step sizes of 2, 4, 8, 16, 32 and 64. Overall, this would lead to 120 parameter pairs. However, this number is reduced because of the overlap constraint for

the step size. Therefore, 93 combinations are considered. Due to this large number, we adapt the evaluation setup as discussed before such that this evaluation becomes feasible.

The results of this evaluation can be found in Figure 24. We realize that parameter pairs with mid-level sized patches and a high overlap produce good scores. Note that the smaller the step size, the higher the overlap and resulting amount of extracted descriptors is. The parameter pair (40/2 (patch size/step size)) achieves a score of 27.65%. All of the top five pairs, which achieve a similar performance, have a patch size of 40, 52 or 64 and a step size of two or four. We further recognize that the pairs with slightly lower or higher patch sizes and a step size of two also achieve satisfying scores. The worst results are obtained with a low overlap or with small patches in general.

These observations substantiate our initial assumption that mid-level patch sizes with a big overlap and therefore a high amount of descriptors are essential for good recognition scores. We further observe that the overlapping parameter pairs clearly outperform the non-overlapping scores by approximately 18%. Therefore, we decide to use overlapping patches.
The scores from the two previous evaluations were obtained by using a modified version of our evaluation setup. Therefore, we use the four best performing parameter pairs from the overlapping evaluation and repeat the experiment with our proper evaluation setup as discussed in Section 4.3. In Figure 25 the obtained scores are shown. We notice that all parameter pairs investigated have a high quality and achieve valuable average scores. This shows that our previous overlap-evaluation leads to useful results although the evaluation setup was limited due to performance reasons. The best mean score is achieved by a patch size of 52 and a step size of two with 36.55%. The subsequent configurations obtain comparable results. However, the computational effort for the combination 52/2 is high. Therefore, we decide to use the efficient 64/4 parameter pair as a similar performance is achieved at $\frac{1}{4}$ of the runtime needed. We introduce this Dense SIFT parameter pair in our final recognition algorithm and in the current evaluation pipeline.

RootSIFT

We further investigate the impact of using the so called RootSIFT strategy [56] as already discussed in Section 3.2.2. The calculation of the RootSIFT strategy given the SIFT descriptor comes with low cost as the additional operations can be performed highly optimized. Therefore, in a last evaluation regarding the descriptors we evaluate the usage of this strategy in our pipeline. The statement in [56] is confirmed as the average score of 36.46% is improved by approximately 1% to 37.31% by using the RootSIFT variant.

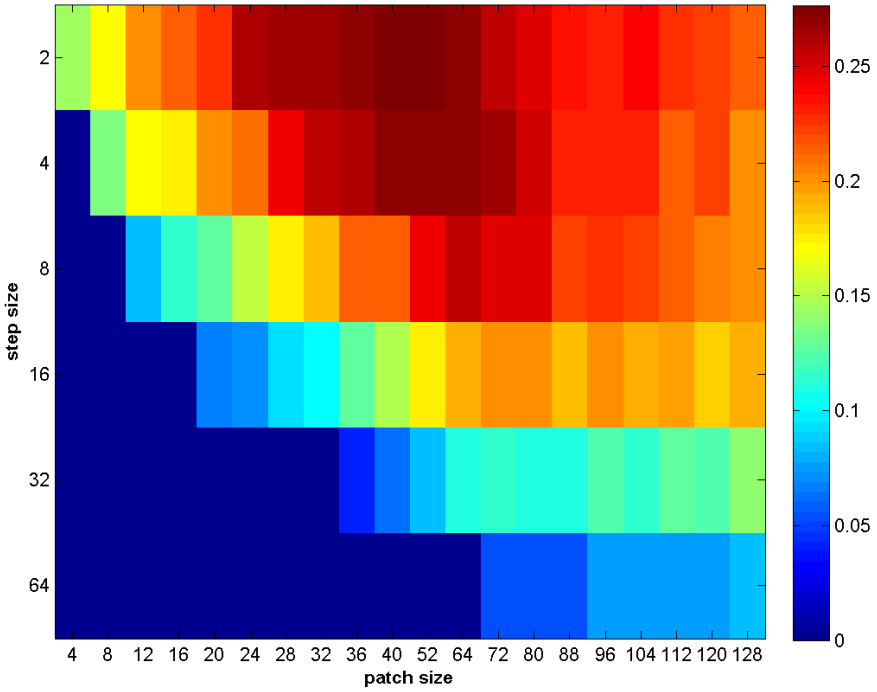


Figure 24: Dense SIFT parameter evaluation for overlapping patches. An overlap only occurs if the step size is smaller than the patch size. Therefore parameter pairs in the lower left corner are not considered. An adapted evaluation setup is used due to performance reasons and the single scores are illustrated as a heatmap.

Therefore and due to the fact that the additional steps can be performed highly efficiently we incorporate this descriptor post-processing step in our final recognition algorithm.

Discussion and Findings

In the previous evaluations regarding the descriptor type we first of all learned that the local gradient-based SIFT descriptor clearly outperforms the other investigated local as well as global methods. Next, we performed two extensive evaluations in order to get optimal parameter pairs for Dense SIFT. We investigated non-overlapping and overlapping patches in this context and learned that a high overlap and a mid-level patch size are essential to achieve a high recognition accuracy. We obtained a patch size of 52 and a step size of 2 as the optimal parameter pair. Nevertheless, due to runtime issues we took the similar performing pair 64/4. The last evaluation result showed that by using the RootSIFT strategy, a performance gain could also be achieved in

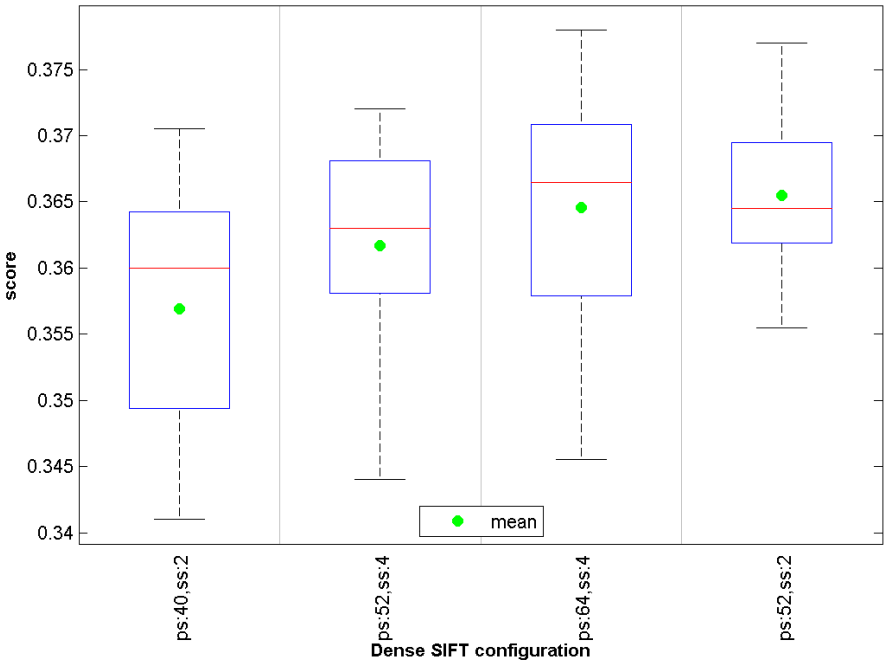


Figure 25: Evaluation result for the four best performing overlapping parameter pairs from the previous evaluation with the proper evaluation setup used. The boxplot visualizations and mean scores are illustrated. A Dense SIFT parameter pair is definded by the patch size (ps) and the step size (ss).

the sketch recognition domain.

4.6 Visual Dictionary Learning

In this section we investigate two main issues regarding dictionary learning for our sketch recognition algorithm. First, a suitable dictionary learning procedure is required to build a solid basis for the rest of Bag-of-Visual-Words (BoVW) pipeline. Then, the impact of using various amounts of visual words is discussed.

From our previous evaluation step we know that most elements in the current evaluation pipeline are efficient. However, we observed that the encoding method used (hard encoding), is computationally expensive as an exact nearest-neighbor search is required. As used in the Dense SIFT parameter experiment (Section 4.5.2), approximated hard encoding makes this step efficient.

Dictionary Learning

As discussed in Section 3.2.3, we compare six dictionary learning methods. Note that we use the associated sparse coding formulation for the evaluation of the K-SVD and the ODL algorithm. Hence, the encoding method, for which the dictionary is learned and optimized for, is incorporated. Default parameter values for both learning algorithms as suggested in [65] are applied. For the probabilistic dictionary consisting of a Gaussian Mixture Model (GMM), we included the improved version of Fisher Vector (FV) encoding (IFV) as discussed in Section 3.2.4. For the rest of the dictionary learning methods we use approximated hard encoding. As already discussed in Section 4.3, for k-means an accelerated algorithm, as introduced in [88], is used. Due to high sketch representation dimension obtained for IFV encoding and the resulting high computational runtime, we only perform one iteration per fold (instead of five). Therefore, in that case the average score of only three iterations is provided.

The evaluation results are illustrated in Figure 26. First of all, we take a look at the three k-means variants. Exact k-means with the k-means++ initialization (in the following k-means++) leads to the highest score of 36.80% within these three methods. However, when using the original initialization method or the more efficient approximated version, similar performances are achieved. When using a random dictionary the performance drops by about 1%. This stays in contrast to [57] in which it is claimed that a random dictionary can be used with similar performance. However, in [57] sparse coding is used in the subsequent encoding step. Next, we see that a GMM dictionary with IFV encoding does not seem to work as well as for image recognition as discussed in Section 3.2.4. To investigate if the amount of Gaussian distributions is sufficient, we doubled the amount and repeated the evaluation. However, the recognition performance remained significantly lower than for k-means++ and is even worse than for the random dictionary. Finally, we compare the two sparse coding-optimized dictionary learning techniques (K-SVD and ODL). Surprisingly, a large performance gap can be seen. We argue that this comes from the different algorithms used to obtain the sparse codes, Orthogonal Matching Pursuit (OMP) for K-SVD and Least Angle Regression (LARS) for ODL. As explained in Section 3.2.4 with LARS it is guaranteed that a global optimal solution is found in contrast to OMP. However, among the investigated learning methods ODL (in combination with sparse encoding) yields to the best results in this evaluation with a score of 39.10%. In contrast to k-means++ a performance gain of almost 2% is achieved. Therefore we see that an encoding-optimized dictionary learning procedure can increase the performance.

Although the dictionary learning step is an offline step, we further discuss the resulting runtime. As ODL processes one descriptor at a time for a fixed number of iterations, this algorithm is efficient and the dictionary can be obtained in a short period of time. We therefore include this learning algorithm in our final recognition algorithm. Further, we also update our current evaluation pipeline and temporarily include sparse encoding. Note that the final choice about the used encoding method is discussed in Section 4.7.

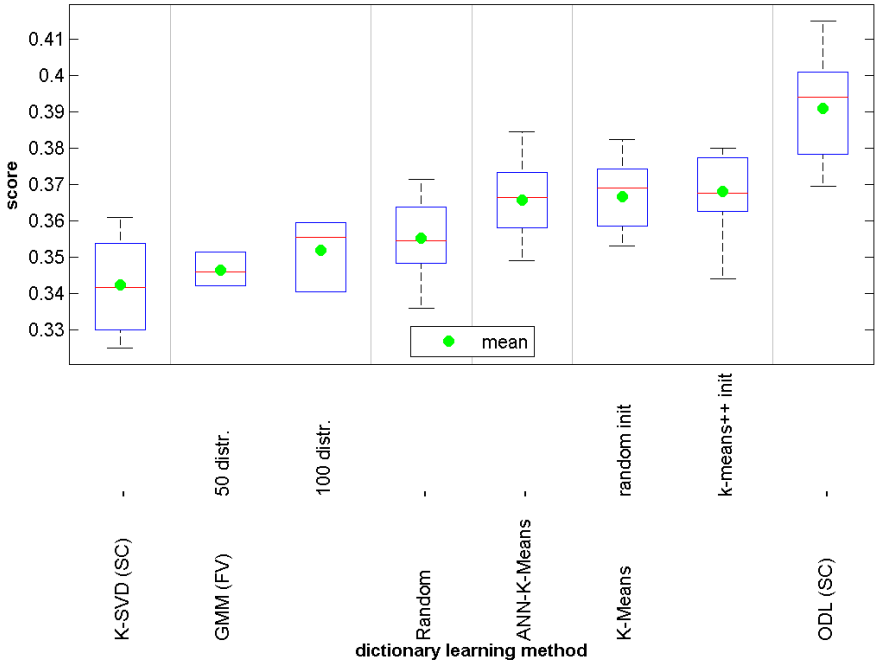


Figure 26: Evaluation for different dictionary learning methods. In the lower caption line the learning approach is displayed. Additionally, if another encoding method rather than approximated hard encoding is applied, the used procedure is denoted in the subsequent brackets, where "FV" means fisher vector encoding and "SC" sparse coding. Above this caption line in some cases additional information about the learning algorithm is declared. The boxplot visualizations as well as the cross-validation mean scores are provided.

Number of Visual Words

As we already found our dictionary learning method, the question about the required number of visual words arises. Therefore, we compare the following dictionary sizes: 100, 250, 500, 1 000 and 1 500. The results of this evaluation can be seen in Figure 27. We clearly see that our initial visual word amount of 500 is already sufficient to achieve a competitive recognition performance.

If more visual words are used, slightly higher performances can be obtained. On the other hand the performance drops for dictionary sizes below 500.

As the amount of visual words is increased, we observe that also the overall evaluation runtime increases linearly. This means that if e.g. 1 000 visual words are used instead of 500, the runtime is also twice as high. As a visual word amount of 500 leads to a competitive performance and at the same time results in an efficient pipeline, we use this amount for our final recognition algorithm.

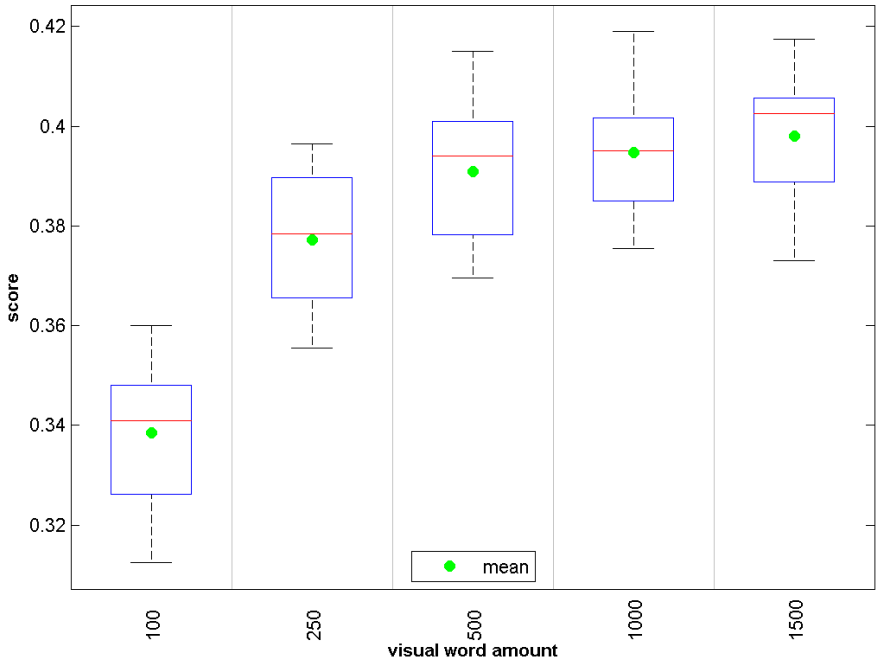


Figure 27: Evaluation concerning various dictionary sizes when the ODL learning algorithm and sparse coding are used. The boxplot visualizations and the cross-validation mean-scores are shown.

In general the more descriptors are provided for the dictionary learning procedure, the higher the quality of the resulting visual vocabulary is. However, runtime issues occur for large descriptor amounts. After a brief evaluation we define this amount to be 1 000 000 as this results in a reasonable runtime of the learning procedure. Therefore, this number of descriptors is obtained at random from the training set.

Discussion and Findings

In this evaluation we investigated the topic of dictionary learning. First, we applied several dictionary learning algorithms (and optionally the appro-

priate encoding method) to our pipeline. We recognized that this can have a major impact on the overall performance. However, we learned that the ODL learning algorithm with sparse coding was both efficient in runtime and achieved the best performing recognition score. Therefore, we decided to use ODL as our dictionary learning algorithm. This demonstrated that using a dictionary learning procedure that is explicitly optimized for the encoding used did have an impact on the recognition performance. In the next evaluation the question of the required number of visual words was answered. Although, the performance slightly increased when more than 500 visual words were used, due to efficiency reasons we chose this dictionary size. Finally, we defined that 1 000 000 randomly chosen descriptors from the training set were used to learn the visual dictionary.

4.7 Descriptor Encoding

An important step towards a powerful and compact sketch representation is the usage of a suitable encoding method. As argued in [57] the encoding method even has a stronger impact on the overall recognition performance than the used dictionary learning method. Therefore, we particularly pay much attention to this pipeline element and perform the following evaluations. First, we want to gain overall insight in which type of encoding works best in sketch recognition. As we only focus on the encoding methods, we use previously learned and fixed dictionaries. We further select the two most promising encoding methods – sparse coding and LLC – and investigate the impact of using the dictionaries optimized for sparse coding by the ODL algorithm. After that, we optimize the parameters of the selected sparse coding method.

Encoding Method for K-Means Dictionaries
We investigate the applicability of the seven encoding methods already defined and explained in Section 3.2.4. Again we use default parameter settings as suggested in the corresponding implementations (see Section 4.2) for all encoding methods. However, for the soft encoding method the uncertainty mode as defined in [45] is used. We pre-calculate the visual dictionaries consisting of 500 visual words by using k-means. For all but the Vector of Locally Aggregated Descriptors (VLAD) and FV encoding methods, these dictionaries are applied. For VLAD encoding we use k-means dictionaries which consist of 64 visual words. We further learned a GMM consisting of 64 Gaussian distributions for IFV encoding. In [74] it was shown that this number is sufficient for these two encoding methods. Due to the fact that for VLAD and IFV a high dimensional sketch representation is obtained, we

again restrict ourselves to one iteration per fold due to runtime issues.

The results of this evaluation can be found in Figure 28. We again see the same phenomenon as during evaluating the dictionary learning (Figure 26). Although closely related, the two algorithms used to solve the corresponding sparse coding optimization problems (OMP and LARS (Sparse in the figure)), produce highly varying results. The OMP algorithm is clearly outperformed by the iterations using LARS by 5%. This indicates that not the K-SVD dictionary learning method but the OMP encoding algorithm is the reason for the low scores. We already argued in that way for the dictionary learning evaluation in Section 4.6. However, the solution of the LARS/sparse coding problem formulation leads to high quality results with an average score of 39.07%. The scores are competitive to the LLC results (39.11%). The soft encoding iterations also lead to similar scores (1% lower than for LLC). The performance gap increases to roughly 2% when the hard encoding variants are considered. Surprisingly, the related VLAD encoding/pooling mechanism performs approximately 3% worse than hard encoding. As in Section 4.6, IFV encoding is not competitive to the best performing encoding methods.

After this evaluation we decide to use LLC and sparse coding for our next evaluation.

Encoding Method for ODL Dictionaries

Although the ODL algorithm learns a dictionary optimized for sparse coding, we also include LLC in this evaluation as the method is based on sparse coding. Further, we want to investigate the possible negative impact when we use a dictionary learned for encoding method A, but use method B. For both encoding methods the same previously learned dictionaries are used. The impact of the adapted dictionaries can be seen in Figure 29. In contrast to the iterations using k-means dictionaries, the performance of sparse coding is increased by almost 1% to 39.8% and therefore outperforms the previously best scoring LLC/k-means combination (39.11%). As expected, the performance of the LLC iterations slightly decreases when ODL dictionaries are used. This clearly indicates that dictionaries designed and built for specific encoding methods improve the performance of the overall approach. However, exactly the same encoding formulation has to be used in the dictionary learning algorithm to obtain this performance gain.

After this evaluation we decide to use sparse coding (the Least Absolute Shrinkage and Selection Operator (LASSO) formulation) as it shows the best performance in our evaluation with the corresponding dictionary learning method. Note however that this encoding mechanism in not as efficient

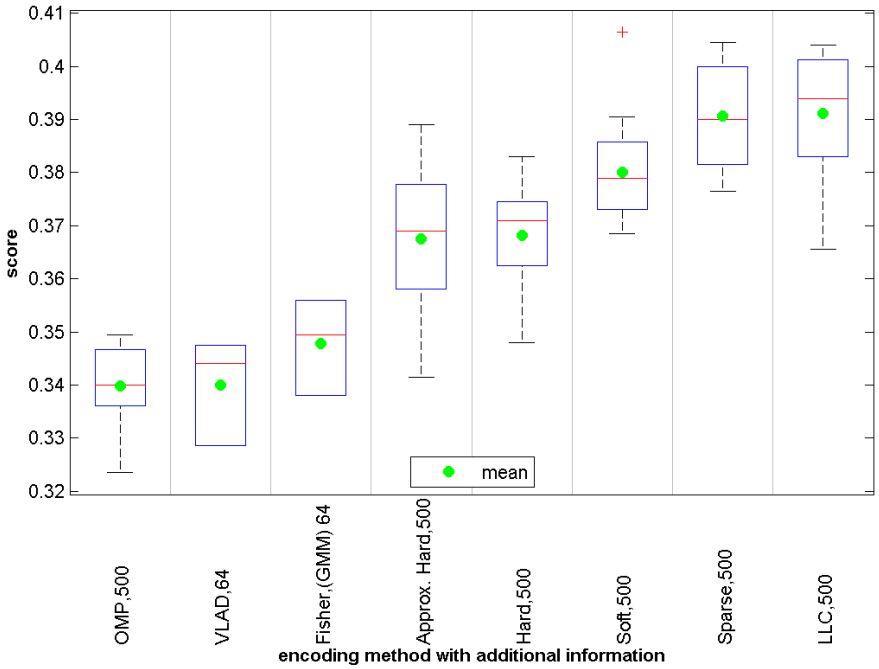


Figure 28: Evaluation regarding different descriptor encoding methods. We use fixed k-means dictionaries with 500 elements for this evaluation (except for VLAD (64 visual words) and FV (GMM with 64 distributions)). The encoding method and its used vocabulary size is provided below the boxplot visualizations. Additionally, the cross-validation mean scores are shown.

as LLC (half the runtime for the complete evaluation) or soft encoding (even $\frac{1}{8}$). Therefore, we suggest these two as alternative encoding methods (with k-means dictionaries used) in applications, where a low runtime is important.

Sparse Coding Parameter Optimization

Finally, we optimize the performance of sparse coding by adapting the available sparsity regularization parameter λ. We obtain the highest score with a parameter value of 0.2. This leads on average to a code per descriptor with six non-zero elements. However, this number varies between $1 - 19$. We also observe that the recognition accuracy can be increased when both for dictionary learning (ODL) and sparse coding a positive code constraint is introduced.

Therefore, we also include the latter two observations in our final recognition algorithm and in the current evaluation pipeline.

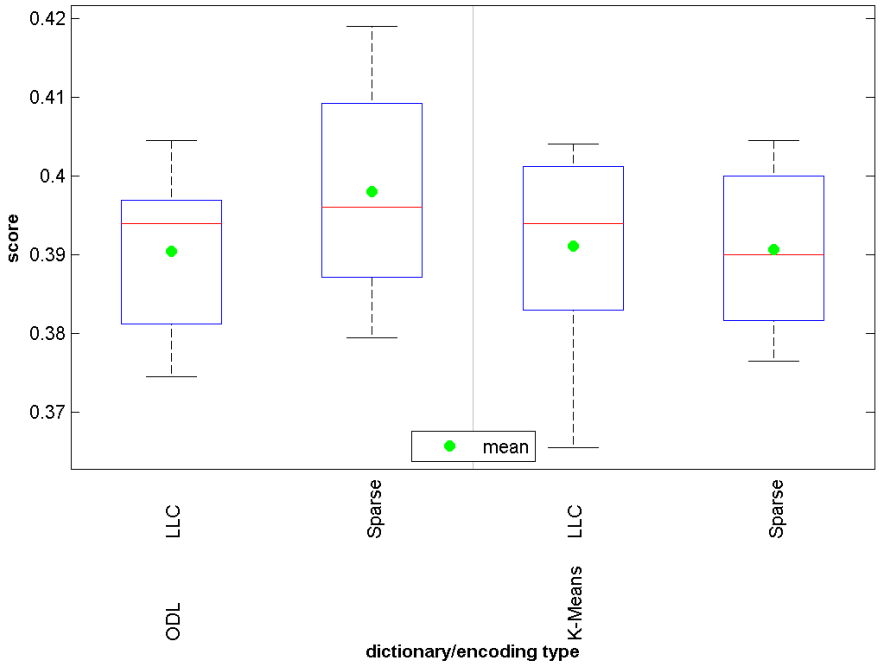


Figure 29: Evaluation regarding the usage of two dictionary learning methods (k-means and ODL) in combination with LLC and sparse coding. The left half illustrates the boxplot visualizations with ODL dictionaries and for the right side k-means dictionaries are used. Additionally the cross-validation mean scores are provided.

Discussion and Findings
By using fixed k-means dictionaries we observed that sparse coding and LLC perform best. Additionally, soft encoding showed a competitive performance. On the other hand we argued that successful encoding/pooling variants for image recognition as VLAD or FV are not suitable and applicable without modifications in the sketch domain, as they were even outperformed by the baseline hard encoding method. We also observed that with a dictionary particularly designed for sparse coding, the performance of the sparse coding approach can further be improved. Finally we investigated parameters for sparse coding to obtain an optimal performance.

4.8 Code Pooling

After we obtained the individual codes for the descriptors of the sketch image, the available information has to be aggregated to form the final sketch representation. As this representation is further used within the classifier,

it should be powerful enough to capture important aspects of the complete sketch. We both investigate pooling methods (as defined and explained in Section 3.2.5) on the single layer and when the spatial pyramid framework is introduced. Further, we discuss runtime issues, since by introducing a spatial pyramid, the sketch representation dimension gets larger by a multiplication factor proportional to the amount of pyramid sub-regions. Therefore, the upcoming classifier has to be able to handle this enlarged vector dimension efficiently.

The results of the pooling step evaluation can be found in Figure 30. First of all we clearly see a significant performance gap whether a spatial pyramid is used or not. With sum pooling, the single-level variant is outperformed by the corresponding spatial pyramid evaluation by almost 3.5%. This is a remarkable performance gain and stays in direct contrast to the observation discussed in [5]. Eitz et al. argued that the spatial pyramid framework does not lead to higher recognition scores in the sketch domain. However, our observed performance gain is in accord with the boost discussed in [15]. For both the single layer and spatial pyramid evaluation, it is surprising that the max pooling strategy is outperformed by sum pooling (and avg pooling in the single layer). Again, the opposite was observed in [20, 32, 76] for the image recognition domain.

Although an increased runtime for the subsequent classifier (RF) learning step by a factor of three is observed, we decide to use the spatial extension with sum pooling due the achieved performance gain. However, we consider sum pooling at a single layer as an efficient alternative for the pooling step.

4.9 Classifier Learning

In this last evaluation section, the goal is to find the most suitable classifier for our already defined sketch representation (as obtained in the previous Sections 4.4-4.8). We therefore first of all compare the three defined classifier types: Kernel SVMs (one-versus-all strategy), Linear SVMs (one-versus-all strategy) and a Random Forest as defined and explained in Section 3.2.6. Further, for the best performing classifier, the linear SVMs, we optimize its tradeoff parameter.

Classifier Type

For this classifier evaluation step, we change our evaluation setting and incorporate the setup which we further use for our final experiments in Section 5. The reason for this is, that we already want to gain knowledge about the final overall performance of our sketch recognition approach. For this

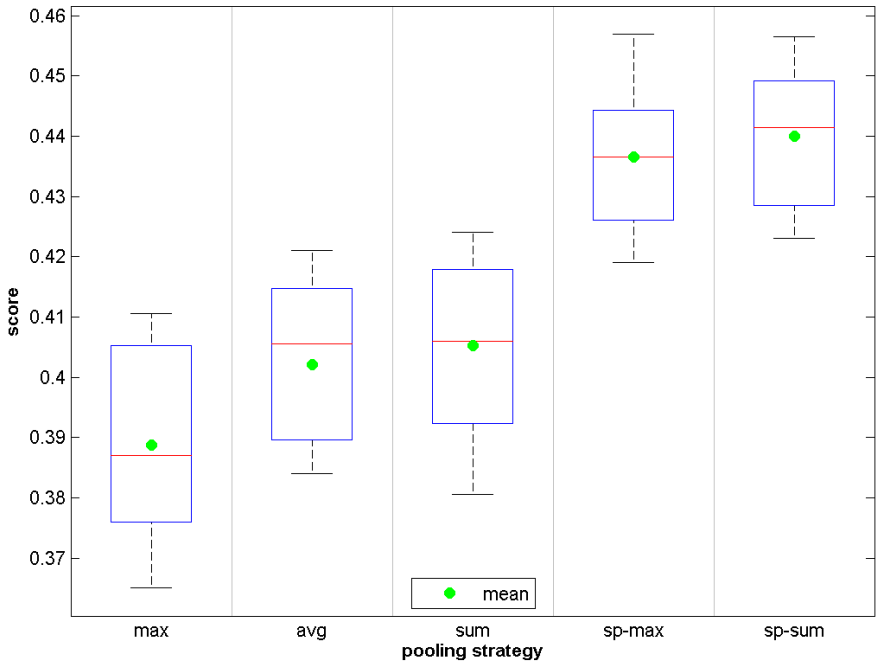


Figure 30: Evaluation for the pooling element of the pipeline. We investigate three pooling methods in a single-layer setting and apply two methods (sum and max) with the spatial pyramid (sp) framework. The boxplot visualizations and the corresponding mean scores are provided.

final setup we split the TU Berlin sketch dataset into four parts at random and keep this split fixed. Then, in the first iteration we choose part one as our evaluation dataset. For this dataset we perform 4-fold cross-validation again by a random split into four parts. As a next step, the dataset size is increased by using two parts and the evaluation procedure is repeated. This is done until all four parts are used for the evaluation procedure. The average recognition scores for each dataset size are discussed. In Section 5.1 this final experiment setup is explained in more detail.

For the SVM variants we use default parameters values for the tradeoff parameter C and the kernel parameter γ (only for kernel SVMs). After an initial RF parameter tuning experiment, we adapt the parameter values of the RF.

The result of the evaluations with the three classifiers types is shown in Figure 31. Surprisingly, the linear SVM based classifier outperforms both remaining types used in this evaluation. On average (over all dataset sizes) the iterations with linear SVMs achieve a mean performance which is roughly 1% better than when the kernel SVM based classifier is used. This gap increases

to about 5% if we consider the RF iterations. Therefore, we can argue that in this setting/domain the RF, even with optimized parameters, is not a suitable classifier type. In general, linear SVMs are known to lead to a lower recognition performance for baseline BoVW approaches with hard/histogram encoding [20]. However, when sparse coding is used, linear SVM based classifiers achieve a similar performance [20]. In this evaluation we even show that this classifier type can outperform kernel based SVMs when our obtained sketch representation is used.

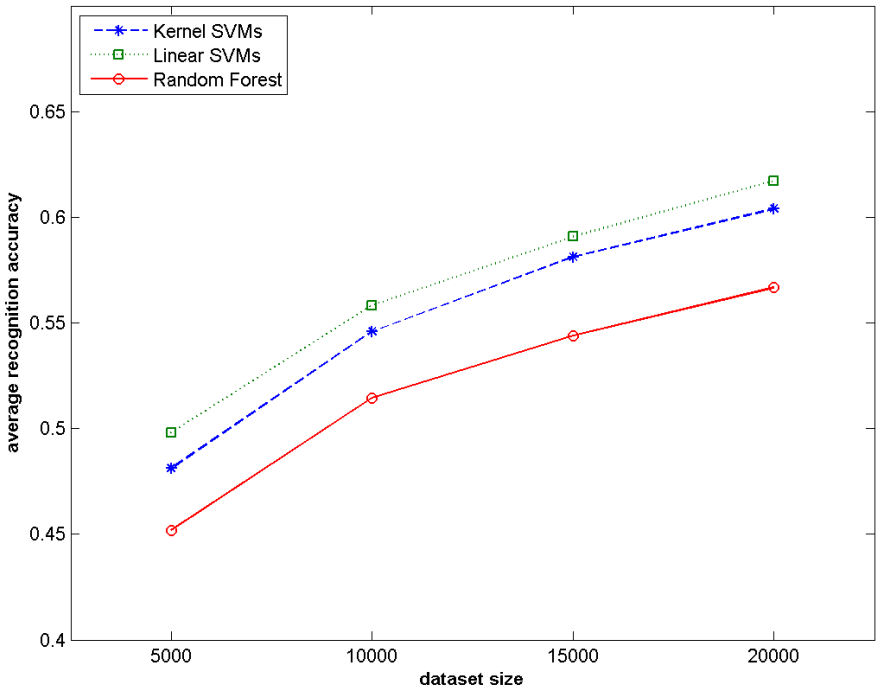


Figure 31: Average 4-fold cross-validation recognition scores for varying dataset sizes and for three different classifier types.

As another surprise, we observe that the classifier training time needed for the RF is much longer than for the linear SVMs. In general, RFs are known to be both efficient in training and classification. Although this is also true for linear SVMs, we argue that the magnitude is spectacular as can be seen in Figure 32. Whereas for the whole evaluation 176 hours are spent to train the RFs, with linear SVMs this time is reduced to 18 minutes. We argue that the enormous RF training runtime comes from the high dimensionality (10 500) of our obtained sketch representation due to the usage of a spatial pyramid. Therefore, the runtime advantages of the RF training procedure are reduced. However, both classifier types are equally efficient for classification. We also

argue that the training as well as the classification runtimes for the kernel based SVMs are reasonable with 220 and 24 hours respectively.

To conclude, the highest recognition accuracy is achieved with the linear SVM based classifier. Further, the runtimes for classifier training and evaluation are outstanding. Therefore, we include this classifier type in our final algorithm.

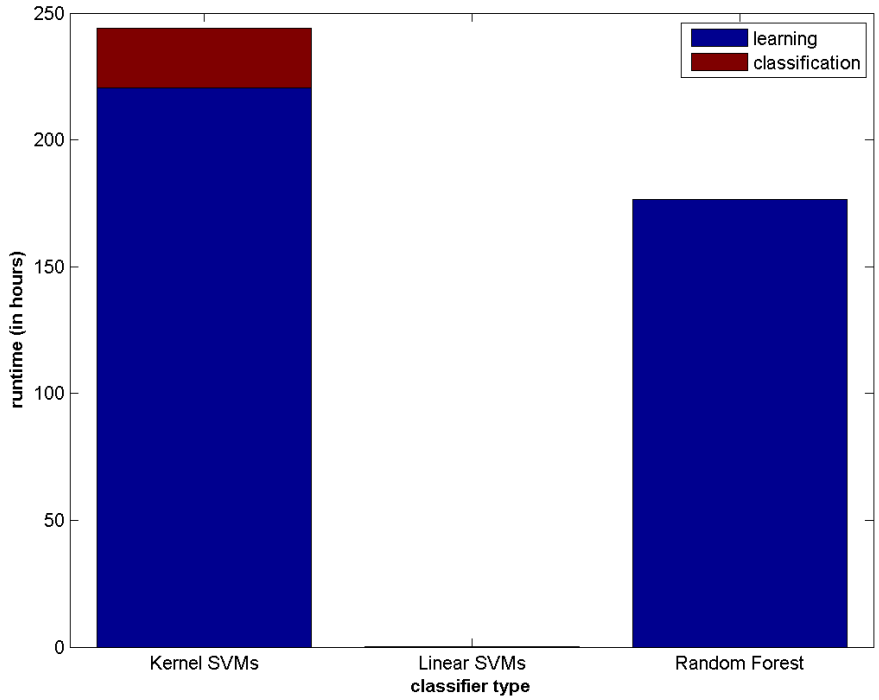


Figure 32: Classifier learning and evaluation runtimes (in hours) for the classifier type evaluation (Figure 31).

Linear SVM Parameter Optimization

As a final step of our evaluation phase, we optimize the linear SVM tradeoff parameter C. We therefore repeat the previous evaluation with various C parameter values in order to tune the classifier. Due to the high efficiency of the classifier learning procedure this evaluation is performed fast. Further, we can use previously obtained and fixed sketch representations. The results for parameter tuning step can be found in Figure 33. Therefore, we identify the ideal tradeoff parameter value to be 1.0. We include this setting in our classifier of our final sketch recognition algorithm.

Discussion and Findings

In this evaluation section we gained knowledge about the most suitable clas-

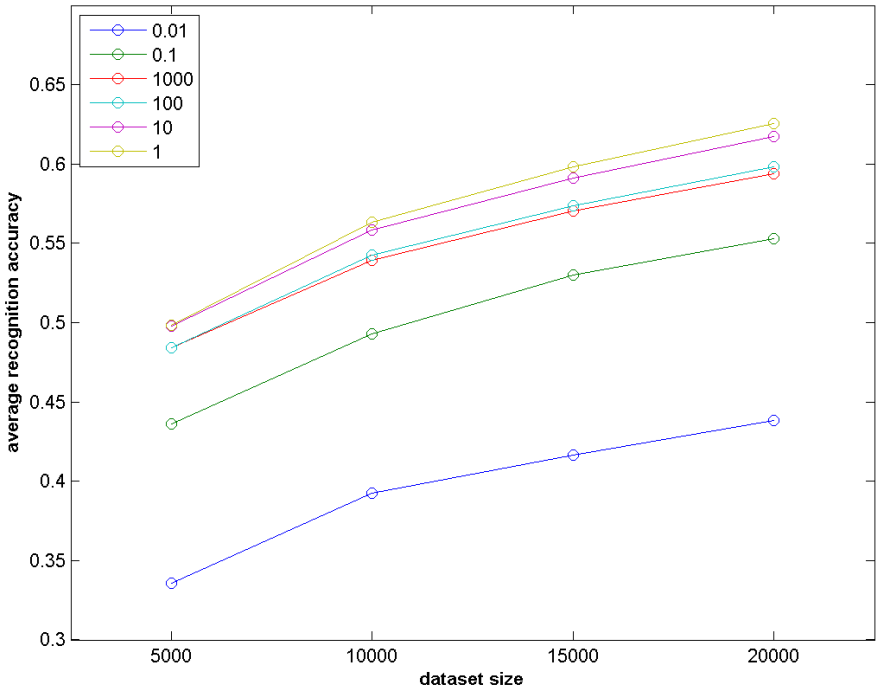


Figure 33: Parameter tuning for the tradeoff parameter C used for the one-versus-all linear SVM classifier.

sifier type for our sketch recognition algorithm. Linear SVMs used with a one-versus-all strategy achieved the best recognition performance. At the same time it was shown that this classifier type was highly efficient both in learning and classification. Whereas, the two other investigated classifier types (kernel SVMs and RFs) needed > 176 hours for classifier learning, this time was reduced to 18 minutes with a linear SVM. We therefore decided to use this highly efficient classifier type with an optimized tradeoff parameter.

4.10 Evaluation Phase Conclusion

In general we learned a lot of important and interesting lessons from the evaluation phase discussed in this section. Further, we gained a deeper knowledge for each element involved in this often used recognition pipeline structure. Our findings are not only restricted to the recognition performance, but also include runtime issues. Therefore, we know which element variants should be chosen for efficient algorithms. We argue that the gained knowledge is not necessarily restricted to the sketch domain but can also help to understand similar image recognition algorithms in more detail. In further consequence, this can be useful to improve such existing image recognition pipelines.

However, after the extensive evaluations performed in this section, we figured out that both investigated factors (recognition performance and runtime efficiency) interfered each other. Therefore we decide to propose two pipeline variants for our final sketch recognition approach. The first one is designed to achieve the best recognition performance at a reasonable runtime. Further, a more efficient but less accurate approach is introduced. We declare both algorithms with their element choices in the following section.

4.11 Final Sketch Recognition Approach

To sum up, we identified two variants of our algorithm, one tuned to optimal recognition performance ($full$) and one tuned to fast predictions ($fast$). These are used in the final experiments in Section 5. In the following we define and describe both algorithms in full detail.

Full Algorithm
For the image pre-processing step we first scale the *grayscale* format sketch image to a fixed size of 256×256 pixels. The sketch content is scaled such that the longer side of its bounding box has a fixed size of 202 pixels. The resulting bounding box is shifted to the image center and the image is smoothed using a Gaussian kernel with a σ of 2.0.

To extract local SIFT descriptors in a dense manner we use the *grid-sampling* strategy. We set the local patch size for SIFT calculation to 64 and use a step size of four. We reject descriptors which do not contain any information, i.e. descriptors for areas without a sketch line. The remaining SIFT descriptors are l_1 normalized. Afterwards, we follow the RootSIFT extension [56] and apply the square root on these normalized descriptors. Finally, we l_2 normalize the resulting descriptors. We retain the patch center locations for each descriptor as additional information.

The visual vocabulary is learned using 1 000 000 randomly sampled descriptors from all sketches of the specified training set without additional restrictions. This means that we do not use e.g. a fixed amount of descriptors per sketch category. We apply this strategy, as sketches of different categories have a varying amount of sketch lines and in further consequence descriptors. By using randomly sampled descriptors we do not obtain a biased dictionary. We incorporate the ODL algorithm to learn our unsupervised dictionary consisting of 500 visual words. For this dictionary learning algorithm we set the amount of iterations to 1 000, use a λ parameter of 0.2 and restrict the dictionary to only include positive visual words. We also restrict the sparse codes obtained in the ODL algorithm to be positive. The final dictionary is created by a l_2 normalization for each visual word.

Sparse coding (LASSO) is included for the descriptor encoding step of the algorithm. We use the same settings/parameter values as for the ODL algorithm, to make use of the optimized dictionary. Therefore, we obtain positive codes with a sparsity regularization parameter λ of 0.2.

To obtain the final sketch representation we use a spatial pyramid with three levels (1×1, 2×2, 4×4) and apply the sum pooling strategy for each of the occurring sub-regions. A concatenation of the 21 region representations yields to our final sketch representation vector with a dimensionality of 10 500 (500 visual words $\times$ 21 pyramid regions).

As a classifier we learn one-versus-all linear SVMs from the sketch representations of the training set. The value of the tradeoff parameter C is set to 1.0 by cross validation.

A visual depiction of the our final sketch recognition algorithm can be found in Figure 34. We discuss the resulting recognition performance as well as runtime issues in detail in Section 5.2.

Fast Algorithm

For our intended real-time applications, we further derived a second algorithm by replacing each pipeline element by a more efficient variant which however achieved reasonable performance in our evaluation phase.

We keep the image pre-processing step as defined before. In the descriptor extraction stage we again use the Dense SIFT setting but increase the step size from four to eight. This further reduces the amount of descriptors extracted by a factor of four. Instead of the ODL algorithm, we use approximated k-means with 100 000 randomly drawn descriptors to learn the visual vocabulary consisting of 500 visual words. For the encoding step we introduce LLC instead of sparse coding. The approximated LLC version, considering five nearest-neighbors with a λ equal to 0.1, is used. For code pooling we reject the usage of a spatial pyramid and only use the sum pooling strategy in a single layer to obtain the final sketch representation. This results in a more compact representation with a dimensionality of 500. The same classifier configuration is used as for the $full$ pipeline.

The recognition performance and the runtimes of this $fast$ algorithm is discussed in detail in Section 5.2. An additional tabular comparison of both introduced sketch recognition algorithms is given in Table 2.

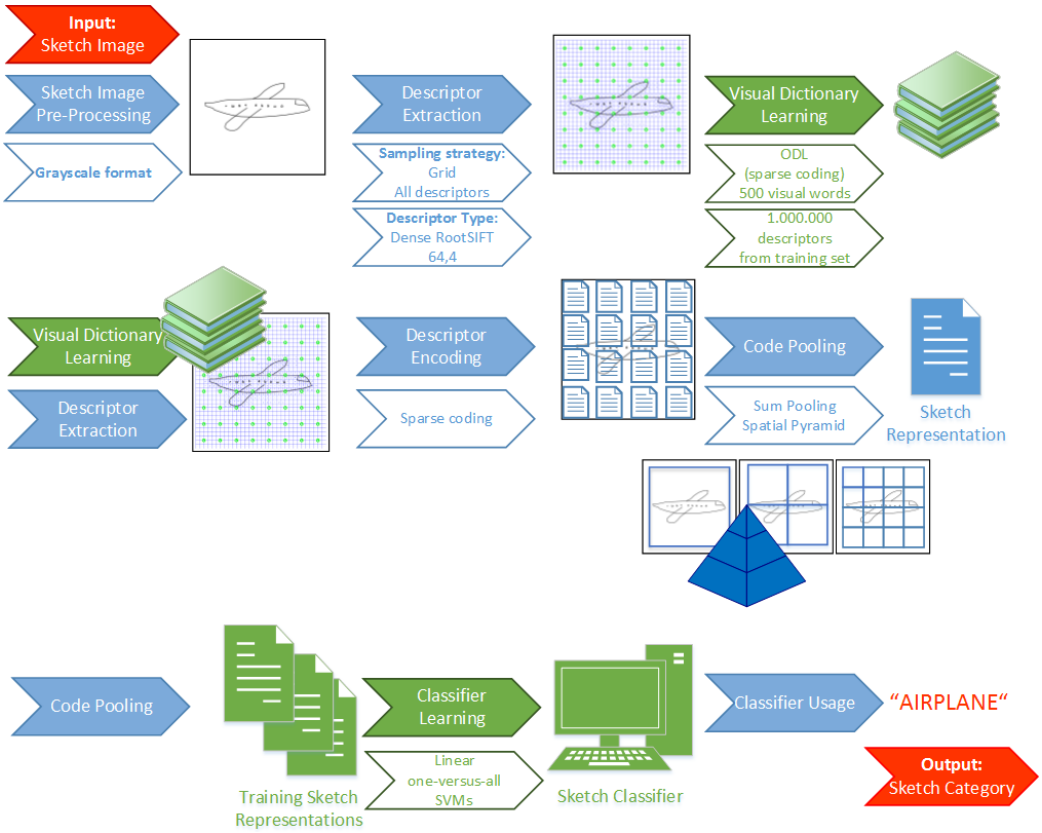


Figure 34: Graphical illustration of our final sketch recognition algorithm (full). Green steps are only performed in the learning procedure. The results of these elements (also green) are used during the classification process.

	Full	**Fast**
Sampling Strategy	Grid	Grid
Step Size	4	8
Descriptor Type	SIFT	SIFT
Patch Size	64	64
Dictionary Learning	ODL	Approximated K-Means
Vocabulary Size	500	500
Learning Amount	1 000 000	100 000
Encoding Method	Sparse Coding	LLC
Pooling Strategy	Sum with Spatial Pyramid	Sum
Sketch Representation Dim.	10 500	500
Classifier	Linear SVMs	Linear SVMs

Table 2: Comparison of our two obtained sketch recognition algorithms.

5 Experimental Results

In this section we perform final experiments for our two sketch recognition algorithms (as defined in Section 4.11), which were designed based on the findings of our evaluation phase (Section 4). First of all we explain our experimental setup in Section 5.1. As a first experiment we compare the recognition performance and runtimes of our two obtained algorithms in Section 5.2. Further, we include the evaluation baseline pipeline as defined in Section 4.3 in this discussion to show the impact of the evaluation phase. Further, we introduce and discuss two slightly adapted versions of our obtained $full$ pipeline and include them in this comparison. After that, we directly compare our two algorithms to related approaches of the sketch recognition domain in Section 5.3. We apply our novel sketch algorithm ($full$) without modifications in the domain of image classification in Section 5.4 and also run an experiment for a competitive image-domain approach on the sketches. Example recognition results for our $full$ algorithm are shown and discussed in Section 5.5. We also explain some interesting recognition statistics and perform a final experiment regarding the classifier responses. We conclude our performed experiments in Section 5.6. Finally, we briefly present our developed augmented reality application based on our novel algorithm in Section 5.7.

5.1 Experimental Setup

For all experiments performed in the sketch domain and in this section we use the TU Berlin sketch dataset [5]. We already explained this dataset in detail in Section 4.1.

In contrast to the setup used in the evaluation phase, we do not restrict ourselves to three (of 10) folds of the dataset anymore. As it is done in [46], we randomly split the TU Berlin sketch dataset with its 20 000 sketches and 250 categories into four equally-sized parts. We keep this split fixed for all experiments on this dataset. Therefore, each part consists of exactly 20 sketches per category. With this split we are able to investigate the performance for different dataset sizes and therefore a different amount of training data.

In the first iteration we only use part one of the split as the current dataset. To get the recognition results for this dataset we perform 4-fold cross-validation. This means that three folds (or 15 sketches for this dataset size) are used for the training procedure and the remaining fold (five sketches) is used to obtain the recognition performance. These four folds are created for each dataset at random. Once the recognition results for the four folds

are available, we increase the amount of sketches used for the current dataset by choosing part one and two. Further, we repeat the 4-fold cross-validation experiment procedure. This is done until all four parts are used for the 4-fold cross-validation experiment.

In that way for each of the four dataset sizes, four recognition results are obtained (overall 16 scores). Note that individual visual dictionaries and sketch classifiers are learned for each iteration. In contrast, in [5], one visual dictionary is created by using descriptors of the complete TU Berlin sketch dataset and applied for experiments. For our four different datasets 15, 30, 45 and 60 sketches are used for the corresponding learning procedures. We report the average recognition scores per dataset size, where the score, as already defined in Section 4.3, measures the percentage of correctly recognized test sketches.

We use the sketch representations of [5] which are publicly available for the Bag-of-Visual-Word (BoVW) based approach of Eitz et al. Further, we learn corresponding sketch classifiers and calculate the recognition scores using our experimental framework in order to get comparable results. When using the kernel based Support Vector Machines (SVMs) as a classifier, we apply the parameters as defined in [5]. For the experiments with linear SVMs we use default parameter values. The results for the ensemble matching approach of Li et al. [46] were kindly provided by Yi Li (only for kernel based SVMs).

For the experiment in the image domain in Section 5.4 we use the Caltech-101 dataset [89]. This dataset consists of 9 144 images of 101 object categories and a background class. The amount of images per class varies between 40 and 800. Further, the images do not have a fixed size and the objects included vary in scale, shape, pose, rotation etc. To get the results for the Locality-constrained Linear Coding (LLC) image recognition approach, we use the publicly available source code of Wang et al. [32]. This approach was already discussed in detail in Section 2.2. A visual dictionary obtained from randomly sampled descriptors from all categories and the whole dataset, which is learned using k-means, is also provided. In contrast to our experimental setup in the sketch domain, we incorporate this strategy for this image recognition experiment using our unmodified $full$ algorithm. As further consequence, the image representations for the whole dataset can be pre-calculated. We define the amounts of images used for each class for the training procedure to be 10, 20 and 30. These training images are chosen at random. For each image amount the experiment is repeated four times. The image representations for the remaining images are used to obtain the recognition results. This strategy was also suggested by the creators of the

Caltech-101 dataset [89]. The final score is obtained by using the average accuracies for all categories.

To obtain the LLC result in the sketch domain, we use the available source code of [32] and adapt it in such a way that the TU Berlin sketch dataset is used.

All experiments were performed on a desktop computer with a 3.2Ghz Quad Core CPU and a memory of 8GB using Matlab R2012b.

5.2 Results of Proposed Sketch Recognition Algorithm

In this section we compare our two sketch recognition pipelines ($full$ and $fast$) defined in Section 4.11 regarding recognition performance and runtime. To further show the impact of our extensive evaluation phase (Section 4) we also consider the evaluation baseline pipeline as defined in Section 4.3. In the following we denote this pipeline as *baseline*. Additionally, we analyze the impact of two slightly adapted versions of the $full$ pipeline, $full\ llc$ and $full\ soft$. We discuss these two algorithms as well as their motivations when we concentrate our discussion on runtime issues. However, we start this section with the recognition performance comparison.

The sketch recognition results can be found in Figure 35. First of all we notice that our evaluation phase yielded a remarkable performance gain. When the whole dataset with 80 sketches per category is used, the performance increases by 22% when our $full$ pipeline is used compared to the *baseline* algorithm. With the $full$ algorithm and the full dataset, 63% of the sketches are recognized correctly. These rates drop to 41% for the *baseline*. This gap decreases for smaller dataset sizes but does not undercut 17%. When we consider the $fast$ algorithm the recognition performance gain is between 10 and 12%. Although we use pipeline elements which are more efficient than the corresponding parts of the *baseline* pipeline, we achieve a recognition rate which is constantly higher by a double-digit value. This is another remarkable result and further points out the quality of our evaluation phase. As we do not restrict ourselves solely to performance but also focus on efficiency we were able to learn about pipeline elements which combine these two aspects. In that way we were able to design an additional runtime-optimized but still accurate recognition pipeline. However, for the comparison of our two obtained algorithms $full$ and $fast$ we notice a performance gap. The range lies between 7 and 9%. This is interesting as we expected this gap to be narrower. We argue that although the individual elements achieve a similar performance, the combination of such elements result in a significantly

lower overall recognition performance. In a more mathematical formulation we can say that the reconstruction error is introduced by one element, transferred to the subsequent elements and reinforced by these elements. When the final classification step is reached it is obvious that a higher overall error and therefore a lower recognition score is achieved. Further, this comes from the fact that fewer descriptors are used due to a larger grid step size. Before we discuss the performance of the remaining two pipelines in the plot in Figure 35 we turn our attention to runtime related issues to understand the motivation behind the algorithm adaptions.

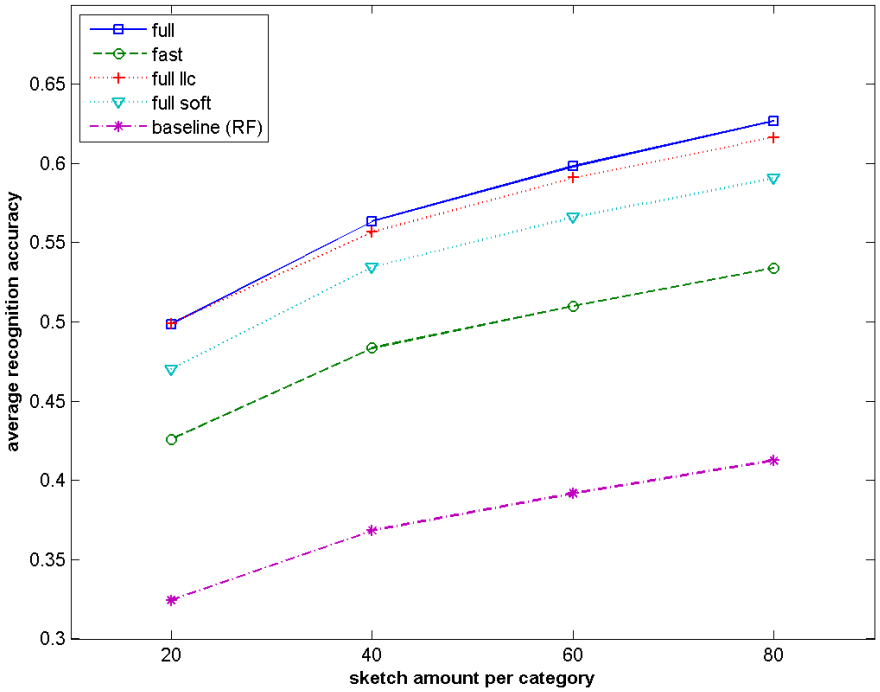


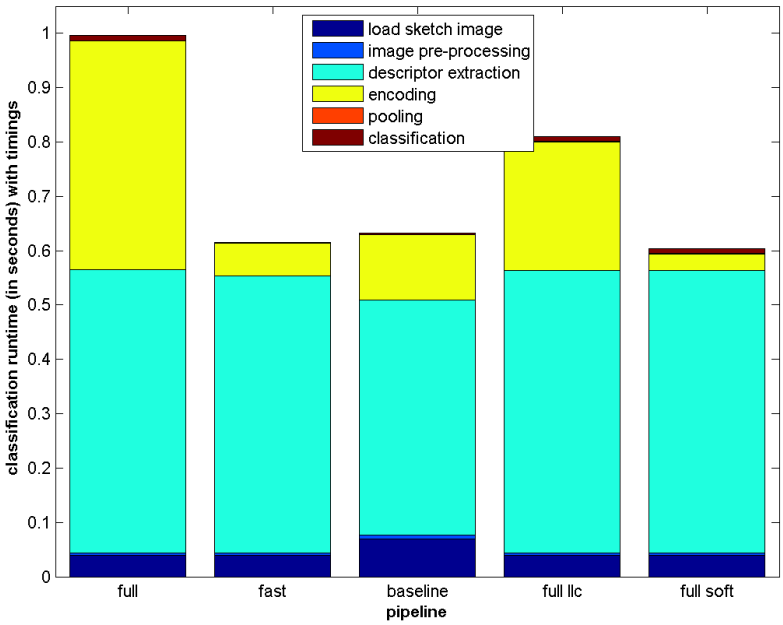
Figure 35: Direct recognition performance comparison of our two obtained recognition pipelines (full and fast) as well as our initial evaluation pipeline (baseline) for different dataset sizes using 4-fold cross-validation. Further, the scores of two slightly adapted versions of the full pipeline, full llc and full soft, are included. For these two pipelines the encoding and (for full soft) the dictionary learning method is/are replaced. Note that for the baseline algorithm a Random Forest is used as a classifier whereas the remaining pipelines use linear one-vs-all SVMs.

For classifying one single sketch image, the timing information was obtained by per-step runtime measurements for 100 randomly selected sketches and the average timings are reported. The diagram regarding the exact classification timings can be found in Figure 36(a). When considering the *full*

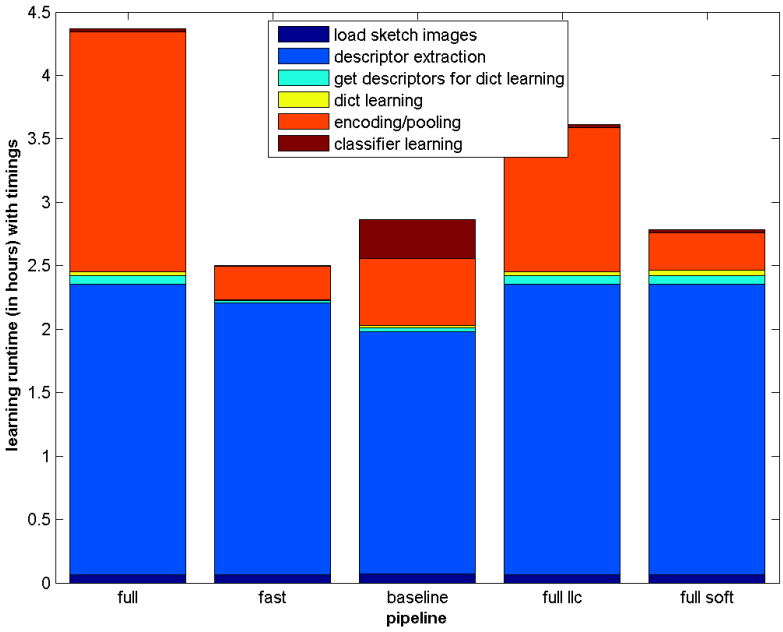
pipeline, the classification of a novel sketch is highly efficient for all but two steps with an average runtime below 40ms (including the sketch image loading). However, the descriptor extraction and the encoding step can be seen as runtime bottlenecks with an average runtime of 0.52 and 0.42 seconds. As these steps are performed using external libraries we are not able to optimize the code. Nevertheless, on average a classification response is obtained for this pipeline within a second. With the $fast$ algorithm a recognition result is obtained within 0.6 seconds on average. This already points out that our efficiency optimized approach has the desired impact. However, the descriptor extraction bottleneck is not solved although fewer descriptors are obtained. Nevertheless, all of the remaining steps are highly efficient, even the encoding mechanism with an average runtime of 60ms. The first reason for this is that a more efficient encoding method (approximated LLC) is used. Secondly, as less descriptors are used per sketch, fewer encoding operations have to be performed. Further, no spatial pyramid is involved in the pooling step which results in a more compact sketch representation. If we discuss the classification runtime of the *baseline* algorithm we see that a recognition result can be obtained in similar time as for the $fast$ pipeline. This is interesting as the *baseline* algorithm only uses a random subset of 1 000 descriptors for each sketch instead of the full set as it is done both in the $full$ and $fast$ algorithm. Due to the fact that a smaller patch size is used, the SIFT descriptor extraction is performed more efficiently and needs on average 0.43 seconds. However, in contrast to the $fast$ algorithm version the encoding step (exact hard encoding) is less efficient although fewer descriptors are used. This is most likely due to the fact that an exact nearest-neighbor search is performed.

The timings for the training procedures using 15 000 sketches can be found in Figure 36(b). For the $full$ algorithm the whole learning procedure obtains a visual vocabulary as well as the classifier in 4.4 hours. We again see that most of the time is spent for descriptor extraction and encoding. When using the $fast$ algorithm the learning procedure is finished after 2.5 hours. Again the overall time is reduced due to the factors described before. With a learning procedure duration of 2.9 hours, the *baseline* approach is less effective even with fewer descriptors per sketch used. We already discussed the reason for that above. However, we observe that the RF learning procedure is time consuming with 19 minutes. A similar observation was made in the classifier evaluation in Section 4.9.

We argue that both the learning and classification runtime for the $full$ pipeline are sufficient for most scenarios regarding user applications such as e.g. a search engine/retrieval system. This further means that we satisfy our efficiency goal for this algorithm both for the learning and recognition



(a) classification procedure timings (for one sketch)



(b) learning procedure timings (for 15 000 sketches)

Figure 36: Timing information (classification and learning) for all investigated pipelines. The individual steps are depicted by different colors and the height of each block represents the average duration spent for this pipeline element.

procedure. However, we further investigate the question if we can design a similar algorithm which is as efficient as the $fast$ pipeline and at the same time achieves competitive performance to the $full$ approach. As we already discussed, the runtime bottleneck of the encoding step can be solved by using a more efficient method. Therefore, for our first investigated approach we replace sparse encoding of the $full$ algorithm by LLC and keep the remaining pipeline fixed. We decide to use LLC as it showed competitive performance to sparse coding even for dictionaries learned by the Online Dictionary Learning for Sparse Coding (ODL) algorithm as discussed in Section 4.7. We denote this pipeline as $full\ llc$. The recognition performance of this approach can be seen in Figure 35. We observe that a similar average recognition rate of 62% is achieved when the full dataset is used. This also represents the performance difference observed in our evaluations according the encoding step in Section 4.7 when ODL dictionaries are used. In Figure 36(a) we see that a novel sketch is recognized within 0.8 seconds for this novel algorithm and the learning procedure takes 3.6 hours (Figure 36(b)). For both the classification and the learning procedure we observe a runtime reduction for the encoding step. However, compared to the $fast$ algorithm version, still a runtime gap is present. Again this is mainly because of the encoding step. Although the same method (LLC) is used, more codes have to be obtained due to the fact that more descriptors are used.

Soft encoding was identified as another competitive and even more efficient encoding method in Section 4.7. However in that evaluation, k-means dictionaries were used. As using ODL in combination with soft encoding does not make sense, we further change the dictionary learning method to approximated k-means. Therefore, in contrast to the $full$ algorithm, the dictionary learning method and the encoding method are different. The resulting pipeline is denoted as $full\ soft$. We observe that for the full dataset 59% of the sketches are recognized correctly (see Figure 35). Therefore, the performance is 4% worse than with the $full$ algorithm. On the other hand, the $fast$ pipeline is outperformed by 6% for this dataset size. In Figure 36(a) we see that the sketch classification process needs 0.6 seconds. This is as efficient as for the proposed $fast$ algorithm. By using soft encoding, the runtime needed for this step is even lower than for LLC with fewer descriptors processed. The runtime needed for one learning procedure of this algorithm is 2.78 hours as can be seen in Figure 36(b). Although this is slightly higher than for the $fast$ pipeline (2.5 hours), the visual dictionary and the classifier are learned faster than for the *baseline* algorithm (2.9 hours). An interesting observation can be made if we take a closer look at the dictionary learning times of the $full$ and the $full\ soft$ pipelines. Although approximated k-means is used, the

time spent for the dictionary learning step is slightly higher than for the ODL algorithm with an equal amount of descriptors used. This is a result of the fact, that although kd-trees are used, the higher the data dimensionality and the data amount, the less efficient this data structure becomes. In contrast, ODL selects and processes one descriptor per iteration with a fixed number of iterations.

We therefore see that the recognition performance highly depends on the encoding method used and on the specialized dictionaries. More powerful encoding methods need more computational runtime but lead to higher recognition performance. On the other hand more efficient encoding schemes result in lower overall recognition scores.

Conclusion and Findings
To conclude this experiment section, we first of all showed that a remarkable performance gain for our two obtained algorithms $full$ (63% with the full dataset) and $fast$ (53%) compared to the evaluation *baseline* pipeline (41%) was achieved. Further, we observed that although both obtained pipelines ($full$, $fast$) were designed to be both efficient and accurate, these two goals interfere each other. The first pipeline ($full$) showed an excellent performance but needed a lot of computational time for the encoding step. For the $fast$ pipeline it was shown that it was more efficient – even more than the *baseline* approach with fewer descriptors used per sketch – but at the cost of a lower recognition performance. We argued, that the gap was the result of the combination of "not best performing" pipeline elements used, although individually these showed a competitive performance in the evaluation phase. We further investigated classification and learning runtimes and noticed that the runtime bottlenecks were the descriptor extraction and the encoding step. For the $full$ pipeline a recognition result could be obtained in less than one second. The learning procedure took 4.4 hours when 15 000 sketch were involved. The prediction of a novel sketch using the $fast$ pipeline was calculated in 0.6 seconds and the learning procedure was finished after 2.5 hours. We argued that the runtimes for both pipelines were suitable for the sketch recognition task. However, we introduced two novel slightly adapted versions of the $full$ pipeline: $full\ llc$ (LLC instead of sparse coding) and $full\ soft$ (approximated k-means and soft encoding instead of ODL and sparse coding). For the $full\ llc$ pipeline it was shown that a competitive recognition performance can be obtained (62%). Further, the runtimes could be decreased to be 0.8 seconds and 3.6 hours. With the $full\ soft$ algorithm the performance dropped to 59%. However, the same efficiency as with the $fast$ pipeline with runtimes of 0.6 seconds and 2.8 hours was achieved.

5.3 Comparison to Related Approaches

In this section we compare our two pipelines $full$ and $fast$ to two related approaches for the domain-independent sketch recognition task. These two approaches are the BoVW-based algorithm of Eitz et al. [5] and the ensemble matching algorithm of Li et al. [46]. We already discussed both approaches in detail in Section 2.3. In contrast to our algorithms, these two algorithms use kernel based SVMs. Therefore, we compare the approaches for both SVM types individually to achieve a fair comparison. As mentioned in the experimental setup (Section 5.1), the sketch representations of the Eitz approach for the TU Berlin sketch dataset are publicly available. Therefore we are able to obtain results for both classifier types using our experimental framework. Results for the ensemble matching approach using kernel based SVMs were kindly provided by Yi Li. Therefore, we are not able to use this approach with linear SVMs.

The recognition performances of the four algorithms using kernel based SVMs are shown in Figure 37. First of all we realize that for all dataset sizes our $fast$ pipeline outperforms the "Eitz sketch representations", which achieves a score of 52.5% when the full dataset is used. This is a surprising result which again points out the quality of our evaluation phase (Section 4). Although, we choose highly efficient and less accurate pipeline elements, the recognition performance gap is between 1.5 and 2.2%. Further, we like to add that in [5], the visual dictionary was obtained from randomly sampled descriptors of the whole dataset. In further consequence, this vocabulary was used to obtain the "Eitz sketch representations". In contrast, we restrict ourselves to descriptors of the training set. With our $full$ pipeline we are able to outperform the "Eitz approach" by 7-8%. This is remarkable, as we do not even use the most suitable classifier as learned the evaluation in Section 4.9.

If only 20 sketches per category are used for this experiment, we are able to surpass the recognition score of the ensemble matching approach [46]. However, as we increase the dataset size, the performance passes our $full$ results. With the full dataset used, a recognition accuracy of 61.5% is achieved with this approach. In contrast, we recognize 60.4% of the sketches correctly with our $full$ algorithm. We argue that this performance gain makes sense for ensemble matching. As more sketches are used, the better matches can be found. For our BoVW-based algorithm this results in a more accurate representation. However, the impact is weakened, as we do not use a star graph of the full sketch, but only descriptors of parts of the sketch, which contain less information. Nevertheless, as mentioned before, these results are obtained by using kernel based SVMs, which are not the

most suitable classifier type as learned in our evaluation phase. However, we achieve a competitive recognition performance. This further means, that we carefully selected our sketch representation such that it is also suitable for different classifier types. The performance gain between the ensemble matching and our *fast* pipeline lies between 4.7 and 7.7%.

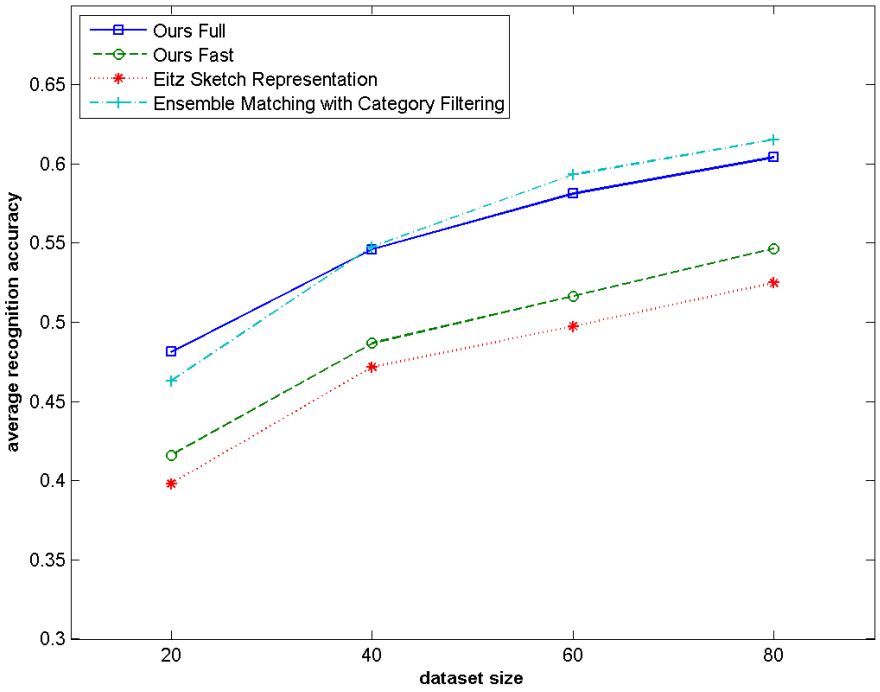


Figure 37: Recognition performance comparison of our two algorithms (full and fast) with two related approaches for domain-independent sketch recognition using our experimental setup. The results for all methods are obtained using kernel based SVMs. Note that in our original algorithms linear SVMs are used. A different amount of sketches per category is used and 4-fold cross-validation is performed for each dataset size. The average recognition scores are illustrated.

Next, we repeat this experiment (without considering the result of the ensemble matching approach) and use linear SVMs as a classifier. The experiment results can be found in Figure 38. We see that the gap between the "Eitz approach" to both of our algorithm grows compared to the experiment before. For the *full* pipeline the performance difference is doubled and amounts to 16-17%. With the *fast* pipeline this gap is between 7 and 9%. We therefore see that the "Eitz representation" leads to a bad recognition performance when linear SVMs are used. Compared to the previous experiment in Figure

37, the performance drops by 6-7%. In contrast, we see that both of our sketch representations work similar with both SVM types. However, if we analyze the performance curves of the $fast$ pipeline, we notice that, by using kernel based SVMs, the scores are slightly higher by 1%. We argue that by spanning our sketch representation to a higher dimensionality with the usage of a spatial pyramid ($full$), a linear separation leads to better classification results than for the compact space (as used in the $fast$ pipeline). In contrast, the performance of the $full$ pipeline is increased by 2% when linear SVMs are used compared to the experimental run with kernel SVMs.

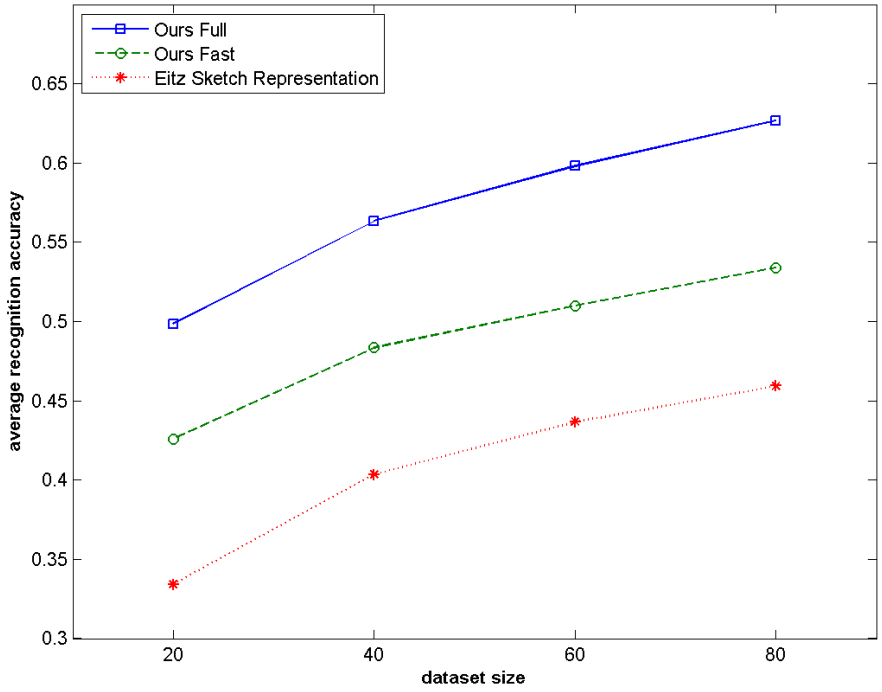


Figure 38: Recognition performance comparison of our two algorithms (full and fast) with one ("Eitz representation") related approach for domain-independent sketch recognition using our experimental setup. The results for all methods are obtained using linear SVMs. Note that the related approach is designed to be used with kernel based SVMs. Further, the scores of the ensemble matching approach [46] are missing due to the fact that no results are available for linear SVMs. A different amount of sketches per category is used and 4-fold cross-validation is performed for each dataset size. The average recognition scores are provided.

In the following we choose the classifier type which lead to the highest recognition scores in the previous two experiments (Figures 37 and 38) for all

four investigated algorithms. Therefore for all but our $full$ pipeline, kernel based SVMs are used. The results of that comparison are shown in Figure 39. We see that with the most suitable classifier our $full$ pipeline is able to outperform the ensemble matching approach [46]. With an amount of 20 sketches used per category, we surpass this approach by 3.5%. This further points out the advantage of BoVW-based approaches compared to star graph matching for small dataset sizes. The performance gap decreases to 0.5% for a dataset size of 60. However, when all 80 sketches per category are used, our $full$ algorithm outperforms the ensemble matching approach again by 1%. This means that with our $full$ pipeline, we achieve, to the best of our knowledge, state-of-the-art performance on the TU Berlin sketch database. Although the ensemble matching approach is competitive concerning recognition performance, it is not efficient. This comes from the fact that a simplified version of our BoVW-based pipeline is used as a pre-processing step (for category filtering). Additionally, the main step of the recognition approach, the actual graph matching, is computational expensive as a query graph has to be compared to a subset of graphs from the database. Although we do not have information about exact runtimes, we argue that this already shows that this approach is not efficient enough to be used in interactive applications. In contrast, for our $full$ recognition algorithm this is indeed possible. For the results obtained with the "Eitz representation" and kernel based SVMs, the gap to our $full$ pipeline varies between 9 and 10%.

Conclusion and Findings
In this section we compared the recognition performances of our two novel algorithms to two related approaches of that domain. Even with our efficient ($fast$) pipeline consisting of low-cost elements we were able to outperform the approach of Eitz et al.[5] which is also based on the BoVW method. Our $full$ pipeline was competitive with the computationally expensive ensemble matching approach of Li et al. [46] for kernel SVMs. With linear SVMs we were able to surpass the recognition scores of this approach. Therefore we achieved, to the best of our knowledge, state-of-the-art performance on the TU Berlin dataset. We further showed that there is a mentionable performance difference for the available "Eitz sketch representations" for the two SVM types. In contrast our two introduced and carefully designed sketch representations achieved a similar performance for both classifier variants.

5.4 Cross-Domain Experiment

As next experiment we evaluate our $full$ sketch recognition algorithm in the image domain on the Caltech-101 dataset [89] without adaptions. The

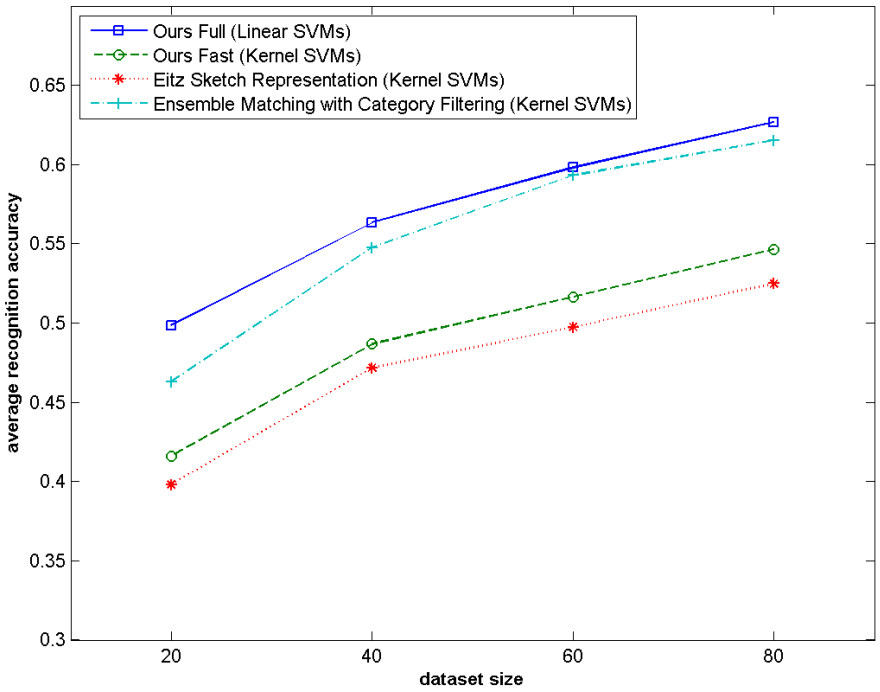


Figure 39: Recognition performance comparison of our two algorithms (full and fast) with two related approaches for domain-independent sketch recognition using our experimental setup. For each algorithm the best performing classifier type from the previous experiments is used as stated in the legend. A different amount of sketches per category is used and 4-fold cross-validation is performed for each dataset size. The average recognition scores are provided. Our full algorithm using linear SVMs achieves state-of-the-art performance on the TU Berlin sketch dataset for any dataset size.

dataset as well as the setup for this experiment were already discussed in Section 5.1. The basic principle for this experiment is the fact that our sketch recognition partly consists of successful image recognition elements. Although our algorithm is specialized to the task of sketch recognition, we argue that reasonable performance is possible for the image domain without pipeline modifications. Further, we reverse the domains and therefore evaluate the original LLC approach [32], which has proven to yield state-of-the-art performance in the image recognition domain, on sketch images.

The image recognition performance of our *full* pipeline is shown in Figure 40. If we compare this performance curve to the original LLC image recognition approach, we clearly see the advantage of the LLC approach in the image domain. We argue, that LLC is designed in a proper way to be

suitable for the image recognition task. In contrast, our pipeline is optimized for the related but still diverse sketch domain. From this aspect we argue that the performance gap of 8% is reasonable. Nevertheless, our sketch pipeline is able to achieve a recognition score of 63% for 30 training images per category in the foreign domain and for a challenging dataset. The original LLC approach however achieves a score of 71% for the same training amount. Finally, we argue that our *full* pipeline is far too specialized for the sketch domain to achieve higher and competitive recognition results compared to such a state-of-the-art method as LLC is.

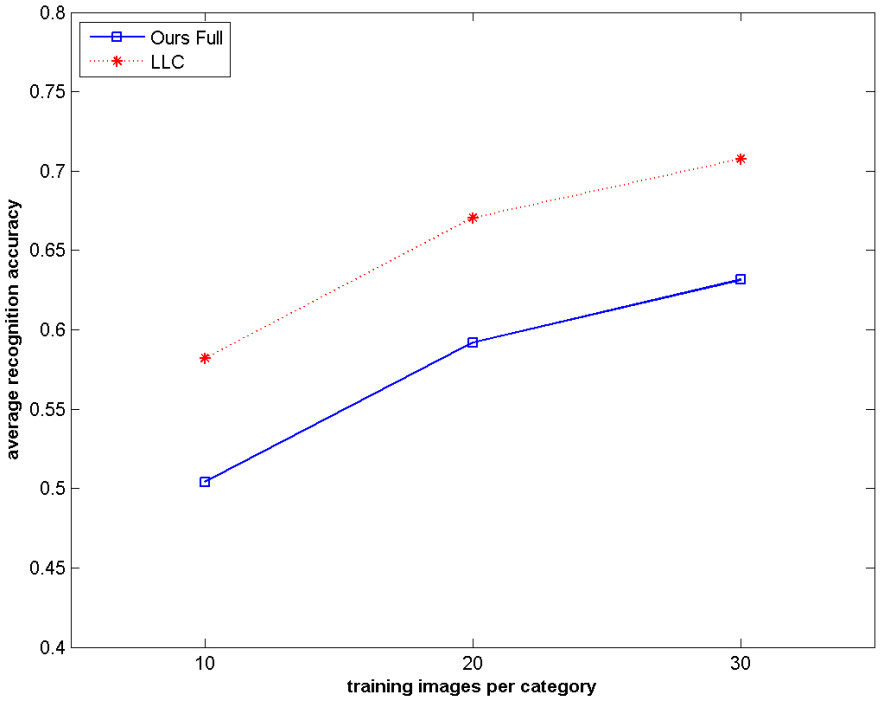


Figure 40: Results of our unmodified sketch-specialized recognition algorithm (full) for the task of image recognition on the Caltech-101 dataset. The original LLC approach [32] is used as a direct competitor. The experiment is performed for three different image amounts per category used for training. For each amount four iterations are calculated and the average scores are presented.

Next, we evaluate the performance of the LLC image recognition approach in the sketch domain. With this experiment we investigate, if the LLC approach is as specialized for the image domain as our approach is for sketches. Therefore, we compare our two pipelines *full* and *fast* to this image domain approach. For this experiment, again our sketch recognition experimental

setup as defined in Section 5.1 was used. The results for this experiment can be found in Figure 41. The LLC approach is outperformed by 10-11% by our *full* approach in the sketch domain. Further, also our *fast* algorithm achieves a higher recognition rate with a performance difference of 2-3%. However, the original LLC approach recognizes 51% of the sketches correctly when the complete dataset is used. This is competitive to the results obtained with the "Eitz sketch representations" and kernel based SVMs in Section 5.3 with 52.5%.

Nevertheless, we argue that the algorithm optimization or design process leads to problems for portable cross-domain approaches. The obtained pipelines are far too much specialized such that a simple application in a different domain is possible without further pipeline modifications.

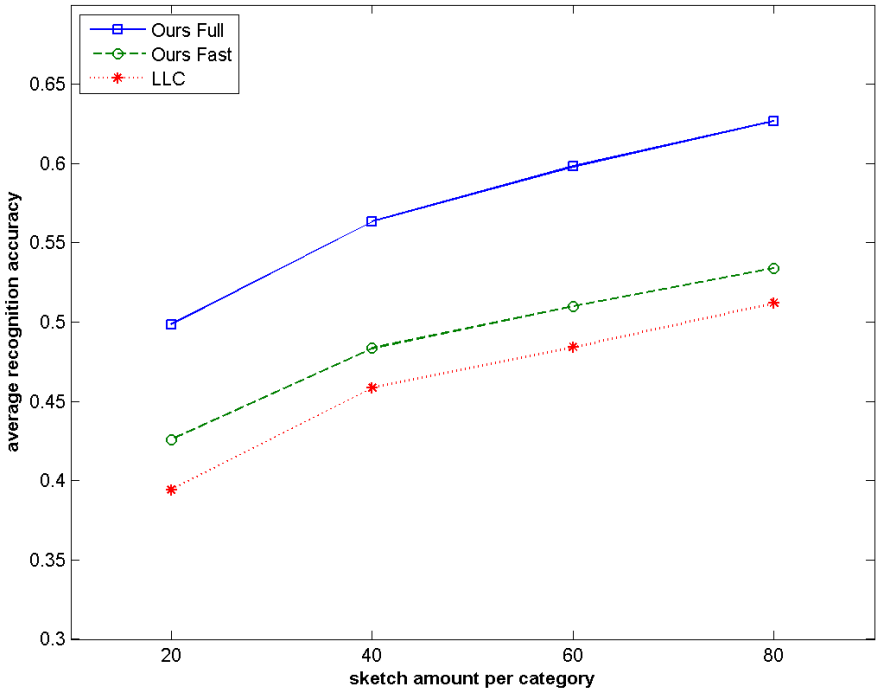


Figure 41: In the domain of sketch recognition we compare the performance of our two obtained pipelines (full and fast), specialized for this task against the foreign domain approach LLC from the image domain. Four different dataset sizes (sketch amount per category) are used, 4-fold cross validation is performed and the average scores are reported.

Conclusion and Findings

In this section we demonstrated the difficulty of transferring an approach specialized for a specific task to another domain, e.g. from sketches to nat-

ural images. We argued that additional modifications are needed to achieve similar performance compared to highly optimized approaches and learned that this is even true if methods from this related domain are integrated. However, we showed that a transfer without further modifications can lead to reasonable but not competitive performance.

5.5 In-Deep Analysis

In this section we discuss the recognition results achieved with our novel *full* algorithm in more detail. We investigate sketch categories which are recognized well with our algorithm as well as categories which lead to low recognition scores. Reasons for the achieved scores are discussed and visual exemplary results are presented. Additionally, a brief comparison between human and computational sketch recognition is given for interesting categories with the results taken from human recognition experiment performed in [5]. Further, statistics of the recognition results are discussed. As a final experiment we investigate the predictions of the applied classifier. We do not only consider the most likely classification result but take a closer look at the k most likely categories. The impact on the recognition performance, if we declare that a correct prediction is made if the ground truth category can be found within these k most likely classifier results, is discussed.

All of the following computational sketch recognition results are obtained by using the best performing iteration from the experiment discussed in Section 5.2 for our *full* pipeline. In this iteration the full sketch dataset with 80 sketches per category are used to obtain the visual dictionary and the sketch classifier. However, as 4-fold cross-validation is performed, only 60 sketches per category are involved in the learning procedure. In further consequence we also take the training/testing dataset split from this iteration. Therefore, for each category, 20 sketch images are used to obtain the recognition result. With this configuration an overall recognition accuracy of 62.92% is achieved. This means that from the 5 000 test sketches of the 250 categories, exactly 3 146 are recognized correctly. In contrast, for the human recognition experiment performed on this dataset, a recognition score of 73% was achieved for all 20 000 sketches [5].

In Figure 42 and 43, 10 of the best performing categories are depicted with exemplary sketches and the corresponding recognition accuracies. Overall 14 categories achieve an accuracy of at least 95%. This score means that one of the 20 sketches is misclassified. Examples for such misclassifications are provided in Figure 43. The complete list of the 14 best performing categories

can be found in Table 3. One might argue that the recognition of categories such as "envelope" or "sun" is trivial, as in general only one meaningful way to draw such sketches exists, that is in a frontal view. However, drawing skills or the thinking of a typical depiction varies as we can see in Figure 42 and 43. Nevertheless, our recognition algorithm is capable of handling such variations. We further want to point out that also challenging categories such as "pear", "sponge bob" or "camel" are present in the list of best performing sketch classes. We argue that the "pear" category is difficult as it could be easily confused with a sketch of an "apple" or a "lightbulb". Note that these types of misclassifications occurred in the human recognition experiment on this dataset performed in [5]. We further argue that the two categories "sponge bob" and "camel" are difficult for various reasons. First, there are multiple meaningful poses to draw the sketches. In addition, a variance in drawing skills and level of details are available as can be seen in Figure 42 and Figure 43. Still these two categories are among the best performing categories. Although the presented best categories also achieved good scores in the human recognition experiment, some categories seem to be more difficult for humans. Examples are the categories "skull" (68.75%) (often classified as "head") and "hour glass" (73.75%) (most misclassifications with "wineglass").

Category	**Score**
pear	100%
t-shirt	100%
hourglass	100%
envelope	100%
sponge bob	100%
sun	100%
pineapple	95%
skull	95%
hamburger	95%
camel	95%
hand	95%
ladder	95%
pumpkin	95%
wine-bottle	95%

Table 3: Categories with a recognition accuracy of at least 95%.

We now turn our attention to categories with a lower recognition perfor-

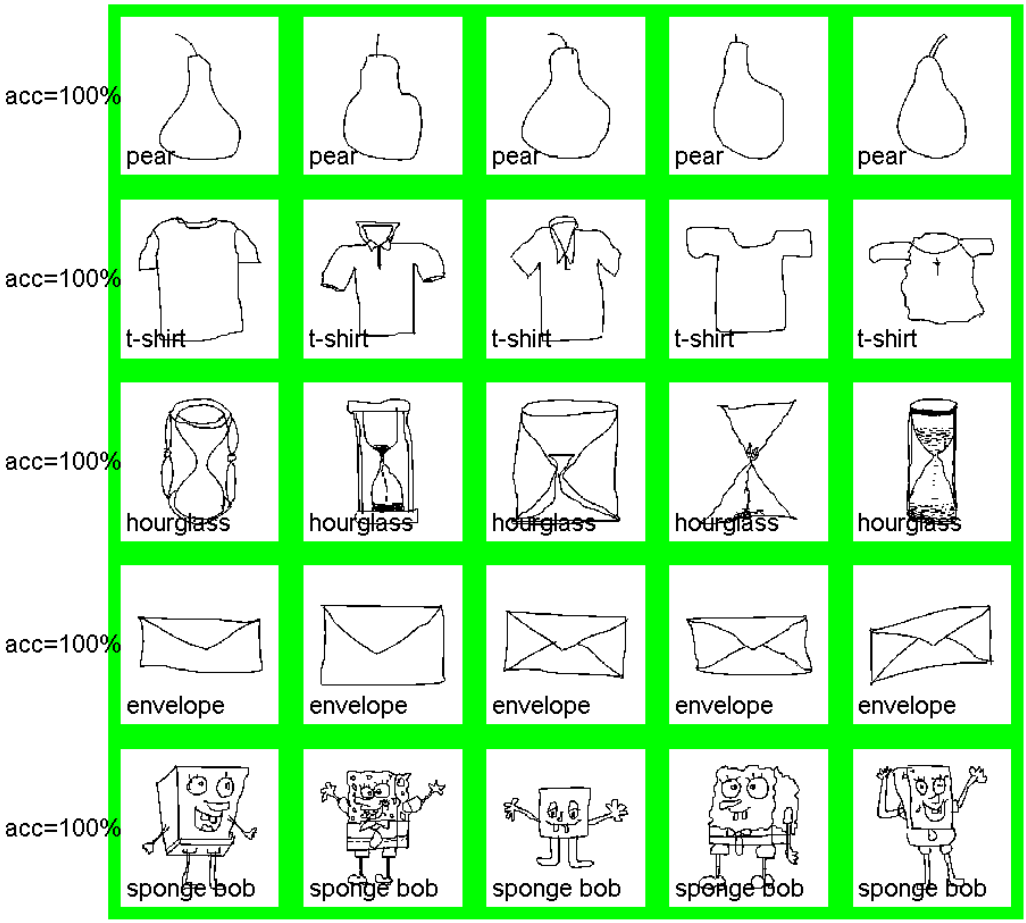


Figure 42: Example sketches for five of the best performing sketch categories for our best performing experiment configuration using our novel recognition algorithm (full). Per sketch class five exemplar sketches are depicted. For each category the recognition accuracy is shown on the left side of the figure.

mance. For 7 sketch categories a recognition accuracy smaller than or equal to 20% is achieved. For our 20 test sketches per category, an accuracy of 20% means that 4 times the correct class is predicted. Example sketches with both positive and negative recognition results for the 7 categories are shown in Figure 44. From the first category, "loudspeaker" and its misclassified sketches, we already recognize that the predicted categories (e.g. "megaphone") are often closely related in terms of their semantic meaning. In this example both objects render/amplify music or voices. Similar "semantic" misclassifications can be found e.g. for the "flying bird", "seagull" and the "race car" sketch categories. In the human recognition experiment [5] sim-

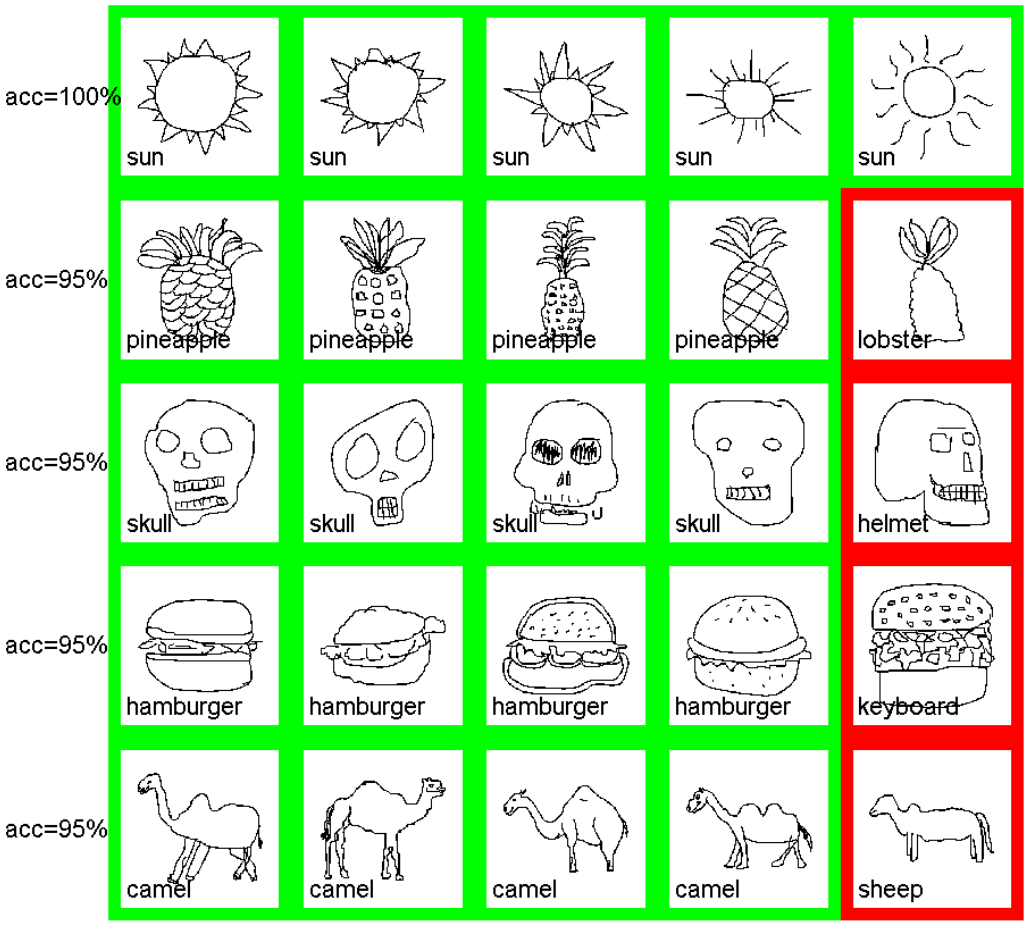


Figure 43: Example sketches for further five of the best performing sketch categories for our best performing experiment configuration using our novel recognition algorithm (full). Per sketch class five exemplar sketches are depicted. For recognition accuracies below 100%, the misclassified sketch is marked with a red frame and additionally, the falsely predicted category is displayed. For each category, the recognition accuracy is shown on the left side of the figure.

ilar problems occurred. Therefore, we argue that certain categories in the dataset are too related to each other from a semantic point of view. Also other misclassifications seem reasonable due to similar visual representations (e.g. a "megaphone" sketch classified as "power outlet" or a "door handle" sketch predicted as "door"). In the human experiment results [5], this can also be observed as a reason for poor recognition scores. On the other hand also bad recognitions such as e.g. "loudspeaker" as "spider" or "flying bird"

as "speed-boat" are available in our obtained recognition results. We argue that such misclassifications are among other factors caused by variances in drawing skills, levels of detail and poses.

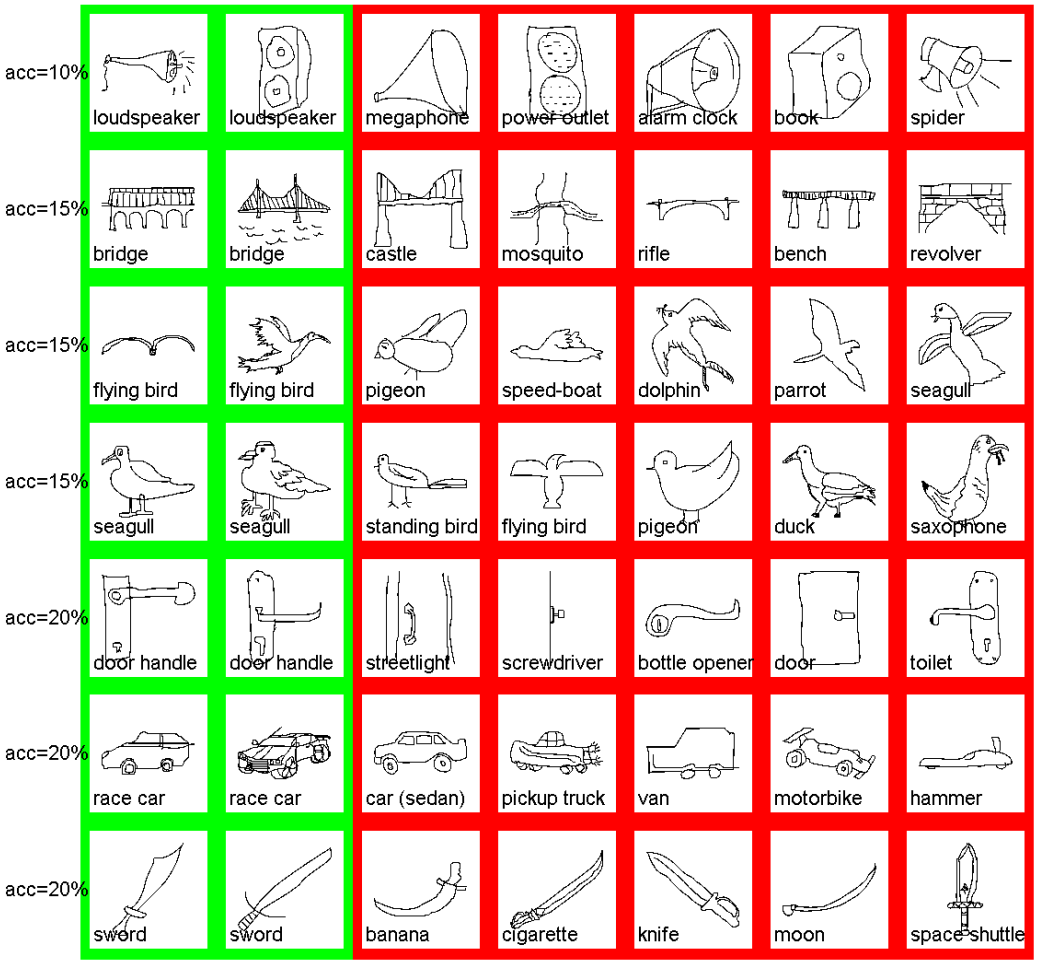


Figure 44: The seven worst performing sketch categories for our best performing experiment configuration using our novel recognition algorithm. For each sketch category seven exemplar sketches are shown. The first two sketches are predicted correctly (marked with a green border). For the remaining five sketches the classifier returns false categories (red border). We select one sketch per wrong category and order them according to their misclassification occurrence. For all sketches the predicted class is given and for all categories the recognition rates are given on the left side of the figure.

Some additional statistics are shown in Table 4. At this point we want to

mention that the probability for a correct random classification is 0.4%, as the dataset consists of 250 categories. As we saw before there are only four categories with a score below 20%. For an amount of 194 categories we achieve a recognition score of at least 50%. The largest branch takes place if we increase the threshold from 60 to 70%. This is reasonable as our overall recognition accuracy is 62.92%. That means, that most categories achieve a score in this area. For six categories all sketches are recognized correctly as already discussed.

Accuracy $\geq$	**Amount of Categories**	**Percentage**
20%	246	98.4%
30%	228	91.2%
40%	219	87.6%
50%	194	77.6%
60%	153	61.2%
70%	91	36.4%
80%	68	27.2%
90%	31	12.4%
100%	6	2.4%

Table 4: The amount as well as the percentage of categories which achieve an accuracy equal or above a certain threshold for our best performing experiment configuration using our novel recognition algorithm (full).

At the end of this section we make an in-deep analysis of the obtained classifiers predictions. As we use the one-versus-all strategy for our linear SVMs, for each of the 250 categories, a likelihood is obtained. These responses state how likely it is that a given sketch representation belongs to certain categories. Obviously, the category of the largest response it taken as the predicted class. However, in this experiment we observe the impact if we not only consider the highest but the k highest SVM responses. This means that we declare a classification to be correct if the ground truth category is found within the k most likely classes. The impact of this strategy for different k values can be seen in Figure 45. We already see that a large improvement is achieved even for small k values, whereas for $k = 1$, 62.92% or 3 146 of the 5 000 test sketches are recognized correctly, this score increases to 73.28% (3 364 sketches) for a $k = 2$ (see Figure 45(b)). Note that the human recognition result for this dataset is 73% [5]. With a k equal to 12, the accuracy rate is 90.14% (4 507). After that the performance increase becomes slower. With a $k = 112$, the accuracy score is above 99% and finally achieves 100%

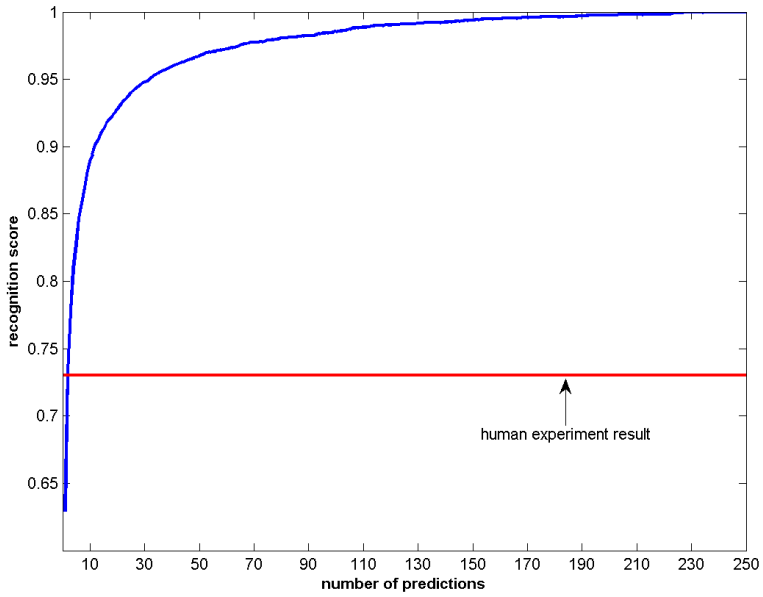
for $k = 242$. We therefore see that although wrong predictions are made, the correct categories are often within the best classifier responses.

In initial observations of the classifier responses for wrong predictions, we recognized that these scores are by far lower than for correct recognitions. We therefore argue that from these response values, in many cases, we can realize if the resulting prediction is correct or not. This could be a fruitful observation for future research in order to apply adapted classifiers in such cases. We discuss the idea of additional specialized classifier in more detail in Section 6.

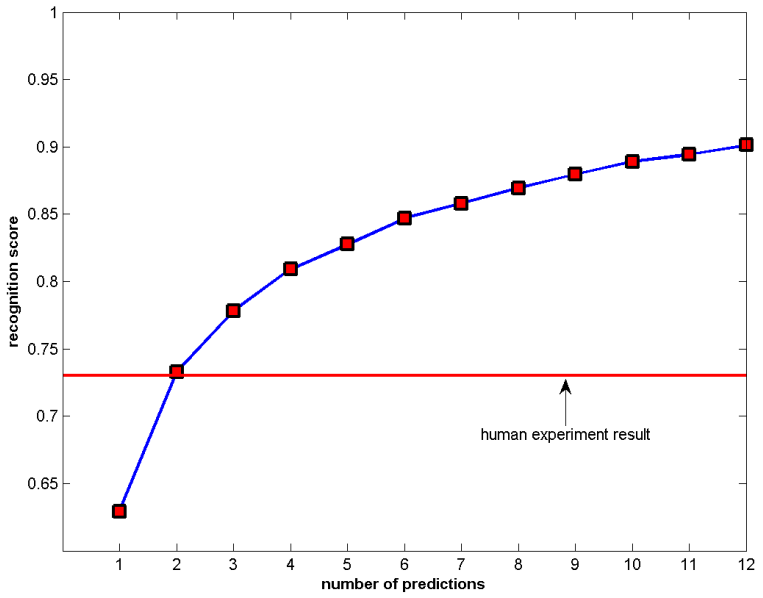
Conclusion and Findings
In this section we showed exemplar recognition results for the best as well as the worst sketch categories using our novel recognition algorithm ($full$). Further, we discussed these categories and pointed out possible reasons for misclassifications. A brief comparison between our obtained computational recognition results and human results from the experiment of Eitz et al. [5] was done for selected categories. We further showed additional statistics for the recognition results. Finally, we investigated the impact of considering the k most promising categories. We defined a correct recognition result if the true category was within these k predictions. For example with $k = 2$ the classification accuracy increased by 10% from 63% (3 146 test sketches) to 73% (3 364), which was the same score achieved for the human recognition experiment. With a k of 12, this rate even became 90% (or 4 507 sketches). We therefore argued that even for misclassifications, the correct category could often be found within a small number of most likely predictions.

5.6 Experiment Conclusion

As a first experiment we compared our two obtained pipelines $full$ and $fast$ as well as our evaluation baseline against each other. We observed that our extensive evaluation phase (discussed in Section 4) led to a remarkable performance improvement of 22% for our $full$ and 10% for the $fast$ algorithm, when compared to the evaluation baseline. Therefore, recognition scores of 63% ($full$) and 53% ($fast$) could be achieved when the full dataset was used. Further, runtime issues were discussed for the three pipelines. When using the $fast$ pipeline, a novel sketch was classified after 0.6 seconds and the whole learning procedure with 15 000 sketches involved, took 2.5 hours. This highlighted the efficiency of the $fast$ pipeline. For the $full$ algorithm, the classification result was obtained within a second. Further, the learning procedure needed 4.4 hours. Although we argued that this is sufficient for most scenarios, we investigated the timings in more detail. Two run-



(a) $k = \{1, \ldots, 250\}$



(b) zoom for $k = \{1, \ldots, 12\}$

Figure 45: Experiment regarding the k highest classifier responses. We define a correct recognition if the true category is found within these k predictions. The impact of increasingly higher k-values is shown. Additionally, the human recognition experiment result of [5] is depicted.

time bottlenecks were detected: the descriptor extraction and the encoding step. Additionally, we investigated the question if we could develop slightly adapted pipeline versions with a similar recognition performance as the $full$ algorithm which is as efficient as the $fast$ approach.

Next we compared our two pipelines $full$ and $fast$ against two related approaches ([5, 46]). As both related approaches were designed for kernel based SVMs we also investigated the impact of our sketch representations used with this classifier type. We observed that both of our pipelines worked similar for the two classifier types. Further, we were able to achieve state-of-the-art performance on the TU Berlin sketch dataset (+1% compared to [46]) with linear SVMs. We then showed that both of our pipelines outperformed the experimental runs which used the "Eitz sketch representations".

Further, the performance of our sketch pipeline in the related domain of image recognition was investigated. Although most elements of the algorithm were taken from the image domain, our pipeline is far too specialized for sketches. As a result of this, our unmodified sketch pipeline was clearly outperformed by a state-of-the-art image recognition approach (LLC [32]). The converse experiment was performed for this approach in the sketch domain which confirmed this argument. Both of our pipelines $full$ and $fast$ were able to outperform this image-domain approach. Therefore, we argued that it is difficult to create a direct cross-domain approach. However, for our $full$ approach we achieved a reasonable performance in the image domain.

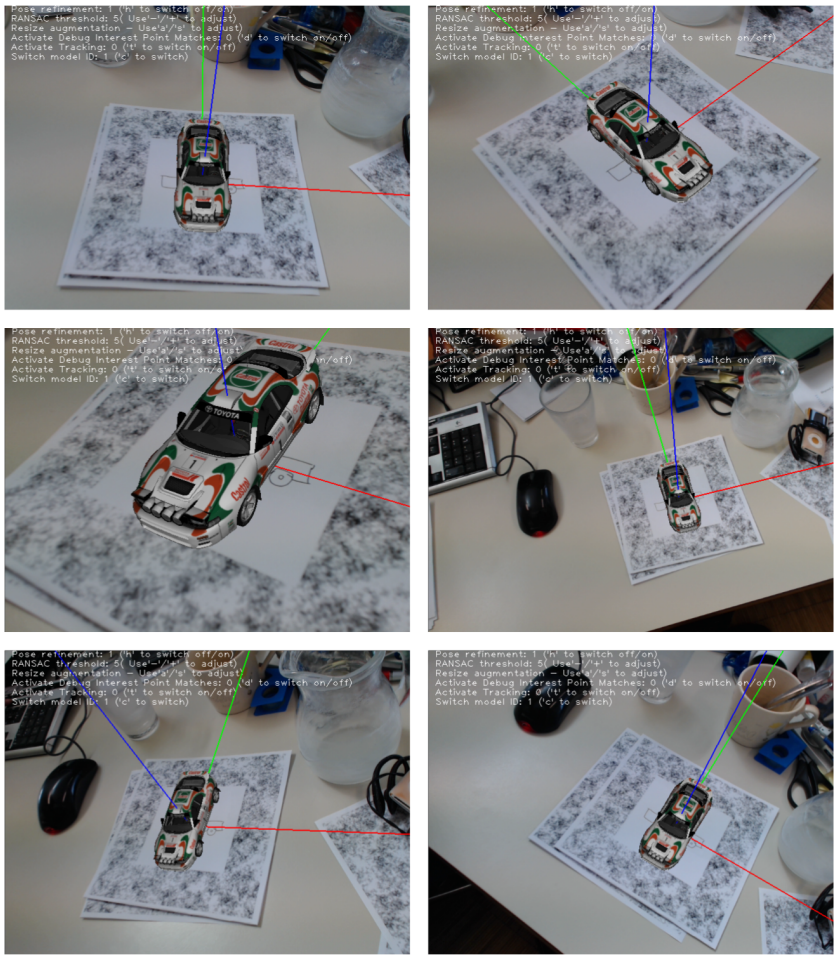
Finally, we presented example recognition results. This was done both for best as well as for worst performing categories (according to their recognition scores). For both types, the results were discussed and investigated. Reasons for certain misclassifications were given. Further, we briefly discussed the performance of the human recognition experiment on the same dataset [5] for selected categories. Next, we presented overall statistics for the 250 categories and finally investigated the impact of considering the k most promising predictions for the recognition result. We showed that even for small k values (e.g. two) a remarkable performance gain could be achieved. This pointed out the suitability of our sketch representations for the chosen classifier and that even for misclassifications the correct category could often be found within the best predictions.

To demonstrate that our introduced state-of-the-art sketch recognition algorithm is suitable for real-life scenarios, we include it into an Augmented Reality application which is briefly discussed in the following subsection.

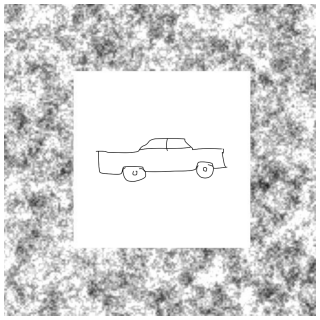
5.7 Augmented Reality Application

Augmented Reality (AR) means that the real world is superimposed by virtual (computer-generated) objects that appear to coexist in the same space as the real word [90]. In this application we track an online drawn sketch, recognize the category, load category from 3D model repository and augment a corresponding 3D model in a live video stream. The sketch to-be-recognized is drawn on a piece of paper in a defined area surrounded by marker symbols to simplify the tracking procedure. To be able to augment a 3D model in a live video stream we need the six Degrees of Freedom (DoF) pose which is extracted by re-identifying the markers. By pressing a button, the obtained image is rectified, pre-processed and provided to our $full$ recognition algorithm. With the recognition result (the sketch category), the corresponding 3D model is loaded from a dataset and augmented in the live video stream. This is done by using the estimated pose from the tracking framework to adapt the viewpoint of the 3D model. In Figure 46 screenshots of our developed application as well as the sketch used can be seen.

For testing, we selected a few categories, that have shown to work well. The application demonstrates that the classification runtime is sufficient to provide the user immediate visual feedback in terms of the corresponding 3D model.



(a) application screenshots



(b) drawn sketch

Figure 46: Screenshots of our AR demo application and the drawn sketch used.

6 Conclusion

In this master thesis we designed two novel sketch recognition algorithms based on the Bag-of-Visual-Words (BoVW) method. In contrast to many related approaches in the field of sketch recognition, our approach is domain-independent. Hence, we did not restrict ourselves to one specific field (e.g. primitives or mathematical formulas). Existing approaches without this constraint are either computationally expensive or only achieve a limited recognition accuracy. In contrast, our algorithm was designed to provide a high recognition rate and to be efficient enough to be used in applications that require immediate recognition feedback. Our first algorithm ($full$) achieves state-of-the-art performance on a large-scale sketch dataset with a reasonable amount of training (4.4 hours) and classification time (less than a second). The second algorithm was designed to be more efficient in both involved phases (2.5 hours/0.6 seconds). However this comes at the cost of a reduced recognition performance which is still competitive to or even outperforms related approaches.

We discussed the relation between sketch and image recognition. Based on this discussion, we reasoned that well studied image recognition methods can be successfully integrated in our BoVW-based sketch domain approach. Therefore, such methods were investigated for each pipeline element commonly used by BoVW approaches. In order to select suitable variants per step, an extensive evaluation was performed. The corresponding results were directly incorporated in our two final sketch recognition algorithms. To the best of knowledge, we are the first to apply this kind of strategy in such large scale, investigating all steps involved in a BoVW-based pipeline. We made important and interesting observations for each pipeline element concerning recognition performance and efficiency. To sum up, for all pipeline elements but image pre-processing we were able to improve the recognition performance compared to our baseline significantly. For descriptor extraction, dense grid sampling and the usage of all available descriptors is important. Further, a local gradient-based descriptor for mid-sized patches is necessary. A powerful encoding with a coupled dictionary learning method is the most suitable choice for these two pipeline elements. Another essential component is the usage of the spatial pyramid framework for code pooling. We observed that also the choice of a primitive, still suitable, classifier can boost the recognition performance. As runtime bottlenecks the descriptor extraction and encoding step were identified. However, we presented alternative encoding methods to solve this bottleneck at similar performance. The remaining steps of the pipeline could be performed highly efficiently. Note that an efficient classifier is needed to be able to process the high di-

mensional sketch representations at this speed. This gained knowledge can further be used to optimize existing recognition approaches. We further argue that this information can also be useful for BoVW-based approaches in additional computer vision domains as e.g. retrieval or detection.

Future Work
Although we performed an extensive evaluation for all pipeline elements, obviously we were not able to test all possible algorithms. Additional approaches and methods could be included in the evaluation. Further, we utilized a step-by-step strategy. This means that the pipeline elements were investigated one after another. Therefore, the overall impact of certain pipelines configurations involving multiple elements was ignored to keep the evaluation in a feasible scope. However, with the learned insights from our evaluation phase, additional strategies can be developed to investigate cross-element relations. In our opinion, the most promising elements for such an analysis are: descriptor extraction, the coupled dictionary learning and encoding step, the pooling method using a spatial pyramid and the choice of the classifier type, as the highest performance gains were achieved for these elements in evaluations. Further, the lessons learned build a solid knowledge base for the future developments in the field of sketch and image recognition. Additionally, it could be used to improve existing image recognition approaches.

The applicability of two recently proposed and promising image recognition methods, in the sketch domain could be an interesting future research direction.

Boureau et al. [76] proposed a supervised dictionary learning algorithm. The basic idea is, that the dictionary should be optimized for the classification task which is not achieved by an unsupervised learning method. As the algorithm is coupled with sparse encoding it could be easily integrated in our sketch recognition algorithm. Further, macro-features are introduced. This is a set of descriptors in a specific neighborhood which is encoded together. Boureau et al. showed that the integration of these ideas improves the image recognition performance. Therefore we argue that this can also be true for our sketch recognition algorithm.

The approach of Jia et al. [91] introduced a novel method in the context of code pooling. Instead of using manually designed spatial regions, this approach learns more adaptive receptive fields. Starting from a large set of possible rectangular spatial areas, the introduced algorithm selects most promising field candidates. Also for the discrimination of certain sketch categories, different spatial areas are more important (e.g. for a sketch of the

class "standing bird", the leg area is important in order to distinguish from a "flying bird" where in general no legs are drawn). Although it could be difficult to apply this approach to such a large scale problem, this could be an interesting and promising research area.

To tackle the poor classification results for visually similar categories, an extended recognition strategy could be introduced. One way could be to search for semantic relations within the k most likely predictions (keyword: "bird" categories). If such a relation is found, additionally learned classifiers for such closely-related categories could be applied (e.g. a special "bird" classifier). This task is known as fine-grained categorization in the computer vision literature. A promising approach is e.g. [92], which classifies different cat and dog breeds.

As a final interesting field for sketch recognition, we identify what we call *incremental recognition*. This means to integrate the temporal information about how the sketches are drawn, in the recognition algorithm. This knowledge is e.g. available for the TU Berlin sketch dataset [5] and could in our opinion lead to a recognition performance boost. One possible method could be to investigate how the sketch representation changes as new strokes are drawn. In [5] it is observed, that a typical way of how sketches are drawn, exists, using a coarse-to-fine strategy. Due to that fact, this could further lead to a recognition approach which is also able to even predict the correct category of unfinished sketches in an online manner (keyword: augmented reality). We believe that this could be a fruitful research area in future development of sketch recognition approaches.

References

[1] Flight safety instructions art. http://tinalee.info/?p=2398, 2011. Accessed: 2013-12-11. 2

[2] Piktogramme auf unseren Straßen. http://www.koelndesign.de/de/referenzprojekte/dsp/piktogramme-auf-unseren-strassen. Accessed: 2013-12-11. 2

[3] Jean Clottes. *Chauvet Cave: The Art of Earliest Times*. University of Utah Press, 2003. 1

[4] Rhino art chauvet cave. http://4c-studio.com/2012/04/24/rhino-art-chauvet-cave/, 2012. Accessed: 2013-09-11. 3

[5] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? In *Proceedings of the Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH)*, 2012. 4, 6, 7, 18, 19, 20, 21, 25, 27, 29, 31, 32, 37, 43, 44, 49, 50, 51, 52, 55, 60, 73, 81, 82, 89, 92, 96, 97, 98, 99, 101, 102, 103, 104, 109

[6] Mark Everingham, Luc Van Gool, Chris Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 5, 7

[7] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007. 8, 50

[8] Josef Sivic and Andrew Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2003. 11, 12, 23, 31, 36, 43

[9] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999. 11

[10] Jiri Matas, Ondrej Chum, Martin Urban, and Tomas Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2004. 11

[11] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004. 11, 13, 19, 26, 28, 29

[12] Stuart Lloyd. Least squares quantization in pcm. *Transactions on Information Theory*, 28(2):129–137, 1982. 11, 31

[13] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Proceedings of the Workshop on Statistical Learning in Computer Vision (ECCV)*, 2004. 12, 19

[14] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2002. 12

[15] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 13, 14, 15, 29, 31, 36, 43, 44, 73

[16] David Lowe. Towards a computational model for object recognition in it cortex. In *Proceedings of the Workshop on Biologically Motivated Computer Vision (BMVC)*, 2000. 13

[17] Antonio Torralba, Kevin Murphy, William Freeman, and Mark Rubin. Context-based vision system for place and object recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2003. 13

[18] Martin Szummer and Rosalind Picard. Indoor-outdoor image classification. In *Proceedings of the Workshop on Content-based Access of Image and Video Databases*, 1998. 13

[19] Anna Bosch, Andrew Zisserman, and Xavier Muñoz. Image classification using random forests and ferns. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2007. 13, 29, 47

[20] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 13, 14, 29, 33, 37, 39, 43, 73, 75

[21] Subhransu Maji, Alexander Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 13

[22] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Transactions on Image Processing*, 15(12):3736–3745, 2006. 14

[23] Julien Mairal, Michael Elad, and Guillermo Sapiro. Sparse representation for color image restoration. *Transactions on Image Processing*, 17(1):53–69, 2008. 14

[24] Jianchao Yang, John Wright, Yi Ma, and Thomas Huang. Image super-resolution as sparse representation of raw image patches. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 14

[25] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Discriminative learned dictionaries for local image analysis. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 14

[26] Xiaofeng Ren and Deva Ramanan. Histograms of sparse codes for object detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 14

[27] Marc'Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 14

[28] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2007. 14

[29] Frank Moosmann, Bill Triggs, and Frederic Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *Proceedings of the Conference on Neural Information Processing Systems*, 2006. 14

[30] Liu Yang, Rong Jin, Rahul Sukthankar, and Frederic Jurie. Unifying discriminative visual codebook generation with classifier training for object category recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 14

[31] Frederic Jurie and Bill Triggs. Creating efficient codebooks for visual recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2005. 14

[32] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 14, 29, 33, 39, 40, 43, 51, 73, 82, 83, 93, 94, 104

[33] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 14, 15, 35, 41

[34] Karen Simonyan, Omkar Parkhi, Andrea Vedaldi, and Andrew Zisserman. Fisher vector faces in the wild. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2013. 15, 41

[35] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010. 15, 35, 41, 42, 44

[36] Brandon Paulson and Tracy Hammond. Paleosketch: Accurate primitive sketch recognition and beautification. In *Proceedings of the International Conference on Intelligent User interfaces*, 2008. 15, 16, 18

[37] Joseph LaViola Jr. and Robert Zeleznik. Mathpad^2: A system for the creation and exploration of mathematical sketches. In *Proceedings of the Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH)*, 2004. 16, 17

[38] Tom Ouyang and Randall Davis. Chemink: A natural real-time recognition system for chemical drawings. In *Proceedings of the International Conference on Intelligent User interfaces*, 2011. 17

[39] Tevfik Sezgin and Randall Davis. Sketch recognition in interspersed drawings using time-based graphical models. *Computers & Graphics*, 32(5):500–510, 2008. 17

[40] Tracy Hammond, Drew Logsdon, Joshua Peschel, Joshua Johnston, Paul Taele, Aaron Wolin, and Brandon Paulson. A sketch recognition interface that recognizes hundreds of shapes in course-of-action diagrams. In *Proceedings of the Conference on Human Factors in Computing Systems*, 2010. 18

[41] Zhenbang Sun, Changhu Wang, Liqing Zhang, and Lei Zhang. Query-adaptive shape topic mining for hand-drawn sketch recognition. In *Proceedings of the International Conference on Multimedia*, 2012. 18

[42] Yang Cao, Changhu Wang, Liqing Zhang, and Lei Zhang. Edgel index for large-scale sketch-based image search. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 18

[43] Mathias Eitz. *Human Object Sketches: Datasets, Descriptors, Computational Recognition and 3d Shape Retrieval*. PhD thesis, Technische Universität Berlin, 2012. 19

[44] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 19, 37

[45] Jan van Gemert, Jan-Mark Geusebroek, Cor Veenman, and Arnold Smeulders. Kernel codebooks for scene categorization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008. 19, 37, 69

[46] Yi Li, Yi-Zhe Song, and Shaogang Gong. Sketch recognition by ensemble matching of structured features. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2013. 19, 27, 29, 30, 60, 61, 81, 82, 89, 91, 92, 104

[47] Hayko Riemenschneider, Michael Donoser, and Horst Bischof. Image retrieval by shape-focused sketching of objects. In *Proceedings of the Computer Vision Winter Workshop*, 2011. 19, 20, 21

[48] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 20, 21

[49] Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, and Marc Alexa. Sketch-based shape retrieval. In *Proceedings of the Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH)*, 2012. 20

[50] Ma Chao, Yang Xiaokang, Zhang Chongyang, Ruan Xiang, and Yang Ming-Hsuan. Sketch retrieval via dense stroke features. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2013. 20, 21, 55, 58

[51] Serge Belongie and Jitendra Malik. Matching with shape contexts. In *Proceedings of the Workshop on Content-based Access of Image and Video Libraries (CBAIVL)*, 2000. 27, 29, 30, 51, 60

[52] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 28, 29, 60

[53] Anna Bosch, Andrew Zisserman, and Xavier Muñoz. Scene classification via plsa. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2006. 28, 29, 60

[54] Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *Transactions on Visualization and Computer Graphics*, 17(11):1624–1636, 2011. 29, 30, 60

[55] Eli Shechtman and Michal Irani. Matching local self-similarities across images and videos. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 30, 60

[56] Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 30, 31, 42, 60, 63, 78

[57] Adam Coates and Andrew Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2011. 31, 39, 66, 69

[58] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Symposium on Discrete Algorithms*, 2007. 32

[59] Jeffrey Beis and David Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 1997. 32

[60] Marius Muja and David Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, 2009. 32

[61] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 32, 33, 37

[62] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: Design of dictionaries for sparse representation. In *Proceedings of the Conference on Signal Processing with Adaptive Sparse Structured Representations (SPARS)*, 2005. 33

[63] Yagyensh Pati, Ramin Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of the Conference on Signals, Systems and Computers*, 1993. 33, 39

[64] Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. Technical report, Technion, 2008. 33, 51

[65] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009. 34, 38, 39, 51, 66

[66] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004. 34, 38

[67] Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1):1–38, 1977. 35

[68] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Improving bag-of-features for large scale image search. *International Journal of Computer Vision (IJCV)*, 87(3):316–336, 2010. 35

[69] Chanop Silpa-Anan and Richard Hartley. Optimised kd-trees for fast image descriptor matching. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 36

[70] Julien Mairal, Marius Leordeanu, Francis Bach, Martial Hebert, and Jean Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008. 38, 39

[71] Michael Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing.* Springer Publishing Company, 1st edition, 2010. 38

[72] David Donoho and Yaakov Tsaig. Fast solution of l1-norm minimization problems when the solution may be sparse. *Transactions on Information Theory*, 54(11):4789–4812, 2008. 39

[73] Kai Yu, Tong Zhang, and Yihong Gong. Nonlinear learning using local coordinate coding. In *Proceedings of the Conference on Neural Information Processing Systems*, 2009. 40

[74] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 40, 41, 69

[75] Relja Arandjelovic and Andrew Zisserman. All about vlad. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 40, 41

[76] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 43, 73, 108

[77] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 47

[78] Juergen Gall, Nima Razavi, and Luc Van Gool. An introduction to random forests for multi-class object detection. In *Proceedings of the International Conference on Theoretical Foundations of Computer Vision*, 2011. 47

[79] Thomas Dean, Mark Ruzon, Mark Segal, Jonathon Shlens, Sudheendra Vijayanarasimhan, and Jay Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 50

[80] Bryan Russell, Antonio Torralba, Kevin Murphy, and William Freeman. Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision (IJCV)*, 77(1):157–173, 2008. 50

[81] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Proceedings of Shape Modeling International*, 2004. 50

[82] Andrea Vedaldi and Brian Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008. 51

[83] Ken Chatfield, James Philbin, and Andrew Zisserman. Efficient retrieval of deformable shape classes using local self-similarities. In *Proceedings of the Workshop on Non-rigid Shape Analysis and Deformable Image Alignment (ICCV)*, 2009. 51, 60

[84] Ken Chatfield, Victor Lempitsky, Andrea Vedaldi, and Andrew Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2011. 51

[85] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. 51

[86] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. 51

[87] Abhishek Jaiantilal. Random forest (regression, classification and clustering) implementation for matlab (and standalone). `http://code.google.com/p/randomforest-matlab/`, 2012. 51

[88] Charles Elkan. Using the triangle inequality to accelerate k-means. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2003. 53, 66

[89] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Proceedings of the Workshop on Generative Model Based Vision (CVPR)*, 2004. 82, 83, 92

[90] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent advances in augmented reality. *Computer Graphics and Applications*, 21(6):34–47, 2001. 105

[91] Yangqing Jia, Chang Huang, and Trevor Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 108

[92] Omkar Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 109