

# Human Object Sketches: Datasets, Descriptors, Computational Recognition and 3d Shape Retrieval

vorgelegt von  
Mathias Eitz, Dipl.-Inf., M.Eng.  
aus Friedrichshafen

von der Fakultät IV - Elektrotechnik und Informatik  
der Technischen Universität Berlin  
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften  
– Dr.-Ing. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. Oliver Brock

Gutachter: Prof. Dr. Marc Alexa

Gutachter: Prof. Tamy Boubekeur, PhD

Tag der wissenschaftlichen Aussprache: 07.12.2012

Berlin 2012  
D83



---

# Abstract

---

Since prehistoric times, humans have used sketching to depict our visual world. Even today, sketching is possibly the only rendering technique readily available to all humans. To understand how humans sketch objects, we perform two experiments. In the first experiment, we analyze the distribution of non-expert sketches of everyday objects such as ‘teapot’ or ‘car’. We ask participants to sketch objects of a given category and gather 20,000 unique sketches evenly distributed over 250 object categories. The second experiment targets 3d shape retrieval, and we gather 1,814 sketches that are related to the categories in an existing dataset of 3d shapes. The sketches in both datasets turn out to be generally quite abstract with large local and global deviations from the original shape. Based on the first sketch dataset, we perform a perceptual study and find that humans can correctly identify the object category of a sketch 73% of the time.

We develop a targeted feature transform for sketches that is based on a bag-of-features approach, yields a compact representation and comes with suitable invariance properties. Using this representation, we develop the first computational recognition method for classifying human object sketches. We compare human performance against the computational model for which we use multi-class support vector machines, trained on the sketch dataset, to classify sketches. The resulting recognition method is able to identify unknown sketches with 56% accuracy (chance is 0.4%). Using the computational model, we demonstrate an interactive sketch recognition system.

Based on the second dataset, we develop a system for 3d object retrieval using sketched feature lines as input. The system employs a view-based approach, matching the input against computer generated line drawings of the objects, using the bag-of-features representation developed earlier. Moreover, we demonstrate how to optimize the parameters of our, as well as other approaches, based on the gathered sketches. In the resulting comparison, we show objectively that our approach performs significantly better than any other system described so far.



---

# Zusammenfassung

---

Seit prähistorischen Zeiten verwenden Menschen Skizzen, um ihre visuelle Welt zeichnerisch abzubilden. Auch heutzutage ist Skizzieren noch immer die einzige Darstellungsmethode, die allen Menschen einfach zugänglich ist. Um herauszufinden, wie Menschen Objekte skizzieren, führen wir zwei Experimente durch. Im ersten Experiment analysieren wir Skizzen alltäglicher Objekte, wie “Teekanne” oder “Auto”. Hierzu bitten wir die Teilnehmer einer Studie solche Objekte zu skizzieren. Insgesamt erhalten wir so einen Datensatz von 20.000 Skizzen, gleichmäßig verteilt auf 250 Kategorien. Das zweite Experiment befasst sich mit der skizzenbasierten Suche von dreidimensionalen Modellen. Dazu sammeln wir 1.814 Skizzen von den Teilnehmern einer weiteren Studie. Die Kategorien dieser Skizzen stammen aus einem existierenden Datensatz dreidimensionaler Modelle. Es zeigt sich, dass die Skizzen in beiden Datensätzen relativ abstrakt gezeichnet sind und dabei mehrheitlich stark von der Geometrie des echten Objektes abweichen. Anhand des ersten Datensatzes führen wir eine Wahrnehmungs-Studie durch und stellen fest, dass Menschen die Kategorie einer Skizze in 73% der Fälle korrekt erkennen können.

Im Folgenden entwickeln wir spezielle Repräsentationen für Skizzen, die auf dem sogenannten “Bag of Features”-Ansatz aufbauen, wenig Speicherplatz benötigen und mit günstigen Invarianzeigenschaften ausgestattet sind. Aufbauend auf dieser Repräsentation entwickeln wir Algorithmen zur maschinellen Erkennung von Skizzen. Wir verwenden hierzu Multi-Class Support Vector Machines, die mittels des ersten Datensatzes trainiert werden. Die daraus resultierende Erkennungsmethode für Objektskizzen weist eine Genauigkeit von 56% auf (Zufall: 0,4%). Mithilfe dieses maschinellen Modells entwickeln wir ein interaktives System zur Skizzenerkennung.

Basierend auf dem zweiten Datensatz entwickeln wir ein System zur skizzenbasierten Suche von dreidimensionalen Modellen. Das System beruht auf einem ansichts-basierten Ansatz, wobei die Skizze des Nutzers mit computer-generierten Zeichnungen der 3D Modelle verglichen wird. Darüber hinaus demonstrieren wir, wie sowohl die Parameter unseres Systems, als auch die von konkurrierenden Ansätzen anhand der Skizzen aus dem zweiten Datensatz optimiert werden können. In einem abschließenden Vergleich zeigen wir objektiv, dass unser Ansatz deutlich bessere Ergebnisse erzielt als andere in der Literatur bisher beschriebene Systeme.



---

# Contents

---

<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Definition: Sketch . . . . .	2
1.2 Prior Research . . . . .	2
1.3 Overview . . . . .	3
1.4 Contributions . . . . .	4
1.5 Outline . . . . .	5
1.6 Publications . . . . .	6
<b>2 Related Work</b>	<b>7</b>
2.1 History . . . . .	7
2.2 Sketch-Based Image Retrieval . . . . .	9
2.3 Sketch-Based Shape Retrieval . . . . .	11
2.4 Sketch Recognition . . . . .	12
<b>3 Sketch Datasets</b>	<b>17</b>
3.1 Experimental Setup . . . . .	18
3.2 A Large Dataset of Human Object Sketches . . . . .	18
3.3 A Large Dataset of Sketches for 3d Shape Retrieval . . . . .	20
<b>4 Human Sketch Recognition</b>	<b>25</b>
4.1 Experimental Setup . . . . .	25
4.2 Human Classification Results . . . . .	26
4.3 Discussion . . . . .	29
<b>5 Sketch Representation</b>	<b>31</b>
5.1 Overview . . . . .	32
5.2 SHOG: Local Histograms of Oriented Gradients . . . . .	33
5.3 GALIF: Gabor Local Line-Based Feature . . . . .	34
5.4 Extracting Local Features . . . . .	35
5.5 Building a Visual Vocabulary . . . . .	37
5.6 Quantizing Features . . . . .	38

5.7	Discussion . . . . .	40
<b>6</b>	<b>Computational Sketch Recognition</b>	<b>41</b>
6.1	Models . . . . .	42
6.2	Recognition Experiments . . . . .	43
6.3	Unsupervised Dataset Analysis . . . . .	48
6.4	Applications . . . . .	50
6.5	Discussion . . . . .	53
<b>7</b>	<b>Sketch-Based Shape Retrieval</b>	<b>55</b>
7.1	Overview . . . . .	56
7.2	View-Based Matching . . . . .	57
7.3	Selecting Views . . . . .	58
7.4	Line Rendering . . . . .	60
7.5	Sampling Local Features . . . . .	60
7.6	Representation . . . . .	61
7.7	Online Querying . . . . .	61
7.8	Benchmarking . . . . .	62
7.9	Optimizing Retrieval System Parameters . . . . .	63
7.10	Evaluation . . . . .	64
7.11	Learning Sketch-Based Shape Retrieval . . . . .	69
7.12	Results . . . . .	69
7.13	Discussion . . . . .	73
<b>8</b>	<b>Conclusions</b>	<b>75</b>
8.1	Future Directions . . . . .	76
<b>A</b>	<b>Parameter Space Evaluation</b>	<b>79</b>
	<b>Bibliography</b>	<b>85</b>



---

# List of Figures

---

1.1	Examples of human sketches . . . . .	1
1.2	Historic and artistic sketches . . . . .	2
2.1	Photograph of SketchPad system . . . . .	8
2.2	Sketch-based image retrieval . . . . .	9
2.3	Gesture recognition . . . . .	13
2.4	Examples of structured sketch domains . . . . .	14
3.1	Instructional examples of sketches for Mechanical Turk workers . .	18
3.2	Screenshot of interactive tool for sketch dataset verification . . . .	20
3.3	Stroke length in sketches over drawing time . . . . .	22
3.4	Distribution of sketching time per category . . . . .	22
3.5	Subsets of sketches from 3d shape retrieval benchmark . . . . .	23
4.1	Selection menu used in human sketch recognition experiment . . .	26
4.2	Per-worker sketch recognition performance . . . . .	26
4.3	Representative sketches with highest human recognition rate . . .	27
4.4	Representative sketches with lowest human recognition rate . . . .	27
5.1	Gaussian derivative filter . . . . .	33
5.2	Gabor filter . . . . .	34
5.3	GALIF feature extraction pipeline . . . . .	36
5.4	Fast histogram construction . . . . .	37
6.1	Accuracy of kNN/SVM models . . . . .	44
6.2	Training set size vs. computational classification accuracy . . . . .	46
6.3	Confusion matrix for selected categories . . . . .	47
6.4	Human vs. computational sketch recognition performance . . . . .	47
6.5	Representative sketches . . . . .	49
6.6	2d layouts of sketches using t-SNE . . . . .	51
6.7	Applications enabled by semantic sketch recognition . . . . .	52
6.8	Computational recognition of artistic/ancient sketches . . . . .	53
6.9	Temporal order of strokes . . . . .	54
7.1	Example 3d scene created using sketch-based retrieval . . . . .	55

## List of Figures

---

7.2	Examples of 3d objects that are difficult to describe . . . . .	56
7.3	Sketch-based 3d shape retrieval pipeline . . . . .	57
7.4	View generation pipeline . . . . .	58
7.5	Best-view selection . . . . .	59
7.6	Comparison of computational line rendering styles . . . . .	61
7.7	Precision/recall plots of sketch-based shape retrieval performance .	70
7.8	Examples of sketch-based shape retrieval query results . . . . .	71
A.1	Tinyimage descriptor parameter optimization . . . . .	79
A.2	Contour plots of GALIF Gabor parameters optimization . . . . .	80
A.3	Contour plots of GALIF intrinsic parameters optimization . . . . .	81
A.4	Contour plots of GALIF extrinsic parameters optimization . . . . .	82
A.5	Retrieval performance evaluation for competing descriptors . . . . .	82
A.6	SHOG Gaussian derivatives parameter optimization . . . . .	83
A.7	Contour plots of SHOG intrinsic parameters optimization . . . . .	83
A.8	Contour plots of SHOG extrinsic parameters optimization . . . . .	84

---

## List of Tables

---

3.1	250 object categories for sketch recognition experiment . . . . .	21
4.1	Hierarchy of categories for human sketch recognition experiment . .	28
6.1	Best parameters for sketch classification models . . . . .	45
6.2	Accuracy of sketch classification models . . . . .	46
7.1	Parameter search ranges for Gabor filter . . . . .	64
7.2	Parameter search ranges for Gaussian derivative filter . . . . .	65
7.3	Parameter search ranges for intrinsic descriptor parameters . . . .	65
7.4	Parameter search ranges for extrinsic system parameters. . . . .	65
7.5	Optimal parameter values for GALIF descriptor . . . . .	68
7.6	Optimal parameter values for SHOG descriptor . . . . .	68



---

## Chapter 1

# Introduction

---

Sketching is a universal form of communication. Since prehistoric times people have rendered the visual world in sketch-like petroglyphs or cave paintings to communicate visual concepts; possibly across borders of culture, country and time. For example, some of the cave paintings in Lascaux, France (see Figure 1.2) date back tens of thousands of years. To draw such sketches, ancient artists often used two colors: red (made from iron oxide) and black (made from charcoal), and they applied the paint using a brush or with a hollow tube [Chalmin et al. 2003]. Such pictographs predate the appearance of language by tens of thousands of years and today the ability to draw and recognize sketched objects is ubiquitous. In fact, recent neuroscience work suggests that simple, abstracted sketches activate our brain in similar ways to real stimuli [Walther et al. 2011].

Despite decades of graphics research, sketching is still the *only* mechanism for most people to render visual content quickly and flexibly [Landay and Myers 2001]. However, there has never been a formal study of how people sketch objects and how well such sketches can be recognized by humans and computers. We examine these topics for the first time and demonstrate applications of computational sketch understanding.

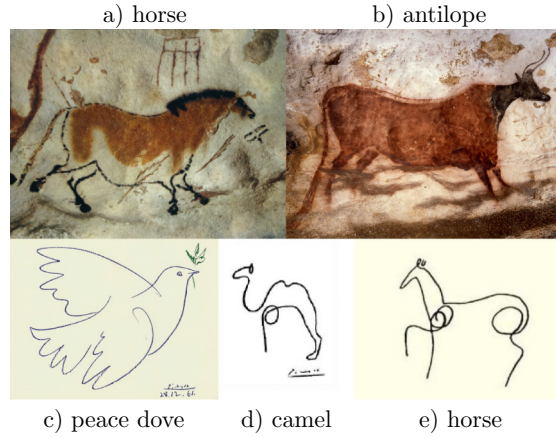
We believe that the concepts introduced in this thesis are potentially widely applicable: compared to spoken language, the visual concept encoded in a sketch appears to be more universal across cultures and time. While the sketch in Figure 1.2 is — even after tens of thousands of years — still easily recogniz-



**Figure 1.1:** *We explore how humans sketch and recognize objects from 250 categories — such as the ones shown above. We then develop suitable representations for such sketches and suggest methods for computational recognition.*

## 1. Introduction

---



**Figure 1.2:** *Top row: cave paintings from Lascaux, France, about 20,000 years old. Bottom row: sketches by Pablo Picasso.*

able as a ‘horse’, the same concept in an ancient language or even a modern foreign language would be hardly understandable to a large fraction of the human population.

### 1.1 Definition: Sketch

In this thesis we use the term ‘sketch’ to mean an abstract pictograph drawn by a non-expert, see Figure 1.1 for several examples. We do not imply any particular medium such as pencil and paper. Sketches are always binary and do not contain any color. We deal with a very similar type of sketches as those painted on the walls of ancient caves and explore if we can make computers ‘understand’ sketches of such everyday objects as ‘horse’, ‘tree’ and ‘house’ as effortlessly as humans do. Although our methods of input are more modern than those of the cave artists — we typically use a mouse or a touchscreen device — our representation is carefully designed such that the resulting computational recognition methods can successfully classify even the cave paintings shown in Figure 1.2, as we demonstrate in Section 6.4.

### 1.2 Prior Research

There exists significant prior research on retrieving images or 3d models based on sketches. The assumption in all of these works is that sketched objects resemble their real-world counterparts in some well-engineered feature space. But this fundamental assumption is often violated since most humans are not faithful artists. Instead, people use *shared*, *iconic* representations of objects (e.g., stick figures) or they make dramatic simplifications or exaggerations (e.g., pronounced ears on rabbits). Thus, to understand and recognize sketches, an algorithm must learn from a training dataset of real *sketches*, not photos or 3d models. Because people represent the same object using differing degrees of realism and distinct drawing styles (see Figure 1.1), we gather large datasets of sketches that adequately sample these variations.

There also exists prior research in sketch recognition which tries to identify predefined glyphs in narrow domains such electronic circuit diagrams [Sezgin and Davis 2008], molecular diagrams [Ouyang and Davis 2011] and musical scores [Rebelo et al. 2010]. We instead identify objects such as ‘snowmen’, ‘ice cream cones’ and ‘giraffes’. This task is hard, because both shape and proportions of a sketched object may be far from that of the corresponding real object, and at the same time sketches are an impoverished visual representation. Nevertheless, humans are amazingly accurate at interpreting such sketches.

### 1.3 Overview

We first define a taxonomy of 250 object categories and acquire a large dataset of human sketches for the categories using crowd-sourcing (Chapter 3). Based on the dataset, we estimate how humans perform in recognizing the categories for each sketch (Chapter 4). We also design two robust visual feature descriptors for sketches (Chapter 5). These features permit not only the computational recognition of sketches (Section 6.2) but also the unsupervised analysis of the dataset (Section 6.3). While we achieve a high computational recognition accuracy of 56% (chance is 0.4%), our study also reveals that humans still perform significantly better than computers at this task.

Our overall approach is broadly similar to recent work in the computer vision community in which large categorical databases of visual phenomena are used to train recognition systems. High-profile examples of this include the Caltech-256 database of object images [Griffin et al. 2007], the SUN database of scenes [Xiao et al. 2010], and the LabelMe [Russell et al. 2008] and Pascal VOC [Everingham et al. 2010] databases of spatially annotated objects in scenes. The considerable effort that goes into building these databases has allowed algorithms to learn increasingly effective classifiers and to compare recognition systems on common benchmarks.

Although our pipeline is similar to many modern computer vision algorithms, we are working in a *new domain* which requires a new, carefully tailored representation. We also need to generate our data from scratch because there are no preexisting large repositories of sketches as there are for images (e.g., Flickr). For this reason, we utilize crowd-sourcing to create a database of human object sketches and hope that it will be as useful to the community as existing databases in other visual domains.

We then apply these concepts to sketch-based retrieval of 3d models. Working with large collections of 3d models requires fast content-based retrieval techniques, especially since public collections are often insufficiently annotated. In this case a keyword based search alone is not promising. While research on example-based retrieval — where users provide a full model as the query — has recently found a lot of interest in the community [Tangelder and Veltkamp 2008], its practical application is difficult, since a suitable example often is not at hand. As an alternative, sketch-based retrieval has been proposed [Löffler 2000; Funkhouser et al. 2003; Chen et al. 2003; Yoon et al. 2010; Shao et al. 2011], where users sketch the desired model as seen from one or more viewpoints. We consider sketch-based retrieval to be even more challenging than example-based retrieval as the query contains only *partial* information about

the *projection* of the shape. Most humans have limited drawing skills, and lines may deviate significantly from that projection. These properties of the input directly translate into four desiderata of sketch-based shape retrieval systems: *partial matching* of feature lines of the shape in *all potential viewing directions* to the sketch, tolerating *global and local deformation*; and, clearly, the retrieval performance has to *scale* to large collections. We present, to our knowledge, the first approach that addresses all of these desiderata.

The approach is based on the visual analysis of meshes: we sample the set of likely view directions, generate line drawings with state of the art line rendering techniques, and encode the line drawings with a bag-of-features approach. This choice is directly related to the requirements. First, rather than trying to match projected lines to shape features in 3d, we exploit current line art rendering techniques. They have reached a mature state, in which almost all lines drawn by humans are also generated by algorithms [Cole et al. 2008]. Second, bag-of-features approaches, which are well known in the image retrieval community [Sivic and Zisserman 2003], use local image descriptors that are independent of location. This is ideal, as it immediately enables partial matching and is resilient to global deformations. We achieve additional resilience to local deformations by quantization of the local image descriptors (identifying so-called “visual words”) and matching based on histograms. This data reduction leads, third, to the desired fast query times.

Overall, this leads to a system with high quality retrieval performance as we demonstrate in our objective evaluation. We also demonstrate the power of our system in Figure 7.1 where we gather all objects for a complete scene in about two minutes. However, we also find that the real-world dataset of sketches gathered in the experiment is challenging for current systems. In particular, our dataset reveals that allowing only closed contour curves for retrieval [Chen et al. 2003] oversimplifies reality: a large majority of our participants’ sketches contain a substantial amount of interior lines. The insights gained from an analysis of our dataset open up several promising areas of further research which we identify in Chapter 8.

We hope that the use of sketching as a visual input modality opens up computing technology to a significantly larger user base than text input alone. This thesis is a first step toward this goal and we release the datasets to encourage future research in this domain.

### 1.4 Contributions

We present the first exploration of human object sketches, with applications to computational sketch recognition and sketch-based 3d shape retrieval. Our main contributions are:

- **Two large large datasets of sketches.** We describe crowd-sourcing experiments to gather two large datasets of human object sketches. The first dataset of 1,814 sketches is designed to be used for 3d shape retrieval. Each sketch in the dataset is associated with an existing category of a given collection of 3d shapes [Shilane et al. 2004]. The second dataset of 20,000 sketches spans 250 object categories. We release both datasets to encourage future research in this domain.



- **An analysis of human sketch recognition performance.** Based on the dataset of 20,000 sketches we perform a crowd-sourcing experiment to analyze how well humans recognize sketches. We find that 73.1% of all sketches are correctly recognized.
- **Two feature transforms optimized for sketches.** Both transforms are tuned to efficiently comparing two abstract human-drawn sketches. The first transform is based on a bank of Gabor filters for orientation estimation while the second transform uses a simpler Gaussian derivative filter for that task. We show objectively that both descriptors outperform other existing transformations.
- **A method for computational sketch recognition.** We introduce the first approach for computational recognition of general human object sketches. We demonstrate a computational recognition accuracy of 56% on the dataset of 250 object categories.
- **A novel approach for sketch-based 3d shape retrieval.** The approach is based on the visual analysis of meshes. We sample the set of likely view directions, generate line drawings with state of the art line rendering techniques, and encode the line drawings using the feature transforms introduced in this thesis. We describe a general approach to determine optimal parameters for such feature transformations and demonstrate that even existing systems can be improved using this approach. We also introduce a large-scale benchmark for sketch-based retrieval systems that is based on the real-world dataset of 1,814 sketches gathered in our perceptual experiment.

## 1.5 Outline

**Chapter 1** We introduce key research questions addressed in this thesis and provide high level background information. We summarize our scientific contributions, outline the thesis and list our publications related to this thesis.

**Chapter 2** We overview related work in the areas of sketch-based image and shape retrieval as well as sketch recognition.

**Chapter 3** We describe the experimental setup used to gather two large datasets of human object sketches. We analyze properties of these datasets in a first attempt to understand how humans sketch objects. The datasets as well as the insights gained from their analysis form the basis of all following experiments.

**Chapter 4** Based on the dataset of human object sketches from the previous chapter, we describe a large-scale experiment in which we analyze how well humans recognize the correct category of such sketches.

**Chapter 5** We describe two novel feature transforms suitable for efficiently and effectively determining similarity between two sketches. The representations are designed according to the insights gained in Chapter 4.

**Chapter 6** We study computational sketch recognition of everyday objects, based on the dataset of 20,000 sketches (Chapter 3). To achieve generalization to unseen instances, we use state of the art supervised machine learning techniques for this task, operating in the feature space introduced in Chapter 5.

**Chapter 7** We propose a novel approach for sketch-based 3d shape retrieval that is based on the visual analysis of meshes: we sample the set of likely view directions, generate line drawings with state of the art line rendering techniques, and encode the line drawings using the representation introduced in Chapter 5. We show how to optimize all retrieval pipeline parameters to achieve optimal retrieval results and objectively demonstrate that the proposed approach outperforms existing approaches from the literature.

**Chapter 8** We conclude our thesis, discuss limitations and applications and give directions for future work.

### 1.6 Publications

The results presented in this thesis have been published as follows:

- The part on sketch-based shape retrieval has been published and presented as Sketch-Based Shape Retrieval [Eitz et al. 2012b] at SIGGRAPH 2012 in Los Angeles, USA. This work has been done in collaboration with Ronald Richter, Kristian Hildebrand and Marc Alexa from TU Berlin, Germany and Tamy Boubekeur from Telecom ParisTech, France.
- The part on sketch recognition has been published and presented as How Do Humans Sketch Objects? [Eitz et al. 2012a] at SIGGRAPH 2012 in Los Angeles, USA. This work has been done in collaboration with James Hays from Brown University, USA and Marc Alexa from TU Berlin, Germany.
- Part of the related work section on sketch-based image retrieval (Section 2.2) is based on Sketch-Based Image Retrieval: Benchmark and Bag-of-Features Descriptors [Eitz et al. 2011a].

---

## Chapter 2

# Related Work

---

“An image is worth a thousand words” perfectly expresses the key insight behind *sketch-based* retrieval and recognition methods: instead of using a few *abstract* keywords, users depict their search intent *visually* as a sketch. While for a keyword-based approach the burden of abstracting the visual search intent into a small set of keywords is on the user, sketch-based input is more direct. As a result, sketch-based interfaces are potentially easier to use for humans, but pose a greater computational challenge as computers now have to ‘understand’ and ‘interpret’ a user’s query. Unfortunately, the general population is not proficient in faithfully depicting the real world using sketches, which makes this task extremely difficult for computers.

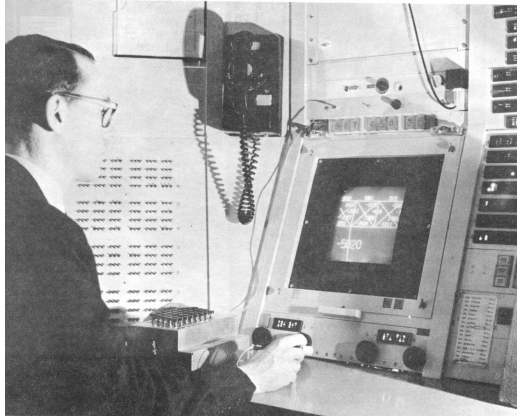
In this chapter, we review relevant prior work that is based on human-drawn sketches as input. Techniques in this domain are often closely related to prior work in text retrieval, content-based image retrieval as well as computer vision and machine learning based object recognition methods. We point out important connections to those areas where necessary but otherwise limit a more detailed review to methods that have been specifically designed for sketch input. Besides the references in this chapter, we provide most references to other relevant literature *in context*. For example, methods for quantizing local features given a visual vocabulary are reviewed in Section 5.6 that covers this topic.

### 2.1 History

The SketchPad system [Sutherland 1964] is the very first system that allows users to *directly* draw shapes on a computer using a pointing device. Before that, users had to *manually type* a shape’s coordinates as text. Sutherland pioneers all essential human-computer interaction techniques that are now ubiquitous in any vector based drawing tool: creating lines (and other shapes) by graphically defining their start- and endpoint, modifying the properties of such shapes as well as copying, pasting and deleting previously created elements by means of pointing and “clicking”.

## 2. Related Work

---



**Figure 2.1:** *SketchPad system operated by Ivan Sutherland creating a drawing of a bridge. Image taken from Sutherland [1964].*

Herot [1976] takes this one step further and proposes the first system that detects lines, corners and over-tracing in *freehand sketches*. Herot proposes to fit splines to the user input, detect corners as the minima of the pen speed function and use latching to overcome slight inaccuracies in a user's sketch.

These techniques for sketch input and “understanding” introduced by Sutherland and Herot have found such widespread adoption that they are nowadays taken for granted. Indeed, all sketches we use in this thesis have been created using software that relies on these concepts. In this thesis, we try take sketch understanding an additional level further: given a freehand sketch, we develop methods to *recognize* what kind of object such a sketch depicts (e.g., ‘house’, ‘airplane’).

Query-by-Pictorial-Example [Chang and Fu 1980a] is an early predecessor of modern sketch-based retrieval approaches. Images are analyzed and split into components such as lines or circles [Chang and Fu 1980b]. Based on this representation a relational query language allows users to select areas or point out parts of an existing image to serve as a query example [Chang and Fu 1980a]. From this point on, the field split into several subareas, all dealing with slightly different constraints: general shape matching systems [Loncaric 1998], where shapes are not necessarily sketches in the sense of this thesis, image retrieval approaches that sometimes incorporate shape features [Tangelder and Velkamp 2008] and explicit sketch recognition approaches that recognize users’ freehand sketches with the ultimate goal of creating an “Electronic Cocktail Napkin” [Gross 1996].

Most modern sketch-based approaches are built on three main concepts: a) representation of a sketch, typically as a compact feature vector in high-dimensional space; b) similarity measure defined over the feature vectors; and c) search/classification algorithms to find/classify a sketch. Most algorithms are built around novel ideas/variations in one or more of these areas to achieve the desired system properties. For example, in our thesis, we develop new representations for sketches but use standard distance metrics. Finally, we employ search and classification techniques that work well with our data-driven approach.



**Figure 2.2:** Example sketch-based image retrieval results from Eitz et al. [2010]. Top left corner: query sketch; top row: semantically meaningful results; bottom row: semantically unexpected results.

## 2.2 Sketch-Based Image Retrieval

For most image collections, queries are often expressed as keywords (or by other means than the images themselves), requiring the images to be tagged. In view of the ever increasing size of image databases, the assumption of an *appropriate* and *complete* set of tags might be invalid, and content-based search techniques become vital. Different types of content-based image queries have been suggested and analyzed: example images [Flickner et al. 1995]; rough, blurry drawings of the desired colors [Jacobs et al. 1995; Wang et al. 1997]; simple outline sketches [Kato et al. 1992; Chan et al. 1997; Matusiak et al. 1998]; and combinations or extensions thereof [Jain and Vailaya 1996; Di Sciascio et al. 1999].

Outline sketches are typically easier and faster to generate than a complete color description of the scene. And they can be generated for arbitrary desired images, while example images may or may not be at hand when searching. In addition, input devices change in favor of sketching as touch-enabled devices become more common. In other words, sketch-based image retrieval (SBIR) is a relevant means of querying large image databases (see Figure 2.2 for an example), content-based video retrieval [Collomosse et al. 2009] or even as a query language for geographic information systems [Egenhofer 1997].

Several approaches for SBIR have been suggested, with earlier approaches often casting the problem as computationally expensive template matching [Del Bimbo and Pala 1997; Anelli et al. 2007] or string matching approaches [Chang et al. 1987; Lopresti and Tomkins 1995; Lopresti et al. 1996]. However, to achieve interactive query response when using large, potentially Internet-scale image collections, it is impossible to compare the sketch to all images in the database directly. Instead, descriptors are extracted in a pre-process and stored in a data structure for fast access.

Very commonly, the descriptors are interpreted as points in high-dimensional space and finding close matches means searching for nearest neighbors in this space. Moreover, image descriptors can be roughly classified into global vs. local descriptors: global descriptors encode specific features of the whole image that suffice to describe the “gist” of a scene [Oliva and Torralba 2001], while many local descriptors need to be extracted for a single image, with each descriptor describing only a small spatially localized region of the image [Lowe 2004]. While the use of local descriptors is a common approach in example-based image retrieval [Squire et al. 1999; Sivic and Zisserman 2003; Jégou et al. 2008; Wu et al. 2009], most SBIR systems up to now still employ global de-

## 2. Related Work

---

scriptors and thus inherit their drawbacks, mainly being not invariant to affine transformations and/or local deformations of a sketch [Kumar Rajendran and Chang 2000; Ip et al. 2001; Chalechale et al. 2004a,b; Eitz et al. 2009, 2010; Springmann et al. 2010]. Recently, approaches that encode a sketch as a set of local patches became popular as those naturally come with desirable invariance properties and lend themselves to fast matching [Eitz et al. 2011a; Hu et al. 2011; Bozas and Izquierdo 2012].

An important design feature for any descriptor based retrieval system is that the distance metric in feature space correlates with perceptual similarity. To gauge this perceptual similarity, ground truth information from user studies is needed. Interestingly, Forsyth [2002] criticizes the design of many existing image retrieval systems for not meeting real users’ needs when they are based on image collections that are comprehensively tagged but are typically unrealistically small. Also, most SBIR systems rely on pre-processing the images to extract a sketch-like representation, often by using the Canny filter [Canny 1986]. Instead, several alternatives could potentially lead to better results, such as learning where humans draw edges [Martin et al. 2004], using filters that generate more human-like results [Kang et al. 2007] or simply post-processing the extracted lines [Barla et al. 2005; Hurtut et al. 2008].

Instead of an example image as in content-based retrieval [Datta et al. 2008], user input for sketch-based retrieval is a simple binary sketch — exactly the setting we consider throughout this thesis. Most existing approaches do not learn from example sketches and thus generally do *not achieve semantic understanding* of a sketch. Retrieval results are purely based on geometric similarity between the sketch and the image content [Chalechale et al. 2005; Hu et al. 2010; Eitz et al. 2011a; Shrivastava et al. 2011; Cao et al. 2011]. This can help make retrieval efficient as it often can be cast as a nearest-neighbor problem [Samet 2006]. However, retrieving perceptually meaningful results can be difficult as users generally draw sketches in an abstract way that is geometrically far from the real photographs or models (though still recognizable for humans as we demonstrate later in this thesis).

## Image Synthesis

Several image synthesis systems build upon the recent progress in sketch-based retrieval and allow users to create novel, realistic imagery using sketched exemplars [Diakopoulos et al. 2004]. Synthesis systems that are based on user sketches alone have to rely on huge amounts of data to offset the problem of geometric dissimilarity between sketches and image content [Eitz et al. 2011b] or require users to augment the sketches with text labels [Chen et al. 2009]. Using template matching to identify face parts, Dixon et al. [2010] propose a system that helps users get proportions right when sketching portraits. Lee et al. [2011] build upon this idea and generalize real-time feedback assisted sketching to a few dozen object categories. Their approach uses fast nearest neighbor matching to find geometrically similar objects [Zitnick 2010] and blends those object edges into rough shadow guidelines. As with other sketch-based retrieval systems, users must draw edges faithfully for the retrieval to work in the presence of many object categories — poor artists see no benefit from the system.

## 2.3 Sketch-Based Shape Retrieval

There exists a huge amount of work on example-based model retrieval, where the input is a 3d shape and the goal is to return similar 3d shapes from a large collection [Tangelder and Veltkamp 2008]. The basic idea in most of these works is to represent shapes in an appropriate feature space. Comparison between shapes is performed in this space given a suitable distance metric. Applications vary widely from Computer Graphics needs [Funkhouser et al. 2003] to retrieving 3d representations of molecules [Ankerst et al. 1999]. A typical challenge for all approaches is to make the shape signature invariant to orientation and scale of a shape [Osada et al. 2002] or even non-rigid transformations [Bronstein et al. 2011]. A wide variety of approaches exist: while Elad et al. [2002] use a global shape descriptor and a distance metric dynamically learned from user feedback, others decompose a shape into a set of views and perform image-based matching on the views [Chen et al. 2003; Furuya and Ohbuchi 2009].

In contrast to content-based 3d shape retrieval, the input to a sketch-based approach is typically only a single human-drawn sketch, depicting the desired object as seen from a particular viewpoint. Such a sketch contains considerably less information than the full model: it describes the model only from a single viewpoint, and it is often abstract and geometrically imprecise. This makes sketch-based 3d shape retrieval a hard (and potentially ambiguous) problem. Up to recently, sketch-based retrieval is often only studied in the context of an example-based retrieval engine [Funkhouser et al. 2003; Chen et al. 2003]. As a consequence, to our knowledge, no benchmark has been established that would allow objective comparison of sketch-based retrieval systems. We hope to alleviate this problem with the benchmark presented in Section 7.8.

One of the earliest references to sketch-based shape retrieval is given by Löffler [2000] who describes a system that lets users refine an initial keyword-based search using a sketch of the desired view. Funkhouser et al. [2003] describe an image based approach. In a pre-processing phase they extract boundary contours from 13 orthographic view directions for each model. They represent each view by a global — but rotation invariant — boundary descriptor and compute best matching models by comparing the corresponding view descriptors to the boundary descriptor computed from the input sketch(es). Chen et al. [2003] describe a system for example-based retrieval that also supports query by sketch. They densely sample view directions to form a “lightfield descriptor”. This descriptor however is only defined for *closed contour* curves, which, as we demonstrate later, is not how humans sketch for shape retrieval. Daras and Axenopolous [2010] describe a unified framework that supports both sketch-based as well as example-based retrieval. They extract 32 views from each model and compute three 2d rotation invariant shape descriptors per view. While a qualitative evaluation demonstrates good retrieval results, they do not perform a quantitative evaluation for sketch-based retrieval. Yoon et al. [2010] propose measuring orientation of sketch lines using a diffusion tensor — as the final descriptor they propose an orientation histogram that globally encodes each view of a model. Finally, Napoléon and Sahbi [2010] introduce a “2d photography to 3d object” retrieval framework that lets users to retrieve 3d models from one or more photographs/sketches.

### Domain Specific Specializations

When designing engineering parts, models are often described by three orthogonal 2d views. Consequently, Pu et al. [2005] extract six views by projection onto the faces of the model’s bounding box. They encode each view image by the distribution of pairwise Euclidean distances between densely sampled random points on the feature lines and employ a Euclidean distance metric to compare histograms of this distribution. Hou and Ramani [2006; 2007] extend this approach: instead of relying on a single feature they learn a classifier based on three shape descriptors. Their system follows a two-tier retrieval approach: first, it displays best matching classes of models and then sorts the models *within each class* according to similarity with the query.

### Sketch-Based Shape Synthesis

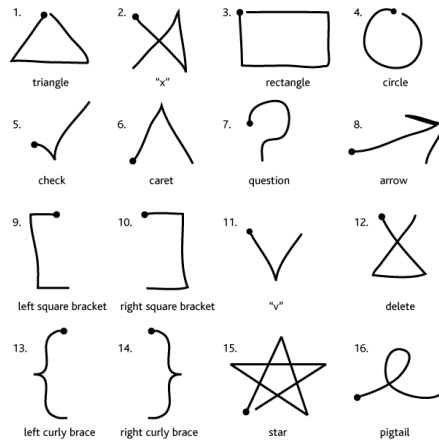
Retrieval can possibly only work when the desired object is actually contained in the collection. If this is not the case, the same sketch input (one or multiple views) could be used to infer the 3d model from the sketches instead [Igarashi et al. 1999; Nealen et al. 2007; Olsen et al. 2009]. This is an extremely difficult and typically under-constrained problem and an alternative can be to compose the desired model from existing parts: users query a collection for parts or even create a complete scene from existing objects, using only rough sketching strokes. Shin and Igarashi [2007] propose a system for interactively composing 3d scenes using existing models. They query models using a sketch-based interface based on contours generated from 16 reference views and encode each view using a Centroid Fourier Descriptor. Lee and Funkhouser [2008] extend this approach to create novel models from *parts* of existing models: a single sketch indicates both shape and placement of a part.

## 2.4 Sketch Recognition

Rough unfinished sketches can spark creativity in designers. Also, the rough nature of such sketches clearly implies that a drawing is not yet exact and the final details still need to be worked out (for example, exact sizes in architectural sketches). The goal of sketch recognition is to produce an “Electronic Cocktail Napkin” [Gross 1996] that combines the ease of creating a rough sketch using pen and paper with the possibility to modify, save and simulate offered by a digital representation. Ideally, users could simply draw a rough freehand sketch and the computer would understand the underlying structure without requiring any additional input.

Sketch recognition has been successfully applied to a variety of problems in domains such as electric circuit diagrams, musical scores, math sketches and molecular diagrams (Figure 2.4). Compared to the sketches we analyze in this thesis, such sketches exhibit a well-defined structure and this structure can be exploited to achieve good recognition results [Davis 2007; Hammond et al. 2008; Johnson et al. 2009].





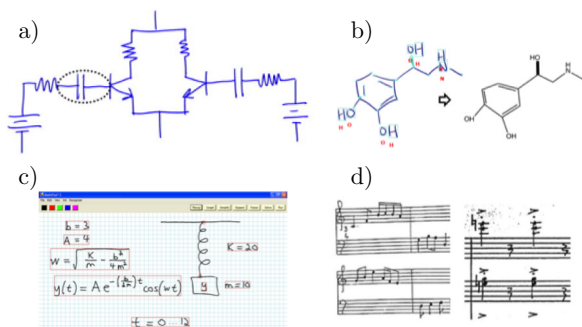
**Figure 2.3:** Set of 16 gestures. Image taken from Wobbrock et al. [2007].

## Primitive Recognition

One of the fundamental problems in sketch recognition is to recognize primitives such as lines, circles or arrows. Such primitives either occur as “building blocks” of larger freehand sketches or as self-contained gestures employed in user interfaces (for example, drawing a “circle” gesture invokes a certain action such as reloading a website). Recognizing primitives in freehand sketches is generally harder than recognizing gestures as it additionally involves some sort of segmentation of a larger sketch into its elements.

Gesture recognizers are typically designed to recognize a small set of single-stroke gestures such as the ones shown in Figure 2.3. Approaches include representing a stroke as a feature vector and performing linear classification [Rubine 1991] or nearest neighbor classification [Li 2010]; template matching on a bitmap representation [Kara and Stahovich 2005]; as well as the “\$1 recognizer” that has been specifically designed to be implementable using as little math as possible [Wobbrock et al. 2007]. Hand-tuned rules for a small set of primitives (line, polyline, circle ellipse, arc, curve, spiral and helix) result in close to perfect recognition results [Paulson and Hammond 2008].

Symbols commonly used in diagrams (e.g., arrow, transistor) are more complex and typically require multiple strokes to be drawn. One of the main challenges for multi-stroke recognizers is to achieve invariance to the temporal order in which strokes are drawn as well as invariance to rotation and scale of a symbol. Lee et al. [2007] represent symbols as graphs where the basic elements (lines and arcs) form the nodes and the edges represent the pairwise relationship between nodes. Invariance is achieved by casting the classification as a graph matching problem. The \$N recognizer [Anthony and Wobbrock 2010] extends the \$1 recognizer [Wobbrock et al. 2007] to multi-stroke recognition but shares the same design goals: simplicity, ease of implementation and trainability using a single example gesture.



**Figure 2.4:** Structured domains for sketch recognition: a) electronic circuit diagrams [Sezgin and Davis 2008]; b) molecular diagrams [Ouyang and Davis 2011]; c) math diagrams [LaViola Jr. and Zeleznik 2004], d) musical score [Rebelo et al. 2010]. Images taken from the respective publications.

## Higher-level Sketch Recognition

Recognition of higher-level structure from freehand sketches (e.g., diagrams, user interface elements, molecules) is often based on inferring their large-scale structure given a parsed set of primitive objects (bottom-up approach). For example, the SILK system [Landay and Myers 2001] for sketching interactive user interface prototypes relies on Rubine’s primitive recognizer [Rubine 1991]. Sezgin et al. [2001] propose a system that segments an arbitrary freehand sketch into low-level geometric descriptions (lines, circles, rectangles, etc.) as an “early processing” step required before true “sketch understanding” can be achieved. Such a representation can then be used to beautify sketches such that wiggly sketches lines become straight digitized lines or constraints between primitives can be enforced [Igarashi et al. 1997; Cheema et al. 2012].

Specializing sketch recognition to specific domains can yield impressive recognition results. Typical domains of interest include: UML diagrams [Hammond and Davis 2002], circuit diagrams [Sezgin and Davis 2007] as well as molecular diagrams [Ouyang and Davis 2011]. Sezgin et al. [2008] also explore if multi-scale temporal information alone can be used to recognize sketched circuit diagrams. They learn temporal ordering of strokes from a manually annotated training dataset gathered from 8 participants and exploit this to train a model for classification. They find that their multi-scale temporal model yields statistically significant better recognition accuracy than a stroke-based baseline model but (by definition) fails in case the temporal order of two unrelated shapes happens to be similar.

Also, several “meta frameworks” for sketch recognition have been proposed that in turn can be customized to achieve certain recognition tasks [Alvarado et al. 2002; Alvarado and Davis 2004; Hammond and Davis 2005].

Sharon et al. [2006] propose to learn a “constellation model” per category. Such a model is defined by a probability distribution over location and size of parts bounding boxes and a probability distribution over distances between pairwise parts. This yields a pretty general model that can be applied to any sketches that consist of a predictable, fixed number of parts (e.g., faces).

One recent approach (adopted in this thesis) is to represent sketches using

*image-based* features (rather than a set of geometric rules that describe strokes and their relationships) [Oltmans 2007; Ouyang and Davis 2011] or even as a combination of image-based and temporal features [Arandjelović and Sezgin 2011]. Sketches are often represented as graphical models to deal with uncertainty in recognizing parts and to incorporate context information into the recognition process [Qi et al. 2005; Sezgin and Davis 2008; Ouyang and Davis 2011; Arandjelović and Sezgin 2011]. Higher level *applications* building on existing sketch recognition work include systems for animating math and physics diagrams [LaViola Jr. and Zeleznik 2004; Cheema and LaViola 2012].

Compared to the sketches analyzed in previous work, the sketches we deal with in this thesis have *considerably less structure* and we wish to discern sketches from a *large number of categories*. This makes the problem very difficult and requires developing suitable representations for such unstructured data as well as learning from large, real-world datasets.



---

## Chapter 3

# Sketch Datasets

---

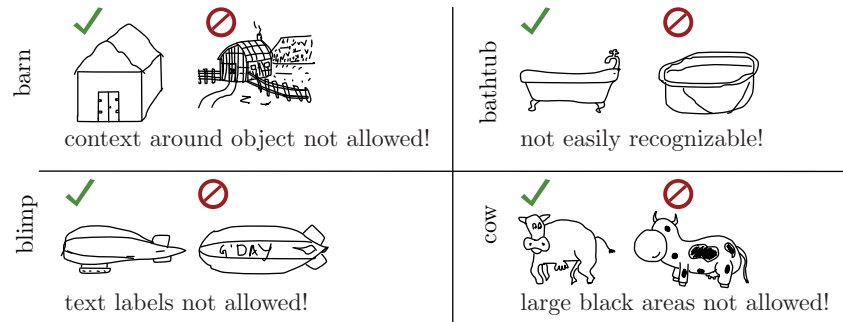
In order to understand how humans sketch objects we perform two large-scale experiments in each of which we gather a large dataset of human sketches. We later apply the insights gained from analyzing these datasets to facilitate computational recognition of human object sketches and improve sketch-based 3d shape retrieval. Ideally, these datasets should adequately sample the variety in drawing styles found in human sketches, which we expect to be potentially quite large. We focus on sketches from the *general population* as opposed to sketches from artists only, as done in previous related experiments [Cole et al. 2008]. As a result, in our experiments, we sample a large number of sketches (tens of thousands) from a large number of different individuals (hundreds).

We gather two distinct sketch datasets: while the first one is very general, spanning a wide range of categories, the second dataset is tailored to sketch-based 3d shape retrieval.

- The first sketch dataset is very general, with each sketch associated to one of 250 object categories such as ‘teapot’, ‘car’ or ‘horse’. We use this dataset to analyze how humans sketch objects, to analyze how well humans recognize each other’s sketches and finally, to train supervised machine-learning algorithms for computational sketch recognition.
- The second dataset of sketches is designed to be used for 3d shape retrieval. Each sketch in the dataset is associated with an existing category of a given collection of 3d shapes [Shilane et al. 2004]. This association is a crucial property of this dataset as it lets us define a benchmark for 3d shape retrieval: when querying with a sketch from a given category, we expect that a “good” system only returns 3d shapes belonging to the same category as the query sketch.

We use the same experimental setup to gather both sketch datasets, which we describe in the following section.

### 3. Sketch Datasets



**Figure 3.1:** *Instructional examples shown to workers on Mechanical Turk. In each field: desired sketching style (left), undesired sketching style (right).*

#### 3.1 Experimental Setup

We ask participants to draw one sketch at a time given a random category name (such as ‘airplane’ or ‘teapot’). For each sketch, participants start with an empty canvas and have up to 30 minutes to create the final version of the sketch. We keep our instructions as simple as possible and ask participants to

“sketch an image [...] that is clearly recognizable to other humans as belonging to the following category: [...].”

We also ask users to draw outlines only and not use context around the actual object. We provide visual examples that illustrate these requirements, see Figure 3.1. We provide undo, redo, clear, and delete buttons for our stroke-based sketching canvas so that participants can easily familiarize themselves with the tool while drawing their first sketch. After finishing a sketch participants can move on and draw another sketch given a new category. In addition to the spatial parameters of the sketched strokes we store their temporal order.

#### Crowd-sourcing

As we wish to sample a large number of sketches from a large number of participants, we rely on crowd-sourcing to generate our sketch dataset. We use Amazon Mechanical Turk (AMT) which is a web-based market where requesters can offer paid “Human Intelligence Tasks” (HITs) to a pool of non-expert workers. In order to ensure a diverse set of sketches within each category, we limit the number of sketches a worker could draw to one per category.

Our sketching task appears to be quite popular on Mechanical Turk — we received a great deal of positive feedback, the time to complete all HITs was low, and very few sketches were unusable, adversarial, or automated responses.

#### 3.2 A Large Dataset of Human Object Sketches

In this section we describe the collection of a dataset of 20,000 human object sketches. This categorical database is the basis for all learning, evaluation, and

applications later in this thesis. We define the following set of criteria for the object categories in the sketch dataset:

**Exhaustive** The categories exhaustively cover most objects that we commonly encounter in everyday life. We want a broad taxonomy of object categories in order to make the results interesting and useful in practice and to avoid superficially simplifying the recognition task.

**Recognizable** The categories are recognizable from their shape alone and do not require context for recognition.

**Specific** Finally, the categories are specific enough to have relatively few visual manifestations. ‘Animal’ or ‘musical instrument’ would not be good object categories as they have many subcategories.

#### Defining a Taxonomy of 250 Object Categories

In order to identify common objects, we start by extracting the 1,000 most frequent labels from the LabelMe [Russell et al. 2008] dataset. We manually remove duplicates (e.g., car side vs. car front) as well as labels that do not follow our criteria. This gives us an initial set of categories. We augment this with categories from the Princeton Shape Benchmark [Shilane et al. 2004] and the Caltech 256 dataset [Griffin et al. 2007]. Finally, we add categories by asking members of our lab to suggest object categories that are not yet in the list. Our current set of 250 categories is quite exhaustive as we find it increasingly difficult to come up with additional categories that adhere to the desiderata outlined above. We list the complete set of categories in Table 3.1.

#### Collecting 20,000 Sketches

We use Amazon Mechanical Turk with the experimental setup described in Section 3.1. In order to prepare for workers that do not strictly follow our criteria, we request more than the desired 20,00 sketches: we submit  $90 \times 250 = 22,500$  HITs, requesting 90 sketches for each of the 250 categories.

As with any crowd-sourced data collection effort, steps must be taken to ensure that data collected from non-expert, untrained users is of sufficient quality. We manually inspect and clean the complete dataset using a simple interactive tool we implemented for this purpose (see Figure 3.2). The tool displays all sketches within a given category on a large screen which lets us identify incorrect ones at a glance. We remove sketches that are clearly in the wrong category (for example, an ‘airplane’ in the ‘teapot’ category), exhibit offensive content or otherwise do not follow the requirements defined in Section 3.1 and Section 3.2 (typically excessive context). We do *not* remove sketches just because they are poorly drawn. As a result of this procedure, we remove about 6.3% of the sketches. We truncate the dataset to contain exactly 80 sketches per category yielding our final dataset of 20,000 sketches. We make the categories uniformly sized to simplify training and testing (e.g., we avoid the need to correct for bias toward the larger classes when learning a classifier).

### 3. Sketch Datasets



**Figure 3.2:** Screenshot of interactive tool for sketch dataset verification for category ‘bench’. The highlighted sketches are marked for removal: a), b), e) wrong category; c) unrecognizable; d) hatching strokes.

### Analysis of Sketches

In total, we receive sketches from 1,350 unique participants who spent a total of 741 hours to draw all sketches. The median drawing time per sketch is 86 seconds with the 10<sup>th</sup> and 90<sup>th</sup> percentile at 31 and 280 seconds, respectively. The participants draw a total of 351,060 strokes with each sketch containing a median number of 13 strokes.

We find that the first few strokes of a sketch are on average considerably longer than the remaining ones, see Figure 3.3. This suggests that humans tend to follow a coarse-to-fine drawing strategy, first outlining the shape using longer strokes and then adding detail at the end of the sketching process. We also find that drawing time can vary considerably between and within categories: the median drawing time for ‘cloud’ is the lowest overall with 30.5 seconds, while the highest overall is for ‘tiger’ (244 seconds). We show a more detailed illustration of the time spent per category in Figure 3.4. In many cases, the sketches in a category with shorter drawing times correspond to simplified, stylized and abstract depictions of the objects (see Figure 3.4 bottom row), while sketches with higher drawing times often appear more realistic (see Figure 3.4 top row).

### 3.3 A Large Dataset of Sketches for 3d Shape Retrieval

In the second experiment we try to provide insight into the following problem: how would an average user of a 3d retrieval system sketch the query? Most

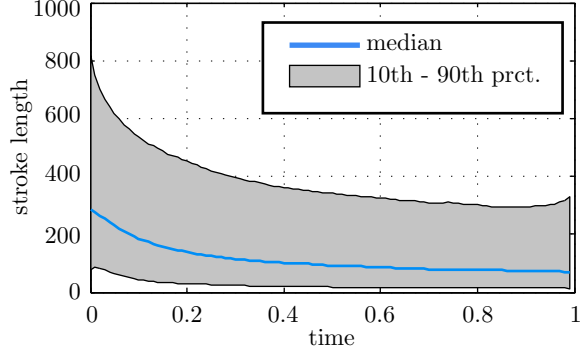


### 3.3. A Large Dataset of Sketches for 3d Shape Retrieval

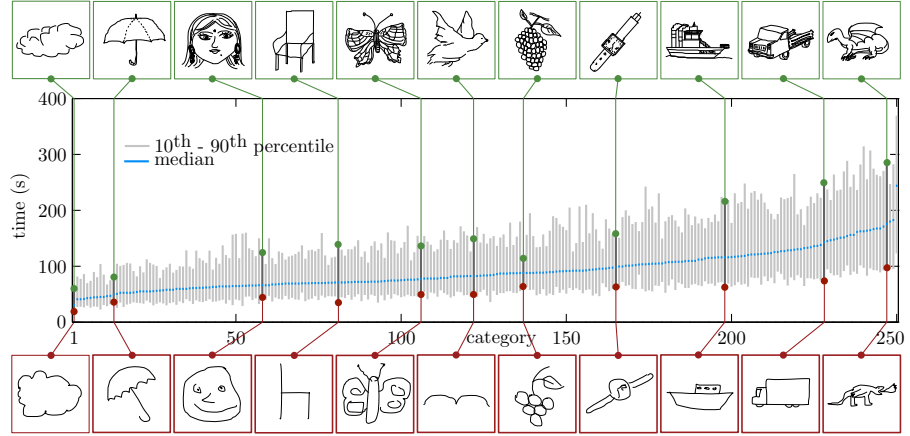
airplane	alarm clock	angel	ant	apple
arm	armchair	ashtray	axe	backpack
banana	barn	baseball bat	basket	bathtub
bear (animal)	bed	bee	beer-mug	bell
bench	bicycle	binoculars	blimp	book
bookshelf	boomerang	bottle opener	bowl	brain
bread	bridge	bulldozer	bus	bush
butterfly	cabinet	cactus	cake	calculator
camel	camera	candle	cannon	canoe
car (sedan)	carrot	castle	cat	cell phone
chair	chandelier	church	cigarette	cloud
comb	computer monitor	computer-mouse	couch	cow
crab	crane (machine)	crocodile	crown	cup
diamond	dog	dolphin	donut	door
door handle	dragon	duck	ear	elephant
envelope	eye	eyeglasses	face	fan
feather	fire hydrant	fish	flashlight	floor lamp
flower with stem	flying bird	flying saucer	foot	fork
frog	frying-pan	giraffe	grapes	grenade
guitar	hamburger	hammer	hand	harp
hat	head	head-phones	hedgehog	helicopter
helmet	horse	hot air balloon	hot-dog	hourglass
house	human-skeleton	ice-cream-cone	ipod	kangaroo
key	keyboard	knife	ladder	laptop
leaf	lightbulb	lighter	lion	lobster
loudspeaker	mailbox	megaphone	mermaid	microphone
microscope	monkey	moon	mosquito	motorbike
mouse (animal)	mouth	mug	mushroom	nose
octopus	owl	palm tree	panda	paper clip
parachute	parking meter	parrot	pear	pen
penguin	person sitting	person walking	piano	pickup truck
pig	pigeon	pineapple	pipe	pizza
potted plant	power outlet	present	pretzel	pumpkin
purse	rabbit	race car	radio	rainbow
revolver	rifle	rollerblades	rooster	sailboat
santa claus	satellite	satellite dish	saxophone	scissors
scorpion	screwdriver	sea turtle	seagull	shark
sheep	ship	shoe	shovel	skateboard
skull	skyscraper	snail	snake	snowboard
snowman	socks	space shuttle	speed-boat	spider
sponge bob	spoon	squirrel	standing bird	stapler
strawberry	streetlight	submarine	suitcase	sun
suv	swan	sword	syringe	t-shirt
table	tablelamp	teacup	teapot	teddy-bear
telephone	tennis-racket	tent	tiger	tire
toilet	tomato	tooth	toothbrush	tractor
traffic light	train	tree	trombone	trousers
truck	trumpet	tv	umbrella	van
vase	violin	walkie talkie	wheel	wheelbarrow
windmill	wine-bottle	wineglass	wrist-watch	zebra

**Table 3.1:** 250 object categories used for sketch recognition experiment.

### 3. Sketch Datasets



**Figure 3.3:** *Stroke length in sketches over drawing time: initial strokes are significantly longer than later in the sketching process. On the x-axis, time is normalized for each sketch.*



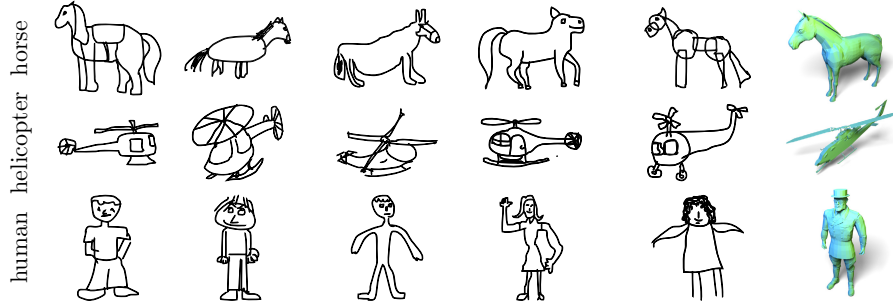
**Figure 3.4:** *Distribution of sketching time. Categories are sorted by their median sketching time. The top and bottom rows show examples of sketches with particularly long or short sketching times, respectively.*

related studies we are aware of gather input from artists [Cole et al. 2008] or contain a significantly smaller number of sketches [Funkhouser et al. 2003].

### Methodology

We ask participants to create an input query sketch given the name of a category only (e.g., ‘airplane’), *without* providing an example rendering. This setup is designed to closely resemble how novel users would use a retrieval system. We expect users to only have a rough model of what they wish to find in mind rather than having an example 3d model at hand that the drawing could be based on. We emphasize that the sketch should be clearly recognizable for other humans.

We perform the experiment on Amazon Mechanical Turk using the experimental setup described in Section 3.1. We ask for all categories that are



**Figure 3.5:** Subsets of the sketches gathered for the sketch-based 3d shape retrieval benchmark. Each sketch corresponds to a specific category (first column) from the Princeton Shape Benchmark [Shilane et al. 2004]. Last column: examples of corresponding 3d shapes from the same category. Note that we did not show the 3d renderings to the participants of the experiment.

defined in the Princeton Shape Benchmark (PSB) [Shilane et al. 2004] and gather sketches for all models in the PSB (i.e. 1,814 sketches). There is no direct association between sketches and models but rather between sketch category and model category. For example, the PSB contains 35 models for the category ‘helicopter’ and we also gather 35 sketches falling into this category. We later exploit this association to define a benchmark for 3d shape retrieval.

### Analysis of Sketches

The majority of sketches makes use of more complex lines than just simple silhouettes and virtually no sketch consists of simple closed boundary curves, see for example the sketches in Figure 3.5. This clearly motivates developing systems for 3d shape retrieval that support arbitrarily complex sketches such as the one presented in Chapter 7.

Also, as expected, sketches show strong abstraction, local and global deformations with respect to the real shape, as well as perspective errors. We can confirm the result from Funkhouser et al. [2003]: users mostly sketch objects from a simple side or frontal view. Simple objects such as a table however often tend to be drawn in perspective (82.6% in our experiments), although typically with significant perspective error.



---

## Chapter 4

# Human Sketch Recognition

---

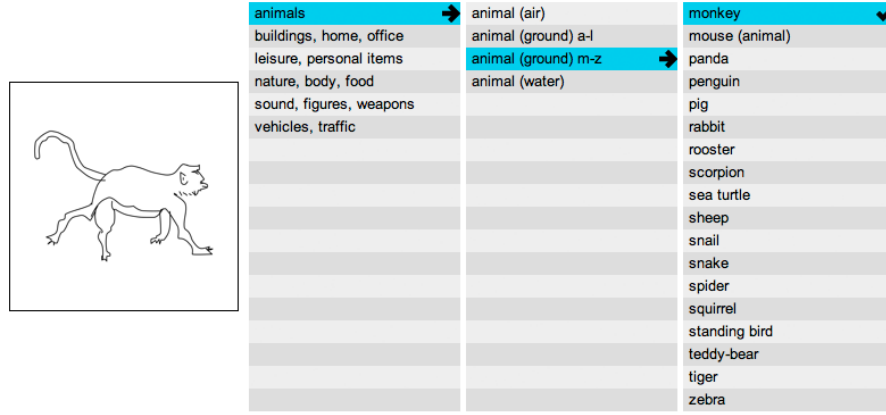
We analyze human sketch recognition performance based on the dataset of 20,000 object sketches gathered in Chapter 3. Our basic questions are the following: given a sketch, what is the accuracy with which humans correctly identify its category? Are some categories easier or more difficult to discern than others? To provide answers to these questions, we perform a second large-scale, crowd-sourced study (again using Amazon Mechanical Turk) in which we ask participants to identify the category of query sketches. This test provides us with an important human baseline which we later compare against our computational recognition method. We invite the reader to try this test on the sketches shown in Figure 1.1: can you correctly identify all categories and solve the riddle hidden in this figure?

### 4.1 Experimental Setup

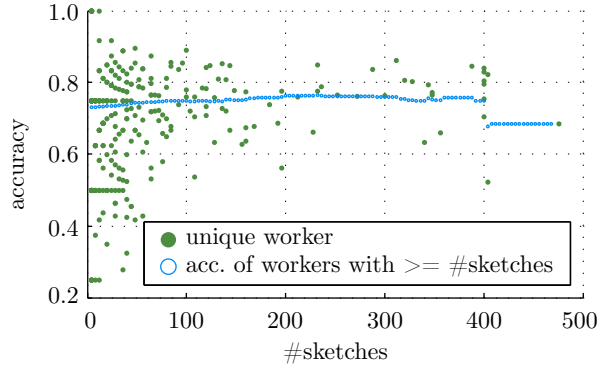
Given a random sketch, we ask participants to select the best fitting category from the set of 250 object categories. We give workers unlimited time, although workers are naturally incentivized to work quickly for greater pay. To avoid the frustration of scrolling through a list of 250 categories for each query, we roughly follow Xiao et al. [2010] and organize the categories in an intuitive 3-level hierarchy, containing 6 top-level and 27 second-level categories such as ‘animals’, ‘buildings’ and ‘musical instruments’. The hierarchy we use is over complete where appropriate (a given category can appear in more than one second-level category) in order to make finding the desired category as easy as possible. For example, ‘apple’ appears both in the second-level category ‘food’ and ‘plants’. We list the complete hierarchy in Table 4.2.

We submit a total of 5,000 HITs to Mechanical Turk, each containing a unique subset of four sketches from the total set of 20,000 sketches. In each HIT we ask workers to sequentially identify four sketches from random categories. This gives us one human classification result for each of the 20,000 sketches. We include several sketches per HIT to prevent workers from skipping tasks that contain ‘difficult’ sketches based on AMT preview functions as

#### 4. Human Sketch Recognition



**Figure 4.1:** Hierarchical selection menu for human sketch recognition experiment. Left: sketch to be classified. Right: hierarchical selection menu from which participants have to choose the “correct” category for the sketch.





















**Figure 4.2:** Scatter plot of per-worker sketch recognition performance. Solid (green) dots represent single, unique workers and give their average classification accuracy (y-axis) at the number of sketches they classified (x-axis). Outlined (blue) dots represent overall average accuracy of all workers that have worked on more than the number of sketches indicated on the x-axis.

this would artificially inflate the accuracy of certain workers. In order to measure performance from many participants, we limit the maximum paid HITs to 100 per worker, i.e. 400 sketches. (However, three workers did 101, 101 and 119 HITs, respectively. This causes the drop in accuracy in Figure 4.2).

























#### 4.2 Human Classification Results

Humans recognize on average 73.1% percent of all sketches correctly. We observe a large variance over the categories: while all participants correctly identified all instances of ‘t-shirt’, the ‘seagull’ category was only recognized 2.5% of the time. We visualize the categories with highest human recognition perfor-

## 4.2. Human Classification Results

t-shirt  100%	snake  99%	comb  99%	flower  99%	eyeglasses  98%	elephant  98%
leaf  98%	sun  98%	wrist-watch  96%	pineapple  96%	trousers  96%	ladder  96%
apple  96%	airplane  96%	butterfly  96%	umbrella  96%	chair  95%	key  95%

**Figure 4.3:** Representative sketches from 18 categories with highest human category recognition rate (bottom right corner of each cell, in percent).

seagull  2.5%	panda  11%	armchair  13%	tire  21%	ashtray  24%	snowboard  25%
flying bird  47%	bear  44%	chair  89%	wheel  44%	cigarette  30%	skateboard  32%
standing bird  24%	teddy bear  30%	couch  3%	donut  16%	bowl  15%	knife  7%
pigeon  14%	dog  8%	bench  1%	fan  6%	bathtub  11%	canoe  3%

**Figure 4.4:** Top row: six most difficult classes for human recognition. E.g., only 2.5% of all seagull sketches are correctly identified as such by humans. Instead, humans often mistake sketches belonging to the classes shown in the rows below as seagulls. Out of all sketches confused with seagull, 47% belong to flying bird, 24% to standing bird and 14% to pigeon. The remaining 15% (not shown in this figure) are distributed over various other classes.

mance in Figure 4.3 and those with lowest performance in Figure 4.4 (along with the most confusing categories). Human errors are usually confusions between semantically similar categories (e.g., ‘panda’ and ‘bear’), although geometric similarity accounts for some errors (e.g., ‘tire’ and ‘donut’).

If we assume that it takes participants a while to learn our taxonomy and hierarchy of objects, we would expect that workers who have done more HITs are more accurate. However, this effect is not very pronounced in our experiments — if we remove all results from workers that have done less than 40 sketches, the accuracy of the remaining workers rises slightly to 73.9%. This suggests that there are no strong learning effects for this task. We visualize the accuracy of each single worker as well as the overall accuracy when gradually removing workers that have classified less than a certain number of sketches in Figure 4.2.

#### 4. Human Sketch Recognition

**Table 4.1:** *Hierarchy of categories for human sketch recognition experiment.*

animals	animal (air)	bee, butterfly, dragon, duck, feather, flying bird, mosquito, owl, parrot, pigeon, standing bird, swan
	animal (ground, a–m)	ant, bear, camel, cat, cow, crocodile, dog, dragon, elephant, frog, giraffe, hedgehog, horse, kangaroo, lion, monkey, mouse
	animal (ground, n–z)	panda, penguin, pig, rabbit, rooster, scorpion, sea turtle, sheep, snail, snake, spider, squirrel, standing bird, teddy-bear, tiger, zebra
	animal (water)	crab, crocodile, dolphin, duck, fish, frog, lobster, octopus, penguin, sea turtle, seagull, shark, swan
buildings, home, office	around the house	bench, door, door handle, fire hydrant, ladder, mailbox, satellite dish, wheelbarrow
	bathroom	bathtub, comb, toilet, toothbrush
	buildings	barn, bridge, castle, church, house, skyscraper, tent, windmill
	electronics, computers	alarm clock, calculator, camera, cell phone, computer monitor, computer-mouse, fan, head-phones, ipod, keyboard, laptop, lightbulb, power outlet, radio, telephone, tv, walkie talkie
	furniture, decor (a–c)	armchair, ashtray, basket, bed, bench, bookshelf, cabinet, candle, chair, chandelier, couch
	furniture, decor (d–z)	door, door handle, fan, floor lamp, hourglass, lightbulb, power outlet, table, tablelamp, vase
	hand tools	axe, bottle opener, flashlight, hammer, ladder, scissors, screwdriver, shovel, syringe
	kitchen, dining	basket, beer-mug, bottle opener, bowl, candle, cup, fork, frying-pan, knife, lighter, mug, spoon, teacup, teapot, wine-bottle, wineglass
	office, work	calculator, computer monitor, computer-mouse, envelope, key, microscope, paper clip, pen, stapler, syringe, telephone
	leisure, personal items	
	clothing	hat, helmet, shoe, socks, t-shirt, trousers
	fun	book, cigarette, feather, pipe (for smoking), present, snowman, sponge bob
	musical instruments	bell, guitar, harp, piano, saxophone, trombone, trumpet, violin
	personal accessoires	backpack, binoculars, cigarette, comb, crown, diamond, eyeglasses, key, lighter, pen, purse, suitcase, umbrella, wrist-watch
	sports	backpack, baseball bat, bicycle, binoculars, boomerang, canoe, parachute, rollerblades, skateboard, snowboard, tennis-racket, tent



nature, body, food	food (a-i)	apple, banana, bread, cake, carrot, crab, donut, fish, grapes, hamburger, hot-dog, ice-cream-cone
	food (j-z)	lobster, mushroom, pear, pineapple, pizza, pretzel, strawberry, tomato
	human body	arm, brain, ear, eye, face, foot, hand, head, human-skeleton, mouth, nose, person sitting, person walking, skull, tooth
	plants	apple, banana, bush, cactus, carrot, flower with stem, grapes, leaf, mushroom, palm tree, potted plant, pumpkin, tree
	the skies	cloud, moon, rainbow, sun
sound, figures, weapons	human-like figure	angel, mermaid, santa claus, snowman, sponge bob, teddy-bear
	sound	bell, head-phones, ipod, loudspeaker, megaphone, microphone, radio
	weapons	axe, cannon, grenade, knife, revolver, rifle, sword
vehicles, traffic	traffic	parking meter, streetlight, tire, traffic light, wheel
	vehicle (air)	airplane, blimp, flying saucer, helicopter, hot air balloon, satellite, space shuttle
	vehicle (ground)	bicycle, bulldozer, bus, car (sedan), crane (machine), motorbike, pickup truck, race car, suv, tire, tractor, train, truck, van, wheel, wheelbarrow
	vehicle (water)	canoe, sailboat, ship, speed-boat, submarine

### 4.3 Discussion

Overall, we find that humans identify the categories of 73.1% of 20,000 sketches correctly. We also find that some categories in our set are particularly hard to discern (e.g., ‘seagull’ vs. ‘pigeon’). This can be due to a number of reasons: in the first experiment (Section 3.2), when asking workers to draw sketches, these sketches can be of arbitrarily poor artistic value making differentiation between two categories indeed virtually impossible. Second, we hypothesize that workers are sometimes “lazy” and happily accept the first choice in the hierarchical selection menu (Figure 4.1) that appears reasonable without looking for further, potentially more appropriate options.

In the experiments we ran on AMT, we did not provide direct incentives for workers to achieve maximum recognition rate. Rather, the incentive was to complete the task as quickly as possible. In conclusion, we thus hypothesize that the human sketch recognition rate of 73.1% is a *lower bound* to the actual maximally possible recognition rate.



---

## Chapter 5

# Sketch Representation

---

We describe two novel methods for computing a sketch representation suitable for efficiently and effectively determining similarity between two sketches. One would expect that a “good” feature transform produces similar features (under a given distance metric) when two sketches are perceptually close. From analyzing the sketch datasets in Chapter 3 we know that humans use a large variety of sketching styles even within a single category. The challenge is to define a sketch representation that ideally is invariant to the variety of styles *within categories* but produces clearly distinct features for sketches from different categories. Finally, given the representation, assessing similarity should be simple and computationally efficient in order to facilitate interactivity in applications. This excludes alternative approaches which use a simpler representation in combination with a more involved distance metric [Rubner et al. 2000]. Instead, in this thesis, we opt to utilize the former method (complex feature transform with a simpler metric) as this lets us shift most of the required computational resources into an offline pre-processing step, making the similarity query performed at runtime very efficient. As a result, all applications proposed in this thesis are interactive.

A sketch’s (possibly downscaled) bitmap representation can be seen as a simple feature transform. Such a feature transform is known (for the case of natural images) to work well when the number of images in the dataset lies in the order of tens of millions [Torralba et al. 2008a]. For our much smaller collections, such a representation works poorly in our experiments (see Figure A.1). Instead we adopt methods from computer vision and represent sketches using local feature vectors that encode *distributions* of image properties [Squire et al. 2000; Sivic and Zisserman 2003]. Specifically, both representations encode the distribution of *line orientation* within a small local region of a sketch. The binning process during construction of the local distribution histograms facilitates better invariance to slight offsets in orientation compared to directly encoding pixel values. The representations differ only in how orientation is estimated but share the remaining feature construction pipeline.

## 5.1 Overview

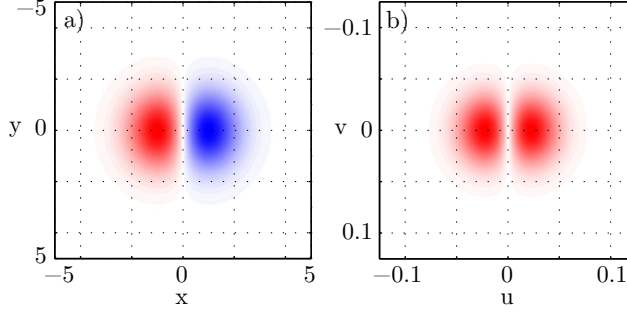
In the remainder of this thesis, we use the following notational conventions:  $k$  denotes a scalar value,  $\mathbf{h}$  a column vector,  $\mathbf{S}$  a matrix and  $\mathcal{V}$  a set. We consider a sketch  $\mathbf{S}$  as a bitmap image, with  $\mathbf{S} \in \mathbb{R}^{m \times n}$ . While the datasets from Chapter 3 are inherently vector-valued, a bitmap-based approach is more general: vector representations are easily rasterized into a bitmap but existing bitmap representations cannot be easily converted into a vector representation. We later exploit this generality of our approach to perform computational sketch recognition on ancient cave paintings (Figure 6.8).

An ideal sketch representation for our purposes is invariant to irrelevant features (e.g., scale, translation and local deformations), discriminative between categories, and compact. More formally, we are looking for a feature space transform that maps a sketch bitmap to a lower-dimensional representation  $\mathbf{x} \in \mathbb{R}^d$ , i.e. a mapping  $f: \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^d$  with (typically)  $d \ll mn$ . In the remainder of this thesis we call the resulting representation  $\mathbf{x}$  either a *feature vector* or a *descriptor*. Ideally, the mapping  $f$  preserves the information necessary for  $\mathbf{x}$  to be distinguished from all sketches in other categories.

Our representations build on *local* image descriptors that encode image content within small local regions of a sketch. To balance accuracy and efficiency, local image regions are commonly represented as descriptors that encode only the information that is essential for retrieval or recognition (rather than using the original bitmap data within that area). Successful representations often capitalize on the *distribution* of the value of interest, such as the SIFT and SURF descriptor [Lowe 2004; Bay et al. 2006]. Another popular approach for designing a descriptor is to perform a change of basis, such that the original data can be faithfully represented using only a *sparse* set of basis elements. The Fourier basis as well as the Wavelet basis [Jacobs et al. 1995] are likely the most commonly used basis transformations. These transformations are not optimal for our type of data, i.e. elongated lines on a constant background. The Curvelet basis is the maximally sparse representation for such data [Candès and Donoho 1999] and, consequently, we design our representations based on ideas taken from this transformation. In the context of feature representation, we relax the requirement of the transformation being a basis (we never want to reconstruct the sketch from its descriptor). We thus approximate ideas from the Curvelet transform — using filters that respond only to image elements with given frequency and orientation — to yield our new feature space transform based on Gaussian derivative filters and Gabor filters.

In the following sections, we describe those two novel feature transforms. They are both explicitly designed for sketches and exhibit the desired invariance properties. For both transforms, we achieve global scale and translation invariance by isotropically rescaling each sketch such that the longest side of its bounding box has a fixed length (scale invariance) and centering each in a  $256 \times 256$  image (global translation invariance).

Both transforms first estimate orientations of sketch lines (Section 5.2 and Section 5.3) and then encode a sketch as a set of local features, each of which encodes a sketch’s content within a (small) rectangular region (Section 5.4). Those local feature definitions form the basis for the final quantization step that is shared by both transforms (Section 5.6). This quantization step has seen a lot of success in the computer vision literature to generate (parts of the)



**Figure 5.1:** Gaussian derivative filter  $\partial G(x, y)/\partial x$  using  $\sigma = 1$ : a) spatial domain, b) frequency domain.

features for tasks such as object based image retrieval [Sivic and Zisserman 2003; Nister and Stewenius 2006; Chum et al. 2007], object recognition [Zhang et al. 2007; Ommer and Buhmann 2010] and scene recognition [Xiao et al. 2010] and proves to be valuable for sketch recognition as well.

## 5.2 SHOG: Local Histograms of Oriented Gradients

We define a feature transform that estimates orientation of sketch lines based on the results of applying a 2d Gaussian derivative filter to the sketch (see Figure 5.1). The idea is to convolve the sketch with the  $x$  and  $y$  partial derivatives of a Gaussian to get a reliable estimate of the gradient of the sketch. Using the gradient, we can eventually compute orientations.

A 2d normalized Gaussian filter is defined as

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (5.1)$$

and its partial derivatives as

$$\partial G(x, y)/\partial x = -x/\sigma^2 G(x, y) \quad (5.2)$$

$$\partial G(x, y)/\partial y = -y/\sigma^2 G(x, y) \quad (5.3)$$

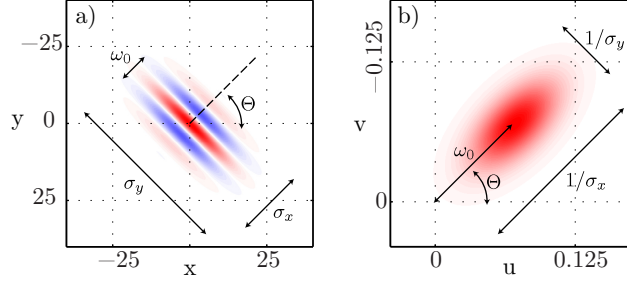
We write  $\mathbf{g}_{xy} = \nabla \mathbf{S}_{xy}$  for the gradient of  $\mathbf{S}$  at coordinate  $(x, y)$  and  $o_{xy} \in [0, \pi)$  for its orientation. We compute the gradient of a sketch by convolution (denoted by  $*$ ) with the two partial Gaussian derivatives filter:

$$\mathbf{g}_{xy} = \begin{pmatrix} \partial G(x, y)/\partial x * \mathbf{S} \\ \partial G(x, y)/\partial y * \mathbf{S} \end{pmatrix} \quad (5.4)$$

We now compute orientation  $o_{xy} \in [0, \pi)$  as

$$o_{xy} = \arccos[(g_y/\|\mathbf{g}\|) \operatorname{sgn}(g_x)] \quad (5.5)$$

We skip computing orientations for coordinates where  $\|\mathbf{g}\| = 0$ . We coarsely bin  $\|\mathbf{g}\|$  into  $k$  orientation bins according to  $o$ , linearly interpolating into the neighboring bins to avoid sharp energy changes at bin borders. We define neighborhood circularly, such that a fraction of the energy that is added to the



**Figure 5.2:** Gabor filter: a) spatial domain, b) frequency domain. Parameters:  $\sigma_x = 5, \sigma_y = 10, \omega_0 = 0.1, \Theta = \pi/4$

last orientation bin is also added to the first orientation bin. This results in  $k$  orientational response images  $\mathbf{R}_i$ , each encoding the fraction of orientational energy of sketch lines at a given discrete orientation value and pixel coordinate.

### 5.3 GALIF: Gabor Local Line-Based Feature

The GALIF feature transform performs orientation estimation using a bank of Gabor filters [Gabor 1946]. A Gabor filter bank is known to be optimal in the sense of achieving maximal joint resolution in the 2d spatial as well as 2d frequency domain [Daugman 1985]. Intuitively speaking, a Gabor filter is optimal to locate a feature with high spatial precision while at the same time exactly measuring its orientation/frequency. There is strong experimental evidence that receptive fields in the visual system of mammals are well described by 2d Gabor filters with appropriate parameters [Jones and Palmer 1987] and Daugman [1985] concludes that

“There is a division of labor among simple cells for the resolution of information along the different axes of information hyperspace, some cells, for example, favoring orientation selectivity at the expense of spatial resolution in one direction, and so on.”

Gabor-like basis functions also naturally emerge when sparse coding natural images [Olsen et al. 2009]. This motivates mimicking nature and developing a feature transform based on Gabor filters for encoding human sketches. Instead of fixing the parameters of the underlying filter bank a priori, we optimize them for the task at hand (see Section 7.9).

#### Gabor Filter

A Gabor filter in the *frequency domain* is defined as:

$$\mathbf{g}(u, v) = \exp \left( -2\pi^2 \left( (u_\Theta - \omega_0)^2 \sigma_x^2 + v_\Theta^2 \sigma_y^2 \right) \right) \quad (5.6)$$

where  $(u_\Theta, v_\Theta) = R_\Theta(u, v)^T$  is the standard coordinate system rotated by angle  $\Theta$ . A Gabor filter can be tuned according to several parameters:

$\omega_0$	:	peak response frequency
$\Theta$	:	filter orientation
$\sigma_x$	:	frequency bandwidth
$\sigma_y$	:	angular bandwidth

We visualize such a filter with its corresponding parameters in Figure 5.2. Note that in the frequency domain the filter is simply a Gaussian (Figure 5.2). Multiplication with the sketch in the frequency domain “masks” all content that does not have the filter’s frequency and orientation. The filter responds only to a subset of the lines in a sketch. The feature transform we propose is based on a set of Gabor filters (filter bank) that all share the same parameters, except orientation.

### Orientation-Selective Filter Bank

To compute our feature space transform, we define a filter bank of Gabor functions  $\mathbf{g}_i$  with  $k$  different orientations (and all other parameters fixed). We then convolve the sketch with the Gabor functions from the filter bank to yield a set of filter response images

$$\mathbf{R}_i = \|\text{idft}(\mathbf{g}_i * \text{dft}(\mathbf{I}))\| \quad (5.7)$$

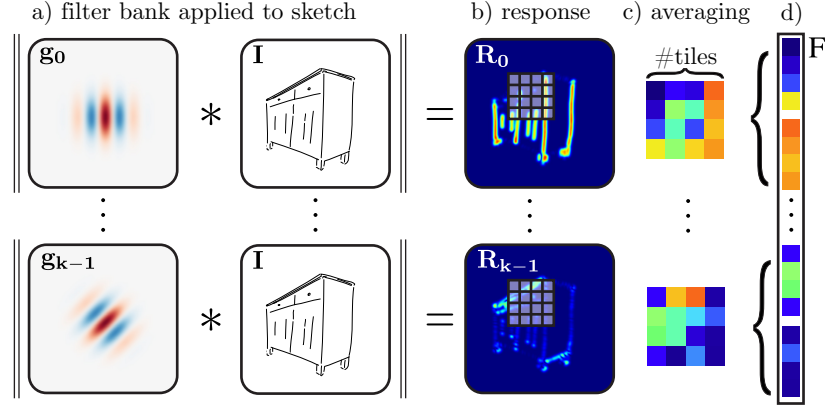
where  $\mathbf{I}$  is the input sketch,  $*$  denotes point-wise multiplication and idft and dft denote the inverse/forward discrete Fourier transform (see Figure 5.3a,b for a visualization).

Note that  $\sigma_y$  determines the amount of overlap between filters  $\mathbf{g}_i$  — depending on the number of orientations in the filter bank. We do not manually fix this value but instead *optimize* it in Section 7.9 such that we achieve optimal retrieval results. Given the number of orientations  $k$  we define the  $\Theta$ ’s used for the filterbank as  $\Theta \in \{0, \pi/k, \dots, (k-1)\pi/k\}$ .

## 5.4 Extracting Local Features

Both the SHOG and GALIF feature transform result in a set of response images  $\mathbf{R}_i$  that encode how much energy a pixel contains at a certain discrete orientation. The two feature transforms differ in how these orientation images are computed, but given the response images they share the same remaining local feature extraction pipeline.

Given a coordinate in image space we consider a regular decomposition of the response images’ area around this coordinate into  $n \times n$  cells  $C_{st}$ . In preliminary experiments, we found this approach to work at least as well as random sampling which in turn has proven superior to keypoint detection [Nowak et al. 2006]. We call the area around a keypoint a “local image patch” and call  $n$  the “number of tiles” (see Figure 5.3b,c). We say  $(x, y) \in C_{st}$  if the pixel with coordinates  $x$  and  $y$  is contained in the cell with index  $(s, t)$ . To achieve invariance of image size, we define the area covered by a local patch (we call this feature size) relative to image area: a feature size of 0.075 means that the local patch covers 7.5% of the image area.



**Figure 5.3:** *GALIF feature extraction pipeline: a) we convolve the input sketch  $I$  with a filter bank of differently oriented Gabor filters  $g_i$  to yield b) response images  $R_i$ . The average responses c) within cells of a local patch form d) a local feature vector  $F$ . The Gabor filters are shown in the spatial domain for visualization purposes.*

We now define a local descriptor  $\mathbf{d}$  as a  $k \times n \times n$  feature vector. In each dimension,  $\mathbf{d}$  stores the average filter response within a cell  $C_{st}$  for orientation  $i$ :

$$\mathbf{d}(s, t, i) = \sum_{(x,y) \in C_{st}} R_i(x, y). \quad (5.8)$$

When inserting a value into  $\mathbf{d}$ , we perform bilinear interpolation in the spatial domain.

We discard features that do not contain sketch lines and finally normalize such that  $\|\mathbf{d}\|_2 = 1$ . This results in a representation that is closely related to the one used for SIFT [Lowe 2004] but stores orientations only [Eitz et al. 2011a]. We visualize the complete local feature extraction pipeline for the GALIF descriptor in Figure 5.3. The pipeline for the SHOG descriptor is similar, only computation of the response images differs.

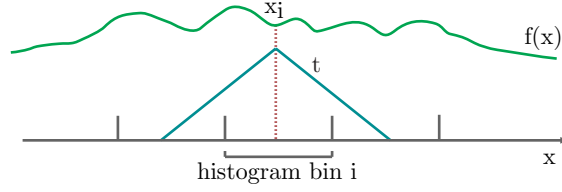
A typical local feature vector resulting from this pipeline is 64-dimensional, using  $4 \times 4$  spatial bins and 4 orientational bins. We evaluate how varying these parameters influences retrieval performance in Section 7.9.

### Local Feature Size and Sampling

While in computer vision applications the size of local patches used to analyze photographs is often quite small (e.g.,  $16 \times 16$  pixels [Lazebnik et al. 2006]), sketches contain little information at that scale and larger patch sizes are required for an effective representation. In our case, we typically use local patches covering an area of around 5% to 25% of the size of  $S$ .

We use  $32 \times 32 = 1,024$  regularly spaced sample points in  $S$  and extract a local feature  $\mathbf{d}$  for each point. The resulting representation is a so-called bag-of-features  $\mathcal{D} = \{\mathbf{d}_i\}$  where a sketch is represented as a large number of local features  $\mathbf{d}_i$  that encode local orientation estimates but do *not* carry any information about their spatial location in the sketch.





**Figure 5.4:** 1d example of fast histogram construction. The total energy in histogram bin  $i$  corresponds to  $(f \star t)(x_i)$  where  $x_i$  is the center of bin  $i$ .

### Accelerating Feature Extraction

Due to the relatively large patch sizes we use, the regions covered by the local features *significantly overlap* and each single pixel gets binned into about 100 distinct histograms. As this requires many image/histogram accesses, this operation can be quite slow.

We speed up building the local descriptors by observing that the total energy accumulated in a single spatial histogram bin (using linear interpolation) is proportional to the convolution of the local image area with a 2d tent function having an extent of two times bin-width. We illustrate this property for the 1d case in Figure 5.4. As a consequence, before creating spatial histograms, we first convolve the response images  $\mathbf{R}_{1...k}$  with the corresponding function (which we in turn speed up using the FFT). Filling a histogram bin is now reduced to a *single* lookup of the response at the center of each bin. This lets us efficiently extract a large number of local histograms which will be an important property later in this thesis.

In the following, we build a more compact representation by quantizing local features against a visual vocabulary [Zhu et al. 2000; Sivic and Zisserman 2003].

## 5.5 Building a Visual Vocabulary

The space of possible local features is huge. For example, a 64-dimensional feature vector has  $256^{64}$  unique instances (each dimension encoded using eight bits). However, the space of *natural* features (i.e. those appearing in actual human sketches) is presumably much more sparsely populated. Additionally, many of the natural local features form *perceptually* similar clusters. This is a result of human invariance to certain image properties making image patches corresponding to slightly different features vectors appear visually identical.

These two properties (sparseness and perceptual similarity) are exploited when constructing a visual vocabulary. Intuitively speaking, a visual vocabulary is the visual equivalent to a vocabulary in a natural language: the elements (visual words) correspond to words in a language and we can (hopefully) discriminatively represent any sketch using just these visual words. A visual vocabulary is always a *model* of the space of local features and as such, very naturally, information is lost. In the ideal case, when the information lost corresponds exactly to the invariance properties of the human visual system, this information loss can even be desirable.

The key idea for constructing such a visual vocabulary is to identify a sufficiently small set of visual words that reasonably represent the original local features sampled from a training dataset [Sivic and Zisserman 2003]. Identifying good representatives is typically achieved by some sort of clustering. Visual words are typically sampled from the local feature space (i.e. in the case of a 64-dimensional space a visual word is 64-dimensional as well) and are typically selected as instances from a training dataset of local features.

It is often difficult to formalize criteria for constructing an optimal visual vocabulary as its quality can typically only be measure indirectly in terms of its influence on retrieval performance or classification accuracy. Consequently, a variety of strategies for constructing visual vocabularies have been proposed, for example based on k-means clustering [Leung and Malik 2001; Sivic and Zisserman 2003; Csurka et al. 2004; Wu et al. 2009], more general Gaussian mixture models [Winn et al. 2005; Li et al. 2011; Chatfield et al. 2011] or based on a vocabulary tree generated using hierarchical k-means clustering [Nister and Stewenius 2006; Agarwal et al. 2009]. Given an existing vocabulary, Cai et al. [2010] show how to learn optimized weights for codebooks such that images within a category achieve a larger similarity measure than across categories.

A final, additional criteria for constructing visual vocabularies is computational efficiency, which can become an important factor when computing large vocabularies from large training datasets [Nister and Stewenius 2006; Philbin et al. 2007; Boix et al. 2012; Avrithis and Kalantidis 2012].

## Our Approach

We follow the baseline approach of [Sivic and Zisserman 2003]: using a training set of  $n$  local descriptors  $\{\mathbf{d}\}$  randomly sampled from our dataset of sketches, we construct a visual vocabulary using k-means clustering [Lloyd 1982], which partitions the descriptors into  $k$  disjunct clusters  $\mathcal{C}_i$ . More specifically, we define our visual vocabulary  $\mathcal{V} = \{\boldsymbol{\mu}_i\}$  to be the set of vectors resulting from minimizing

$$\mathcal{V} = \arg \min_{\{\boldsymbol{\mu}_i\}} \sum_{i=1}^k \sum_{\mathbf{d}_j \in \mathcal{C}_i} \|\mathbf{d}_j - \boldsymbol{\mu}_i\|^2 \quad (5.9)$$

with

$$\boldsymbol{\mu}_i = 1/|\mathcal{C}_i| \sum_{\mathbf{d}_j \in \mathcal{C}_i} \mathbf{d}_j. \quad (5.10)$$

Each cluster center  $\boldsymbol{\mu}_i$  forms one visual word and those are the basis for computing the final representation of a sketch which we describe in the following section.

## 5.6 Quantizing Features

Given a visual vocabulary, the local features extracted from a sketch are quantized against the vocabulary, identifying the visual words contained in the sketch. Given this information, a sketch is encoded as its frequency histogram of visual words [Sivic and Zisserman 2003], potentially applying some spatial pooling strategies to add back spatial information into the representation [Lazebnik et al. 2006].

Quantization of a local feature is typically achieved using hard assignment to the best-matching nearest neighbor in the visual vocabulary [Sivic and Zisserman 2003], using probabilistic distance measures [van Gemert et al. 2008, 2010; Philbin et al. 2008; Chatfield et al. 2011; Li et al. 2011] or, more recently, using sparse coding approaches where each local feature is represented as a sparse linear combination of visual words [Yang et al. 2009, 2010b,a; Boix et al. 2012]

We adopt those strategies and represent a sketch as a *frequency histogram* of visual words  $\mathbf{h}$  — this is the final representation we use throughout this thesis. As a baseline we use a standard histogram of visual words using a “hard” assignment of local features to visual words [Sivic and Zisserman 2003]. We compare this to using “soft” kernel-codebook coding for constructing the histograms [van Gemert et al. 2008; Philbin et al. 2008; Chatfield et al. 2011].

### Hard Quantization

We represent a sketch as histogram of visual word frequency. We compute visual words by hard assignment (quantization) to an entry in the visual vocabulary  $\mathcal{V}$ . We quantize a local feature  $\mathbf{d} \in \mathcal{D}$  by finding the entry in the visual vocabulary with smallest Euclidean distance, representing it as the index  $q(\mathbf{d})$  of the closest visual word [Sivic and Zisserman 2003].

In particular, we define the entries  $h_j$  of the final histogram of visual words  $\mathbf{h}$  as:

$$h_j = |\{q(\mathbf{d}) = j\}|. \quad (5.11)$$

where  $q(\mathbf{d})$  is a *scalar function* that returns the index of the closest cluster center  $\boldsymbol{\mu}$  in the visual vocabulary:

$$q(\mathbf{d}) = \arg \min_j \|\boldsymbol{\mu}_j - \mathbf{d}\|. \quad (5.12)$$

### Soft Quantization

The idea behind kernel codebook coding (we also call this soft quantization in this thesis) is that a feature vector may be equally close to multiple visual words but this information cannot be captured in the case of hard quantization [van Gemert et al. 2008; Philbin et al. 2008; Chatfield et al. 2011; Liu et al. 2011]. Following these ideas we use a kernelized distance between descriptors that encodes weighted distances to all visual words — with a rapid falloff for distant visual words. More specifically, we define our histogram  $\mathbf{h}$  as:

$$\mathbf{h}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{d}_i \in \mathcal{D}} q(\mathbf{d}_i) / \|q(\mathbf{d}_i)\|_1 \quad (5.13)$$

where  $q(\mathbf{d})$  is a *vector-valued quantization function* that quantizes a local descriptor  $\mathbf{d}$  against the visual vocabulary  $\mathcal{V}$ :

$$q(\mathbf{d}) = [K(\mathbf{d}, \boldsymbol{\mu}_1), \dots, K(\mathbf{d}, \boldsymbol{\mu}_k)]^T. \quad (5.14)$$

We use a Gaussian kernel to measure distances between samples, i.e.

$$K(\mathbf{d}, \boldsymbol{\mu}) = \exp(-\|\mathbf{d} - \boldsymbol{\mu}\|^2 / 2\sigma^2). \quad (5.15)$$

Note that in Equation 5.13 we normalize  $\mathbf{h}$  by the number of samples. As a result, the representation is not sensitive to the total amount of local features in a sketch, but rather to local structure and orientation of lines. We use  $\sigma = 0.1$  in our experiments.

### 5.7 Discussion

Both feature transforms are intimately related: they both rely on the concept of estimating orientation of sketched feature lines and encoding the distribution of orientation within small local regions. The feature transforms differ in how the orientation estimation is performed, i.e. using a Gaussian derivative filter in case of the SHOG descriptor and a bank of Gabor filters in case of the GALIF feature. While the SHOG transform is governed by only a single filter parameter (standard deviation of the Gaussian filter), the GALIF transform offers more flexibility and thus allows us to carefully tune its parameters to achieve optimal results for sketches. We carefully evaluate in Chapter 7 if this additional flexibility can be exploited to yield a superior representation.

Interestingly, higher order Gaussian derivative filters appear visually quite similar to Gabor kernels (with important theoretical differences) [ter Haar Romeny 2003]. We only use a first order Gaussian derivative filter for the SHOG transform but will analyze later if this is already enough to yield a good representation. The GALIF feature transform is related to the successful global GIST descriptor [Oliva and Torralba 2001]: it also uses a bank of Gabor filters to estimate content orientation at a certain frequency but at a single frequency only and encodes the result using a large number of local descriptors rather than a single global one.

---

## Chapter 6

# Computational Sketch Recognition

---

Sketching as a means of human communication is astonishingly universal across culture and time: even after tens of thousands of years, we can still easily recognize the ancient cave paintings depicted in Figure 1.2. Sketching is also universally used as a means to quickly visualize design ideas, for example, in the movie, automotive and fashion industry as well as in architecture.

A natural question arises from this: can we make computers understand sketches as reliably and effortlessly as humans do? This task is extremely challenging as sketching skills vary a lot among the general population. Some drawings can be extremely abstract and far from the original (see Chapter 3). Additionally, instances of the same object can be visually quite dissimilar but need to be reliably identified as belonging to the same category.

In the existing literature, computational sketch recognition has so far only been studied in very structured domains such as electric circuit diagrams [Sezgin and Davis 2008] or molecular diagrams [Ouyang and Davis 2011] where impressive recognition results have been achieved. Instead, we are interested in making computers understand unstructured sketches of ideally any objects from our visual world such as ‘cars’, ‘tables’ and ‘horses’.

In case reliable computational sketch recognition at the level of human accuracy was possible this would enable a multitude of applications:

- Sketching as a means of interacting with computers for hundreds of millions of illiterate people.
- Improved sketch-based visual search [Chalechale et al. 2005; Eitz et al. 2011a] in cases where shape is more descriptive than keywords.
- Image synthesis applications such as Sketch2Photo [Chen et al. 2009], PhotoSketcher [Eitz et al. 2011b] ShadowDraw [Lee et al. 2011] and HelpingHand [Lu et al. 2012] could potentially benefit from semantic understanding of a sketch to achieve better synthesis results.
- Semantic understanding could lead to better sketch-based 3d modeling tools [Igarashi et al. 1999; Karpenko and Hughes 2006; Nealen et al. 2007],

for example by using the automatically computed semantic information to search for priors for the model being created.

## 6.1 Models

To achieve generalization to unseen instances, we opt to use state of the art supervised machine learning techniques to learn models for computational sketch recognition. As a baseline technique, we employ standard k-nearest-neighbor (kNN) classification which is fast, easy to implement, and does not require an explicit training step. We compare this to multi-class support vector machines (SVM), a popular and effective supervised learning technique [Schölkopf and Smola 2002]. All classification algorithms are trained using the dataset from Chapter 3 and operate in the feature space described in Chapter 5.

### Nearest-Neighbor Classification

Given a histogram  $\mathbf{h}$  we find its  $k$  nearest neighbors (kNN) in feature space using a distance function  $d$ . We classify  $\mathbf{h}$  as belonging to the category that the majority of  $k$  nearest neighbors belongs to [Cover and Hart 1967]. In case of ties (i.e. when two or more classes receive exactly the same number of votes) we classify  $\mathbf{h}$  according to its nearest neighbor from the tied groups. The kNN approach is particularly fast, as it does not require an initial training step.

### SVM Classification

We train *binary* SVM classifiers [Schölkopf and Smola 2002] to make the following decision: does a sketch belong to category  $i$  or does it rather belong to any of the remaining categories? We say  $\text{cat}(\mathbf{h}) = i$  to denote that the sketch corresponding to histogram  $\mathbf{h}$  belongs to category  $i$ . More specifically, for each category  $i$  we learn a classifier function

$$c^i(\mathbf{h}) = \sum_j \alpha_j^i K(\mathbf{s}_j^i, \mathbf{h}) + b^i \quad (6.1)$$

with support vectors  $\mathbf{s}_j$ , weights  $\alpha_j$  and bias  $b$  determined during the SVM training phase. In particular, we use the SMO algorithm for training SVMs [Platt 1998]. As before,  $K(\cdot, \cdot)$  is a kernel (in our experiments, Gaussian) that measures similarity between a support vector and the histogram  $\mathbf{h}$  that we wish to classify. Given a training dataset, we use the sketches from category  $i$  as the positive examples and all the remaining sketches as the negative examples.

SVMs are inherently *binary* classifiers but we wish to distinguish 250 categories from each other. We train 250 classifiers — one for each category, each able to discern its category from the union of the remaining categories (one-vs-all approach). To decide  $\text{cat}(\mathbf{h})$  we classify a sketch as belonging to the category that yields the largest classification response, i.e.

$$\text{cat}(\mathbf{h}) = \arg \max_{i=1, \dots, 250} c^i(\mathbf{h}). \quad (6.2)$$

## 6.2 Recognition Experiments

In this section we determine the best parameters for computational sketch recognition. In all cases, we train on a subset of the sketches while evaluating on a disjunct testing set (cross-validation). We always use 3-fold cross-validation: we partition the respective dataset into three parts, use two parts for training and the remaining part for testing. We use stratified sampling when creating the folds to ensure that each subset contains (approximately) the same number of category instances. We report performance of a model using its accuracy, averaged over all three folds, i.e. the ratio of correctly classified samples to the total number of positive samples.

### Dataset & Features

We use the complete 250 category dataset described in Chapter 3 and rasterize each sketch into a grayscale bitmap of size  $256 \times 256$ . We extract  $28 \times 28 = 784$  local features sampled on a regular grid from each sketch. We use both the SHOG feature transform (Section 5.2) and the GALIF feature transform (Section 5.3) and later evaluate which one is most suitable for the recognition task.

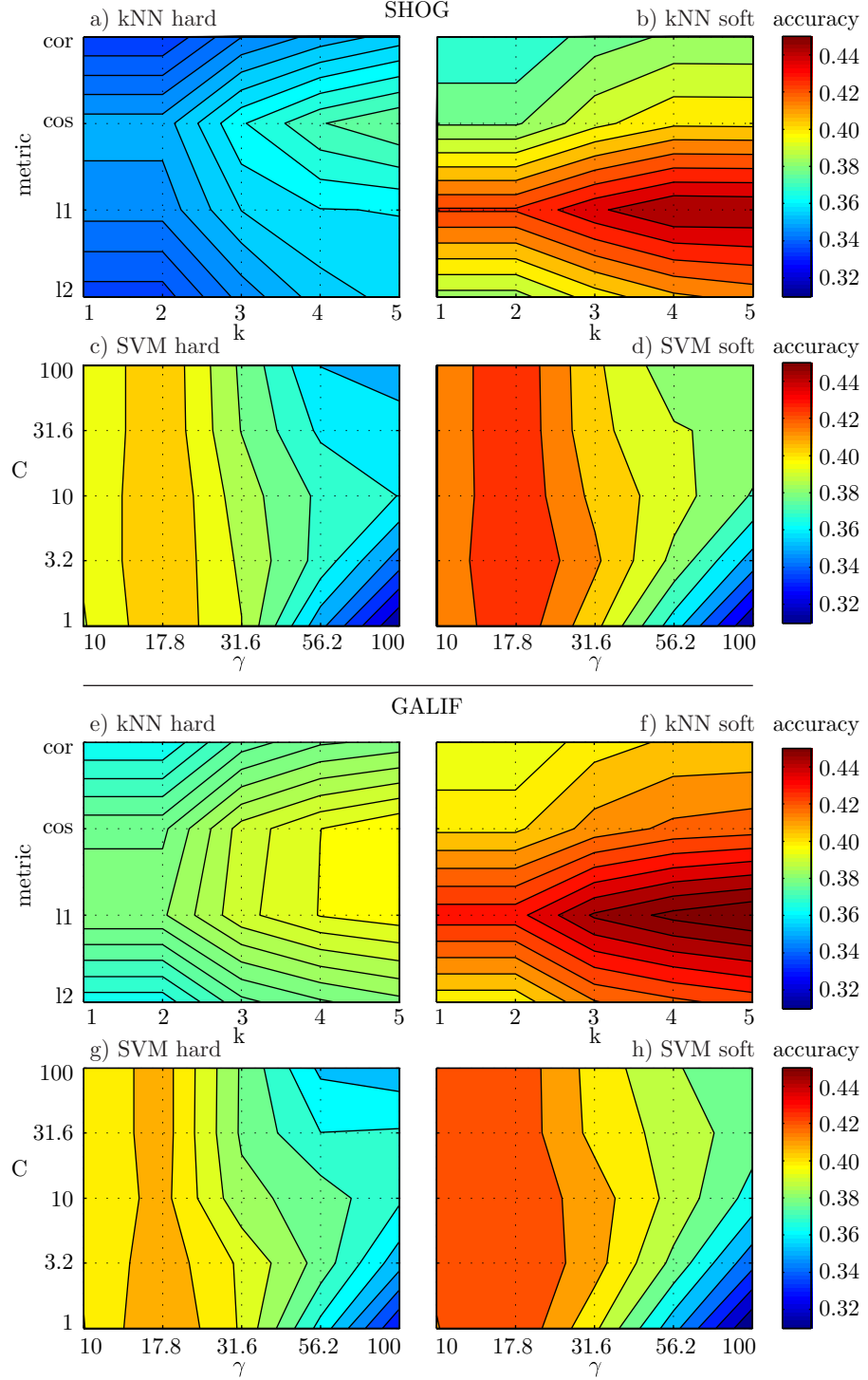
To create a visual vocabulary, we need to learn from a large number of samples. For computational reasons, we generate the visual vocabulary  $\mathcal{V}$  (Equation 5.9) from a randomly sampled subset of all  $20,000 \times 784$  local features (we use  $n = 1,000,000$  features). We set the number of visual words in the vocabulary to 500 as a coarse evaluation indicates good classification rates at a moderate cost for quantization.

We use hard quantization (Equation 5.11) as well as soft quantization (Equation 5.13) to compute a histogram of visual words from the local features of a sketch and later evaluate which one works best for the task at hand. Independently of the feature transform (SHOG vs. GALIF) and quantization method (hard vs. soft), the final representation of a sketch always is a 500-dimensional histogram of visual words. Before training (and classification), we normalize the data such that the features are centered at their mean and have unit standard deviation.

### Model Search

Classification performance of both the kNN as well as the SVM classifier can be quite sensitive to user-chosen model parameters. Therefore we first identify the best performing model parameters for each case. We perform a standard grid search over the 2d space of model parameters. For kNN we use  $k \in \{1, \dots, 5\}$  (number of nearest neighbors used for classification) and distance measures  $d = \{l_1, l_2, \text{cosine}, \text{correlation}\}$ . For SVM we use  $\gamma \in \{10, \dots, 100\}$  (the Gaussian kernel parameter) and  $C \in \{1, \dots, 100\}$  (the regularization constant), both with logarithmic spacing. We use a 1/4 subsample of the whole dataset to speed-up the search for SVM, for kNN we use the full dataset. Given the dataset, we measure average accuracy from 3-fold cross validation.

It turns out that both models are indeed quite sensitive to the parameters chosen: the kNN model with hard quantization performs best when using a cosine distance metric combined with 5 nearest neighbors to vote for the best



**Figure 6.1:** Accuracy of kNN/SVM recognition models using SHOG descriptor (a – d) and GALIF descriptor (e – f) over a range of model parameters. Note that for the kNN models we use the full dataset for 3-fold cross-validation, while for the SVM model we use a 25% subset due to performance reasons.



**Table 6.1:** *Best parameters for kNN/SVM computational sketch recognition models. Note that in this table the SVM model appears to achieve slightly lower performance than the kNN model because only 1/4 the amount of training data is used (due to computational reasons).*

		kNN			SVM		
		k	metric	accuracy	$\gamma$	C	accuracy
SHOG	soft	4	$l_1$	0.45	17.8	3.2	0.43
	hard	5	cosine	0.38	17.8	10	0.41
GALIF	soft	5	$l_1$	0.46	17.8	3.2	0.43
	hard	5	cosine	0.40	17.8	1	0.41

category (Figure 6.1a,e). In contrast to that, when using soft quantization, an  $l_1$  distance metric performs significantly better than any other option we tested for both the SHOG and GALIF descriptor (see Figure 6.1,b,f).

In the case of SVMs, we achieve highest accuracy for  $\gamma = 17.8$ ,  $C = 3.8$  and soft quantization. The SVM model with hard quantization behaves similarly to changes in parameters as the model using soft quantization, although overall performance is significantly lower (see Figure 6.1c,g).

The behavior of both the kNN as well as SVM model over the range of parameters is pretty consistent for both the SHOG as well as GALIF descriptor. The GALIF descriptor achieves slightly higher overall recognition accuracy in case of a kNN model while results are virtually identical for a SVM model. We list the resulting best model parameters in Table 6.1 and use those in the remainder of this thesis.

### Influence of Training Set Size

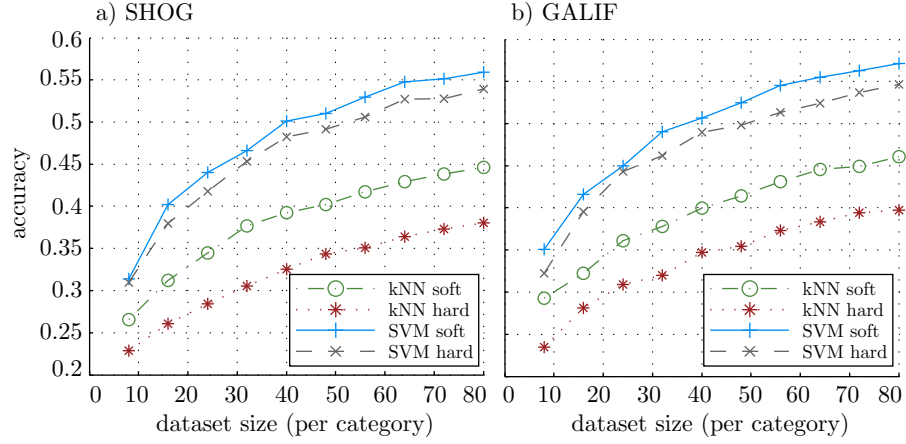
To identify how the amount of training data influences classification accuracy, we split the whole dataset into increasingly larger sub-datasets (8, 16, ..., 80 sketches per category). For each of those sub-datasets, we measure average 3-fold cross-validation accuracy.

We perform the evaluation over dataset size for any combination of classification model (kNN vs. SVM); feature transform (GALIF vs. SHOG) and quantization method (hard vs. soft). This results in a total of eight curves which we visualize in Figure 6.2.

It turns out that computational classification accuracy highly depends on the number of training instances available. This is to be expected as the sketches in many categories are highly diverse (see for example the sketches in the layouts shown in Figure 6.6). The performance gain for larger training set sizes becomes smaller as we approach the full size of our dataset. This suggests that the dataset is large enough to capture most of the variance within each category.

### Human vs. Computational Classification

A manual inspection of the computational as well as human classification results reveals that the confusions humans make are often of hierarchical nature:



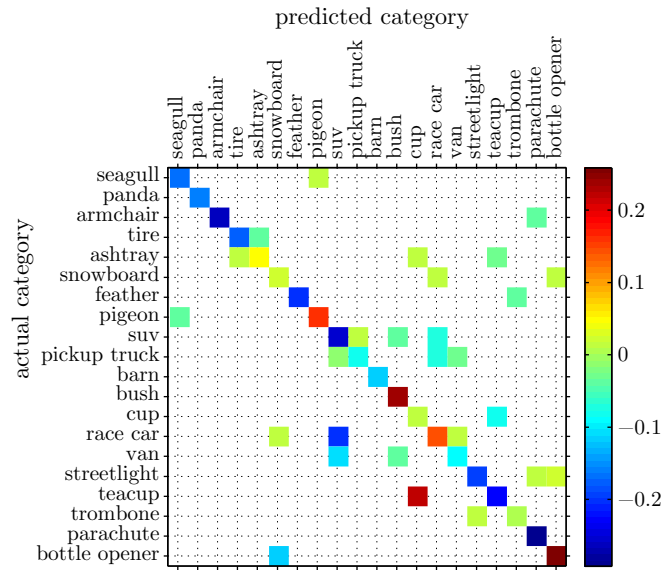
**Figure 6.2:** Influence of training set size on computational sketch classification accuracy for SHOG descriptor (Section 5.2) and GALIF descriptor (Section 5.3). All models use best parameters determined using grid search (see Table 6.1). ‘hard’ and ‘soft’ refer to one nearest-neighbor (Equation 5.11) and kernel-codebook quantization (Equation 5.13) methods, respectively.

**Table 6.2:** Average 3-fold cross-validation accuracy of sketch classification models trained using best performing model parameters (see Table 6.1) using the full dataset of 20,000 sketches. Due to 3-fold cross-validation the actual dataset size for training is 13,333 sketches.

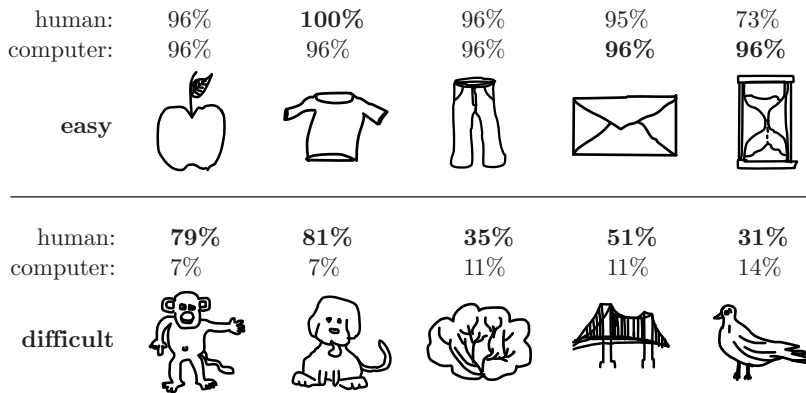
	SHOG		GALIF	
	hard	soft	hard	soft
kNN	0.38	0.45	0.40	0.46
SVM	0.54	0.56	0.55	0.57

for example, ‘bear’ and ‘teddy bear’ are often confused with ‘panda’ (see Figure 4.4). We hypothesize that the participants in our experiment were satisfied once they found a category that matches reasonably well, and may not have noticed the presence of semantically similar categories in our taxonomy. And, indeed, computational classification can perform better in such cases: for ‘arm-chair’ and ‘suv’ the computational model achieves significantly higher accuracy than humans. To visualize such cases, we show the *difference* between human and computational confusion in Figure 6.3 (for the set of categories with overall lowest human recognition performance).

In Figure 6.4 we compare human to computational sketch recognition accuracy for the five sketches with best and the five sketches with lowest computational recognition accuracy, respectively. We observe that human sketch recognition accuracy can be significantly higher than computational sketch recognition accuracy for some categories (e.g., 79% vs. 7% for ‘monkey’; 81% vs. 7% for ‘dog’). This suggests that the quality of the sketch dataset (Chapter 3) is high enough such that sketches from these categories are indeed discernible for humans. However, these categories are very challenging for our current models,



**Figure 6.3:** Confusion matrix for selected categories showing *difference* between human and computational classification performance. Positive (red) entries mean humans are better. We hide the zero-level for clarity.



**Figure 6.4:** Human vs. computational sketch recognition performance. Top row: sketches with highest computational recognition accuracy. Bottom row: sketches with lowest overall computation recognition accuracy.

which suggests that even better computational models and features are needed to achieve accuracy comparable to the human level. Computational classification — in the case of a failure — sometimes makes predictions that are far off from what a human would possibly predict. It seems that despite the large dataset humans are still much better at generalizing from a smaller number of samples.

### Computational Classification Summary

Our experiments with computational classification clearly demonstrate that an SVM model with kernel codebook coding for constructing the histograms is superior to other methods. A kNN classification performs *significantly* worse for this task. Also, for both models and feature transforms, soft quantization performs significantly better than hard quantization (the difference between both quantization methods is more pronounced for the kNN model). In the remainder of this thesis, we exclusively utilize the best performing computational model (RBF SVM with kernel codebook coding) to implement several applications.

### 6.3 Unsupervised Dataset Analysis

In this section we perform an automatic, unsupervised analysis of the sketch dataset making use of the feature space developed in Chapter 5. In this feature space each sketch is represented as a 500-dimensional frequency histogram of visual words  $\mathbf{h}$  using soft quantization (see Equation 5.13). We would like to provide answers to the following questions:

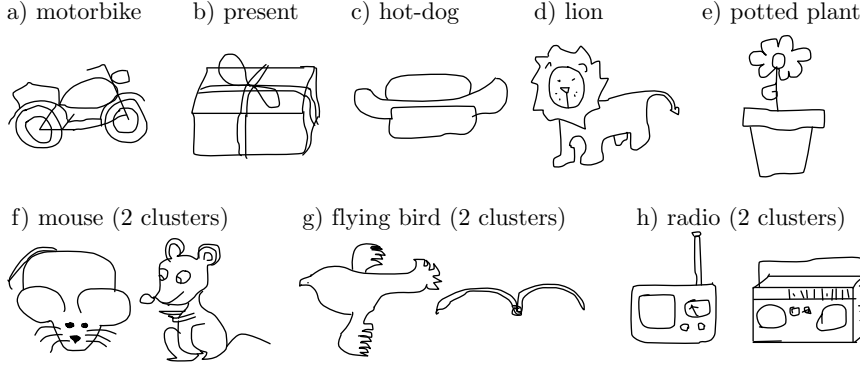
- What is the distribution of sketches in the proposed feature space? Ideally, we would find clusters of sketches in this space that clearly represent our categories, i.e. we would hope to find that features *within* a category are close to each other while having large distances to all other features.
- Can we identify iconic sketches that are good representatives of a category? Such iconic representatives could “enable effective summarization, visualization, and browsing” [Raguram and Lazebnik 2008].
- Can we visualize the distribution of sketches in feature space? This would help build an intuition about the representative power of the feature transforms developed in Chapter 5.

#### Mean-Shift Clustering

The feature space we operate in is sparsely populated (only 20,000 points in a high-dimensional space). This makes clustering in this space a difficult problem. Efficient methods such as k-means clustering do not give meaningful clusters as they use rigid, simple distance metrics and require us to define the number of clusters beforehand. Instead, we use variable-bandwidth mean-shift clustering with locality sensitive hashing to speed up the underlying nearest-neighbor search problem [Georgescu et al. 2003]. Adaptive mean-shift estimates a density function in feature space for each histogram  $\mathbf{h}$  as:

$$f(\mathbf{h}) = \frac{1}{n} \sum_{i=1}^n \frac{1}{b_i^d} K \left( \|\mathbf{h} - \mathbf{h}_i\|^2 / b_i^2 \right),$$

where  $b_i$  is the bandwidth associated with each point and  $K$  is again a Gaussian kernel. Given this definition, we compute the modes, i.e. the maxima of  $f$  by using an iterative gradient ascent approach. As is standard, we assign all features  $\mathbf{h}$  that are mean-shifted close to the same maximum to a common cluster.



**Figure 6.5:** A sample of the representative sketches automatically computed for each category. Bottom row: categories that produce multiple clusters and thus more than one representative.

### Intra-Category Clustering Analysis

We perform a local analysis of our dataset by independently running adaptive mean-shift clustering on the descriptors of each individual category. The resulting average number of clusters within our 250 categories is 1.39: the sketches *within* most categories in our dataset are reasonably self-similar. We can, however, identify categories with several distinct clusters in feature-space — for these categories people seem to have more than one iconic representation. We visualize examples of such categories in the bottom row of Figure 6.5.

### Iconic or Representative Sketches

We denote by  $\mathcal{C}_i$  the set of all descriptors belonging to cluster  $i$ . To identify *iconic* sketches that might be good representatives of a cluster and thus the corresponding category, we propose the following strategy: a) compute the average feature vector  $\mathbf{a}_i$  from all features in that cluster:

$$\mathbf{a}_i = 1/|\mathcal{C}_i| \sum_{\mathbf{h}_j \in \mathcal{C}_i} \mathbf{h}_j.$$

And b) given  $\mathbf{a}_i$  find the closest actual descriptor — our final representative  $\mathbf{r}_i$  — in that cluster:

$$\mathbf{r}_i = \arg \min_{\mathbf{h}_j \in \mathcal{C}_i} \|\mathbf{h}_j - \mathbf{a}_i\|^2.$$

The sketches corresponding to the resulting  $\mathbf{r}_i$ 's are often clear representative sketches of our categories and we show several examples in Figure 6.5.

### Dimensionality Reduction

To visualize the distribution of sketches in the feature space we apply dimensionality reduction to the feature vectors from each category. We seek to reduce their dimensionality to two dimensions such that we can visualize their distribution in 2d space, see Figure 6.6. Using standard techniques such as PCA

or multi-dimensional scaling for dimensionality reduction results in crowded plots: many data points fall close together in the mapped 2d space, resulting in unhelpful layouts.

Van der Maaten and Hinton [2008] propose t-distributed stochastic neighbor embedding (t-SNE), a dimensionality reduction technique that specifically addresses this crowding problem: t-SNE computes a mapping of distances in high-dimensional space to distances in low-dimensional space such that smaller pairwise distances in high-dimensional space (which would produce the crowding problem) are mapped to larger distances in 2d while still preserving overall global distances.

We apply t-SNE to the feature vectors from all categories separately. The resulting layouts are an intuitive tool for quickly exploring the distinct types of shapes and drawing styles used by humans to represent a category. Some of the observed variations are continuous in nature (e.g. sketch complexity), but others are surprisingly discrete — there tend to be one or two canonical viewpoints or poses in a given category (see Figure 6.6 for several examples).

## 6.4 Applications

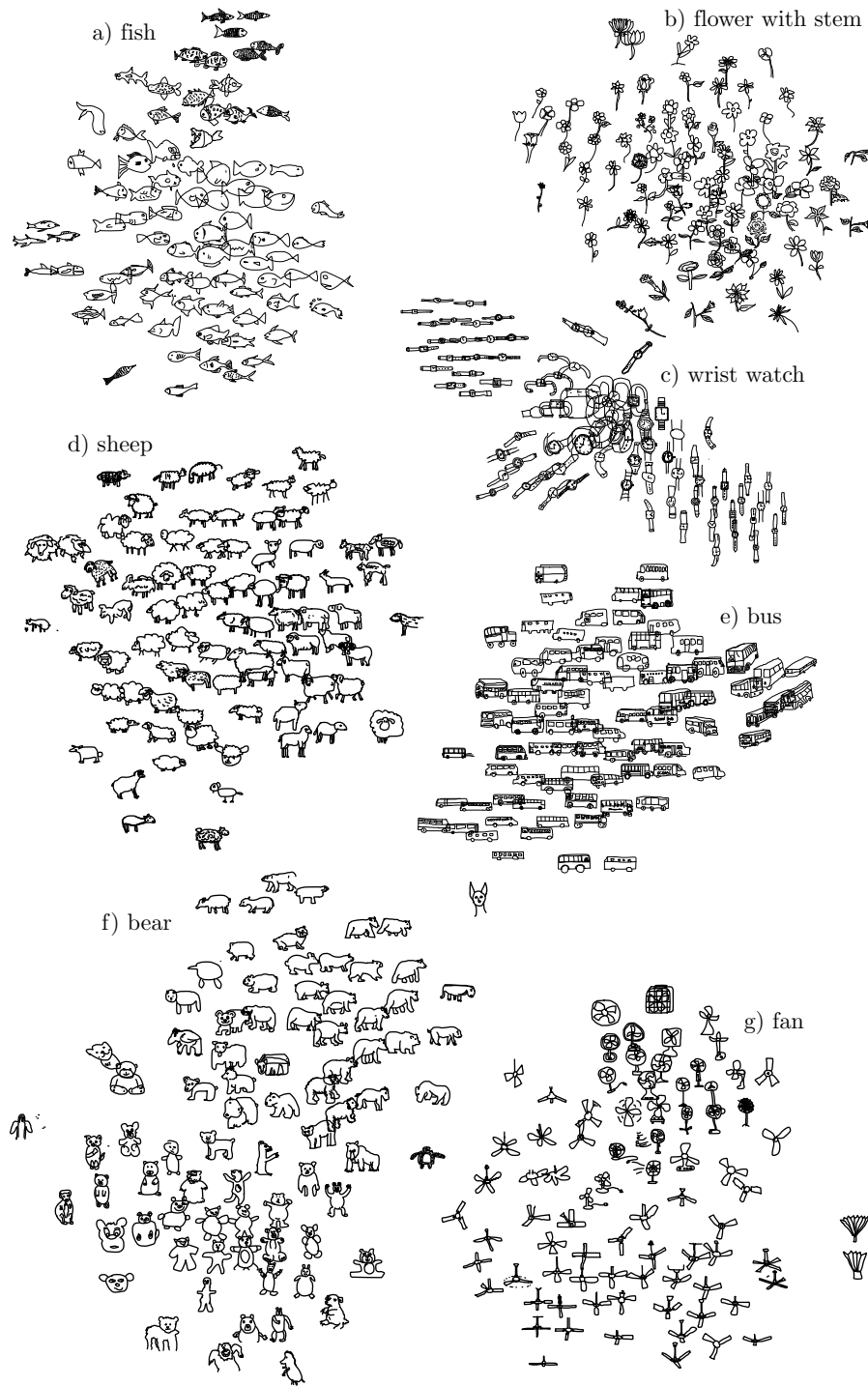
Computational sketch recognition lets us design several interesting applications. We build upon on the multi-class support vector sketch recognition engine (Section 6.1) using the SHOG feature transform and the corresponding best model parameters (Table 6.1). We train the model on the full dataset of 20,000 sketches (Section 3.2).

### Interactive Sketch Recognition

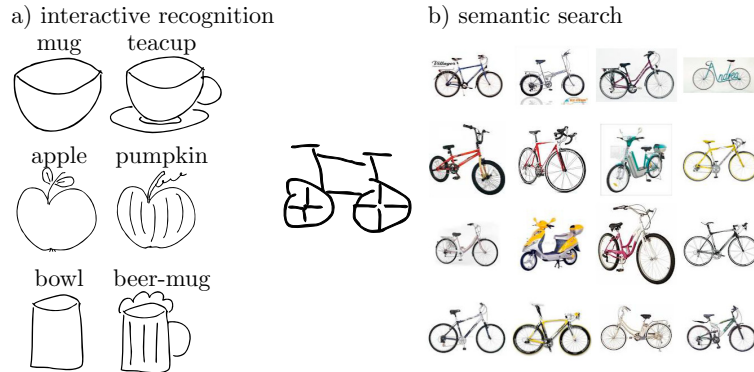
We propose an interactive sketch recognition engine that recognizes human object sketches in real-time. After each stroke a user draws, we run the following recognition pipeline:

1. Extract local features from the sketch (Section 5.4) and quantize them against the visual vocabulary (Equation 5.13). This gives us a histogram of visual words  $\mathbf{h}$ .
2. Classify  $\mathbf{h}$  using the SVM model (Equation 6.2).
3. Display the top-20 categories with highest scores (rather than simply displaying the single best category).

The resulting application is highly interactive (about 100 ms for the complete recognition pipeline on a modern computer). Users report that the application is fun and intuitive to use. In case the classification is not as expected or the desired classification does not come up on rank one, we observe that users explore small modifications of their sketch until the classification is finally correct. Users are often amazed that even their very first sketch is correctly recognized — our system supports a large variety of sketching styles by learning from a large real-world set of sketches. As sketching is very natural to humans, there is basically no learning required to successfully use our system. The interactive application works well with either a touchscreen (as on a tablet) or with a mouse.



**Figure 6.6:** Automatic layout of sketches generated by applying *t*-SNE dimensionality reduction [van der Maaten and Hinton 2008] to the feature vectors within a category.



**Figure 6.7:** Applications enabled by semantic sketch recognition: a) stable interactive recognition that reliably adapts its classification (left column) when adding additional detail (right column). b) semantic sketch-based image search, the only input is the sketch on the left.

Recognition is also reasonably stable: as soon as a sketch is close to completion, oversketching, additional strokes, and even a small number of random background scribbles typically do not influence the recognition. On the other hand, if there are only slight differences between two categories, a single stroke can make all the difference: adding a handle to a mug reliably turns it into a teacup, sketching vertical indentations on an apple turns it into a pumpkin (see Figure 6.7).

### Semantic Sketch-Based Image Retrieval

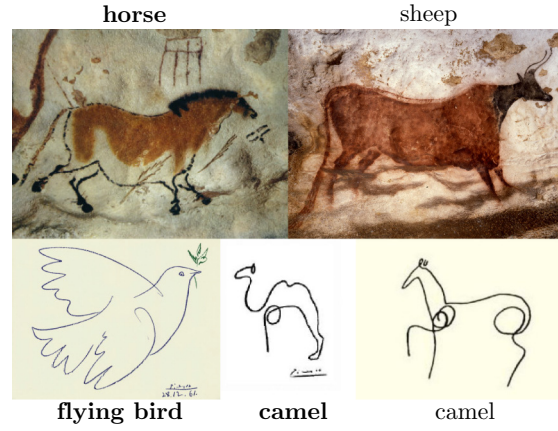
Compared to existing sketch-based retrieval approaches [Chalechale et al. 2005; Eitz et al. 2011a; Shrivastava et al. 2011], we are now able to identify the *semantic* meaning of what a user sketches — even if the sketch is geometrically far from its corresponding real-world shape. This is a situation where traditional sketch-based retrieval engines naturally have problems.

We propose the following extension to sketch-based image retrieval: a) perform classification on the user sketch and query a traditional keyword based search engine using the determined category; b) (optionally) re-order the resulting images according to their geometric similarity to the user sketch. For step b) we can use any traditional sketch-based search engine. We demonstrate step a) of this approach on a dataset of 125,000 images downloaded from Google Images (for each category the top 500 images returned when searching for the keyword), see Figure 6.7 for an example.

### Recognizing Artistic and Historical Sketches

A challenging evaluation for recognition systems is whether the method can generalize to styles not seen at training time. If sketch representations are really a universal, shared vocabulary for humans then our algorithm should be able to recognize existing pictographs from other domains. We present a limited example of this by running our sketch recognition pipeline on two





**Figure 6.8:** *Computational recognition of artistic/ancient sketches. Predictions by our system are shown along with the corresponding input sketch (correct predictions are marked with bold font).*

famous sets of sketches — animal sketches by Pablo Picasso, and ancient cave paintings from Lascaux, France (we converted the cave paintings into sketches by manually tracing their feature lines). Our system predicts the dove, camel and horse correctly, and the antelope, which is not part of our categories, is classified as a sheep (see Figure 6.8).

## 6.5 Discussion

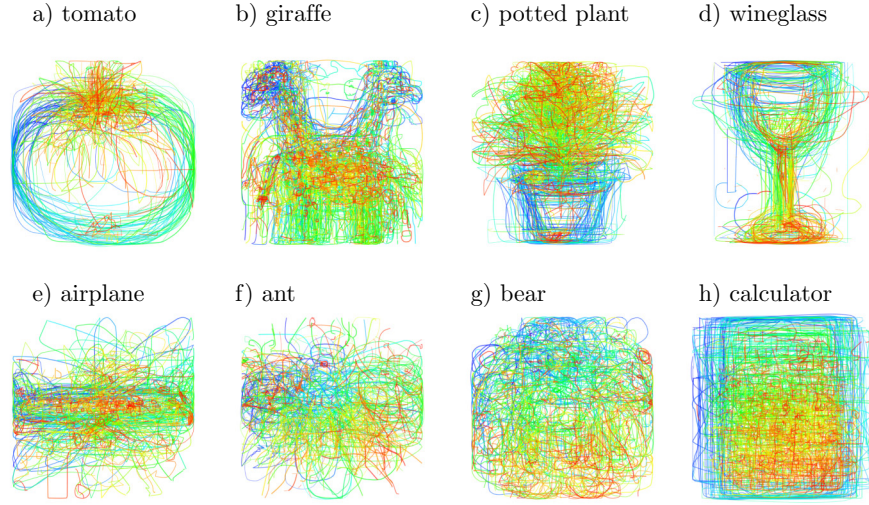
The feature space for sketches proposed in Chapter 5 builds upon the successful bag-of-features model. However, this approach also comes with certain limitations, which we discuss in this section.

### Spatial Layout of Strokes

The features do not encode spatial location, although the meaning of certain features might be dependent on context in the sketch. We experimented with the well known spatial pyramid representation [Lazebnik et al. 2006] (an overcomplete hierarchy of bags-of-words) but it did not significantly improve performance. We hypothesize that better representations of spatial information would significantly improve performance, but those representations might be distinct from the features developed for the image domain. We have not analyzed if it is advantageous to make features rotation invariant. It appears this would serve recognition at least for some categories, where sketches are clearly rotated/reflected versions of each other, see for example Figure 6.6 and Figure 6.9.

### Temporal Information

The raw stroke-based sketch data contains information about the temporal order in which humans have drawn those sketches. The order of strokes seems to



**Figure 6.9:** *Temporal order of strokes averaged for all sketches within a category. We color-code normalized time, blue: beginning, red: end. In the first row we show examples that are recognizable even from the average which suggests that humans are pretty consistent when drawing those objects. In the second row we show examples that are difficult to recognize from the average because a variety of different shapes/orientations is used to depict those objects.*

be quite consistent for certain types of sketches, see Figure 6.9 for several examples. Recent research shows that a plausible order can even be automatically generated [Fu et al. 2011]. In our current representation we do not exploit temporal order of strokes. We have performed initial experiments with descriptors that additionally encode temporal order but have so far only achieved small improvements at the cost of a much higher-dimensional representation. Nevertheless, we believe that this is an interesting and fruitful area for further research. Note that our dataset can be directly exploited for such experiments as it comes with full temporal information.

### Sketch Representation

We have chosen to compute the bag-of-features from *rasterized* sketches. While this is general and accurate, there is no generative process to map from the features back to a sketch. A stroke-based model might be more natural and facilitate easier synthesis applications such as simplification, beautification, and even synthesis of novel sketches by mixing existing strokes.

---

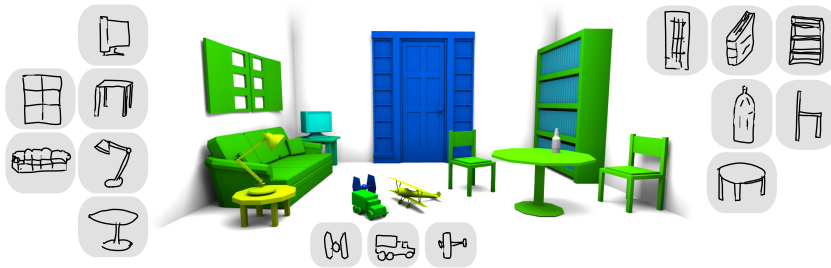
## Chapter 7

# Sketch-Based Shape Retrieval

---

A sketch-based interface for 3d shape retrieval can be essential in situations where the desired shape is easy to sketch but difficult to describe. For example, the exact shape of a 3d model of a vase (see Figure 7.2) is difficult to describe using just a few keywords. Additionally, it is highly unlikely that a collection is tagged so carefully that each model comes with keywords that accurately describe its shape.

Our approach for sketch-based retrieval of 3d models is based purely on visual analysis of the meshes in a collection. This is motivated by the observation that a large part of our perception of shapes stems from their salient features, usually captured by dominant lines in their display [Hoffman and Singh 1997]. Recent research on such feature lines has shown that a) people mostly draw the same lines when asked to depict a certain model, and b) the shape of an object is well represented by the set of feature lines generated by recent line drawing algorithms [Cole et al. 2009]. Consequently, we suggest an image based approach for 3d shape retrieval that exploits the similarity between human sketches and the results of state of the art line drawing algorithms. We make use of successful techniques from other domains where appropriate and provide the following novel contributions:



**Figure 7.1:** *A complete scene with objects retrieved using our sketch-based system in a total time of about two minutes.*



**Figure 7.2:** *Examples of 3d objects that are difficult to describe using just a few keywords but can be easily sketched: human figures (a) and vases (b). For both examples it is highly unlikely that each model comes with keywords that accurately describe its shape.*

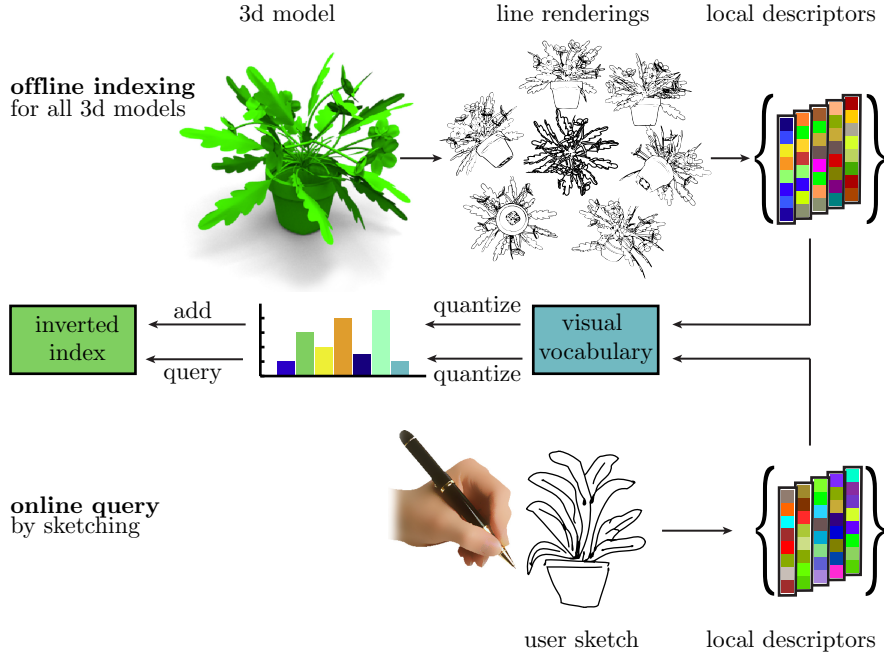
- A large-scale benchmark for sketch-based retrieval systems. The benchmark is based on a real-world dataset of 1,814 sketches gathered from a large variety of participants in a perceptual experiment. We provide this dataset as a free resource.
- New feature transforms based on Gaussian derivative filters/a bank of Gabor filters that are tuned to the requirements of sketch-based shape retrieval (see Chapter 5). These descriptor outperforms other existing transformations.
- A general approach to determine optimal parameters for such feature transformations. We demonstrate that even existing systems can be improved using this approach.

Overall, this leads to a system with high quality retrieval performance as we demonstrate in an objective evaluation using a large variety of real-world user sketches. We also demonstrate the power of our system in Figure 7.1 where we gather all objects for a complete scene in about two minutes. However, we also find that the real-world dataset of sketches gathered in the experiment is challenging for current systems. In particular, our dataset reveals that allowing only closed contour curves for retrieval [Chen et al. 2003] oversimplifies reality: a large majority of our participants’ sketches contain a substantial amount of interior lines.

### 7.1 Overview

We build our retrieval engine upon a *bag-of-features* (BoF) model [Squire et al. 1999; Sivic and Zisserman 2003], which has become the method of choice for affine invariant image retrieval. The basic idea of this approach is to compare images based on a histogram of features. In Chapter 5 we define the representation we use in detail. For the reader’s convenience we summarize again an outline of the required steps:

1. Select the location and size of features in the images.



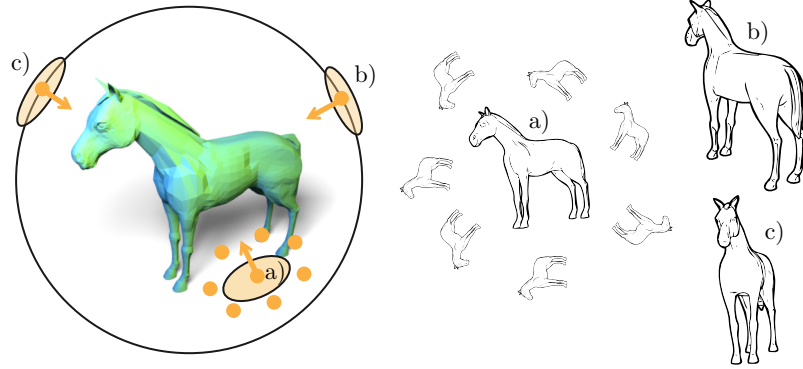
**Figure 7.3:** *Sketch-based 3d shape retrieval pipeline. Our retrieval engine, as most modern visual search engines relies on features extracted from each shape represented as a set of numerical values. Those descriptors are then compared against the ones extracted from the query at search time.*

2. Transform the pixel set of the feature into a (usually) smaller dimensional feature vector.
3. Find the closest match of this vector in a set of predetermined clusters.
4. For the whole image, count the occurrences of cluster matches.

The histogram of cluster matches generates a signature for the image. Compared to the full sketch this signature is low dimensional and facilitates fast matching. The set of clusters is usually derived by clustering the feature vectors found in the images of the database.

## 7.2 View-Based Matching

It is not immediately clear how to use this pipeline for the retrieval of 3d objects based on queries sketched in 2d. Our main idea in this regard is to *generate a set of 2d sketch-like drawings from the objects* for each object in the database. We argue that there are several reasons to perform matching in 2d rather than trying to directly align a user sketch to the 3d shape: the input to the system is 2d and contains large errors which might not even be physically plausible in 3d. Additionally, most sketches depict shapes that are not exactly part of the database, so a perfect alignment of a view to the sketch is impossible. Finally,



**Figure 7.4:** *View generation pipeline. Left: we place virtual cameras a) – c) on the bounding sphere of a mesh, looking into the center of the mesh (in practice we choose a much larger number). Right: this results in the corresponding line-rendered views a) – c). Choosing random up-vectors for each camera results in similar views for nearby viewpoints (dots around camera a)), with each having a different orientation (horses around rendering a)). This approximates a globally rotation invariant representation.*

there is experimental evidence that this resembles how humans recognize 3d objects [Bülthoff and Edelman 1992].

Matching in 2d turns the problem into a comparison of a single query image to several two-dimensional projections per object in the database. This general approach still leaves several design choices for the different steps:

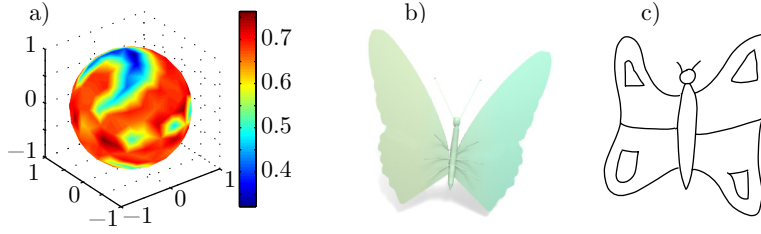
1. for an object, define the set of projection directions
2. for the projection, define a certain line drawing style
3. for a line drawing, define the set of sampling locations
4. for a drawing sample, define the feature transform

The first two items are only necessary in a preprocessing phase, in which we generate the set of clusters from the database. The database forms a finitely sized “visual vocabulary”  $\mathcal{V}$ . This yields a representation for each projection in the database by a distribution of “visual words” (see Chapter 5 for details and Figure 7.3 for a visualization). In the query phase, we compute a similar distribution for the input and compare it against the elements of the database.

In the following sections we describe several details of our search engine. We lay out different choices for the design and make the final decision on the best design choice later, based on an evaluation against the sketches gathered in the experiment (Section 3.3).

### 7.3 Selecting Views

As we have no a priori knowledge about which viewpoint a user chooses when sketching an object, it is vital that the underlying retrieval system encodes all



**Figure 7.5:** Best-view selection: a) best view probability map predicted by the SVM model, b) predicted best view, c) user sketch

*potential* viewpoints. To reduce the set of candidates to a feasible number, we follow existing approaches [Chen et al. 2003] and only consider views that result in the complete model being rendered to screen. Specifically, we consider view directions towards the barycenter of the 3d shape which reduces their definition to points on a sphere. We randomly choose a camera up-vector for each viewpoint. Consequently, nearby view-points have different orientations assigned to them. This approximates a globally rotation-invariant indexing of the shapes. We visualize this pipeline in Figure 7.4. We evaluate two possible strategies for selecting the viewpoints:

### Uniformly Distributed Views

We generate  $d$  *uniformly distributed* directions on the unit sphere using k-means clustering. Starting from a highly tessellated triangle mesh of a unit sphere  $M = \{V, T\}$ , with  $V$  a set of vertices and  $T$  a set of triangles, we select a set  $S$  of  $d$  random seed vertices among  $V$  and perform Lloyd relaxations iteratively. After convergence, we return the resulting Voronoi cell centers as the view directions  $v_i$ . We use  $d \in \{7, 22, 52, 102, 202\}$ . The number of samples  $d$  is an important parameter and we determine its optimal value in Section 7.9.

### Perceptually Best Views

While a uniform sampling does guarantee that we sample all *possible* viewpoints, it is intuitively clear that humans do not draw them with equal probability — when sketching a cow, we would probably rather choose a side-view than a viewpoint from the bottom. We ask if we can exploit this intuition computationally: can we *learn* a model of viewpoint preference for sketch-based shape retrieval? We would then only use those views to represent a 3d model that are likely to be sketched by humans. Compared to a set of densely sampled viewpoints, our hope is to achieve both faster and better retrieval results:

1. Less views would be needed to represent a shape, this could potentially speed up the search.
2. We would learn a visual vocabulary only from views that are likely to be sketched by users — this could lead to a higher quality visual vocabulary only containing elements that actually get sketched.



Recent work on viewpoint selection shows that human view preference can be highly correlated with several simple measures, such as silhouette length and projected area [Dutagaci et al. 2010; Secord et al. 2011]. We follow those approaches and make use of the models in the training set of the Princeton Shape Benchmark [Shilane et al. 2004] for all of which we manually define both a best as well as a worst viewpoint. We extract the following three image based measures [Secord et al. 2011] for each best and worst view in this training set:

- silhouette length in image space, relative to image area
- projected area relative to image area
- smoothness of depth distribution over the model.

For a given view, direct linear combinations of these values does not provide a meaningful score, so we use a non-linear classification. More precisely, we learn a “best view classifier” from the training set using support vector machines (SVM) with radial-basis function kernels [Schölkopf and Smola 2002]. We use 5-fold cross-validation to determine best SVM model parameters before the actual training step. This results in a general model of viewpoint preference which we employ to predict good viewpoints for the meshes in the “test” set of the PSB: we densely sample uniform view directions (using the k-means method) and for each view direction  $v_i$  predict its probability  $p_i = p(v_i)$  of being a best view [Wu et al. 2004]. This results in a smooth scalar field over the sphere (see Figure 7.5a) and we select best views as local maxima (determined over the one-ring neighborhood) with  $p_i > 0.5$ . We visualize such a prediction in Figure 7.5.

## 7.4 Line Rendering

We use the generated uniform/predicted view directions  $v_i$  as input for view-dependent line drawing algorithms and render the views using the following line types:

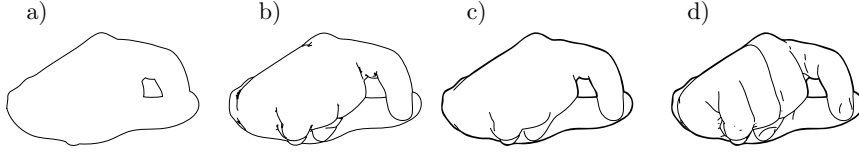
1. silhouettes, depicting the 2d closed boundary of the rendering
2. occluding silhouettes, depicting all points on the mesh where with normals orthogonal to the view direction
3. suggestive contours [DeCarlo et al. 2003], depicting lines where a contour would appear after a small change in viewpoint. We use suggestive contours exclusively together with occluding silhouettes.
4. Canny lines [Canny 1986] from the depth image

We illustrate those styles for a given model in Figure 7.6.

## 7.5 Sampling Local Features

Following Chapter 5 we encode each view image as a bag of small local image patches transformed into an appropriate feature space (see Figure 5.3). As the local features do not carry any spatial information, this representation is commonly called a “bag-of-features”. We generate  $32 \times 32 = 1,024$  key-points evenly distributed over the image by sampling on a regular grid.





**Figure 7.6:** Comparison of different line rendering approaches: (a) silhouettes (SH), (b) Canny lines from depth image (CFD), (c) occluding contours (OC) and (d) suggestive contours (SC)

## 7.6 Representation

We use the hard quantization strategy outlined in Section 5.6 to generate frequency histograms of visual words as our final representation for a sketch. We generate a visual vocabulary using k-means clustering (Equation 5.9). As the training data, we randomly sample one million local features from all models and views in order to cover a wide variety of possible local features. The set of resulting cluster centroids forms the visual vocabulary where each entry (visual word) is a representative of the local features in its corresponding cluster. The size of the visual vocabulary, i.e. the number of clusters, is an important parameter that strongly influences retrieval performance and we determine its optimal value in Section 7.9.

We represent each view as a histogram of visual word frequency. We quantize all local features from a given sketch against the visual vocabulary, representing them as the index of their closest visual word according to Equation 5.12. We define the entries of the final histogram of visual word representation  $\mathbf{h}$  that encodes a view according to Equation 5.11. Each dimension  $j$  in the feature vector corresponds to a visual word and encodes the *number* of those words appearing in a sketch. This representation is typically very sparse, as the number of distinct features occurring in a given sketch is usually much lower than the size of the vocabulary. We store the resulting histogram in an inverted index datastructure [Witten et al. 1999; Zobel and Moffat 2006] in order to achieve quick lookups during the query stage.

## 7.7 Online Querying

At runtime, users draw a query sketch and submit it to the retrieval pipeline. We perform the following steps on the query sketch: we first extract local descriptors, quantize them against the visual vocabulary and finally represent the sketch as a (sparse) histogram of visual word occurrences (those steps are identical to how views are represented in the offline indexing stage, see Figure 7.3).

**Tf-idf weighting function** The entries  $h_j$  of a histogram do not necessarily need to be raw word counts — and it is indeed common to represent  $h_j$  as a function of the “importance” of the  $j^{th}$  word. We use the tf-idf model (*term frequency-inverse document frequency*) [Salton and Buckley 1988; Witten et al. 1999] to define importance of a visual word. The idea is that a word is important if it appears often in a sketch (high term frequency) but at the same

time less distinctive if is a common word in the collection (inverse document frequency). We follow Sivic et al. [2003] and use the following tf-idf function to compute term weights:

$$h_j = (h_j / \sum_i h_i) \log(N/f_j) \quad (7.1)$$

where  $N$  denotes the total number of views in the collection and  $f_j$  the frequency of visual word  $j$  in the whole collection. We have also experimented with simpler, computationally less expensive definitions, such as the constant function (which amounts to simply counting the number of words that occur in both documents) but found this formulation to achieve better retrieval results.

**Similarity metric** We employ a vector space model to define similarity between two visual word occurrence histograms [Salton et al. 1975; Witten et al. 1999]. Let  $\mathbf{h}$  and  $\bar{\mathbf{h}}$  be two histograms of visual words (representing two images). We define their similarity as

$$s(\mathbf{h}, \bar{\mathbf{h}}) = \langle \mathbf{h}, \bar{\mathbf{h}} \rangle / \|\mathbf{h}\| \|\bar{\mathbf{h}}\|. \quad (7.2)$$

Intuitively, two images are considered similar, if their histograms (seen as high-dimensional vectors) point into the same direction. Note that the histograms are normalized — this is important in order not to favor histograms with higher word counts.

**Retrieving models** Given a histogram  $\mathbf{h}$  computed from a user sketch, we retrieve similar models in two steps: first, we find similar views, querying the inverted index. This operation amounts to computing Equation 7.2 between  $\mathbf{h}$  and the views in the collection. This is fast, as only those views in the index need to be checked that share visual words with  $\mathbf{h}$ . The result is a set of best-matching views and we return the set of models in the order of their corresponding best matching views.

## 7.8 Benchmarking

Since each sketch is associated with one category from the PSB, benchmarking a sketch-based system now becomes analogous to benchmarking an example-based system using the PSB. For a query sketch from the experiments, we count the number of retrieved models belonging to the same category as the sketch. We use this data to compute precision and recall (as well as any other metric). This is a standard procedure in information retrieval [Salton 1992] and enables an objective comparison of retrieval engines. Since we use exactly the classification from the PSB, we can directly use all evaluation tools that come with the PSB.

**Train/test dataset** The PSB defines a split into training/test dataset (907/907 models) and we accordingly split the benchmark sketch dataset into a training/test dataset (907/907 sketches). Later, we use the training dataset when optimizing system parameters, while evaluating on the test dataset.

Overall, the sketch dataset we gathered defines a general and challenging benchmark for SBSR systems which we hope will help make research results in this field more comparable and thus encourage further research. We provide the whole dataset as a free resource.

## 7.9 Optimizing Retrieval System Parameters

The parameter-space for the proposed shape retrieval pipeline (as well as for other similar systems) has many dimensions, including: local feature size, visual vocabulary size, number of view directions sampled for each model, line types used to render the views as well as the parameters underlying the Gabor filter bank employed in the GALIF feature transform. In particular, our system has nine main parameters (see Table 7.1, Table 7.2, Table 7.3 and Table 7.4), each of which can significantly influence retrieval performance.

A dense sampling of this parameter space to find the best parameter combination is computationally prohibitive: sampling only 10 parameter values along each dimension of the parameter space results in  $10^9$  combinations, each of which takes 1 to 10 hours to evaluate on a modern multi-core machine. Evaluating a single parameter combination requires running all steps illustrated in Figure 7.3 for both the views extracted from the models as well as the sketches in the benchmark dataset.

In particular, for the model dataset those steps are: computing features from the views, computing a visual vocabulary from a sample of those features, quantizing the features to form visual word occurrence histograms and computing an inverted index from those histograms. For the benchmark sketch dataset those steps are the same except that we use the vocabulary computed on the view dataset and we do not need to build an inverted index. Next, we compute pairwise distances between the histograms of the benchmark dataset and the histograms of the 3d shape collection. Finally, from the pairwise distances, we compute the desired “goodness” measure as well as precision/recall curves.

However, the wrong choice for just a single parameter out of the nine main parameters can significantly impact overall performance. While experienced researchers often make surprisingly good guesses about “best” parameter values, it remains impossible to optimize the combination of parameters by hand.

In the following section, we explain a simple iterative visualization strategy to evaluate and optimize the retrieval pipeline. We perform the complete optimization on the train dataset split. To measure system performance at a parameter combination, we use the fraction of 1-nearest neighbors belonging to the same class as the query sketch [Shilane et al. 2004], averaged over all sketches in the benchmark dataset (i.e. a measure of 1 would be a perfect result). In other words, we average the success of the retrieval compared to the human data (Section 3.3). This is supposed to make the retrieval perform close to what humans expect. However, the optimization strategy outlined below is clearly independent of this measure.

### Optimization Strategy

We advocate a human guided visual gradient descent strategy. The idea is to color-code the performance resulting from varying two parameters, while

**Table 7.1:** *Parameter search ranges for Gabor filter used in the GALIF descriptor. For each parameter, we take six logarithmically spaced samples. This results in  $6 \times 6 \times 6$  sample points for each of which we compute system performance.*

	$lw$	$\lambda$	$\omega_0$
start : end	0.005 : 0.05	0.05 : 1	0.05 : 0.25

keeping all other parameters fixed. This color code allows us to pick good combinations of the two parameters. Then these parameters are fixed and others are varied. This goes on until no further improvement is visible.

We have found that humans are good in performing two tasks in this search that computers so far usually fail in:

- Identifying interesting *combinations* of parameters. It turns out some parameters are good to optimize together, while others are not. Knowing the design of the system helps in identifying these groups.
- Avoid inspecting local minima or undefined/unsuccessful regions in the parameter space. Since each evaluation of the system requires a long time, going down the right path saves days to weeks in the search of the best parameter combination, in our experience.

We group parameters into three independent groups:

**filter parameters** corresponding to the filter of the underlying image transform, i.e.  $\lambda$ ,  $lw$ ,  $\omega_0$  in the case of the GALIF descriptor (see Table 7.1) and  $\sigma$  in the case of the SHOG descriptor (see Table 7.2).

**intrinsic feature parameters** that define the remaining core feature transform parameters, i.e. number of orientations, number of tiles and feature size (see Table 7.3). Those parameters are the same for both the GALIF and SHOG descriptor.

**extrinsic parameters** that define the remaining retrieval system parameters but are not directly related to the feature transform. We evaluate vocabulary size, number of views per model and linetype used when rendering a model during view generation (see Table 7.4). By definition, those parameters are the same for both the GALIF and SHOG descriptor.

In the following, we use this strategy to optimize not only our system but also all competitors. It turns out that some of the parameters given for these systems in the respective publication are not optimal, showing that parameter optimization is important, yet far from trivial.

## 7.10 Evaluation

Our strategy for evaluating feature transform and system parameters is the same for both the GALIF and SHOG descriptor. We start by evaluating filter parameters while fixing intrinsic descriptor parameters as well as extrinsic system parameters to reasonable values determined using the informed best guess

**Table 7.2:** *Parameter search range for standard deviation  $\sigma$  of Gaussian derivate filter used in the SHOG descriptor. We take twenty logarithmically spaced samples.*

	values
standard deviation $\sigma$	$\{.010, .015, .022, .033, .050, .074, .11, .16, .25, .37, .55, .81, 1.2, 1.8, 2.7, 4.0, 6.0, 8.9, 13.4, 20.0\}$

**Table 7.3:** *Parameter search ranges for intrinsic descriptor parameters. Parameters and ranges are the same for both the GALIF and SHOG descriptor.*

	values
#orientations	$\{1, 2, 4, 8, 16\}$
#tiles	$\{1, 2, 4, 8\}$
feature size	$\{.01, .05, .1, .125, .15, .2, .25, .3, .4, .5\}$

**Table 7.4:** *Parameter search ranges for extrinsic system parameters. Parameters and ranges are the same for both the GALIF and SHOG descriptor.*

	values
vocabulary size	$\{50, 100, 500, 1000, 2500, 5000, 10000\}$
#views	$\{7, 22, 52, 102, 202\}$
linetypes	$\{\text{sil, occ, suggcont, canny}\}$

strategy. We show the fixed values of this first evaluation step in the second column of Table 7.5 and Table 7.6.

### Filter Parameters

While both the GALIF and SHOG descriptor share many parameters, the main difference between both is the underlying filter for estimating orientation of sketch lines.

**GALIF: bandwidth and peak-frequency** To make evaluation results invariant to the size of a sketch, we first define  $linewidth = \sigma_x/w$  where  $w$  denotes the side-length of a sketch (in pixels) and  $\lambda = \sigma_x/\sigma_y$ . We evaluate over those parameters instead of  $\sigma_{x,y}$  directly. As expected the GALIF descriptor is sensitive to the correct choice of those parameters: line drawings naturally contain mostly high-frequency content and the descriptor’s performance increases with peak frequency  $\omega_0$ . The optimal setting for  $linewidth$  and  $\lambda$  is coupled to the choice of the  $\omega_0$  of the Gabor filter (see Figure A.2). The optimal combination of parameter values turns out to be  $linewidth = 0.02$ ,  $\lambda = 0.3$  and  $\omega_0 = 0.13$  (see also Table 7.5).

**SHOG: standard deviation** The SHOG filter is governed by a single parameter  $\sigma$  that defines the standard deviation of the Gaussian derivative filter. This parameters determines how much high-frequency detail is filtered out “be-

fore” computing partial derivatives. Intuitively, for the case of strictly binary sketches, smoothing is necessary to achieve smoothly varying gradients orientations along the sketch lines. The evaluation shows that best results are achieved for small parameter values, we achieve the best results for  $\sigma = 1.81$  and observe reduced performance for significantly larger values (see Figure A.6).

## Intrinsic Feature Parameters

We now fix filter parameters for both the SHOG and GALIF descriptor to best values determined in the first evaluation round (see third column of Table 7.5 and Table 7.6) and evaluate over the remaining intrinsic descriptor parameters.

**Number of orientational filters** We analyze using 1, 2, 4, 8 and 16 orientations. Using less than 4 orientations results in significantly reduced retrieval performance for both descriptors (see Figure A.3 and Figure A.7). In that case the descriptor is no longer discriminative enough — it can only encode vertical and horizontal lines.

In case of the GALIF descriptor, retrieval performance drops only slightly when using more than four orientations (see Figure A.3) — but at the cost of a much higher-dimensional descriptor.

In case of the SHOG descriptor, using 8 or 16 orientations results in significantly reduced retrieval performance (see Figure A.7). We offer the following explanation: a larger number of orientations makes the feature transform sensitive to small deviations in orientation of lines — this may be undesirable because of human inaccuracy in drawing.

Overall, using 4 orientations consistently yields the best performing features in our experiments.

**Number of tiles** We evaluate using 1, 2, 4, and 8 tiles to subdivide a local image patch, see Figure A.3 and Figure A.7. Note: 2 tiles means the local image area is subdivided into  $2 \times 2$  cells.

Results are consistent among both descriptors: using only 1 tile results in poor performance. Each local feature is then encoded using only a single dimension per orientation — not enough to yield a discriminative feature vector.

Overall, we achieve best retrieval performance when using 4 tiles. Note that the dimensionality of the descriptor grows quadratically with the number of tiles and indeed, using 8 tiles results in reduced retrieval performance.

**Local feature size** We find that — compared to approaches working with natural images — for both descriptors, we consistently achieve optimal results when using relatively large local feature sizes. Performance is optimal between feature size 0.1 and 0.3. Using a small feature size  $\leq 0.05$  significantly reduces performance, as the information in a single feature is no longer discriminative and typically encodes only single line segments (see Figure A.3 and Figure A.7).

We achieve overall highest performance for feature size 0.2 in case of the GALIF descriptor and for feature size 0.125 in case of the SHOG descriptor.

### Extrinsic System Parameters

We now fix filter parameters and descriptor intrinsic parameters to best values determined previously (see Table 7.5 and Table 7.6, fourth column) and optimize the remaining system parameters.

**Vocabulary size** When using 1,000 visual words or more, we consistently achieve good retrieval performance for both descriptors (see Figure A.4 and Figure A.7). Using more words can lead to better performance but there is a tradeoff to be considered: using a larger vocabulary makes computing a descriptor more expensive in two ways: a) it requires more time as we need to quantize against a larger vocabulary — this can be undesirable for an interactive system. And b) the dimensionality of the descriptor becomes higher — this is typically not so much of an issue as long as the descriptor is sparse. We find that a good compromise between speed and performance is achieved for a vocabulary size of 2,500 visual words, which also results in overall best performance for both descriptors.

**Number of sampled views** Our analysis shows that the sampling of view directions should be dense enough to capture enough data about the model — using only 7 views per model severely reduces retrieval performance (see Figure A.4 and Figure A.7). A good sampling is reached for around 100 uniformly distributed view directions, with the overall maximum at 202 views for both descriptors. Our perceptual best view selection generates an average of 14.4 views per model, achieving similar results to using 22 regularly sampled views, see Figure 7.7b.

**Line types** Our evaluation of line types used to render views shows that occluding contours and suggestive contours perform better than outlines or Canny lines from the depth image (see Figure A.4 and Figure A.7). Interestingly, the additional information contained in suggestive contours does not provide a significant boost compared to using occluding contours only.

Since most of the real-world sketches gathered in our experiment clearly contain more types of lines rather than just silhouettes, this explains the reduced performance of silhouettes only and justifies our decision to develop a system that can deal with arbitrary line-types.

Overall, we can report that the following parameters result in a system with a good performance/speed ratio: vocabulary size: 1,000, number of views: 102, line-type: suggestive contours. For those settings, our performance measure (fraction of correct 1-nearest neighbors) is 0.288 (GALIF) and 0.278 (SHOG).

### Optimizing Existing Approaches

Our parameter optimization strategy as well as the benchmark are very general: we demonstrate this by optimizing parameters for two existing sketch-based shape retrieval systems. Yoon et al. [2010] propose a global descriptor based on the diffusion tensor. Their descriptor is governed by two parameters: 1) number of histogram bins and 2) number of tiles. They propose using a single tile with 18 histogram bins to encode a view. This is not ideal as visualized in Figure A.5. Our optimization finds a more favorable parameter combination

**Table 7.5:** Optimal parameter values for GALIF descriptor as determined by evaluation strategy described in Section 7.9 over the parameter ranges listed in Table 7.1, Table 7.3 and Table 7.4. Bold font denotes variable parameters and the particular values are the resulting best parameter values determined by evaluation. Standard font denotes fixed parameter values. The bottom row shows retrieval performance for the given parameter combination.

	filter	intrinsic	extrinsic
lw	<b>0.02</b>	0.02	0.02
$\lambda$	<b>0.3</b>	0.3	0.3
$\omega_0$	<b>0.13</b>	0.13	0.13
#orient	4	<b>4</b>	4
#tiles	4	<b>4</b>	4
feature size	0.125	<b>0.2</b>	0.125
#views	102	102	<b>202</b>
line type	suggc	suggc	<b>suggcont</b>
vocabulary	1000	1000	<b>2500</b>
value	0.271	0.288	0.308

**Table 7.6:** Optimal parameter values for SHOG descriptor as determined by evaluation strategy described in Section 7.9 over the parameter ranges listed in Table 7.2, Table 7.3 and Table 7.4. Bold font denotes variable parameters and the particular values are the resulting best parameter values determined by evaluation. Standard font denotes fixed parameter values. The bottom row shows retrieval performance for the given parameter combination.

	filter	intrinsic	extrinsic
$\sigma$	<b>1.81</b>	1.81	1.81
#orient	4	<b>4</b>	4
#tiles	4	<b>4</b>	4
feature size	0.125	<b>0.125</b>	0.125
#views	102	102	<b>202</b>
line type	suggc	suggc	<b>suggcont</b>
vocabulary	1000	1000	<b>2500</b>
value	0.292	0.278	0.313



with significantly higher retrieval performance (12 tiles with 4 bins each, resulting in a  $12 \times 12 \times 4$ -dimensional descriptor). We also analyzed the spherical harmonics descriptor proposed by Funkhouser et al. [2003], which is defined by two parameters: number of circular functions and number of coefficients. Our evaluation shows that this descriptor performs best when using 8 functions and 16 coefficients — very close to the original parameters  $16 \times 32$  (see Figure A.5).

## 7.11 Learning Sketch-Based Shape Retrieval

One of the main advantages of the kNN based sketch-based 3d shape retrieval approach employed so far is that it is completely unsupervised: it does not require any annotations or labels to come with the models. Retrieval is purely data-driven, making this method applicable to arbitrary datasets. Additionally, the kNN approach is efficient and fast and lets us handle large collections of models in an interactive way.

Despite those apparent advantages, one natural question to ask is: can we *learn* sketch-based shape retrieval in the sense that we learned to classify sketches in Chapter 6? While this clearly requires an annotated training dataset of sketches, it potentially could yield even better retrieval results than the unsupervised approach discussed in this chapter.

### Experiment

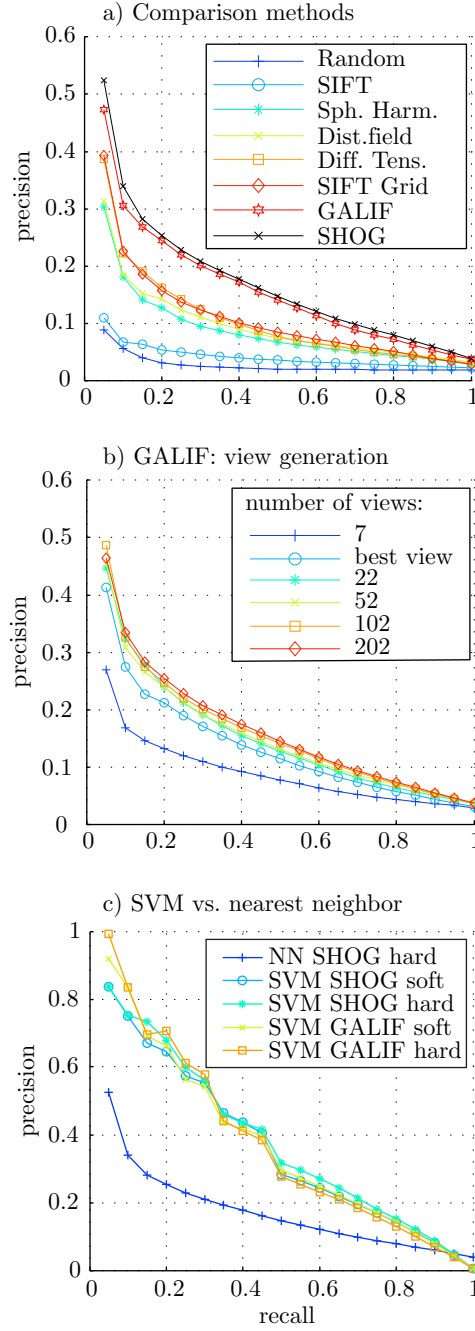
We try to answer this question by running the following experiment: we use the SVM based computational sketch recognition pipeline discussed in Section 6.1 to learn a classification model of human object sketches, but this time using the PSB sketch dataset (Section 3.3). Given the computational recognition model and an unknown query sketch, we first classify the sketch and then return 3d shapes according to the classification.

Again, we learn the model on a training dataset and evaluate on a distinct test dataset. While the PSB comes with a predefined split into a training/test dataset, one peculiarity of this split is that it is *not stratified*, i.e. not all categories that appear in the training dataset also occur in the test dataset. To overcome this, we use a stratified split for this experiment, making sure that all categories appear in both the test and training dataset, with equal numbers. To compute a computational model, we use the descriptor parameters as well as the best performing model parameters as determined in Section 6.2.

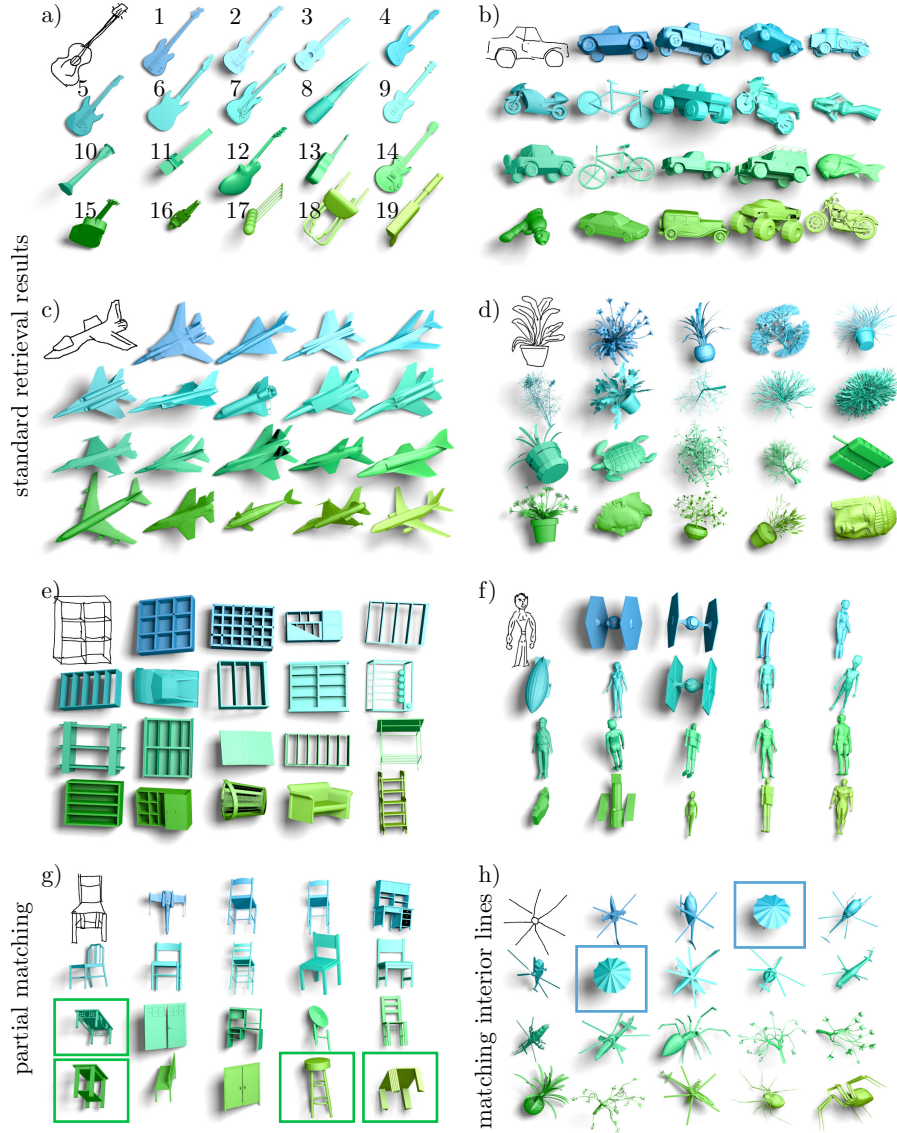
Compared to the kNN approach discussed earlier in this chapter, the views and shapes are not required to perform a query. Instead, we classify the query sketch as belonging to one of the 161 categories contained in the PSB. As the result of a query, we return one model per category exactly in the ranking that the 161 one-vs-rest classifiers suggest. This in turn allows us to compute precision/recall plots that are broadly comparable to the ones computed for the unsupervised approach.

## 7.12 Results

In this section we compare our proposed system (using optimal parameters as determined in Section 7.10) to previous work and analyze its properties



**Figure 7.7:** Detailed evaluation of retrieval performance on test dataset (precision/recall, higher curves are better): a) comparison with previous work using optimized system parameters; b) GALIF descriptor: influence of view generation methods; c) nearest neighbor retrieval — as in a) and b) — vs. SVM classification. The NN SHOG hard curve is the best performing one from a).



**Figure 7.8:** Examples of sketch-based query results using our system. For each query sketch (top left of a cell), we show the top 19 results with a color indicating their rank (blue being the highest, see a)). The “chair” example (g) illustrates partial matching (i.e., tables are retrieved). The “man” example (f) exhibits failure cases, with the highest ranked objects not matching the desired object. Note that many of the remaining sketches are perfect matches, though. Finally, we show an abstract query in h) that matches interior lines of the retrieved umbrellas.

— such as partial matching. We perform all evaluations on the *test dataset* (see Section 3.3). For the learning based retrieval approach, the test dataset differs from that used for the kNN approach as described Section 7.11.

### Comparison to Other Systems

We compare our approach (based on the GALIF and SHOG descriptor) to three other leading sketch-based retrieval systems [Funkhouser et al. 2003, 2004; Yoon et al. 2010]. We use standard precision/recall plots to visualize our results (Figure 7.7). For each system we compute precision/recall values averaged over all 907 sketches from the test dataset. To make the comparison as fair as possible, we use the *best parameters* for each approach as determined in Section 7.9. Additionally, we evaluate performance of the popular image descriptor SIFT [Lowe 2004] on sketches. We use two variants: a) the complete scale-space feature detection and extraction pipeline (SIFT) and b) a single-scale grid sampled approach (SIFT Grid), using exactly the same sampling parameters and feature size as for the GALIF descriptor.

Our proposed system clearly outperforms all existing approaches, see Figure 7.7. We additionally visualize the high quality of our results in Figure 7.8. Note that some existing systems *cannot handle* interior lines in sketches [Chen et al. 2003] and thus cannot be evaluated against the real-world sketches in our benchmark. We believe that this is not a limitation of the benchmark, but rather an indication that modern retrieval systems should not be artificially limited to closed boundary curves as input.

Notably, using the SIFT scale-space keypoint detection to compute features results in poor retrieval performance (see Figure 7.7): on average only few key-points are detected, resulting in an imprecise representation of a sketch by its histogram. However, several optimized parameters of the GALIF/SHOG descriptors turn out to have a connection to the SIFT descriptor: both use four tiles to subdivide a local patch. The optimal parameter of 4 orientations also lets us draw an interesting connection: binary sketches contain only information about the *orientation* of lines, while photographs (for which the SIFT descriptor has been designed) contain *directional* information. In that sense the optimal angular resolution for our descriptors turns out to be identical to the 8 directions used in the SIFT descriptor.

### Partial Matching

As we cannot expect users to sketch all of the lines appearing in a computer-generated line-drawing, a retrieval system should be able to reliably retrieve a model from only a *subset* of its representative lines (partial matching). Our system naturally supports this without using computationally expensive sliding window approaches. The feature transforms encode the distribution of visual words in a sketch — and this is invariant to the position of the individual features in a sketch. The similarity measure in Equation 7.2 essentially computes a weighted count of the number of features that are shared across two sketches: if one sketch contains only a subset of the strokes of the second sketch, the two histograms are similar in the regions encoding the shared strokes and the system returns a partial match. We demonstrate this partial matching behavior in Figure 7.8.

### Supervised vs. Unsupervised Retrieval

Compared to the unsupervised kNN model, the SVM model achieves superior results (see Figure 7.7) but, by definition, at the cost of requiring an annotated dataset for training the computational model.

The precision/recall curves for both the soft quantized and hard quantized variants of both descriptors do not differ significantly. This appears odd at first, as we conclude in Section 6.2 that soft quantization works considerably better than hard quantization for computational classification. However, the training set size is rather small in case of the PSB (on average 5.6 instances per class). For such small training set sizes, recognition accuracy for soft and hard quantization is very similar (see also Figure 6.2).

### Interactive Application

We run our retrieval engine in a graphical user interface: users sketch a shape, hit the search button, and the display shows a collection of matching models. The system's retrieval speed (using all 1,814 models from the PSB) is currently at only a few milliseconds for performing the search, meaning the system could accommodate a significantly larger database.

## 7.13 Discussion

One interesting observation from evaluating the number of orientations used for the GALIF descriptor vs. the SHOG descriptor is that for a larger number of orientations (8 or 16) retrieval performance of the GALIF descriptor stays about the same (see Figure A.3) while retrieval performance of the SHOG descriptor significantly drops (see Figure A.7). Such a drop in performance for a larger number of orientations is what we intuitively expect: we lose invariance to slight differences in orientation of a line. The behavior of the GALIF descriptor for a large number of orientations thus appears surprising at first.

We can offer the following explanation: for the GALIF descriptor, the bandwidths of the underlying Gabor filter (see Equation 5.6) have been optimized using a fixed parameter of four orientations (because we cannot not exhaustively sample the complete parameter space as discussed in Section 7.9). This results in a relatively large bandwidth parameter for the angular component of the Gabor filter which we continue to use when using a larger number of orientations. As a result, the filters in the orientational filter bank significantly overlap and return very similar responses for neighboring filter orientations. In other words: a GALIF feature vector computed using 16 orientations does not contain significantly more information than the one with only four orientations. As a result, retrieval performance stays roughly the same. In both cases however, it appears that the best number of orientations is four and there is no benefit in using a larger number.

### Limitations

Our approach depends on the quality of the representation that it uses to generate line art — poor models pose a significant problem for line rendering

techniques that depend on derivatives on the model. Many of the models in the PSB are not connected, i.e. they are polygon soups. Our evaluation, however, also shows that resorting to Canny lines on the depth map, which gives good results on any polygonal model, results in retrieval performance that is reasonably close to that of more sophisticated lines types. However, we expect the gap between Canny lines and, e.g., suggestive contours to become larger with increased quality of the underlying models.

Matching is based on *geometric similarity*, while humans might expect a more semantic behavior of the system. We tend to quickly recognize the semantic category (cow, airplane) that a sketch depicts — if the retrieved models do not fall into this category, we would quickly dismiss those results as poor — although geometrically the matches might actually be quite good. This behavior is also encoded in the benchmark we use: only matches within the same category are counted, a geometrically identical match (and in this sense very good match) from a different category is counted as a negative result.

---

## Chapter 8

# Conclusions

---

This thesis describes the first large scale exploration of human object sketches. We have collected two novel, large datasets of sketches for computational recognition and 3d shape retrieval. We have publicly released both datasets in the hope that they will be of help for the community to further improve sketch-based human computer interaction.

We have used the first dataset to evaluate human sketch recognition accuracy (73%). We have demonstrated that — given such a large dataset — reasonable classification rates can be achieved for computational sketch recognition (56%). We have used the second dataset to establish the first large-scale benchmark for sketch-based 3d shape retrieval and proposed new methods for sketch-based shape retrieval that outperform existing approaches.

To our knowledge, we are the first to collect a significant number of sketches for the evaluation of shape retrieval performance. Our dataset is based on the freely available and widely accepted set of models from the Princeton Shape Benchmark [Shilane et al. 2004]. This makes our benchmark *easily applicable* in any sketch-based shape retrieval project. We make the set of benchmark sketches available as a free resource and hope that this helps making comparisons between approaches easier as well as more reliable.

Our dataset shows that artificially limiting the input to closed boundary curves [Chen et al. 2003] is, quite simply, not how humans would like to draw for shape retrieval. Although our evaluation shows that rendering additional computer generated lines results only in a slight improvement of retrieval performance, it is important that a system actually *technically supports* interior lines.

Although the proposed approach achieves significantly better performance than any of the previous approaches we evaluated, a brief look into the sketches we have collected suggests that sketch-based shape retrieval for realistic inputs is still a very hard problem. The main technical ingredients of our approach, bag-of-features and the new descriptor for line-art renderings, relate to the variance and deficiencies in this type of input. The underlying feature transform is based on Gabor filters — as is the global GIST descriptor [Oliva and

Torralba 2001, 2006] successfully employed in image retrieval. One of the main differences is that we do not fix the filter bank parameters (as is the case for the GIST descriptor) but rather learn optimal parameter values suitable for sketches. This strategy is general and we are interested in seeing its applications in other domains as well.

Overall, the SHOG (Section 5.2) and GALIF (Section 5.3) feature transforms turn out to achieve comparable performance levels. This is a bit surprising at first, as the Gabor filter underlying the GALIF feature transform is highly tunable, which we exploit to find its best performing parameters for the overall feature transform (Section 7.9). However, higher order Gaussian derivative filters start to appear very similar to Gabor kernels (with important theoretical differences) [ter Haar Romeny 2003]. As we locally average over the outputs of the filters to compute features and eventually quantize these, small differences in filter output might not be crucially important. Rather, a stable estimation of orientation is required, which both feature transforms are certainly capable of.

We have not discussed the user interface we are using for shape retrieval as it is not yet aiding the search interaction. There is clearly room for improvement, such as optimizing the layout of the results or learning from individual users or the user community as a whole. Different users might sketch different types of lines which we could exploit to improve retrieval results. Despite these possible ways of improving sketch-based shape retrieval, we agree with many researchers that rather than using one search mode in isolation, combining text-queries and context-based shape search with sketch-based search could be a potentially fruitful direction for further research.

Finally we hope that better computational understanding of sketches will lead to better computer accessibility. Virtually everybody is able to sketch a face or recognize a sketched face. Writing and reading, which today are still the standard way of communicating with computers, are much less widespread. By some definitions, functional illiteracy, even in first-world countries, is up to 20% of adults. If computers were to understand sketches as we do, sketching would give a much larger audience access to the data that has been gathered digitally over the last decades.

### 8.1 Future Directions

Our work relies heavily on computer vision and machine learning techniques, although applied to a novel domain. While we have introduced novel representations tailored to the requirements of this domain and shown that those outperform existing approaches, several existing techniques known to work well for photographs could potentially be applicable (with modifications) for sketches as well. We give an overview over promising techniques in the following section.

### Representation, Classification and Search

When even more sketch data was available, potentially better and more compact descriptors could be learned [Torralba et al. 2008b]. Other features than just gradient orientation could be integrated into the representation [Bileschi and Wolf 2007], pairs of local patches could form visual phrases [Zhang et al.



2009; Chum and Matas 2010] and finding the most distinct subset of local features could also be an option [Turcot and Lowe 2009]. As an alternative to modifying the representation itself, employing better distance metrics could also help improve performance [Wu and Rehg 2009]. If this is not possible, learning an “optimal” distance metric from the data [Weinberger et al. 2006] might be possible.

Both classification and 3d shape retrieval rely on an unordered bag-of-features model. Putting back some spatial information into the representation could potentially benefit classification and retrieval accuracy [Lazebnik et al. 2006; Viitaniemi and Laaksonen 2009; Cao et al. 2010; Jégou et al. 2010].

While we have shown that an SVM approach clearly outperforms kNN classification on the representation we use, Boiman et al. [2008] show that by performing kNN in the unquantized space of local descriptors superior performance can be achieved. Classification speed could be potentially improved by learning hierarchical taxonomies [Griffin and Perona 2008] and better voting schemes building on the output of the binary SVM classifiers could further improve classification accuracy [bo Duan and Keerthi 2005]. Finally, among the multitude of classifiers available, we have evaluated two popular ones: a kNN classifier as the baseline for its simplicity and SVMs which have found widespread adoption and for which stable implementations are available. Nevertheless, many other options are possible and might even yield better classification results. For a recent overview see Domingos [2012].

For sketch-based shape retrieval, an inverted index might potentially not be the ideal data structure in terms of retrieval speed, as the frequency histograms it operators on are significantly less sparse than for classical text retrieval. For sketches, the visual vocabulary is much smaller and contains significantly more words for a query. As a result, other (approximate) nearest neighbor search techniques might be useful to scale sketch-based shape retrieval to significantly larger collections (millions of model, resulting in hundreds of millions of views) [Weber et al. 1998; Liu et al. 2004; Samet 2006; Andoni and Indyk 2008; Muja and Lowe 2009; Chum et al. 2008, 2009; Lee et al. 2010].

## **Sketch Synthesis**

We feel that sketch synthesis is an interesting, unexplored problem. How could the computer generate distinctive sketches that are immediately recognizable by humans? If this question can be answered, many new applications could benefit from the research we have started here. Additionally, if we had a generative model of sketches, the insights learned from that could potentially also be of value for better understanding the sketch recognition problem.

## **Sketch Simplification and Beautification**

We also believe that the computational model could be useful for supervised simplification and beautification of sketches. Simplification has been well-studied in certain graphics domains such as 3d geometry [Garland and Heckbert 1997]. The general strategy for such techniques is to remove complexity (e.g., delete edges) while staying as close as possible to the original instance according to some geometric error metric. Such unsupervised heuristics have no semantic understanding of each instance and therefore will often make simplifications

which are geometrically modest but perceptually jarring (e.g., smoothing away the face of a statue). We believe that an ideal simplification algorithm would consider the *semantic meaning* of each instance when deciding which simplifications to perform. Specifically, we argue that in the case of a sketch, the best strokes to remove are those that are *not required for successful recognition*. Strokes vital for recognition (e.g., the trunk of an elephant) should be preserved.

### Evaluation

Measuring performance and optimizing parameters of the sketch-based shape retrieval system proposed in this thesis is based on the notion of precision/recall. This is a widely accepted measure in information retrieval to evaluate such systems [Salton 1992]. It is easy to implement, fast to evaluate and only requires that each object in the dataset is assigned to one particular category. Slaney [2011] argues that precision/recall might not be the “right” measure for multimedia datasets: as collections grow larger (and thus contain millions of potentially correct results for a given query), the notion of recall becomes nonsensical. Defining precision as the ratio of results in the correct category enforces a very strict binary success/failure measure on the search. Especially in the case of fuzzy user sketches such a binary measure might be too strict. A 3d model might be considered an incorrect result during evaluation — because it formally does not belong to the same category as the sketch — while perceptually being very close to the shape. This suggests a need for “better” evaluation methods, using measures that better take into account user preference, perhaps similar to the click-through ratio employed in modern Internet multimedia search engines [Slaney 2011].

A final open question is how universal sketching and sketch recognition is among humans. Our sketches come from Amazon Mechanical Turk workers all over the world, but it is certainly not a uniform sample of different cultures, ages, genders and artistic expertise. How do these factors and many others affect sketching? Are the stylizations different cultures use for a certain object similar and even mutually recognizable?

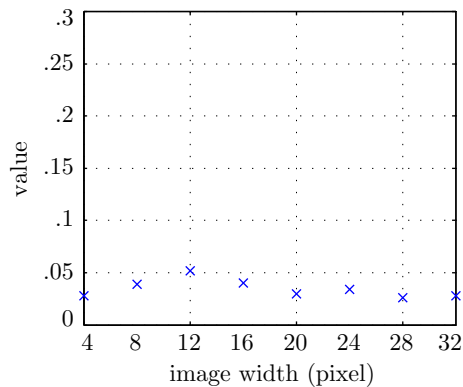
---

## Appendix A

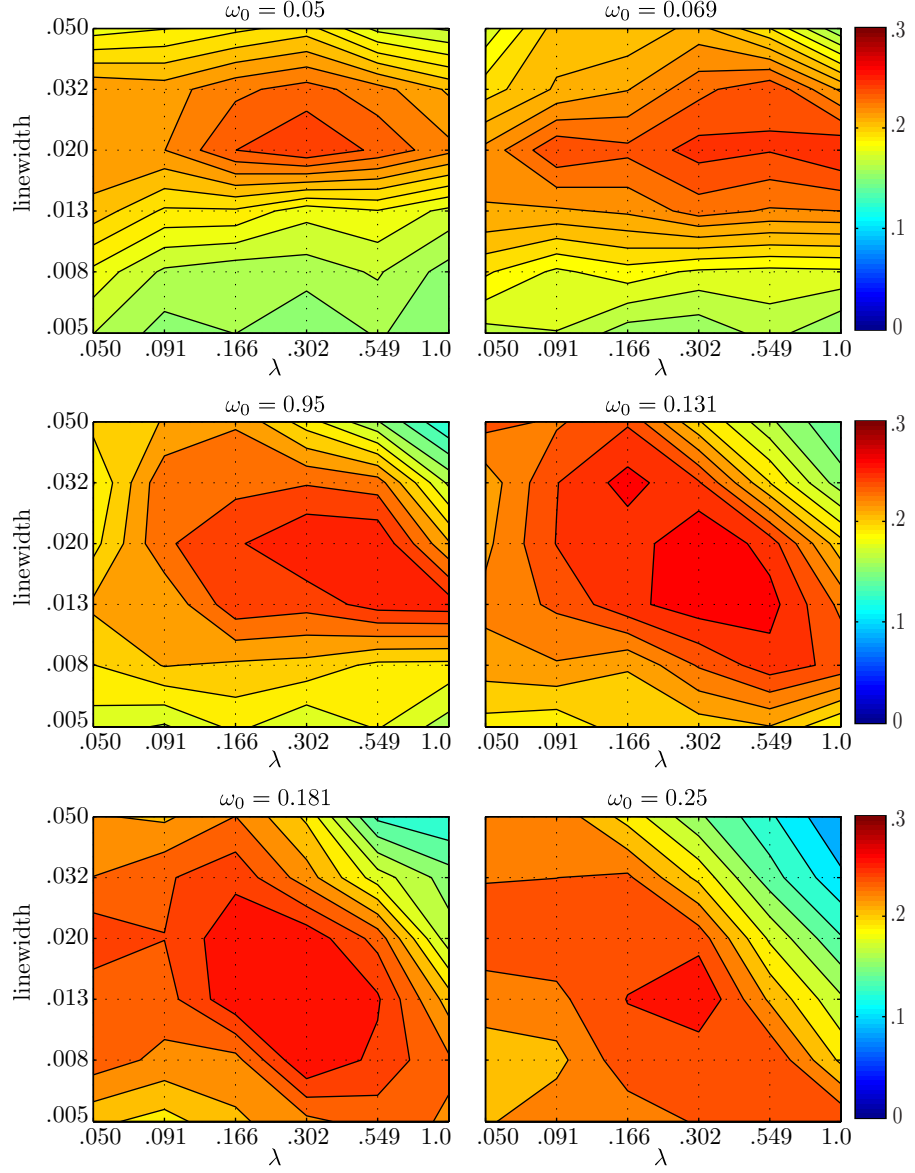
# Parameter Space Evaluation

---

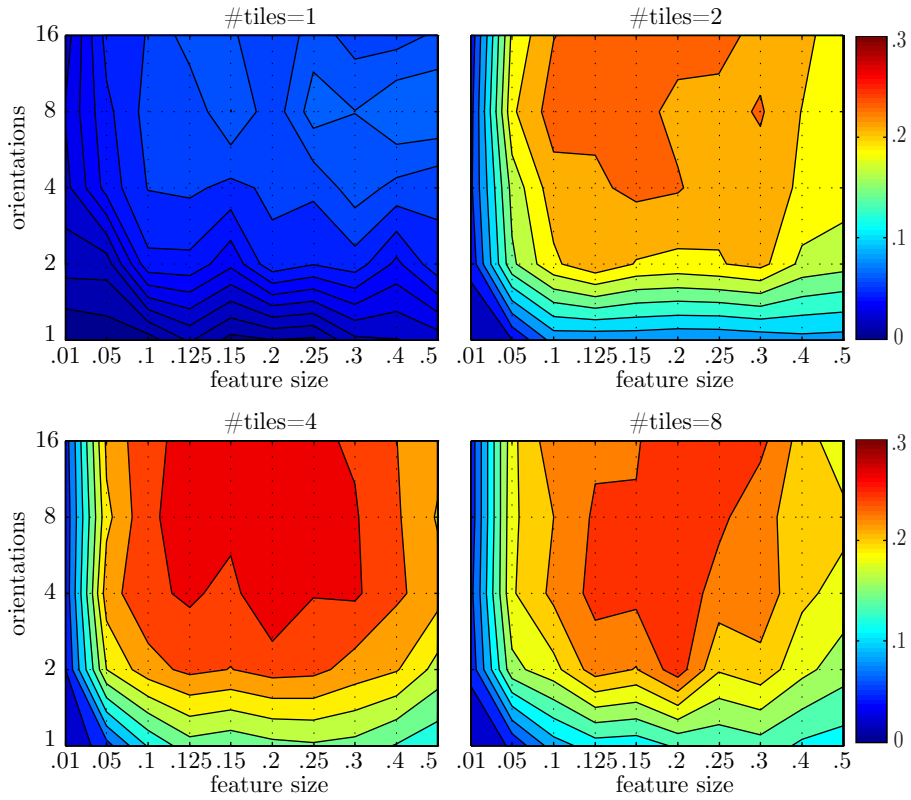
We show all descriptor parameter evaluation plots for the sketch-based shape retrieval problem (Chapter 7) generated according to the optimization strategy outlined in Section 7.9. The evaluation function in all plots is the ratio of correct nearest neighbors (higher values are better). All plots use the same scale to make them easily comparable with each other.



**Figure A.1:** Retrieval performance evaluation of tinyimage descriptor (down-scaled sketch image directly used as feature vector).

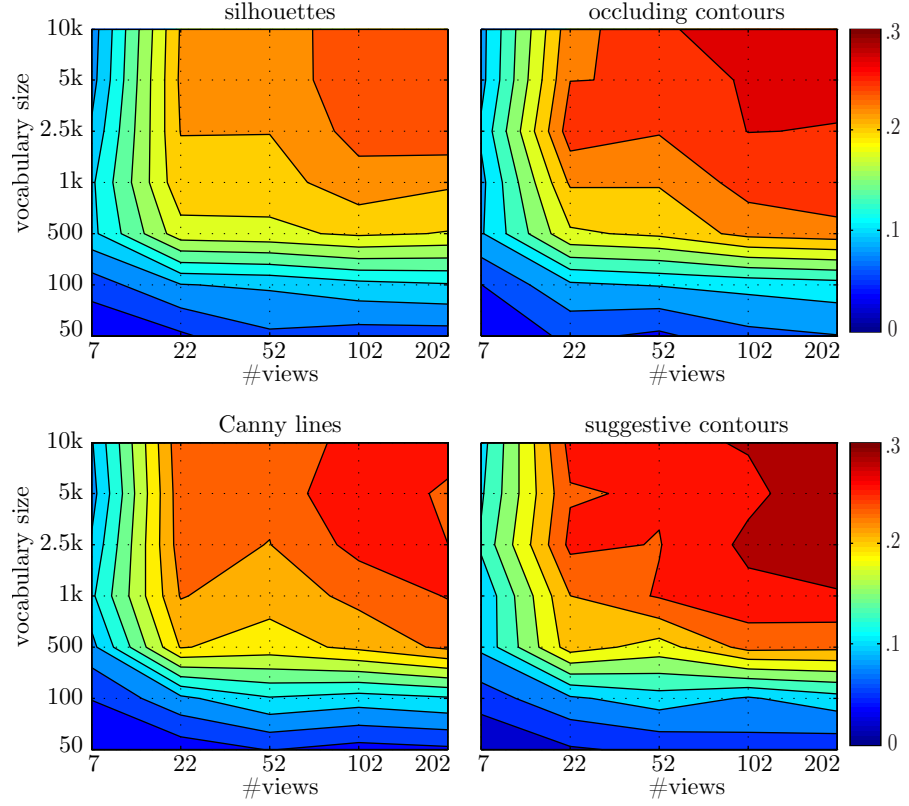


**Figure A.2:** Retrieval performance evaluation of GALIF descriptor with respect to its three intrinsic parameters that define the underlying Gabor filter. X-axis: filter bandwidth  $\lambda$ ; y-axis: linewidth; across plots: peak frequency  $\omega_0$ .

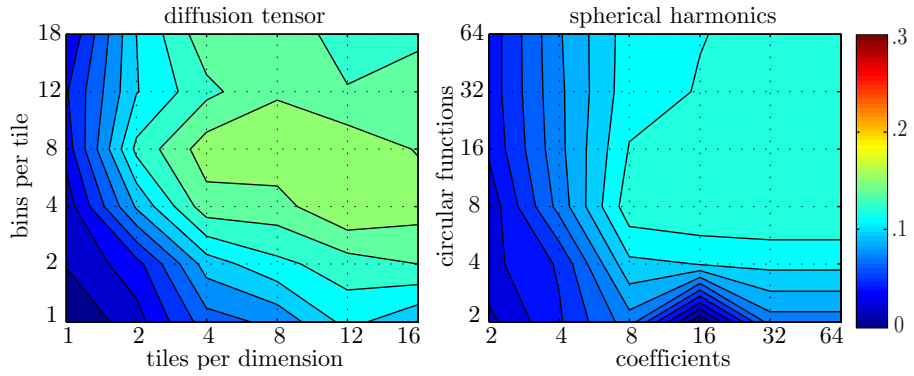


**Figure A.3:** Retrieval performance evaluation of GALIF descriptor with respect to its remaining three intrinsic parameters. X-axis: local feature size in fraction of the total sketch area; y-axis: number of orientations in the local histogram; across plots: number of tiles in a local feature.

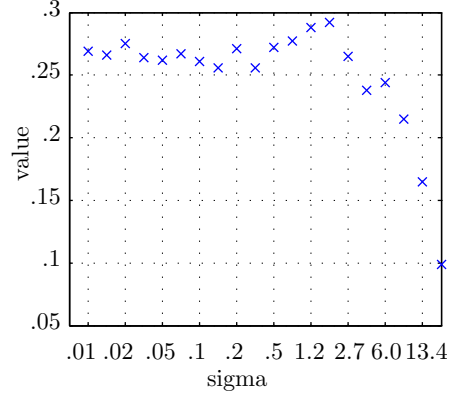
## A. Parameter Space Evaluation



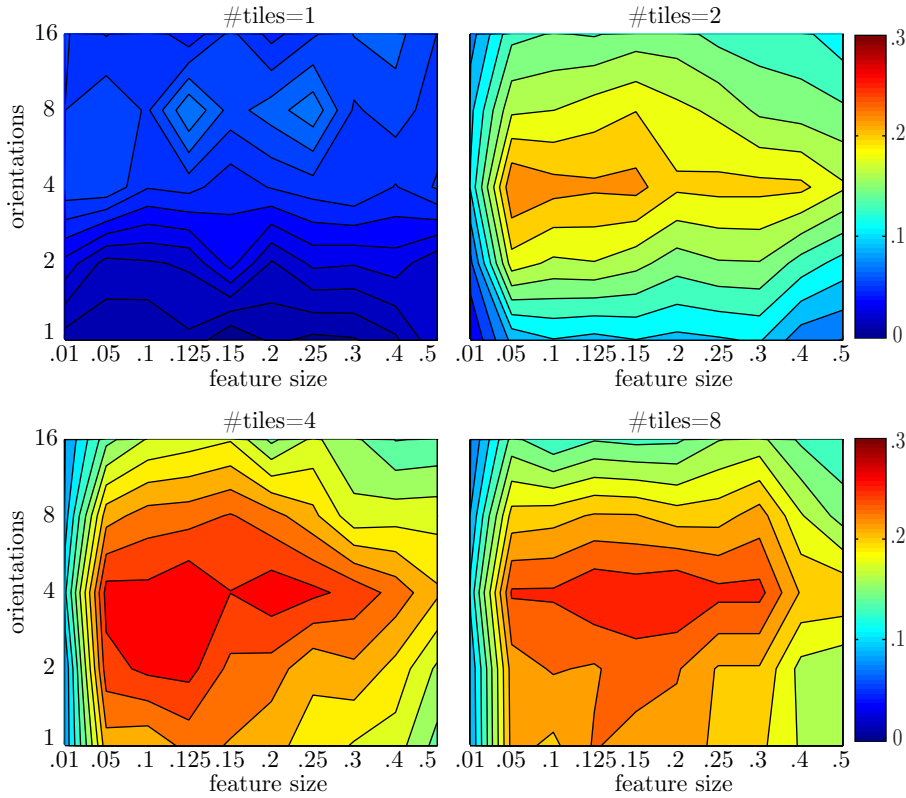
**Figure A.4:** Retrieval performance evaluation of GALIF descriptor with respect to extrinsic parameters of the retrieval pipeline. X-axis: number of views per 3d model; y-axis: vocabulary size for quantization; across plots: line-types used to render a view.



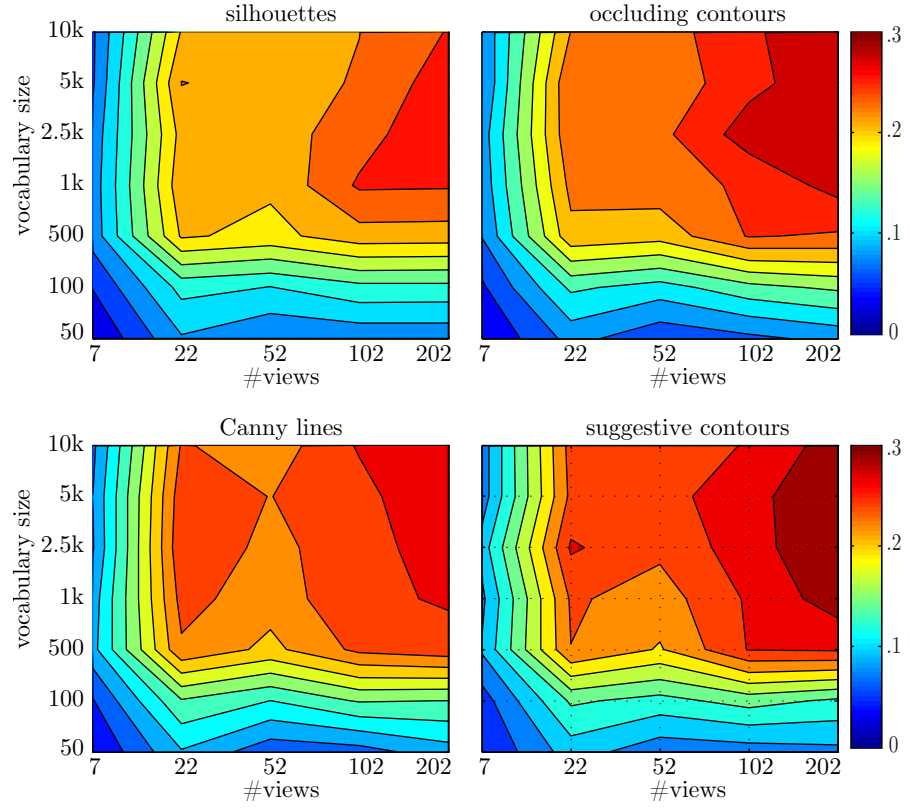
**Figure A.5:** Retrieval performance evaluation of competing approaches over their respective free parameters. Left: diffusion tensor descriptor [Yoon et al. 2010]; right: spherical harmonics descriptor [Funkhouser et al. 2003].



**Figure A.6:** Retrieval performance evaluation of SHOG descriptor with respect to standard deviation  $\sigma$  of its underlying Gaussian derivatives filter. Note the logarithmic scale on the x-axis.



**Figure A.7:** Retrieval performance evaluation of SHOG descriptor with respect to its remaining three intrinsic parameters. X-axis: local feature size in fraction of the total sketch area; y-axis: number of orientations in the local histogram; across plots: number of tiles in a local feature.



**Figure A.8:** Retrieval performance evaluation of SHOG descriptor with respect to extrinsic parameters of the retrieval pipeline. X-axis: number of views per 3d model; y-axis: vocabulary size for quantization; across plots: line-types used to render a view.



---

## Bibliography

---

- S. Agarwal, N. Snavely, I. Simon, S. Seitz, and R. Szeliski. Building Rome in a day. In *International Conference on Computer Vision*, pages 72–79, 2009.
- C. Alvarado and R. Davis. Sketchread: a multi-domain sketch recognition engine. In *User Interface Software and Technology*, pages 23–32, 2004.
- C. Alvarado, M. Oltmans, and R. Davis. A framework for multi-domain sketch recognition. In *AAAI Spring Symposium on Sketch Understanding*, pages 1–8, 2002.
- A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1): 117–122, 2008.
- M. Anelli, L. Cinque, and E. Sangineto. Deformation tolerant generalized hough transform for sketch-based image retrieval in complex scenes. *Image and Vision Computing*, 25(11):1802–1813, 2007.
- M. Ankerst, G. Kastenmüller, H.-P. Kriegel, and T. Seidl. 3d shape histograms for similarity search and classification in spatial databases. In *International Symposium on Advances in Spatial Databases*, pages 207–226, 1999.
- L. Anthony and J. Wobbrock. A lightweight multistroke recognizer for user interface prototypes. In *Graphics Interface*, pages 245–252, 2010.
- R. Arandjelović and T. Szeghin. Sketch recognition by fusion of temporal and image-based features. *Pattern Recognition*, 44(6):1225–1234, 2011.
- Y. Avrithis and Y. Kalantidis. Approximate gaussian mixtures for large scale vocabularies. In *European Conference on Computer Vision*, pages 15–28, 2012.
- P. Barla, J. Thollot, and F. Sillion. Geometric clustering for line drawing simplification. In *Eurographics Symposium on Rendering Techniques*, pages 183–192, 2005.
- H. Bay, T. Tuytelaars, and L. J. V. Gool. SURF: speeded up robust features. In *European Conference on Computer Vision*, pages 404–417, 2006.

- S. Bileschi and L. Wolf. Image representations beyond histograms of gradients: the role of gestalt descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
- K. bo Duan and S. S. Keerthi. Which is the best multiclass SVM method? an empirical study. In *International Workshop on Multiple Classifier Systems*, pages 278–285, 2005.
- O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.
- X. Boix, G. Roig, C. Leistner, and L. Van Gool. Nested sparse quantization for efficient feature coding. In *European Conference on Computer Vision*, pages 744–758, 2012.
- K. Bozas and E. Izquierdo. Large scale sketch based image retrieval using patch hashing. In *Advances in Visual Computing*, pages 210–219, 2012.
- A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov. Shape Google: geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics*, 30(1):1:1–1:20, 2011.
- H. Bülthoff and S. Edelman. Psychophysical support for a two-dimensional view interpolation theory of object recognition. *Proceedings National Academy of Sciences*, 89(1):60–64, 1992.
- H. Cai, F. Yan, and K. Mikolajczyk. Learning weights for codebook in image classification and retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2320–2327, 2010.
- E. J. Candès and D. L. Donoho. Curvelets – a surprisingly effective nonadaptive representation for objects with edges. In *International Conference on Curves and Surfaces*, pages 105–120, 1999.
- J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.
- Y. Cao, C. Wang, Z. Li, L. Zhang, and L. Zhang. Spatial-bag-of-features. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3352–3359, 2010.
- Y. Cao, C. Wang, L. Zhang, and L. Zhang. Edgel index for large-scale sketch-based image search. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–768, 2011.
- A. Chalechale, A. Mertins, and G. Naghdy. Edge image description using angular radial partitioning. *Vision, Image and Signal Processing*, 151(2): 93–101, 2004a.
- A. Chalechale, G. Naghdy, and P. Premaratne. Image database retrieval using sketched queries. In *International Conference on Image Processing*, volume 1, pages 433–436, 2004b.

- A. Chalechale, G. Naghdy, and A. Mertins. Sketch-based image matching using angular partitioning. *IEEE Transactions on Systems, Man and Cybernetics*, 35(1):28–41, 2005.
- E. Chalmin, M. Menu, and C. Vignaud. Analysis of rock art painting and technology of palaeolithic painters. *Measurement Science and Technology*, 14(9):1590–1597, 2003.
- Y. Chan, Z. Lei, D. Lopresti, and S. Y. Kung. A feature-based approach for image retrieval by sketch. *Multimedia Storage and Archiving Systems II*, 3229:220–231, 1997.
- N.-S. Chang and K.-S. Fu. Query-by-pictorial-example. *IEEE Transactions on Software Engineering*, 6:519–524, 1980a.
- N.-S. Chang and K.-S. Fu. A relational database system for images. In *Pictorial Information Systems*, pages 288–321, 1980b.
- S.-K. Chang, Q.-Y. Shi, and C.-W. Yan. Iconic indexing by 2-d strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(3):413–428, 1987.
- K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *British Machine Vision Conference*, pages 76.1–76.12, 2011.
- S. Cheema and J. LaViola. Physicsbook: a sketch-based interface for animating physics diagrams. In *International Conference on Intelligent User Interfaces*, pages 51–60, 2012.
- S. Cheema, S. Gulwani, and J. LaViola. Quickdraw: improving drawing experience for geometric diagrams. In *Human Factors in Computing Systems*, pages 1037–1064, 2012.
- D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung. On visual similarity based 3d model retrieval. *Computer Graphics Forum (Proceedings Eurographics)*, 22(3):223–232, 2003.
- T. Chen, M. Cheng, P. Tan, A. Shamir, and S. Hu. Sketch2Photo: Internet image montage. *ACM Transactions on Graphics (Proceedings SIGGRAPH ASIA)*, 28(5):124:1–124:10, 2009.
- O. Chum and J. Matas. Unsupervised discovery of co-occurrence in sparse high dimensional data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3416–3423, 2010.
- O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: automatic query expansion with a generative feature model for object retrieval. In *IEEE International Conference on Computer Vision*, pages 1–8, 2007.
- O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *British Machine Vision Conference*, pages 50.1–50.10, 2008.

- O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: finding a (thick) needle in a haystack. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 17–24, 2009.
- F. Cole, A. Golovinskiy, A. Limpaecher, H. S. Barros, A. Finkelstein, T. Funkhouser, and S. Rusinkiewicz. Where do people draw lines? *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 27(3):88:1–88:11, 2008.
- F. Cole, K. Sanik, D. DeCarlo, A. Finkelstein, T. Funkhouser, S. Rusinkiewicz, and M. Singh. How well do line drawings depict shape? *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 28:28:1–28:9, 2009.
- J. P. Collomosse, G. McNeill, and Y. Qian. Storyboard sketches for content based video retrieval. In *IEEE International Conference on Computer Vision*, pages 245–252, 2009.
- T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.
- G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Workshop on Statistical Learning in Computer Vision*, 2004.
- P. Daras and A. Axenopoulos. A 3d shape retrieval framework supporting multimodal queries. *International Journal of Computer Vision*, 89(2):229–247, 2010.
- R. Datta, D. Joshi, J. Li, and J. Wang. Image retrieval: ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):1–60, 2008.
- J. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Optical Society of America, Journal, A: Optics and Image Science*, 2(7):1160–1169, 1985.
- R. Davis. Magic paper: sketch-understanding research. *IEEE Computer*, 40(9):34–41, 2007.
- D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella. Suggestive contours for conveying shape. *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 22(3):848–855, 2003.
- A. Del Bimbo and P. Pala. Visual image retrieval by elastic matching of user sketches. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):121–132, 1997.
- E. Di Sciascio, G. Mingolla, and M. Mongiello. Content-based image retrieval over the web using query by sketch and relevance feedback. In *Visual Information and Information Systems*, pages 123–130, 1999.
- N. Diakopoulos, I. Essa, and R. Jain. Content based image synthesis. In *Conference on Image and Video Retrieval*, pages 299–307, 2004.

- D. Dixon, M. Prasad, and T. Hammond. iCanDraw?: using sketch recognition and corrective feedback to assist a user in drawing human faces. In *International Conference on Human Factors in Computing Systems*, pages 897–906, 2010.
- P. Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.
- H. Dutagaci, C. P. Cheung, and A. Godil. A benchmark for best view selection of 3d objects. In *ACM Workshop on 3d Object Retrieval*, pages 45–50, 2010.
- M. Egenhofer. Query processing in spatial-query-by-sketch. *Journal of Visual Languages and Computing*, 8(4):403–424, 1997.
- M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa. A descriptor for large scale image retrieval based on sketched feature lines. In *Sketch-Based Interfaces and Modeling*, pages 29–36, 2009.
- M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa. An evaluation of descriptors for large-scale image retrieval from sketched feature lines. *Computers & Graphics*, 34(5):482–498, 2010.
- M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa. Sketch-based image retrieval: benchmark and bag-of-features descriptors. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1624–1636, 2011a.
- M. Eitz, R. Richter, K. Hildebrand, T. Boubekeur, and M. Alexa. Photosketcher: interactive sketch-based image synthesis. *IEEE Computer Graphics and Applications*, 31(6):56–66, 2011b.
- M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 31(4):44:1–44:10, 2012a.
- M. Eitz, R. Richter, T. Boubekeur, K. Hildebrand, and M. Alexa. Sketch-based shape retrieval. *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 31(4):31:1–31:10, 2012b.
- M. Elad, A. Tal, and S. Ar. Content based retrieval of VRML objects: an iterative and interactive approach. In *Eurographics Workshop on Multimedia*, pages 107–118, 2002.
- M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.
- M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Xanker. Query by image and video content: the QBIC system. *IEEE Computer*, 28(9):23–32, 1995.
- D. A. Forsyth. Benchmarks for storage and retrieval in multimedia databases. In *Storage and Retrieval for Media Databases*, pages 240–247, 2002.

- H. Fu, S. Zhou, L. Liu, and N. Mitra. Animated construction of line drawings. *ACM Transactions on Graphics (Proceedings SIGGRAPH ASIA)*, 30(6):133:1–133:10, 2011.
- T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. A search engine for 3D models. *ACM Transactions on Graphics*, 22(1):83–105, 2003.
- T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 23(3):652–663, 2004.
- T. Furuya and R. Ohbuchi. Dense sampling and fast encoding for 3d model retrieval using bag-of-visual features. In *International Conference on Image and Video Retrieval*, pages 26:1–26:8, 2009.
- D. Gabor. Theory of communication. Part 1: the analysis of information. *Journal of the Institution of Electrical Engineers*, 93(26):429–441, 1946.
- M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH*, pages 209–216, 1997.
- B. Georgescu, I. Shimshoni, and P. Meer. Mean shift based clustering in high dimensions: a texture classification example. In *IEEE International Conference on Computer Vision*, pages 456–463, 2003.
- G. Griffin and P. Perona. Learning and using taxonomies for fast visual categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.
- G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007.
- M. D. Gross. The electronic cocktail napkin: a computational environment for working with design diagrams. *Design Studies*, 17(1):53–69, 1996.
- T. Hammond and R. Davis. Tahuti: A geometrical sketch recognition system for UML class diagrams. In *AAAI Spring Symposium on Sketch Understanding*, pages 59–66, 2002.
- T. Hammond and R. Davis. LADDER, a sketching language for user interface developers. *Computers & Graphics*, 29(4):518–532, 2005.
- T. Hammond, B. Eoff, B. Paulson, A. Wolin, K. Dahmen, J. Johnston, and P. Rajan. Free-sketch recognition: putting the chi in sketching. In *Human Factors in Computing Systems*, pages 3027–3032, 2008.
- C. F. Herot. Graphical input through machine recognition of sketches. *Computer Graphics*, 10(2):97–102, 1976.
- D. Hoffman and M. Singh. Saliency of visual parts. *Cognition*, 63(1):29–78, 1997.
- S. Hou and K. Ramani. Sketch-based 3d engineering part class browsing and retrieval. In *Sketch-Based Interfaces and Modeling*, pages 131–138, 2006.

- S. Hou and K. Ramani. Classifier combination for sketch-based 3d part retrieval. *Computers & Graphics*, 31(4):598–609, 2007.
- R. Hu, M. Barnard, and J. Collomosse. Gradient field descriptor for sketch based retrieval and localization. In *IEEE International Conference on Image Processing*, pages 1025–1028, 2010.
- R. Hu, T. Wang, and J. Collomosse. A bag-of-regions approach to sketch-based image retrieval. In *IEEE International Conference on Image Processing*, pages 3661–3664, 2011.
- T. Hurtut, Y. Gousseau, F. Cheriet, and F. Schmitt. Pictorial analysis of line-drawings. In *Symposium on Computational Aesthetics in Graphics*, pages 123–130, 2008.
- T. Igarashi, S. Matsuoka, S. Kawachiya, and H. Tanaka. Interactive beautification: a technique for rapid geometric design. In *User Interface Software and Technology*, pages 105–114, 1997.
- T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3d freeform design. In *Proceedings of SIGGRAPH*, pages 409–416, 1999.
- H. Ip, A. Cheng, W. Wong, and J. Feng. Affine-invariant sketch-based retrieval of images. In *Computer Graphics International*, pages 55–61, 2001.
- C. E. Jacobs, A. Finkelstein, and D. H. Salesin. Fast multiresolution image querying. In *Proceedings of SIGGRAPH*, pages 277–286, 1995.
- A. Jain and A. Vailaya. Image retrieval using color and shape. *Pattern recognition*, 29(8):1233–1244, 1996.
- H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European Conference on Computer Vision*, pages 304–317, 2008.
- H. Jégou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, 87(3):316–336, 2010.
- G. Johnson, M. Gross, and J. Hong. Computational support for sketching in design: a review. *Foundations and Trends in Human Computer Interaction*, 2(1):1–93, 2009.
- J. Jones and L. Palmer. An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology*, 58(6):1233–1258, 1987.
- H. Kang, S. Lee, and C. Chui. Coherent line drawing. In *Non-Photorealistic Animation and Rendering*, pages 43–50, 2007.
- L. Kara and T. Stahovich. An image-based, trainable symbol recognizer for hand-drawn sketches. *Computers & Graphics*, 29(4):501–517, 2005.
- O. A. Karpenko and J. F. Hughes. Smoothsketch: 3d free-form shapes from complex sketches. *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 25(3):589–598, 2006.

- T. Kato, T. Kurita, N. Otsu, and K. Hirata. A sketch retrieval method for full color image database-query by visual example. In *International Conference on Pattern Recognition*, pages 530–533, 1992.
- R. Kumar Rajendran and S. Chang. Image retrieval with sketches and compositions. In *IEEE International Conference on Multimedia and Expo*, pages 717–720, 2000.
- J. Landay and B. Myers. Sketching interfaces: toward more human interface design. *IEEE Computer*, 34(3):56–64, 2001.
- J. J. LaViola Jr. and R. Zeleznik. MathPad<sup>2</sup>: a system for the creation and exploration of mathematical sketches. *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 23(3):432–440, 2004.
- S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2169–2178, 2006.
- D. Lee, Q. Ke, and M. Isard. Partition min-hash for partial duplicate image discovery. In *European Conference on Computer Vision*, pages 648–662, 2010.
- J. Lee and T. Funkhouser. Sketch-based search and composition of 3d models. In *Sketch-Based Interfaces and Modeling*, pages 97–104, 2008.
- W. Lee, L. Burak Kara, and T. Stahovich. An efficient graph-based recognizer for hand-drawn symbols. *Computers & Graphics*, 31(4):554–567, 2007.
- Y. Lee, C. Zitnick, and M. Cohen. ShadowDraw: real-time user guidance for freehand drawing. *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 30(4):27:1–27:10, 2011.
- T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44, 2001.
- Y. Li. Protractor: a fast and accurate gesture recognizer. In *Human Factors in Computing Systems*, pages 2169–2172, 2010.
- Z. Li, K. Yap, and X. Chen. Beyond bag of words: Combining generative and discriminative models for natural scene categorization. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 965–968, 2011.
- L. Liu, L. Wang, and X. Liu. In defense of soft-assignment coding. In *IEEE International Conference on Computer Vision*, pages 2486–2493, 2011.
- T. Liu, A. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. *Advances in Neural Information Processing Systems*, 17:825–832, 2004.
- S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.



- J. Löffler. Content-based retrieval of 3d models in distributed web databases by visual shape information. In *International Conference on Information Visualization*, pages 82–87, 2000.
- S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, 1998.
- D. Lopresti and A. Tomkins. Temporal domain matching of hand-drawn pictorial queries. In *Conference of The International Graphonomics Society*, pages 98–114, 1995.
- D. Lopresti, A. Tomkins, and J. Zhou. Algorithms for matching hand-drawn sketches. In *International Workshop on Frontiers in Handwriting Recognition*, pages 233–238, 1996.
- D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- J. Lu, F. Yu, A. Finkelstein, and S. DiVerdi. HelpingHand: example-based stroke stylization. *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 31(4):46:1–46:10, 2012.
- D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004.
- S. Matusiak, M. Daoudi, T. Blu, and O. Avaro. Sketch-based images database retrieval. In *Advances in Multimedia Information Systems*, pages 185–191, 1998.
- M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009.
- T. Napoléon and H. Sahbi. From 2d silhouettes to 3d object retrieval: contributions and benchmarking. *EURASIP Journal in Image and Video Processing*, pages 1:1–1:22, 2010.
- A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. FiberMesh: designing freeform surfaces with 3d curves. *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 26(3):41:1–41:9, 2007.
- D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, 2006.
- E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *European Conference on Computer Vision*, pages 490–503, 2006.
- A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

- A. Oliva and A. Torralba. Building the gist of a scene: the role of global image features in recognition. *Progress in Brain Research*, 155:23–36, 2006.
- L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge. Sketch-based modeling: a survey. *Computers & Graphics*, 33(1):85 – 103, 2009.
- M. Oltmans. *Envisioning sketch recognition: a local feature based approach to recognizing informal sketches*. PhD thesis, Massachusetts Institute of Technology, 2007.
- B. Ommer and J. Buhmann. Learning the compositional nature of visual object categories for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):501–516, 2010.
- R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Transactions on Graphics*, 21(4):807–832, 2002.
- T. Ouyang and R. Davis. ChemInk: a natural real-time recognition system for chemical drawings. In *International Conference on Intelligent User Interfaces*, pages 267–276, 2011.
- B. Paulson and T. Hammond. PaleoSketch: accurate primitive sketch recognition and beautification. In *International Conference on Intelligent User Interfaces*, pages 1–10, 2008.
- J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, Jan 2007.
- J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: improving particular object retrieval in large scale image databases. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.
- J. C. Platt. Sequential minimal optimization: a fast algorithm for training support vector machines. In *Advances in Kernel Methods — Support Vector Learning*. MIT Press, 1998.
- J. Pu, K. Lou, and K. Ramani. A 2d sketch-based user interface for 3d CAD model retrieval. *Computer-Aided Design and Applications*, 2(6):717–725, 2005.
- Y. Qi, M. Szummer, and T. Minka. Diagram structure recognition by bayesian conditional random fields. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 191–196, 2005.
- R. Raguram and S. Lazebnik. Computing iconic summaries of general visual concepts. In *IEEE Workshop on Internet Vision*, pages 1–8, 2008.
- A. Rebelo, G. Capela, and J. Cardoso. Optical recognition of music symbols. *International Journal on Document Analysis and Recognition*, 13:19–31, 2010.
- D. Rubine. Specifying gestures by example. *Computer Graphics (Proceedings SIGGRAPH)*, 25(4):329–337, 1991.

- Y. Rubner, C. Tomasi, and L. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.
- B. Russell, A. Torralba, K. Murphy, and W. Freeman. LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1):157–173, 2008.
- G. Salton. The state of retrieval system evaluation. *Information Processing & Management*, 28(4):441–449, 1992.
- G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.
- G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.
- H. Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.
- B. Schölkopf and A. Smola. *Learning with Kernels*. The MIT Press, 2002.
- A. Secord, J. Lu, A. Finkelstein, M. Singh, and A. Nealen. Perceptual models of viewpoint preference. *ACM Transactions on Graphics*, 30(5):109:1–109:12, 2011.
- T. Sezgin and R. Davis. Sketch recognition in interspersed drawings using time-based graphical models. *Computers & Graphics*, 32(5):500–510, 2008.
- T. M. Sezgin and R. Davis. Sketch interpretation using multiscale models of temporal patterns. *IEEE Computer Graphics and Applications*, 27(1):28–37, 2007.
- T. M. Sezgin, T. Stahovich, and R. Davis. Sketch based interfaces: early processing for sketch understanding. In *Workshop on Perceptive User Interfaces*, pages 1–8, 2001.
- T. Shao, W. Xu, K. Yin, J. Wang, K. Zhou, and B. Guo. Discriminative sketch-based 3d model retrieval via robust shape matching. *Computer Graphics Forum (Proceedings Pacific Graphics)*, 30(7):2011–2020, 2011.
- D. Sharon and M. van de Panne. Constellation models for sketch recognition. In *Sketch-Based Interfaces and Modeling*, pages 19–26, 2006.
- P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton shape benchmark. In *Shape Modeling International*, pages 167–178, 2004.
- H. Shin and T. Igarashi. Magic canvas: interactive design of a 3-d scene prototype from freehand sketches. In *Graphics Interface*, pages 63–70, 2007.
- A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros. Data-driven visual similarity for cross-domain image matching. *ACM Transactions on Graphics (Proceedings SIGGRAPH ASIA)*, 30(6):154:1–154:10, 2011.

- J. Sivic and A. Zisserman. Video Google: a text retrieval approach to object matching in videos. In *IEEE International Conference on Computer Vision*, pages 1470–1477, 2003.
- M. Slaney. Precision-recall is wrong for multimedia. *IEEE Multimedia*, 18(3): 4–7, 2011.
- M. Springmann, D. Kopp, and H. Schuldt. QbS: searching for known images using user-drawn sketches. In *International Conference on Multimedia Information Retrieval*, pages 417–420, 2010.
- D. Squire, W. Mueller, H. Mueller, and J. Raki. Content-based query of image databases, inspirations from text retrieval: inverted files, frequency-based weights and relevance feedback. In *Scandinavian Conference on Image Analysis*, pages 7–11, 1999.
- D. Squire, W. Müller, H. Müller, and T. Pun. Content-based query of image databases: inspirations from text retrieval. *Pattern Recognition Letters*, 21(13):1193–1198, 2000.
- I. Sutherland. SketchPad: a man-machine graphical communication system. In *AFIPS Conference Proceedings*, pages 323–328, 1964.
- J. Tangelder and R. Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia Tools and Applications*, 39:441–471, 2008.
- B. M. ter Haar Romeny. *Front-end vision and multi-scale image analysis*. Springer, 2003.
- A. Torralba, R. Fergus, and W. Freeman. 80 million tiny images: a large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008a.
- A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008b.
- P. Turcot and D. Lowe. Better matching with fewer features: the selection of useful features in large database recognition problems. In *ICCV Workshop on Emergent Issues in Large Amounts of Visual Data*, pages 2109–2116, 2009.
- L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.
- J. van Gemert, J.-M. Geusebroek, C. Veenman, and A. Smeulders. Kernel codebooks for scene categorization. In *European Conference on Computer Vision*, pages 696–709, 2008.
- J. van Gemert, C. Veenman, A. Smeulders, and J. Geusebroek. Visual word ambiguity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1271–1283, 2010.
- V. Viitanen and J. Laaksonen. Spatial extensions to bag of visual words. In *ACM International Conference on Image and Video Retrieval*, pages 37:1–37:8, 2009.

- D. Walther, B. Chai, E. Caddigan, D. Beck, and L. Fei-Fei. Simple line drawings suffice for functional MRI decoding of natural scene categories. *Proceedings National Academy of Sciences*, 108(23):9661–9666, 2011.
- J. Z. Wang, G. Wiederhold, O. Firschein, and S. X. Wei. Wavelet-based image indexing techniques with partial sketch retrieval capability. In *IEEE International Forum on Research and Technology Advances in Digital Libraries*, pages 13–24, 1997.
- R. Weber, H. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *International Conference on Very Large Data Bases*, pages 194–205, 1998.
- K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 1473–1480, 2006.
- J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *IEEE International Conference on Computer Vision*, pages 1800–1807, 2005.
- I. Witten, A. Moffat, and T. Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 1999.
- J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a \$1 recognizer for user interface prototypes. In *User Interface Software and Technology*, pages 159–168, 2007.
- J. Wu and J. Rehg. Beyond the Euclidean distance: creating effective visual codebooks using the histogram intersection kernel. In *IEEE International Conference on Computer Vision*, pages 630–637, 2009.
- T. Wu, C. Lin, and R. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.
- Z. Wu, Q. Ke, M. Isard, and J. Sun. Bundling features for large scale partial-duplicate web image search. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 25–32, 2009.
- J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, 2010.
- J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1794–1801, 2009.
- J. Yang, K. Yu, and T. Huang. Efficient highly over-complete sparse coding using a mixture model. In *European Conference on Computer Vision*, pages 113–126, 2010a.
- J. Yang, K. Yu, and T. Huang. Supervised translation-invariant sparse coding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3517–3524, 2010b.

- S. Yoon, M. Scherer, T. Schreck, and A. Kuijper. Sketch-based 3d model retrieval using diffusion tensor fields of suggestive contours. In *International Conference on Multimedia*, pages 193–200, 2010.
- J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: a comprehensive study. *International Journal of Computer Vision*, 73(2):213–238, 2007.
- S. Zhang, Q. Tian, G. Hua, Q. Huang, and S. Li. Descriptive visual words and visual phrases for image applications. In *International Conference on Multimedia*, pages 75–84, 2009.
- L. Zhu, A. Zhang, A. Rao, and R. Srihari. Keyblock: an approach for content-based image retrieval. In *International Conference on Multimedia*, pages 157–166, 2000.
- C. Zitnick. Binary coherent edge descriptors. In *European Conference on Computer Vision*, pages 170–182, 2010.
- J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38(2):1–56, 2006.