Pacific Graphics 2011
Bing-Yu Chen, Jan Kautz, Tong-Yee Lee, and Ming C. Lin
(Guest Editors)

Volume 30 (2011), Number 7

Discriminative Sketch-based 3D Model Retrieval via Robust Shape Matching

Tianjia Shao* Weiwei Xu† Kangkang Yin‡ Jingdong Wang† Kun Zhou§ Baining Guo†

* Tsinghua University † Microsoft Research Asia ‡ National University of Singapore § Zhejiang University

Abstract

We propose a sketch-based 3D shape retrieval system that is substantially more discriminative and robust than existing systems, especially for complex models. The power of our system comes from a combination of a contour-based 2D shape representation and a robust sampling-based shape matching scheme. They are defined over discriminative local features and applicable for partial sketches; robust to noise and distortions in hand drawings; and consistent when strokes are added progressively. Our robust shape matching, however, requires dense sampling and registration and incurs a high computational cost. We thus devise critical acceleration methods to achieve interactive performance: precomputing kNN graphs that record transformations between neighboring contour images and enable fast online shape alignment; pruning sampling and shape registration strategically and hierarchically; and parallelizing shape matching on multi-core platforms or GPUs. We demonstrate the effectiveness of our system through various experiments, comparisons, and user studies.

1. Introduction

Recent advances in scanning technologies and modeling tools have dramatically increased the quantity and complexity of publicly available 3D geometric models. For example, TurboSquid, an online commercial repository of 3D models, contains over 200,000 models. Yet the query methods available there are rather basic: one can either search models via keywords, or browse models pre-organized into different categories. The seminal work of Funkhouser et al. [2003] illustrates the simplicity and power of sketch-based interfaces for 3D model retrieval. It not only suggests an easy-to-learn and intuitive-to-use search method, but also opens up a new door to user-driven 3D content creation. Users can now find, reuse, and alter existing contents, all by sketching [SI07, LF08, CCT*09].

However, the progress of sketch-based model retrieval has been slow. There are fundamentally two challenges. First, representing 3D objects using 2D drawings is inherently ambiguous. Most 3D search engines utilize shape descriptors that measure global geometric properties of objects [Lof00, ETA01, OFCD02, FMK*03]. Global shape signatures are fast to compare for shape matching, but cannot discriminate well large amounts of similar models, especially for models with complex interior structures. The proliferation of complex 3D models calls for 2D shape representations that are more discriminative than before.

Second, the success of sketch-based interfaces highly depends on the quality of the query sketch, and unfortunately users who do not have an artistic background usually cannot draw very well. Figure 1(b) is an illustration of this point. Malformed strokes, such as broken or repeat strokes, and noise and outliers representing funny details and decorations, are all common in sketches drawn by novice users. Shape distortions, including translation, rotation, scaling and shearing, are also inevitable in hand drawings. Users are typically only good at expressing visual semantics rather than exact shapes and locations. Therefore, noise and distortions have to be dealt with to better capture users' real drawing intent, which can potentially help improve the quality of retrieval. Furthermore, non-trained users not only draw poorly, but also draw slowly, especially for complicated multi-stroke objects. The ability to query with incomplete sketches can greatly improve search efficiency, and foster a progressive search style where interactive visual feedbacks can be leveraged to enhance the query precision.

† e-mail:{wwxu,jingdw,bainguo}@microsoft.com
‡ e-mail:kkyin@comp.nus.edu.sg
§ e-mail:kunzhou@acm.org

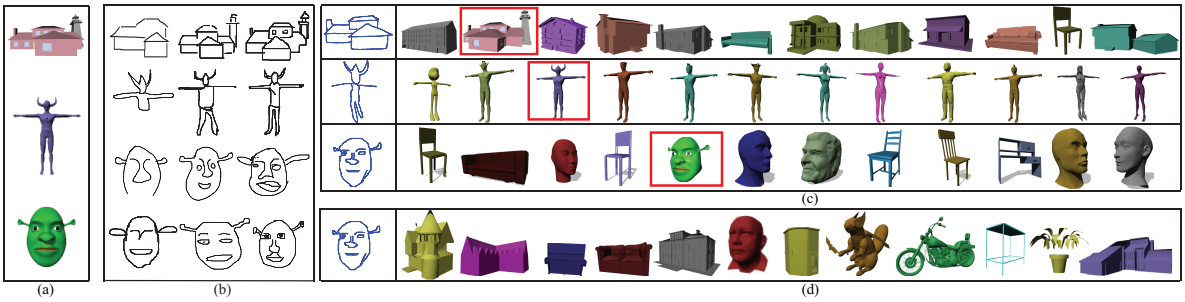

Figure 1: *User study: (a) front-view pictures of the target models are given to users as reference. (b) Some user-drawn partial and complete sketches, which all successfully retrieved the targets using our method. (c) Three example query sketches and their corresponding retrieval results using our method. (d) The retrieval results using a search algorithm similar to [FMK*03].*

In response to the above challenges, we propose

- vectorized contours as the 2D shape representation, and a shape similarity measure computed locally from these contours. Together they enable more discriminative and partial shape comparison.
- a sampling-based robust shape matching algorithm with acceleration strategies that are critical for performance .

More specifically, our shape representation and similarity measure are more discriminative than global shape descriptors and more precise than unorganized local features. They can also filter certain types of noise and outliers, and naturally support partial matching not well-defined on global shape descriptors.

Our sampling-based robust shape matching is the key to coping with noise, distortions, malformed strokes, and partial queries. Our algorithm follows the RANSAC framework [FB81] and is robust to noise when a sufficient number of sample points are tested. However, we test samples in a deterministic fashion to produce consistent query results. Moreover, our algorithm utilizes confined affine transformations, affine transformations that are close to similarity transformations, to handle small affine distortions such as non-uniform scaling and shearing. The limited affinity lowers the requirement of high-quality user drawings, improves the discriminative power of the search algorithm, as well as avoids possible confusing retrieval results from unconstrained affine-invariant matching.

Sampling-based registration incurs a high computational cost, however. We devise two critical acceleration techniques to achieve interactive performance. First, we introduce *transformation graphs*, which encode the transformations between neighboring nodes. We organize vectorized contour images into *k*NN graphs based on their similarity, and precompute transformation graphs which later enable fast online shape alignment. Second, to reduce the cost of dense sampling and registration, we prune samples using geometry invariants and align shapes in a hierarchical fashion. We also implement parallelized versions of our algorithms on multi-core platforms and GPUs that are evermore commonplace today. To the best of our knowledge, our work is the first successful application of sampling-based matching for sketch-based retrieval of 3D models.

We test the discriminative power, robustness, and performance of our system with a medium-scale database of 5,000 3D models. Query precision and user satisfaction are significantly improved compared to previous methods. On average a single query takes less than two seconds on our eight-core desktop, and less than one second on our mid-range Graphics card. Our system is adequate for in-house repositories or in-game catalogs, and provides a solid point of departure for discriminative and robust Internet-scale shape retrieval and re-ranking.

Figure 2 shows the flowchart of our system. Database preprocessing is done offline as described in Section 3. The similarity measure and sampling-based shape matching are detailed in Section 4, together with critical acceleration schemes. Section 5 demonstrates the effectiveness of our system through a variety of experiments, comparisons, and user studies. Finally, we discuss the limitations and possible future directions of this work.

2. Related Work

3D Shape Retrieval: Various global shape descriptors, such as Topology information [HSKK01], statistics of shapes [OFCD02], and distance functions [KCD*02, PSG*06], have been developed for 3D shape retrieval. We refer the interested readers to survey papers for more complete discussions [TV08]. Applications in shape segmentation and example-based modeling have stimulated algorithms that support part-in-whole shape retrieval, through a similarity measure that integrates a distance field over the entire mesh surface [FKS*04]. Recently a more ambitious goal is to design deformation-invariant shape descriptors [Rus07]. A set of local features is proposed for priority-driven partial matching [FS06]. 3D local features have also been incorporated into the bag-of-words model for 3D shape retrieval [FO09, BBGO11].

2D Shape Matching: Sketch-based interfaces are intuitive

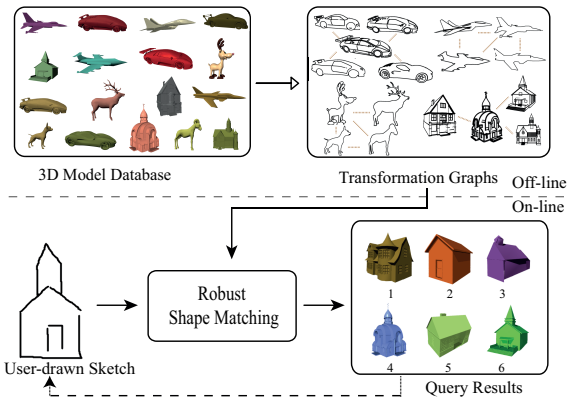

Figure 2: *The proposed sketch-based model retrieval system. In the offline stage, 3D models are parameterized into 2D contours, and then organized into transformation graphs based on their similarity. At runtime, the shape matching algorithm compares query sketches with contour images in the transformation graphs. The user can iteratively refine her sketch based on visual feedbacks from the returned models.*

to use for 3D shape retrieval and modeling [FMK*03, IMT99]. Reconstructing 3D models directly from 2D sketches is a challenging task, however. Thus a common practice is to convert 3D models into 2D representations, and then investigate the similarity between the query sketch and the planar representations using 2D shape matching methods. Many 2D shape signatures have been proposed [CNP04]. The Princeton search engine computes a Fourier descriptor of the boundary distance transform, which is rotation and translation invariant [FMK*03]. Boundary information alone cannot discriminate internal structures though. In [HR07], Fourier descriptors plus 2.5D spherical harmonic coordinates and Zernike moments are used as the classifier. Diffusion tensor, which characterizes the boundary direction information, is also applied to sketch-based 3D shape retrieval [YSSK10]. They all require, however, complete input sketches. More recently, bag-of-features is investigated for sketch-based shape retrieval [EHBA10]. It is not obvious how to handle affine distortions in the bag-of-words framework. Affine-invariant image contour matching for object recognition does not handle noise and outliers in hand-drawn strokes [MCH10].

Our 2D vectorized contours and similarity measure extend the above mentioned distance functions, with more focus on handling inferior drawings, supporting a progressive sketching style, and improving the discriminative power. Distance-based matching methods, however, generally have poor indexing efficiency. Various data structures have been proposed to improve its efficiency, including the k nearest neighbors (kNN) graph [SK02]. We organize 2D contours into kNN graphs as well, and augment the graph edges with precomputed registration between neighboring nodes.

Shape Registration: The task of shape registration is to align two shapes in a shared coordinate system. Most shape registration algorithms are based on searching point correspondences. Local descriptors, such as spin images, shape contexts, and multi-scale SIFT-like features, have been proposed to locate corresponding points for computing an initial alignment [BMP02, LG05]. Then refinement algorithms aim to improve the initial alignments, usually with an optimization framework, such as the well-known ICP algorithm for optimal rigid transforms between shapes [BM92]. Aligning details calls for non-rigid registration mechanisms, such as thin-plate splines, and part-wise or point-wise transformations [IGL03, BR07]. However, it is difficult to find correspondences robustly with local shape descriptors alone, especially for noisy input. State-of-the-art alignment algorithms [IR99, AMCO08] sample wide bases, i.e., any three distant non-collinear points, randomly from the source shape, and try to match them to all or selected bases in the target mesh. The cost of dense sampling and registration, however, is not affordable in one-to-many matching applications like ours. We adapt the wide base sampling and registration approach to robust 2D shape matching, and push its performance into the interactive regime.

3. Database Preprocessing

We construct a database containing 5,000 3D models, including animals, plants, humans, aircrafts, houses, furniture, tools and devices etc. We start from all the models in the Princeton Shape Benchmark (`http://shape.cs.princeton.edu/benchmark/`), which contains 1815 models. We then expand the database with models of the same categories from the INRIA GAMMA 3D mesh research database (`http://www-roc.inria.fr/gamma/download/`). Appendix A of the supplemental material reports the detailed constitution of our database.

3.1. 2D Shape Representation for 3D Models

We achieve 3D shape matching by comparing their 2D representations. Each model is firstly normalized and oriented [FMK*03]. We then render perspective 2D contours of each model from seven selected views: three canonical views (front, side, and top), plus four corner views from the top corners of its bounding cube. We use the word *contour* to refer to the aggregation of boundaries, silhouettes, suggestive contours [DFRS03], and salient feature lines [HPW05]. These contours represent the 2D shape of a 3D model from a particular pre-selected viewpoint, for example Figure 3(b). Contour images are directly read from the GL render buffer, with pixels on the contours labeled black.

We then vectorize the contour images by fitting line segments to the black pixels with Robust Moving Least-Squares (RMLS) [FCOS05]. RMLS is not only robust with respect to noisy contours, but also able to detect sharp features such as corner points. The original contour images are turned into

polylines as shown in Figure 3(c). From polylines we then generate a thickened contour image as illustrated in Figure 3(d). The polylines grow from one pixel to δ_w pixels on both sides, similar to a feathering effect in image processing. Each black pixel on a widened line records the line segment it associates with, its distance to the original line shaft, and the polyline it belongs to. In case one pixel lies in the feather areas of multiple line segments, we pick the closest shaft for it. White pixels are not associated with any line segments and carry no information, thus are discarded by our image compression step using perfect spatial hashing [LH06]. The full database of 5,000 3D models takes about 3GB and their contour images about 1.5GB to store.

Note that unlike methods that use distance functions or transforms, our vectorized contours only consider pixels near important and existing features. This not only alleviates the inherent large memory consumption of previous methods, but also helps to ignore noisy strokes and pixels near insignificant details that may not be present in similar models.

3.2. Transformation Graphs

When a user inputs a query sketch, our shape matching algorithm detailed in §4 will select the best matching contour images and return their corresponding 3D models. A straightforward matching algorithm would be to independently compare the query sketch with each contour image of all 3D models in the database. However, this will be computationally prohibitive even for medium-sized databases, if a robust and accurate shape matching method is used. We thus organize contour images into kNN graphs based on their similarity. These graphs further embed the registration among the database contour images, and can greatly enhance the retrieval speed (§4.3).

The nodes of a transformation graph are contour images of all the 3D models from a specific view. Therefore there are seven graphs in total. We connect each graph node with its $k = 20$ nearest neighbors. The distance between nodes is determined by their similarity score computed as will shortly be described in §4.1. To reduce the computational cost of graph construction, we locate nearest neighbors by first comparing 3D shape distributions [FMK*03] to quickly filter out dissimilar models. Only the contour images of the top-100 matching models are further ranked using our shape matching algorithm (§4.2), and the 20 nearest neighbors are selected. The affine transform between two neighboring nodes estimated from the shape matching step is then recorded in their linking edge.

The seven transformation graphs of 5,000 models currently take about four hours to compute on a Dell desktop with dual quad-core Intel Xeon E5540 processors, and about 18MB to store. When additional models are added to the database individually, we can simply query with their contours and link them to the top matches. When models are added in batches, we can either rebuild the graphs or use more advanced graph updating techniques [HY07].

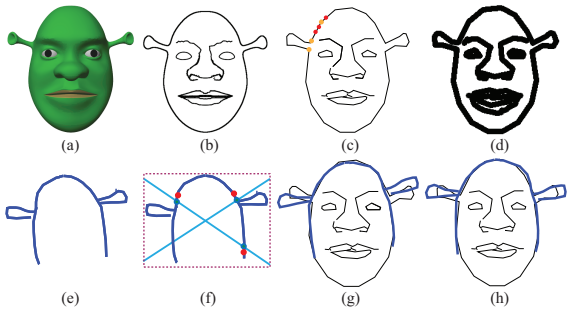

Figure 3: *2D shape representation and matching. (a) A 3D head model. (b) GL-rendered contours of its front view. (c) Vectorized contours and sampled points. Orange dots are interest points. To avoid clutter, we only show a few of the sampled points. (d) Thickened contour image. (e) One partial sketch from users. (f) One wide base (red dots). (g) Coarse alignment using only interest points. (h) Refined alignment between the sketch and the contour image.*

4. Sampling-based Robust Shape Matching

We propose a novel sampling-based 2D shape matching method to estimate the similarity between query sketches and contour images of 3D models. The same shape matching algorithm is also used to organize contour images into transformation graphs during database preprocessing. We will first describe our similarity measure designed for vectorized contours, and then detail the sampling-based robust shape registration method.

4.1. 2D Similarity Measure

Our 2D shape similarity function only considers shape elements that lie close to each other within a distance threshold δ_w. Specifically, a user-drawn sketch S is first vectorized just as what has been done to the contours read from the frame buffer. We then sample a set of points p_i from S as shown in Figure 3(c). The samples are drawn uniformly from the start to the end of each polyline. Now given a contour image T of a 3D model, the similarity score $f(S,T)$ between S and T is defined as a weighted sum of $f(p_i,T)$, the proximity of each point p_i to T, as follows:

$$f(S,T) = \frac{1}{m}\sum_{i=1}^{m} w_i f(p_i,T) \tag{1}$$

where m is the number of samples in S, and $w_i = 1$ by default for regular strokes. The point-to-shape distance is defined as

$$f(p,T) = \begin{cases} e^{\left(\frac{-d(p,l_p^T)}{\delta_d} + \frac{-|k(l_p^S)-k(l_p^T)|}{\delta_k}\right)} & d(p,l_p^T) < \delta_w \\ 0 & otherwise \end{cases} \tag{2}$$

where l_p^S denotes the line segment in S that point p belongs to, and l_p^T refers to the closest line segment to p in T. $d(p,l_p^T)$ is the distance between p and l_p^T, and $k(l_p^S)$ and $k(l_p^T)$ are the slopes of l_p^S and l_p^T. The negative exponential function maps a smaller distance to a larger value, so that shapes close to each other will have higher similarity scores.

Note that when the point-to-line distance $d(p,l_p^T)$ is greater than δ_w, the point is given a zero score and ignored in the similarity computation. This helps to deal with noisy hand drawings. When the point-to-line distance is within the threshold, we mark the polyline that p belongs to in S as a matched polyline. We then scale the sum $f(S,T)$ in Equation 1 by a concave function of the ratio between the length of matched polylines and the total length of all polylines. Thus simpler models rank higher than complex models that match equally well to features present in the sketch drawn so far. As more strokes are added and the complexity of the sketch progresses, the ordering will gradually reverse because complex models will score higher with more matching features.

4.2. Sampling-based 2D Shape Registration

The above similarity score represents shape similarity only when computed from shapes properly aligned. We thus need to search for an affine transformation that best aligns a sketch S with a target contour image T. The basic procedure is to pick a base p_0,p_1,p_2, i.e., three non-collinear points, from S, and another base p_0',p_1',p_2' from T. The 2D affine transformation $\mathscr{T}=\{\mathscr{A},t\}$ that transforms the base of S into the base of T can be computed as follows:

$$\begin{aligned}\mathscr{A} &= \left(\sum_i (p_i'-o')^t(p_i'-o')\right)\left(\sum_i (p_i-o)^t(p_i-o)\right)^{-1}\\ t &= o'-\mathscr{A}o \end{aligned} \quad (3)$$

where o and o' are the mean position of p_0,p_1,p_2 and p_0',p_1',p_2' respectively. $\mathscr{T}$ is then used to transform S to an intermediate sketch S', and the similarity score $f(S',T)$ is computed. A high score suggests that a good registration between S and T has been found, and that they are highly similar shapes. Many bases from both shapes need to be sampled and compared to achieve a robust estimation. We then record the transformation that yields the highest similarity score as the final registration.

A naive implementation of the above registration process comparing all possible bases, however, will lead to an algorithm of time complexity $O(m^3n^3)$, where m is the number of points sampled from S, and n is the number of points sampled from T. To accelerate the performance, we develop (a) an effective base pruning strategy based on the wide-base rule and similarity congruency; (b) a hierarchical alignment scheme to combine fast coarse estimation with local refinement; (c) parallel CPU and GPU implementations. These three accelerations combined significantly improve the performance as well as robustness of the naive algorithm.

4.2.1. Base Pruning

Any three non-collinear points sampled from the contours can serve as a base. However, many of them form rather small triangles and are not stable for registration [AMCO08]. Wide-bases, points that are sufficiently distant and form a relatively large triangle, however, are more robust to noise [GMO94]. We first compute the Oriented Bounding Box (OBB) of S, then draw the diagonals of the OBB. We pick the end points (blue points in Figure 3(f)) among all the intersection points on the diagonals, and locate the closest sample points (red points in Figure 3(f)). The chosen sample points form four triangles at the most, and the three points that make the largest triangle are chosen as a wide base. If we wish to sample $\bar{m}$ wide bases from S, we can rotate the OBB diagonals $\bar{m}$ times, with an angle increment of $\pi/\bar{m}$ each time.

For each chosen wide base from S, testing its registration with all bases in T still results in unacceptable performance. Instead, we only select promising bases from T for further examination. Given a base $\{a,b,c\}$ of S, we compute a similarity-invariant tuple (r,θ) as follows:

$$\begin{aligned} r &= \|b-a\|/\|c-a\| \\ \theta &= angle(\overrightarrow{ab},\overrightarrow{ac}) \end{aligned} \quad (4)$$

It is easy to verify that (r,θ) is invariant under similarity transforms. For a base $\{a,b,c\}$ of S and a base $\{a',b',c'\}$ of T, they are similarity congruent if $(r,\theta)\approx(r',\theta')$. In case only two points $\{a',b'\}$ are given, we can compute c' conveniently so that $\{a,b,c\}$ and $\{a',b',c'\}$ are similarity congruent. That is, we first rotate vector $\overrightarrow{a'b'}$ by angle θ, and then scale the rotated vector by ratio r.

So after a wide base $\{a,b,c\}$ is picked from S, we traverse all the 2-point pairs $\{a',b'\}$ in T, and compute c' as described above using (r,θ). All samples in T that are close to c' within a specified threshold δ_w are chosen as the third point of a candidate matching base. Since the contours are already thickened by δ_w, we can easily verify the existence of c' on the thickened 2D contours. This pruning strategy is in spirit similar to that of [AMCO08]. However, we use similarity congruency rather than affine congruency to limit the search space of the affine transformations. The radius of the circular window δ_w determines how much distortion our affine transformations can accommodate. Note that even though we use 3-point bases, only 2-point pairs in T are traversed. Therefore, the time complexity of traversing all possible bases in T for one sampled base in S is reduced from $O(n^3)$ to $O(kn^2)$, where k is in proportion to δ_w.

As mentioned in the introduction, limited affinity is desirable in several ways. Affine distortions are common in hand

Algorithm 1 : **HAlign**(*SAllPts,SInterestPts,TAllPts,TInterestPts*)
Input: all sampled points on S; interest points on S; all sampled points on T; interest points on T
Output: the best affine transformation to align S and T

```
ss ← 0 {initialize the similarity score}
𝒯 ← null {initialize the affine transformation}
for i = 1 to m̄ do
    wb ← FindWideBase(SInterestPts)
    (𝒯′, ss′) ← AlignBases(TInterestPts, wb)
    if (ss′ > ss) then
        𝒯 ← 𝒯′ {update the best transformation}
        ss ← ss′ {update the best score}
    end if
end for
for i = 1 to m̄ do
    wb ← FindWideBase(SAllPts)
    (𝒯′, ss′) ← RefineAlignment(TAllPts, wb, 𝒯)
    if (ss′ > ss) then
        𝒯 ← 𝒯′
        ss ← ss′
    end if
end for
return (ss, 𝒯)
```

drawings so we have to deal with them in shape matching. But unconstrained affine matching can cause confusing retrieval results. For example, a square can be perfectly matched to an elongate rectangle via an unconfined affine transformation. However, the users who draw a square, most likely a distorted square rather than a perfect square, might be confused to see skinny rectangles displayed in the query results. We will further illustrate this point in one of the user studies (§5.1). Our above base pruning approach originates from similarity congruency, and thus naturally supports limited affinity.

4.2.2. Hierarchical Alignment

The cost $O(kn^2)$ can still be prohibitive when n is large for a complex shape T. We therefore develop a two-tier alignment scheme as shown in Algorithm 1. A coarse registration is first estimated quickly from only important feature points in T. A common practice in shape analysis is to choose points of high curvature as important feature points. We have detected corner points between line segments when we vectorize contours. In addition, we consider intersection points of line segments as well. We use *interest points* to refer to all the corner and intersection points. The number of interest points $\tilde{n}$ is on average about one fourth of n in our experiments. The first for-loop in Algorithm 1 corresponds to this fast estimation of $\mathscr{T}$ with interest points only, and Figure 3(g) shows an initial alignment estimated from this step.

Then a refinement process further improves the coarse registration as shown in the second for-loop of Algorithm 1. Given a base $\{a,b,c\}$ from S, we first transform $\{a,b,c\}$ into $\{a',b',c'\}$ using $\mathscr{T}$, then use samples of T within a circular window of radius δ_r centered at a',b',c' for possible better alignment. This tuning step typically takes near constant time, and Figure 3(h) illustrates the refinement effect.

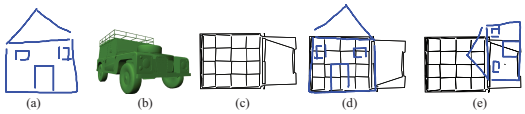

Figure 4: *Transformation Graphs can help eliminate false positives. (a) Sketch of a house. (b) A vehicle model. (c) Contour image of (b)'s top view. (c) The best alignment between (a) and (c) by direct comparison has a high similarity score 0.4263. (d) The best alignment through transformation graphs scores 0.1878.*

Par.	Description	Value
δ_w	distance threshold in Sec. 3.1	8 pixels
δ_d	variance of distance in Eq. 2	3.0
δ_k	variance of slope in Eq. 2	0.3
δ_r	refinement threshold in Sec. 4.2.2	5 pixels
$\bar{m}$	number of tested bases in Alg. 1	12
n_s	number of seeds in Sec. 4.3	132

Table 1: *Parameters used for all experiments.*

4.2.3. Parallel Implementations

The proposed sampling-based matching algorithm is easy to parallelize: the comparisons of different contours are independent, and the registration of different bases from the same pair of shapes is parallelizable. Today multi-core machines are common, and we can easily achieve an eight-fold speedup on an eight-core desktop. Graphics cards are also readily available. We have achieved a significant speedup using a CUDA implementation of the proposed algorithm on our single-core NVIDIA GeForce GTX 285. Detailed performance statistics are shown in Table 2 and will be explained in the Results section.

4.3. Transformation Graph Assisted Retrieval

A straightforward implementation of the retrieval system aligns a query sketch to each stored contour image independently on the fly, and this is simply too slow for our application. We can actually achieve much faster registration between the input and contour images, with the help of precomputed transformation graphs. In the preprocessing stage, in addition to constructing graphs as described in §3.2, we also cluster the graph nodes by a simple region growing method. An initial image is randomly selected from a graph as the seed, and we traverse the whole graph following the best-first order. When the similarity between the seed and the current node is lower than a chosen threshold, we start a new cluster with the current image as a new seed.

At runtime, the hierarchical alignment scheme is first applied to register the query sketch with all the n_s seeds, and the results are sorted and pushed into a heap. We then traverse

Model: #Points	Chair: 170		Car: 215		House: 187		Human: 111	
perf. / alg. comps	time	AP	time	AP	time	AP	time	AP
Baseline	2061.16	0.610	1681.24	0.490	2303.32	0.261	2106.56	0.295
Hierarchical Alignment	236.48	0.574	177.83	0.402	401.57	0.246	320.03	0.299
Transformation Graph	10.39	0.716	5.94	0.670	12.93	0.414	9.68	0.373
CPU Parallelism	1.35	0.716	0.80	0.670	1.67	0.414	1.21	0.373
GPU Acceleration	0.80	0.716	0.47	0.670	0.93	0.414	0.66	0.373

Table 2: *Performance: timing in seconds, and quality measured by average precision (AP).*

the graphs in the best-first order. The current best matching image is popped up, and its neighbors are examined with respect to the query sketch. To obtain the affine transform between the query and a neighbor of the current node, however, we only need to concatenate the transformation between the query to the current node, which is already known, with the transformation between the current node and its neighbor, which is precomputed and stored on the graph edge. A full-blown registration is not necessary anymore. Nevertheless, to limit error accumulation, we treat the composite transformation only as an initial guess, and perform a refinement procedure same as described in Algorithm 1. Table 2 reports the significant performance gain achieved by the transformation graph assisted retrieval scheme.

A pleasant surprise is that the retrieval quality based on transformation graphs is also improved, as shown by the average precision in Table 2. This is counter intuitive on the first thought. Concatenating transformations accumulates errors, and should have negative effects to the registration quality, if having any effect at all. A reasonable explanation is that the graphs built on our similarity measure approximate the local manifold inside the geometry space of all database models. Thus, navigation through the graphs follows paths on the manifold from the query to the best matching shapes. This helps to filter out false positives and improves precision. Our finding is consistent with the results reported in manifold ranking methods [ZWG*04]. As illustrated in Figure 4, the direct registration between two unrelated models might occasionally succeed numerically with poor semantic quality.

5. Results

Parameters: We test our shape retrieval system via various query examples with the same set of parameters shown in Table 1. Although these parameters are manually tuned, we found the landscape of performance vs. parameters rather smooth. The query results are not super sensitive to any of the parameters. Alternatively an offline procedure that explores the parameter space in a systematic fashion may further improve the system performance.

Performance: Figure 5 plots the average precision-recall curves tested on the full database. The left diagram illustrates

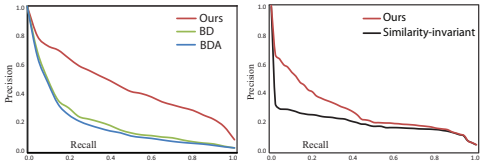

Figure 5: *Average precision recall curves: (left) query results with 36 contour images of representative database models, illustrating the better discriminative power of our method; (right) query results with 50 hand-drawn sketches from the user study, showing the necessity of affine transformation in shape matching.*

the better discriminative power of our method, using noise-free contour images of 36 representative models of various categories as the query inputs. The selected models are attached in Appendix B of the supplemental material. The BD method refers to the boundary-descriptor based method of [FMK*03]. The BDA method refers to our augmented version of the BD method, where not only boundaries but also interior contours are used in computing the shape descriptor. The results in Figure 1(d) is generated with BDA. Note that the statistics of BD and BDA does not differ significantly, but for cases where interior contours are important, such as the Shrek head, BDA can generate slightly better results sometimes. The right diagram of Figure 5 shows the necessity of incorporating affine transformation in dealing with distortions in hand-drawn sketches. The precision of our shape matching algorithm is better than its pure similarity-transformation based counterpart. The 50 query sketches are obtained from the second user study detailed in Section 5.1.

Table 2 reports the detailed performance statistics with four of the models used in the above precision-recall experiment. Timing is measured on a desktop PC of 8GB RAM with dual quad-core Intel Xeon E5540 processors, and a NVIDIA GeForce GTX 285 graphics card. The baseline algorithm registers the sketch with contour images using our method described up to §4.2.1. The completely naive algorithm without base pruning is just too slow and erratic, because of the existence of many narrow bases. We then add different algorithmic components in turn to see their effect on the system performance. All components achieve significant speedups, and the most effective one is the transformation graphs. The quality of retrieval is measured by av-

Figure 6: *Example of negative (blue) and positive (red) strokes. Top: Initial search results. Middle: Adding negative strokes to suppress the unwanted back support style. Bottom: Adding positive strokes to emphasize the armrests.*

erage precision (AP): for each query image we compute its precision-recall curve, from which we obtain its average precision.

Sketch-based retrieval: Figure 6 illustrates a progressive search within 136 chair models, and the use of negative and positive strokes. On occasions where users want to exclude a certain feature, we supply negative strokes, with stroke weight $w_i = -1$ in Equation 1, to suppress the undesired parts. This is similar to the NOT Boolean operator in keyword-based text search. Positive strokes have weight $w_i = 2$. The blue negative strokes in the middle row aim to remove the unwanted style of back support in the top row, and the red positive strokes in the bottom row emphasize the armrests.

Partial shape matching is one of the key features supported by our 2D shape representation and matching scheme. Given complex models as shown in Figure 1(a), it is difficult for a non-trained user to complete her sketch with one stroke. Using our system, the target models are returned within the top matches when queried with the partial sketches shown in the leftmost column of Figure 1(b). The ability to search from partial sketches provides fast feedback and improves query efficiency, for general users as well as for artists.

Example-based modeling: Today shape retrieval is commonly used to support example-based modeling [SI07, LF08]. We integrate one of the state-of-the-art 3D deformation techniques [KSvdP09] into our query system to assist users in creating new models from retrieved examples. The user chooses the most similar 3D model to her sketch, and the deformation engine automatically deforms the model to match its contours to the sketch. In this setting, we can circumvent difficulties like occlusions and topology ambiguities typical for a modeling system that directly goes from 2D sketches to 3D models. Figure 8 illustrates such an example.

5.1. User Studies

The first user study we performed is to validate the limited affinity design. We prepared four pairs of 2D shapes with different degrees of affine distortions. We recruited sixty subjects and divided them into four groups. Each group was asked to judge the similarity of only one pair of shapes to avoid possible interference. Figure 7 shows the shape pairs and the users' choices. Users do tolerate affine distortions, but only to a certain degree. Although it is still difficult to quantify how perceived similarity degrades with respect to increased distortions, this user study clearly implies that shapes with more distortions should be ranked lower, and shapes with severe distortions should be filtered out. Our limited affinity shape matching captures these features naturally.

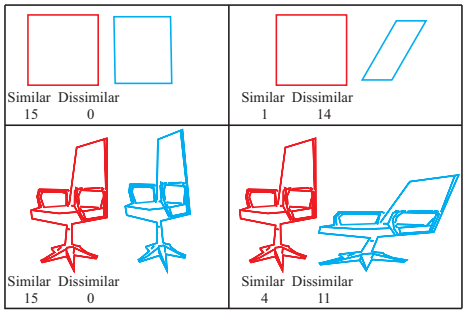

Figure 7: *The influence of affine distortion to perceived similarity. For each shape pair, the original shape (red) is on the left, and the distorted shape (cyan) is on the right. The number under the similar and dissimilar labels reports the number of subjects who agree with the specific choice. This study shows that users do tolerate affine distortions, but only to a certain degree.*

Our second user study asked users to retrieve target models within the top 24 matches, i.e., the first two pages of the retrieval results, as fast as possible and within five minutes. Ten graduate students with a science and engineering background were recruited, and five of them are females. For each test query, we give users a color print-out of the front view of the target model as reference. The tests were conducted on the same desktop for performance measurement, with a 24" widescreen monitor. During a pilot study we found that most novice users draw worse with our Wacom sketch pad than with a mouse, so we simply chose mouse for the user study. We trained each subject for five minutes: a demonstration of our system for about two minutes, then a user practice session about three minutes when they could get our help if needed. The five target models are: bicycle, sailboat, house, Shrek head, and ultraman. These models are all used in calculating the precision/recall curves in Figure 5. All subjects were able to retrieve the target models within five minutes with our system. Figure 1(b) shows some of the user-drawn sketches; and Appendix C of the supplemental material contains all fifty sketches. Table 3 reports the time spent by each subject and the number of search attempts until they succeeded.

We also performed post-study interviews with each subject

	A	B	C	D	E	F	G	H	I	J
Ultraman	38/1	64/1	49/1	57/1	43/1	86/1	53/1	31/1	48/1	58/1
Bicycle	24/1	66/1	58/1	37/1	73/1	69/2	67/2	39/2	87/2	30/1
House	55/1	83/1	65/1	109/2	63/1	225/3	65/2	64/1	96/1	217/4
Sailboat	97/3	42/1	87/3	105/2	69/1	174/4	41/2	65/3	114/3	39/1
Shrek	88/2	207/3	247/5	57/1	101/3	287/5	202/3	87/2	111/2	38/1

Table 3: *User study statistics. Number pairs indicate total query time in seconds and number of search attempts. The number of search attempts equals the number of times a user hits the search button on the GUI.*

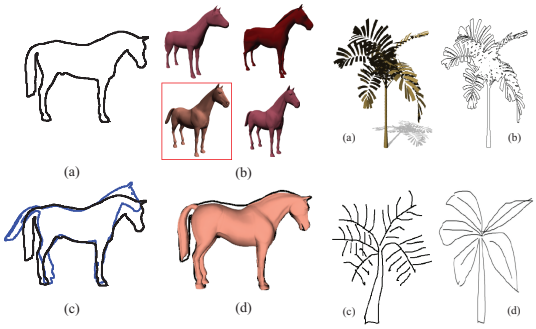

Figure 8: *Example-based modeling. (a) A query sketch extracted from a photo. (b) Retrieval results. (c) The input and the contour of the boxed model overlapped. (d) The chosen model deformed to match the query.*

Figure 9: *Failure case. (a) A plant model. (b) The contour image of the plant. (c)(d) Two user-drawn query sketches. Neither can retrieve the plant.*

soliciting their suggestions and feedbacks. All of them felt the retrieval ability of our system is satisfactory; and the response time is acceptable or fast enough. One complained that drawing curves with a mouse was difficult. Two subjects said they could not draw the relative scale and position of geometric parts very well, and they hoped the system could somehow handle it. We have also tried to carry out the same study with the BD method. However, we stopped after testing with five users only. The success rate of BD was rather low about 10%. The subjects often gave up the task, sometimes even before they ran out of time, after seeing that little progress could be made by revising their sketches.

6. Discussion and Future Work

Recent advances in 3D scanning technologies and modeling tools have dramatically increased the complexity of geometric models available on the web. These models pose new challenges for sketch-based shape retrieval systems. Input sketches tend to contain more strokes, noise, and distortions. We propose the use of vectorized contours for 2D shape representation. We have also developed a robust sampling-based shape matching algorithm. Retrieval results from our methods are significantly better than those of previous systems. Through various critical acceleration schemes, most importantly the transformation graphs and registration pruning, we successfully pushed our system performance into the interactive regime. With the fast progresses in parallel hardware platforms today, be it CPU or GPU based, we expect the performance even faster in the near future.

There are several limitations of this work that are relatively easy to address. We only use contour images from seven viewpoints to compare with the query sketch. Incorporating more views is straightforward, although the query time and storage do increase linearly with the number of preselected views. The cost of transformation concatenation and similarity computation is currently neglectable, so we traverse the full graphs. With a properly designed stop criteria, we may also terminate the graph traversal earlier and thus reduce the number of nodes visited. This should be beneficial to the application of our search algorithm to large scale databases.

We see many exciting but more difficult avenues for future improvements. Currently most of our failure cases happen when there is a mismatch between the contours automatically extracted from the rendered models, and the user's mental image of an object. Figure 9 shows such an example. The contours of a tree in Figure 9(b) are literally the contours of individual leaves, yet users usually just draw stems, such as Figure 9(c), or virtual contours of each branch, such as Figure 9(d). This suggests different fusions of multiple features and representations, for different categories of objects. After all, the way we draw a table is quite different from the way we draw a tree.

Shape matching based on global transformations may miss users' real intent. Taking the face matching in Figure 1(c) as an example again, although most people are able to sketch down important facial features and organs, non-artistic users seldom draw proportions and locations accurately. We envision a part-wise post-refinement procedure that further partitions the whole sketch into independent parts, e.g., connected contours, and then refines the alignment of each part. Ideally this part-wise refinement procedure should be exposed to users as an option that can be turned on and off as needed. Along this line, feature-aware or part-aware shape matching may also be attainable within our framework.

References

[AMCO08] AIGER D., MITRA N. J., COHEN-OR D.: 4-points congruent sets for robust pairwise surface registration. *ACM Trans. Graph. 27*, 3 (2008), Article 85. 3, 5

[BBGO11] BRONSTEIN A. M., BRONSTEIN M. M., GUIBAS L. J., OVSJANIKOV M.: Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Trans. Graph. 30* (2011), Article 1. 2

[BM92] BESL P. J., MCKAY N. D.: A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2 (1992), 239–256. 3

[BMP02] BELONGIE S., MALIK J., PUZICHA J.: Shape matching and object recognition using shape contexts. *PAMI 24* (2002), 509–522. 3

[BR07] BROWN B. J., RUSINKIEWICZ S.: Global non-rigid alignment of 3-d scans. *ACM Trans. Graph. 26*, 3 (2007), Article 21. 3

[CCT*09] CHEN T., CHENG M.-M., TAN P., SHAMIR A., HU S.-M.: Sketch2photo: internet image montage. *ACM Trans. Graph. 28* (2009), Article 124. 1

[CNP04] CHALECHALE A., NAGHDY G., PREMARATNE P.: Sketch-based shape retrieval using length and curvature of 2d digital contours. In *IWCIA'04* (2004), pp. 474–487. 3

[DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive contours for conveying shape. *ACM Trans. Graph. 22*, 3 (2003), 848–855. 3

[EHBA10] EITZ M., HILDEBRAND K., BOUBEKEUR T., ALEXA M.: Sketch-based 3d shape retrieval. In *SIGGRAPH'10 Talks* (2010), p. Article 5. 3

[ETA01] ELAD M., TAL A., AR S.: Content based retrieval of vrml objects: an iterative and interactive approach. In *the sixth Eurographics workshop on Multimedia 2001* (2001), pp. 107–118. 1

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6 (1981), 381–395. 2

[FCOS05] FLEISHMAN S., COHEN-OR D., SILVA C. T.: Robust moving least-squares fitting with sharp features. *ACM Trans. Graph. 24*, 3 (2005), 544–552. 3

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. *ACM Trans. Graph. 23*, 3 (2004), 652–663. 2

[FMK*03] FUNKHOUSER T., MIN P., KAZHDAN M., CHEN J., HALDERMAN A., DOBKIN D., JACOBS D.: A search engine for 3d models. *ACM Trans. Graph. 22*, 1 (2003), 83–105. 1, 2, 3, 4, 7

[FO09] FURUYA T., OHBUCHI R.: Dense sampling and fast encoding for 3d model retrieval using bag-of-visual features. In *CIVR'09* (2009), p. Article 26. 2

[FS06] FUNKHOUSER T., SHILANE P.: Partial matching of 3d shapes with priority-driven search. In *SGP'06* (2006), pp. 131–142. 2

[GMO94] GOODRICH M. T., MITCHELL J. S. B., ORLETSKY M. W.: Practical methods for approximate geometric pattern matching under rigid motions. In *SCG'94* (1994), pp. 103–112. 5

[HPW05] HILDEBRANDT K., POLTHIER K., WARDETZKY M.: Smooth feature lines on surface meshes. In *SGP'05* (2005), p. Article 85. 3

[HR07] HOU S., RAMANI K.: Calligraphic interfaces: Classifier combination for sketch-based 3d part retrieval. *Computers and Graphics 31*, 4 (2007), 598–609. 3

[HSKK01] HILAGA M., SHINAGAWA Y., KOHMURA T., KUNII T. L.: Topology matching for fully automatic similarity estimation of 3d shapes. In *SIGGRAPH'01* (2001), pp. 203–212. 2

[HY07] HACID H., YOSHIDA T.: Incremental neighborhood graphs construction for multidimensional databases indexing. In *Canadian Conference on AI* (2007), pp. 405–416. 4

[IGL03] IKEMOTO L., GELFAND N., LEVOY M.: A hierarchical method for aligning warped meshes. In *3DIM'03* (2003), pp. 434–441. 3

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In *SIGGRAPH'99* (1999), pp. 409–416. 3

[IR99] IRANI S., RAGHAVAN P.: Combinatorial and experimental results for randomized point matching algorithms. *Computational Geometry 12*, 1-2 (1999), 17–31. 3

[KCD*02] KAZHDAN M., CHAZELLE B., DOBKIN D., FINKELSTEIN A., FUNKHOUSER T.: A reflective symmetry descriptor. In *ECCV'02* (2002), pp. 642–656. 2

[KSvdP09] KRAEVOY V., SHEFFER A., VAN DE PANNE M.: Modeling from contour drawings. In *SBIM'09* (2009), pp. 37–44. 8

[LF08] LEE J., FUNKHOUSER T.: Sketch-based search and composition of 3d models. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling* (2008), pp. 20–30. 1, 8

[LG05] LI X., GUSKOV I.: Multi-scale features for approximate alignment of point-based surfaces. In *SGP'05* (2005), p. 217. 3

[LH06] LEFEBVRE S., HOPPE H.: Perfect spatial hashing. *ACM Trans. Graph. 25*, 3 (2006), 579–588. 4

[Lof00] LOFFLER J.: Content-based retrieval of 3d models in distributed web databases by visual shape information. In *ICIV'00* (2000), pp. 82–87. 1

[MCH10] MAI F., CHANG C., HUNG Y.: Affine-invariant shape matching and recognition under partial occlusion. In *ICIP* (2010), pp. 4605–4608. 3

[OFCD02] OSADA R., FUNKHOUSER T., CHAZELLE B., DOBKIN D.: Shape distributions. *ACM Trans. Graph. 21*, 4 (2002), 807–832. 1, 2

[PSG*06] PODOLAK J., SHILANE P., GOLOVINSKIY A., RUSINKIEWICZ S., FUNKHOUSER T.: A planar-reflective symmetry transform for 3d shapes. *ACM Trans. Graph. 25*, 3 (2006), 549–559. 2

[Rus07] RUSTAMOV R. M.: Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *SGP'07* (2007), pp. 225–233. 2

[SI07] SHIN H., IGARASHI T.: Magic canvas: interactive design of a 3-d scene prototype from freehand sketches. In *GI'07* (2007), pp. 63–70. 1, 8

[SK02] SEBASTIAN T. B., KIMIA B. B.: Metric-based shape retrieval in large databases. In *ICPR'02* (2002), pp. 291–296. 3

[TV08] TANGELDER J. W., VELTKAMP R. C.: A survey of content based 3d shape retrieval methods. *Multimedia Tools Appl. 39* (2008), 441–471. 2

[YSSK10] YOON S. M., SCHERER M., SCHRECK T., KUIJPER A.: Sketch-based 3d model retrieval using diffusion tensor fields of suggestive contours. In *MM'10* (2010), pp. 193–200. 3

[ZWG*04] ZHOU D., WESTON J., GRETTON A., BOUSQUET O., SCHÖLKOPF B.: Ranking on data manifolds. In *Advances in Neural Information Processing Systems 16* (2004). 7