Sketch-Based Shape Retrieval

Mathias Eitz* TU Berlin
Ronald Richter TU Berlin
Tamy Boubekeur Telecom ParisTech – CNRS
Kristian Hildebrand TU Berlin
Marc Alexa TU Berlin

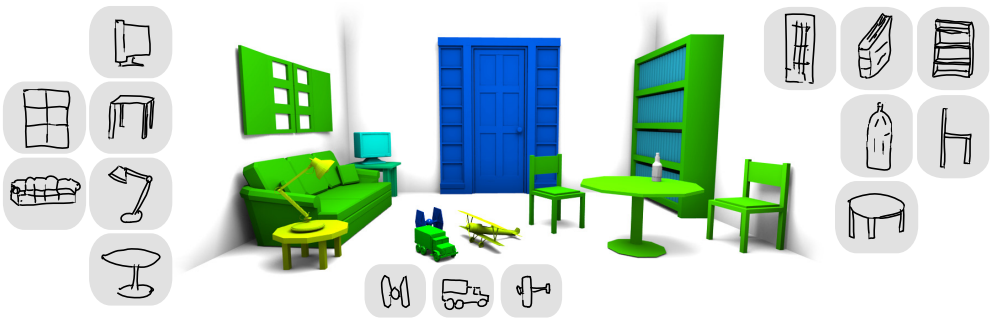

Figure 1: *A complete scene with objects retrieved using our sketch-based system in a total time of about two minutes.*

Abstract

We develop a system for 3D object retrieval based on sketched feature lines as input. For objective evaluation, we collect a large number of query sketches from human users that are related to an existing data base of objects. The sketches turn out to be generally quite abstract with large local and global deviations from the original shape. Based on this observation, we decide to use a bag-of-features approach over computer generated line drawings of the objects. We develop a targeted feature transform based on Gabor filters for this system. We can show objectively that this transform is better suited than other approaches from the literature developed for similar tasks. Moreover, we demonstrate how to optimize the parameters of our, as well as other approaches, based on the gathered sketches. In the resulting comparison, our approach is significantly better than any other system described so far.

CR Categories: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—Retrieval models; I.3.3 [Computer Graphics]: Picture/Image Generation—Line and Curve Generation;

Keywords: Shape Retrieval, Visual Search, Bag-of-Features, Local Descriptors

Links: DL PDF

1 Introduction

Working with large collections of 3D models requires fast content-based retrieval techniques, especially since public collections are often insufficiently annotated. In that 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 good example is often not at hand. Instead, 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 additionally deviate significantly from that projection. These properties of the input directly translate into desiderata of sketch-based shape retrieval systems: a) *partial matching* of feature lines of the shape in b) *all potential viewing directions* to the sketch, tolerating c) *global and local deformation*; and, clearly, the retrieval performance has to d) *scale* to large collections.

We present, to our knowledge, the first approach that addresses all of the 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.

We make use of successful techniques from other domains where appropriate and provide the following novel contributions:

- A large-scale benchmark for sketch-based retrieval systems. The benchmark is based on a real-world dataset of 1,914 sketches gathered from a large variety of participants in a perceptual experiment. We provide this dataset as a free resource.

*e-mail: m.eitz@tu-berlin.de

- A new feature transform based on a bank of Gabor filters that is tuned to the requirements of sketch-based shape retrieval. This 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 our objective evaluation as well as the accompanying video using a large variety of real-world user sketches. We also demonstrate the power of our system in Fig. 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 Sec. 8.

2 Related work

While there exists a huge amount of work on example-based model retrieval (we refer the reader to the survey of Tangelder et al. [2008]), 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 later in this paper.

2.1 Sketch-based model retrieval

One of the earliest references to sketch-based model retrieval is given by Löffler [2000] who describe 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 orthographics 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 quantitive evaluation for sketch-based retrieval. Yoon et al. [2010] propose measuring orientation of sketch lines using the diffusion tensor – as the final descriptor they propose an orientation histogram that globally encodes each view of a model.

2.2 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] 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, they show best matching classes of models and second, they sort the models *within each class* according to similarity to the query.

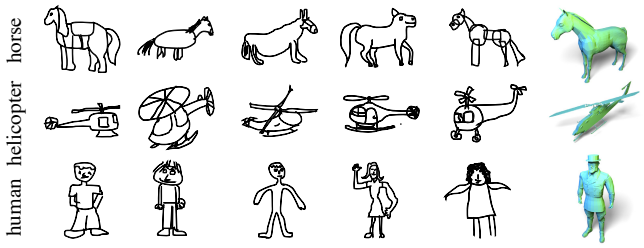

Figure 2: *Subsets of the sketches gathered for the benchmark, each sketch corresponds to a specific category from the PSB. Examples of corresponding 3D shapes are shown on the right.*

2.3 Sketch-based synthesis

Using sketch-based interaction to create new content is an intriguing idea: users could assemble an object from existing 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. Sketch-based interaction is also used to synthesize novel images from a sketch and additional keyword label [Chen et al. 2009] or sketch alone [Eitz et al. 2011]. Lee et al. [2011] interactively synthesize a shadow that helps users create better sketches.

3 How do people sketch for 3D retrieval?

We report on a large-scale experiment which tries to provide insight into the following problem: how would an average user of a 3D retrieval system sketch the query? Most 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].

3.1 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. We emphasize that the sketch should be clearly recognizable for other humans. We perform the experiment on Amazon Mechanical Turk and provide participants with a web-based drawing tool that supports the usual undo/redo/erase functionality. This setup is designed to closely resemble how novel users would use a retrieval system.

We ask for all categories that are defined in the Princeton Shape Benchmark (PSB) [Shilane et al. 2004] and make sure that we gather sketches for all models in the PSB (i.e. 1,914 sketches). There is no direct association between sketches and models but rather between sketch category and model category. We later exploit this association to define a benchmark for shape retrieval.

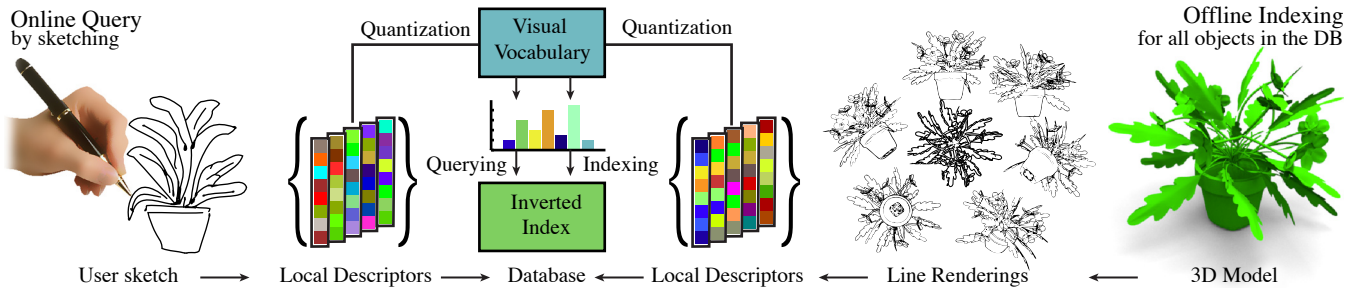

Figure 3: Overview of our 3D shape search engine. *The offline shape indexing and online querying stages differ only in the way we generate input feature lines: automatic line rendering from 3D shapes (right) or hand-sketched by users (left).*

3.2 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 e.g. the sketches in Fig. 2. This clearly motivates developing systems that support arbitrarily complex sketches such as the one presented in the following sections. 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 e.g. a table however often tend to be drawn in perspective (82.6% in our experiments), although typically with significant perspective error.

3.3 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 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). In the remainder of this paper 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.

4 Bag-of-features shape retrieval

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. This requires several steps:

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

4.1 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 data base. 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 data base, so a perfect alignment of a view to the sketch is impossible. Finally, 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 data base. This general approach still leaves several design choices for the different steps: a) for an object, define the set of projection directions; b) for the projection, define a certain line drawing style; c) for a line drawing, define the set of sampling locations and d) for a drawing sample, define the feature transform. We note that the first two items are only necessary in a preprocessing phase, in which we generate the set of clusters from the data base. The data base forms a finitely sized "visual vocabulary" $\mathcal{V}$. This yields a representation for each projection in the data base by a distribution of "visual words" $v_i \in \mathcal{V}$ (see Fig. 3). In the query phase, we compute a similar distribution for the input and compare it against the elements of the data base.

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 (Sec. 3). Our major contribution, the feature transformation, is described in an extra section.

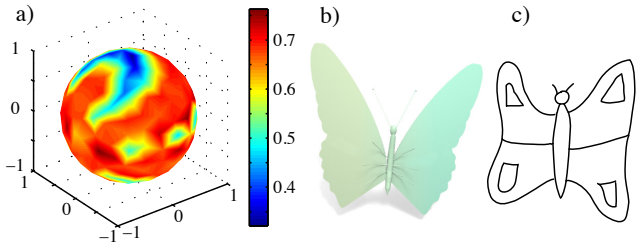

Figure 4: *Best-view selection: a) best view probability map predicted by the SVM model, b) predicted best view, c) user sketch*

4.2 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 *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 viewpoints have different orientations assigned to them. This approximates a globally rotation-invariant indexing of the shapes. 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 $\boldsymbol{M} = \{\boldsymbol{V}, \boldsymbol{T}\}$, with $\boldsymbol{V}$ a set of vertices and $\boldsymbol{T}$ a set of triangles, we select a set $\boldsymbol{S}$ of d random seed vertices among $\boldsymbol{V}$ and perform Lloyd relaxations iteratively. After convergence, we return the resulting Voronoi cell centers as the view directions v_i. We use $\mathrm{d} \in \{7, 22, 52, 102, 202\}$. The number of samples d is an important parameter – we determine its optimal value in Sec. 6.

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 PSB for all of which we manually define both a best as well as a worst viewpoint. We extract the following three image based measures for each best and worst view in this training set: a) silhouette length in image space, relative to image area b) projected area relative to image area and c) smoothness of depth distribution over the model.

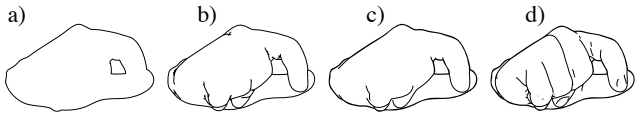

Figure 5: *Comparison of different line rendering approaches: (a) silhouettes (SH), (b) Canny lines from depth image (CFD), (c) occluding contours (OC) and (d) and Suggestive Contours (SC)*

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. This results in a smooth scalar field over the sphere Fig. 4a) 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 Fig. 4.

4.3 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] and 4) Canny lines [Canny 1986] from the depth image. We illustrate those styles for a given model in Fig. 5.

4.4 Sampling local features

We encode each view image as a bag of many small local image patches transformed into an appropriate feature space (see Fig. 7 for an illustration). 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.

4.5 Representation

We generate a visual vocabulary using k-means clustering [Lloyd 1982]. 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 $\mathcal{C} = \{\boldsymbol{c}_j\}$ forms the visual vocabulary where each entry $\boldsymbol{c}_j$ (visual word) represents the local features in cluster j. The size of the visual vocabulary $|\mathcal{C}|$, i.e. the number of clusters, is an important parameter that strongly influences retrieval performance and we determine its optimal value in Sec. 6.

We represent each view as histogram of visual word frequency. We quantize all local features $\boldsymbol{x}_i$ from a given sketch against the visual vocabulary, representing them as the index q_{ij} of their closest visual word:

$$q_{ij} = \arg\min_j \|\boldsymbol{x}_i - \boldsymbol{c}_j\| . \quad (1)$$

We now define the entries h_j of the final histogram of visual word

representation $\boldsymbol{h}$ that encodes a view as:

$$h_j = |\{q_{ij}\}| \,. \tag{2}$$

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 occuring 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] in order to achieve quick lookups during the query stage.

4.6 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 Fig. 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*) [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: $h_j = (h_j / \sum_i h_i) \log(N/f_j)$ 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 occurance histograms [Witten et al. 1999]. Let $\boldsymbol{h}$ and $\bar{\boldsymbol{h}}$ be two histograms of visual words (representing two images). We define their similarity as

$$s(\boldsymbol{h}, \bar{\boldsymbol{h}}) = \langle \boldsymbol{h}, \bar{\boldsymbol{h}} \rangle / \|\boldsymbol{h}\| \|\bar{\boldsymbol{h}}\|. \tag{3}$$

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 $\boldsymbol{h}$ computed from a user sketch, we retrieve similar models in two steps: first, we find similar views, querying the inverted index. This operation accounts to computing Eqn. (3) between $\boldsymbol{h}$ 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 $\boldsymbol{h}$. 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.

5 GALIF: Gabor local line-based feature

Our search relies on local image descriptors that encode image content within a small local region of a sketch. To balance accuracy and efficiency, local image regions are commonly represented as descriptors, encoding only the essential information in a region.

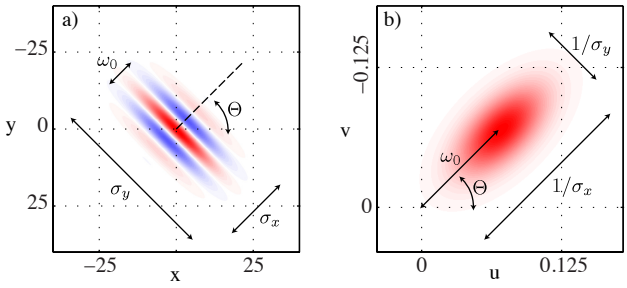

Figure 6: *Gabor filter: a) spatial domain, b) frequency domain. Parameters:* $\sigma_x = 5, \sigma_y = 10, \omega_0 = 0.1, \Theta = \pi/4$

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 descriptor 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 Gabor filters.

5.1 Gabor filter

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

$$\boldsymbol{g}(u, v) = \exp\left(-2\pi^2 \left((u_\Theta - \omega_0)^2 \sigma_x^2 + v_\Theta^2 \sigma_y^2\right)\right) \tag{4}$$

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

ω_0	:	peak response frequency
Θ	:	filter orientation
σ_x	:	frequency bandwidth
σ_y	:	angular bandwidth

We visualize such a filter with its corresponding parameters in Fig. 6. Note that in the frequency domain the filter simply is a Gaussian, see Fig. 6a and Eqn. (4). Multiplication with the sketch in the frequency domain "masks" all content that does not posess the right frequency and orientation: the filter responds only to a subset of the lines in a sketch.

5.2 Orientation-selective filter bank

To compute our feature space transform, we define a filter bank of Gabor functions $\boldsymbol{g}_i$ 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

$$\boldsymbol{R}_i = \|\mathrm{idft}(\boldsymbol{g}_i * \mathrm{dft}(\boldsymbol{I}))\| \tag{5}$$

where $\boldsymbol{I}$ is the input sketch, $*$ denotes point-wise multiplication and idft and dft denote the inverse/forward discrete Fourier transform (see Fig. 7a,b for a visualization).

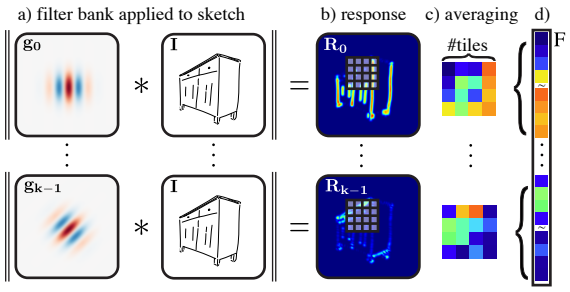

Figure 7: *GALIF feature extraction pipeline: a) we convolve the input sketch $\boldsymbol{I}$ with a filter bank of differently oriented Gabor filters $\boldsymbol{g}_i$ to yield b) response images $\boldsymbol{R}_i$. The average responses c) within cells of a local patch form d) a local feature vector $\boldsymbol{F}$.*

Note that σ_y determines the amount of overlap between filters $\boldsymbol{g}_i$ – depending on the number of orientations in the filter bank. We do not manually fix this value but instead *optimize* it in the next section such that we achieve optimal retrieval results. Given the number of orientations k we define the Θ's used for the filterbank as $\Theta \in \{0, \pi/k, \ldots, (k-1)\pi/k\}$.

5.3 Local GALIF feature definition

Given a keypoint coordinate in image space we consider a regular decomposition of the response images' area around this coordinate into $n \times n$ cells C_{st}. We call this area a local image patch and call n the "number of tiles" (see Fig. 7b,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 the 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.

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

$$\boldsymbol{F}(s, t, i) = \sum_{(x,y) \in C_{st}} \boldsymbol{R}_i(x, y). \tag{6}$$

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

We discard features that do not contain sketch lines and finally normalize such that $\|\boldsymbol{F}\|_2 = 1$. We visualize the complete local feature extraction pipeline in Fig. 7.

6 Optimizing retrieval-system parameters

The parameter-space for the proposed 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 feature transform. However, the wrong choice for just a single parameter 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, 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 (Sec. 3). This is supposed to make the retrieval perform close to what humans expect. However, the optimization approach outlined below is clearly independent of this measure.

6.1 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 keeping all other parameters fixed (see Fig. 8). 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. 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.

6.2 Local feature parameters

To evaluate the influence of the local GALIF feature transform (Sec. 5) on retrieval performance we fix the remaining extrinsic retrieval system parameters to vocabulary size = 1000, num views = 102 and line type = Suggestive Contours.

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 ω_0. The optimal setting for $linewidth$ and λ is coupled to the choice of the ω_0 of the Gabor filter (see Fig. 8a–d). The optimal combination of parameter values turns out to be $linewidth = 0.02$, $\lambda = 0.3$ and $\omega_0 = 0.13$.

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 (see Fig. 8f). In that case the descriptor is no longer discriminative enough – it can only encode vertical and horizontal lines. When using more than four orientations, retrieval performance starts to drop slightly – at the additional cost of a much higher–dimensional descriptor. We offer the following explanation: a high 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 yields the best performing feature in our experiments.

Number of tiles We evaluate using 1, 2, 4, and 8 tiles to subdivide a local image patch (note: 2 tiles means the local image area is subdivided into 2×2 cells). 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.

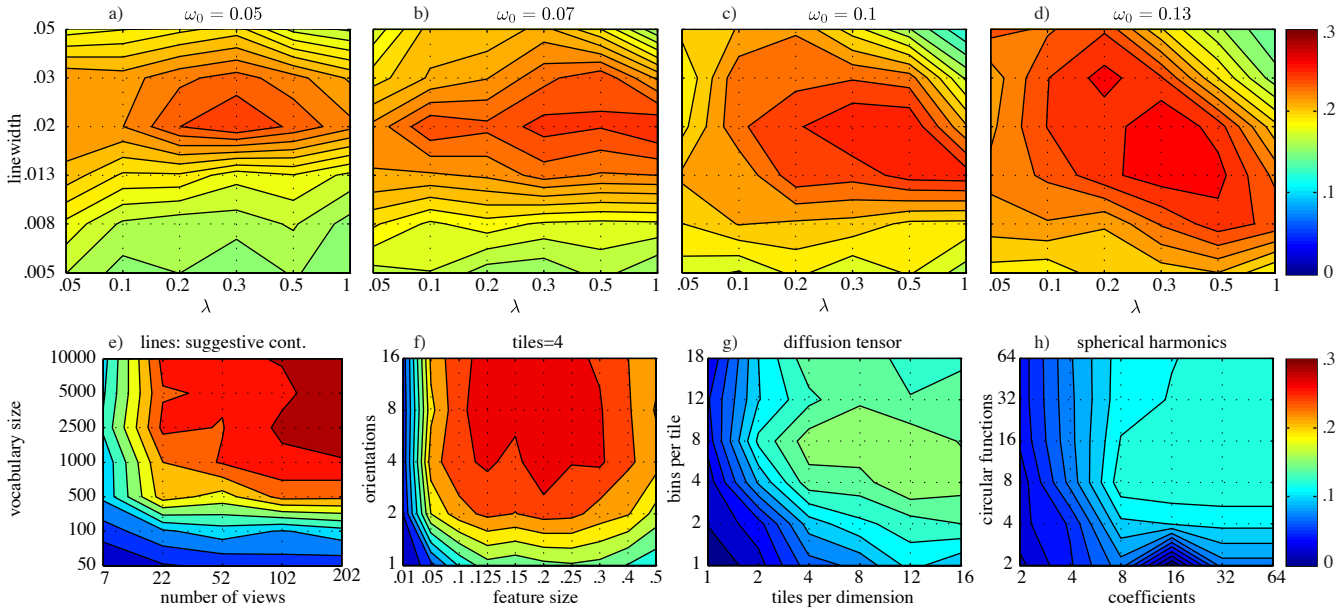

Figure 8: *a)–d) evaluation of GALIF parameters defining the shape of the underlying Gabor filter. e)–f) evaluation of extrinsic system parameters using fixed intrinsic parameters. g) evaluation of optimal parameters for Diffusion Tensor and h) Spherical Harmonics descriptor.*

Local feature size We find that – compared to approaches working with natural images – we achieve optimal results when using relatively large local feature sizes: performance is optimal between feature size 0.1 and 0.3, see Fig. 8f for the visualization. We achieve overall highest performance for feature size 0.2. 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.

6.3 Retrieval system parameters

We now fix descriptor intrinsic parameters to best values determined in Sec. 6.2 and optimize the remaining system parameters.

Vocabulary size When using 1,000 visual words or more, we achieve good retrieval performance, see Fig. 8e. Using more words generally leads 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.

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. An optimal sampling is reached for around 100 uniformly distributed view directions, see Fig. 8e. When using more directions performance begins to saturate. Our perceptual best view selection generates an average of 14.4 views per model, achieving similar results to using 22 regularly sampled views, see Fig. 9b.

Line types Our evaluation of line types used to render views shows that occluding contours and suggestive contours perform better than outlines. Interestingly, the additional information contained in suggestive contours does not provide a significant boost compared to using occluding contours only.

Overall, we can report that the following parameters result in a system with a good performance/speed ratio: vocabulary size: 2.500, number of views: 100, line-type: suggestive contours. For those settings, our performance measure (fraction of correct 1-nearest neighbors) is 0.288.

6.4 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 Fig. 8g. Our optimization finds a more favorable parameter combination with significantly higher retrieval performance (12 tiles with 4 bins each, resulting in a $12\times12\times4$-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 optimally when using 8 functions and 16 coefficients – very close to the original parameters 16×32 (see Fig. 8h).

7 Results

In this section we compare our proposed system (using optimal parameters as determined in Sec. 6) to previous work and analyze its properties – such as partial matching. We perform all evaluations on the test dataset (see Sec. 3).

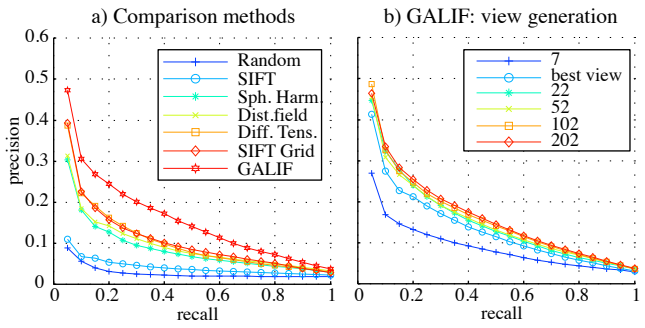

Figure 9: *Detailed evaluation of retrieval performance on test dataset (higher curves are better): a) comparison with previous work using optimized system parameters; b) GALIF descriptor: influence of view generation methods.*

7.1 Comparison to other systems

We compare our approach to three other leading sketch-based retrieval systems. We use standard precision/recall plots to visualize our results, see Fig. 9a,b. 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 Sec. 6. 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 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 Fig. 9a. We visualize the high quality of our results in Fig. 10 as well as the accompanying video. 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 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 Fig. 9): on average only few keypoints are detected, resulting in an imprecise representation of a sketch by its histogram. However, several optimized parameters of the GALIF descriptor 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 descriptor turns out to be identical to the 8 directions used in the SIFT descriptor.

7.2 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 GALIF feature transform encodes 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 Eqn. (3) 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 contained in the second sketch, the two histograms are similar in the areas encoding the shared strokes and the system returns a partial match. We demonstrate this partial matching behavior in Fig. 10 as well as in the accompanying video.

7.3 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 1,914 models from the PSB) is currently at only a few milliseconds for performing the search, meaning the system could accommodate a significantly larger data base.

7.4 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 derivates 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.

8 Conclusions

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. 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] 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

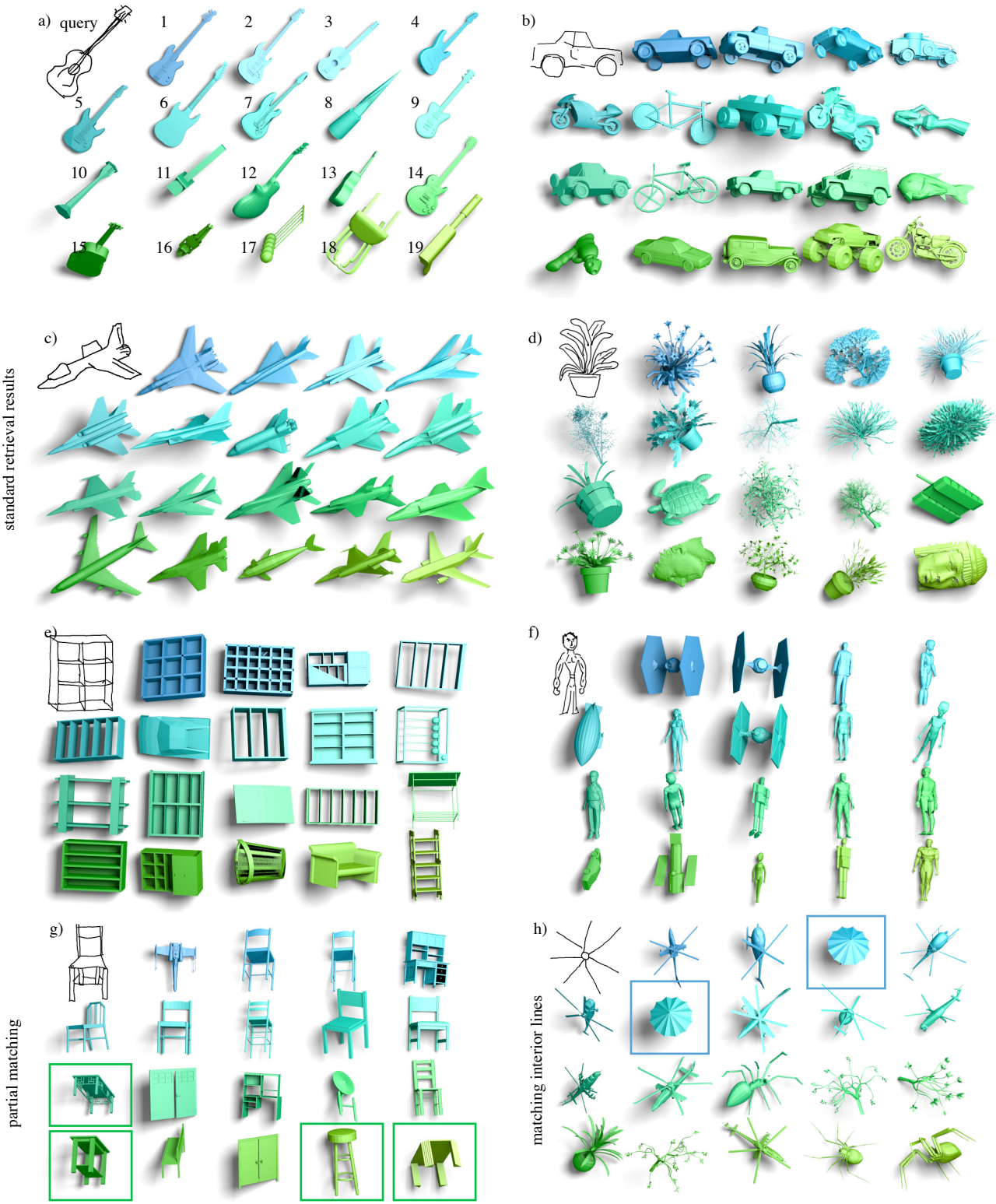

Figure 10: *Examples of sketch-based query results using our system. For each sketch (top left of a cell), we show the top 19 results with a color indicating their rank (blue being the highest, see a)). The "chair" example (g) illustrates partial matching (i.e., tables are retrieved). The "man" example (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.*

parameter values suitable for sketches. This strategy is general and we are interested in seeing its applications in other domains as well.

We have not discussed the user interface we are using 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.

Acknowledgements We thank all participants of our experiments for their sketches, the anonymous reviewers for their constructive comments and James Hays for his help with Amazon Mechanical Turk. This work has been supported in part by the ERC-2010-StG 259550 XSHAPE grant, the REVERIE E.U. project, the 3DLife N.o.E. and the iSpace&Time ANR project.

References

BAY, H., TUYTELAARS, T., AND GOOL, L. J. V. 2006. SURF: Speeded up robust features. In *ECCV*, 404–417.

BÜLTHOFF, H., AND EDELMAN, S. 1992. Psychophysical support for a two-dimensional view interpolation theory of object recognition. *Proc. National Academy of Sciences 89*, 1, 60–64.

CANDÈS, E. J., AND DONOHO, D. L. 1999. Curvelets – a surprisingly effective nonadaptive representation for objects with edges. In *Int'l. Conf. Curves and Surfaces*, 105–120.

CANNY, J. 1986. A computational approach to edge detection. *IEEE TPAMI 8*, 6, 679–698.

CHEN, D.-Y., TIAN, X.-P., SHEN, Y.-T., AND OUHYOUNG, M. 2003. On visual similarity based 3d model retrieval. *Comput. Graph. Forum (Proc. Eurographics) 22*, 3, 223–232.

CHEN, T., CHENG, M., TAN, P., SHAMIR, A., AND HU, S. 2009. Sketch2Photo: internet image montage. *ACM TOG (Proc. SIGGRAPH ASIA) 28*, 5, 124:1–124:10.

COLE, F., GOLOVINSKIY, A., LIMPAECHER, A., BARROS, H. S., FINKELSTEIN, A., FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2008. Where do people draw lines? *ACM TOG (Proc. SIGGRAPH) 27*, 3, 88:1–88:11.

DARAS, P., AND AXENOPOULOS, A. 2010. A 3D shape retrieval framework supporting multimodal queries. *IJCV 89*, 2, 229–247.

DECARLO, D., FINKELSTEIN, A., RUSINKIEWICZ, S., AND SANTELLA, A. 2003. Suggestive contours for conveying shape. *ACM TOG (Proc. SIGGRAPH) 22*, 3, 848–855.

DUTAGACI, H., CHEUNG, C. P., AND GODIL, A. 2010. A benchmark for best view selection of 3D objects. In *Proc. ACM workshop on 3D object retrieval*, 45–50.

EITZ, M., RICHTER, R., HILDEBRAND, K., BOUBEKEUR, T., AND ALEXA, M. 2011. Photosketcher: interactive sketch-based image synthesis. *IEEE CG&A 31*, 6, 56–66.

FUNKHOUSER, T., MIN, P., KAZHDAN, M., CHEN, J., HALDERMAN, A., DOBKIN, D., AND JACOBS, D. 2003. A search engine for 3D models. *ACM TOG 22*, 1, 83–105.

HOU, S., AND RAMANI, K. 2006. Sketch-based 3D engineering part class browsing and retrieval. In *Sketch-Based Interfaces and Modeling*, 131–138.

JACOBS, C. E., FINKELSTEIN, A., AND SALESIN, D. H. 1995. Fast multiresolution image querying. In *Proc. SIGGRAPH 95*, 277–286.

LEE, J., AND FUNKHOUSER, T. 2008. Sketch-based search and composition of 3D models. In *Sketch-Based Interfaces and Modeling*, 97–104.

LEE, Y., ZITNICK, C., AND COHEN, M. 2011. ShadowDraw: real-time user guidance for freehand drawing. *ACM TOG (Proc. SIGGRAPH) 30*, 4, 27:1–27:10.

LLOYD, S. P. 1982. Least squares quantization in PCM. *IEEE Trans. Information Theory 28*, 2, 129–137.

LÖFFLER, J. 2000. Content-based retrieval of 3D models in distributed web databases by visual shape information. In *Int'l. Conf. Information Visualization*, 82–87.

LOWE, D. 2004. Distinctive image features from scale-invariant keypoints. *IJCV 60*, 2, 91–110.

OLIVA, A., AND TORRALBA, A. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV 42*, 3, 145–175.

PU, J., LOU, K., AND RAMANI, K. 2005. A 2D sketch-based user interface for 3D CAD model retrieval. *Computer-Aided Design and Applications 2*, 6, 717–725.

SCHÖLKOPF, B., AND SMOLA, A. 2002. *Learning with Kernels*. The MIT Press.

SECORD, A., LU, J., FINKELSTEIN, A., SINGH, M., AND NEALEN, A. 2011. Perceptual models of viewpoint preference. *ACM TOG 30*, 5, 109:1–109:12.

SHAO, T., XU, W., YIN, K., WANG, J., ZHOU, K., AND GUO, B. 2011. Discriminative sketch-based 3D model retrieval via robust shape matching. *Computer Graphics Forum (Proc. Pacific Graphics) 30*, 7, 2011–2020.

SHILANE, P., MIN, P., KAZHDAN, M., AND FUNKHOUSER, T. 2004. The Princeton Shape Benchmark. In *Proc. Shape Modeling International*, 167–178.

SHIN, H., AND IGARASHI, T. 2007. Magic canvas: interactive design of a 3-D scene prototype from freehand sketches. In *Proc. Graphics Interface*, 63–70.

SIVIC, J., AND ZISSERMAN, A. 2003. Video Google: a text retrieval approach to object matching in videos. In *ICCV*, 1470–1477.

SQUIRE, D., MUELLER, W., MUELLER, H., AND RAKI, J. 1999. Content-based query of image databases. In *Scand. Conf. on Image Analysis*, 143–149.

TANGELDER, J., AND VELTKAMP, R. 2008. A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications 39*, 441–471.

WITTEN, I., MOFFAT, A., AND BELL, T. 1999. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann.

YOON, S., SCHERER, M., SCHRECK, T., AND KUIJPER, A. 2010. Sketch-based 3d model retrieval using diffusion tensor fields of suggestive contours. In *Int'l. Conf. Multimedia*, 193–200.