Model-driven Sketch Reconstruction with Structure-oriented Retrieval

Lei Li\textsuperscript{1} Zhe Huang\textsuperscript{2} Changqing Zou\textsuperscript{3,4,*} Chiew-Lan Tai\textsuperscript{1} Rynson W.H. Lau\textsuperscript{2} Hao Zhang\textsuperscript{3} Ping Tan\textsuperscript{3} Hongbo Fu\textsuperscript{2}

\textsuperscript{1}Hong Kong UST \textsuperscript{2}City University of Hong Kong \textsuperscript{3}Simon Fraser University \textsuperscript{4}Hengyang Normal University

Abstract

We propose an interactive system that aims at lifting a 2D sketch into a 3D sketch with the help of existing models in shape collections. The key idea is to exploit part structure for shape retrieval and sketch reconstruction. We adopt sketch-based shape retrieval and develop a novel matching algorithm which considers structure in addition to traditional shape features. From a list of retrieved models, users select one to serve as a 3D proxy, providing abstract 3D information. Then our reconstruction method transforms the sketch into 3D geometry by back-projection, followed by an optimization procedure based on the Laplacian mesh deformation framework. Preliminary evaluations show that our retrieval algorithm is more effective than a state-of-the-art method and users can create interesting 3D forms of sketches without precise drawing skills.

Keywords: sketch interpretation, sketch-based shape retrieval, structure matching, 3D sketch, mesh deformation, user interface

Concepts: ●Computing methodologies → Shape modeling;

1 Introduction

Sketching is a vital and frequently adopted form of artistic expression. For many people, free-hand sketching is arguably the most natural and accessible means to visually depict a 3D object or concept. To more fully exploit sketching as a visual form, it may be even more interesting if a sketch can serve as not only a static and final depiction, but also a 3D entity to be interacted with and viewed from different angles. There exist two main categories of approaches to exploring such an idea: interactive 3D sketching [Bae et al. 2008] and reconstruction from completed sketches [Zou et al. 2015; Zheng et al. 2016]. Current systems in the latter category, which our work belongs to, typically take accurate engineering drawings [Zou et al. 2015] or predefined 3D plane knowledge [Zheng et al. 2016] as input. In contrast, our approach is model-driven and can manage even rough sketches from users.

We propose a structure-oriented system that elevates 2D sketches into 3D immediately after their creation, taking advantage of 3D models from large shape collections available online. Our key insight is that if we understand the 3D characteristics of the sketched object, we may find a suitable model to serve as a 3D proxy. An essential part of that understanding is the part structure of an object. More specifically, a user sketches an object part by part using our interface, which is supported by a preprocessed repository of 3D man-made models of the same category and endowed with structural information. To lift the user’s completed sketch into 3D, our system consists of two stages (Fig. 1): shape retrieval and sketch reconstruction.

In the retrieval stage, a 3D model resembling the sketch is retrieved to serve as a 3D proxy, contributing 3D information for reconstructing what the user has sketched. Our shape retrieval approach jointly considers structure and shape matching to improve performance within a specified object category, yielding part-level correspondences between the sketch and the model as a retrieval by-product. In the reconstruction stage, we further leverage the correspondences descending from the retrieval to extract suitable planes from model parts for back-projecting the sketch into 3D. A straightforward back-projection approach, however, usually produces disconnected 3D sketch parts due to imprecise drawing and foreshortening perception bias (Fig. 1-(f)). We formulate these issues as a quadratic energy minimization problem to derive a feasible reconstruction result.

To sum up, our contributions are threefold: (i) applying shape collections to the analysis and reconstruction of rough sketches; (ii) developing a shape retrieval algorithm that combines part structure and conventional shape features; (iii) casting 3D planar sketch reconstruction as a quadratic energy optimization procedure based on the Laplacian mesh deformation framework.

2 Database Preprocessing

We collected different categories of man-made model collections from online public datasets [Shilane et al. 2004; Shen et al. 2012]. All models in each category were pre-segmented into meaningful parts and aligned with consistent upright and front-facing orienta-
3 Shape Retrieval

In this section, we describe how an input sketch is matched to the projected contours of 3D models with the same object category in our database for shape retrieval. The matching process takes both the shape and the structural similarity into account, and produces a ranked list of the database models.

The user sketches on a canvas under the drawing mode in our system, and s/he needs to confirm when finishing a semantic part, e.g., the back or the gas lift of a chair. We assume that the 3D counterpart of each sketch part envisioned by the user is hosted on a plane in 3D space. When the drawing is completed, the user switches to the annotation mode to mark connections among strokes of sketch parts (Fig. 1–(b)).

We symbolically represent the input sketch as a graph \( G = (S, V, L) \), where \( S = \{ s^i \} \) is the set of strokes, \( V = \{ v^i \} \) is the set of parts, part \( v^i = \{ s^k \} \) is defined as the set of its constituent strokes, and \( L = \{ (v^i, v^j) \} \) is a set of part pairs that have connections annotated by the user. The 2D projection of a 3D model is represented similarly, except that \( S \) consists of linked pixels, resulting from rendering the contours of each part separately, and \( L \) is directly derived from the part connections in the model. In the remaining sections, we refer the input sketch as \( G = (S_S, V_S, L_S) \), and the projection of a 3D model as \( M = (S_m, V_m, L_m) \). We now describe how to perform structure matching (Sec. 3.1) and shape matching (Sec. 3.2) of \( G \) and \( M \) in detail.

3.1 Structure Matching

The goal of structure matching is to evaluate whether the projection of a database model is structurally similar to the input sketch. To this end, we find correspondences between the parts in \( V_S \) and \( V_m \), and assess their plausibility. Since one-to-many and many-to-one mappings among the parts may exist, common graph matching methods [Livi and Rizzi 2013] do not suffice. We develop our own structure matching approach by first introducing a pairwise cost of part mapping and then evaluating the structural difference.

Pairwise cost. We first compute a pairwise cost of mapping \( v^i_S \) to \( v^i_m \), and then find the correspondences between \( V_S \) and \( V_m \) such that the total cost is approximately minimized (see Fig. 2). We observe that (i) if we superimpose \( V_S \) on \( V_m \) (or vice versa), the corresponding parts should be close to each other; (ii) the shapes of the corresponding parts should be similar. In structure matching, we would like to focus on comparing the positions and sizes of the semantic parts. So, we replace each part by a best-fit ellipse in the pairwise cost computation. Therefore, we define the pairwise cost of matching two parts as the weighted sum of the distance and shape difference of their best-fit ellipses.

Correspondence estimation. With the pairwise cost, the correspondence problem can be formulated as an optimal linear assignment problem. However, some parts may have no correspondence if the sketch and the model projection have different numbers of parts. Hence, we devise an iterative assignment approach, which allows many-to-one and one-to-many correspondences, to make sure that all parts have correspondences. Suppose that the sketch has fewer parts than the model (i.e., \( |V_S| < |V_m| \)). We first map \( V_S \) to a subset \( V'_m \subset V_m \) that minimizes the cost and \( |V'_m| = |V_m'| \), using optimal linear assignment. We then remove \( V'_m \) from \( V_m \), keeping \( V_m \) unchanged, and repeat the assignment step again. This process stops when \( V_m \) becomes empty. The case of \( |V_S| > |V_m| \) is handled similarly.

After the computation of part-level correspondences between the input sketch and the model projection, we evaluate their structure similarity using the correspondences, during which two factors are taken into account: relative positions of corresponding parts and part merging.

1) Relative position. If \( V_S \) and \( V_m \) are structurally similar, their parts should possess similar spatial arrangement, that is, corresponding parts should have similar relative positions. Precisely, if \( v^i_S \) and \( v^j_S \) are connected \( ((v^i_S, v^j_S) \in L_S) \), and \( v^a_m \) and \( v^b_m \) are their corresponding parts in the model, then the relative position cost is simply defined as \( E_r = \sum_{i,j} (p(v^i_m) - p(v^j_m)) - (p(v^i_m) - p(v^j_m)) \), where \( p(\cdot) \) is the position of a part.

2) Part merging. When the sketch and the model projection have different numbers of parts, one element in \( V_S \) might correspond to multiple elements in \( V_m \), or vice versa. This might be due to the difference in segmentation granularity. For example, some users might treat the base and the attached wheels of an office chair as multiple elements in \( V_S \). Therefore, we allow the merging of parts \( V'_m \subset V_m \) (Note that \( |V'_m| > = 2 \); otherwise no merging cost is applied.) If \( v_m^i \) is similar to elements in \( V'_m \) combined, the part merging is possibly due to different segmentation granularities, which shall incur zero merging cost. Otherwise, the part merging should reflect incorrect correspondences with a cost. The cost increases with the pairwise distances between the merged parts since combining distant parts together will lead to large changes in the structure. Specifically, the cost is defined as \( E_m = \sum_{n=2}^{N} \sum_{i,j} (|p(v^i_m) - p(v^j_m)|)^2 \), where \( MST(V'_m) \) is the minimum spanning tree connecting the centers of all the parts in \( V'_m \). \( MST \) is used here so that the cost increases linearly with the number of merged parts. The cost can be defined similarly for the case of mapping a subset of parts of the sketch to a single part of the model projection. \( E_m \) is a binary indicator function, which is computed by measuring the dissimilarity of \( v^i_m \) and \( V_m \), to indicate whether the merging is implausible or not.

Given the definitions above, the part merging cost is \( E_m = \sum_{v'^i_m \rightarrow v_m^i} E_0(v^i_m, V_m) + \sum_{v_m^i \rightarrow v'^i_m} E_0(v^i_m, V_m) \), where \( v^i_m \) stands for the mapping from a sketch part to a subset of model parts, similarly for \( v_m^i \rightarrow v'^i_m \). Finally, the structure difference is measured as \( E_r + E_m \).
3.2 Shape Matching

Shape matching aims at comparing pure geometry details of the query sketch to the projections of the database models, i.e., $S_g$ and $S_m$. We define shape similarity using the Gabor feature, inspired by [Eitz et al. 2012]. The sketch or the database model projection is first rasterized into a fixed-size square image, whose Gabor feature is then extracted in the same way as [Eitz et al. 2012]. The extracted feature vectors are normalized into unit length, and compared using cosine distance $E_f = 1 - h(S_g) \cdot h(S_m)$, where $h(S_g)$ and $h(S_m)$ are the normalized features of $S_g$ and $S_m$.

Complete matching cost. Based on the previous analysis, the complete cost of matching a sketch $S$ and a database model projection $M$ is $E_c = w_1 \log(E_r + E_m) + w_2 E_f$. We apply the logarithm to the structural cost to make its magnitude comparable to the shape matching cost $E_f$. We set $w_1 = 0.2$ and $w_2 = 0.8$. During sketch-based retrieval, this cost is computed between the input sketch and every projection of each model in the database, and the matched models are finally returned as a ranked list (see Fig. 3).

4 Sketch Reconstruction

In this section, we describe our semi-automatic reconstruction method that elevates a user’s 2D sketch $G$ into 3D, with the help of 3D information from a model. Let $M$ denote the 3D mesh of a selected model from the ranked list. Given part-level correspondences between $G$ and $M$ originating from the retrieval algorithm, the main idea is to firstly back-project $G$ onto the 3D planes distilled from $M$ (Sec. 4.1) before an optimization procedure that deforms the back-projection result subject to constraints of user specification and geometric derivation (Sec. 4.2). The user may need to fine-tune the view of $M$ to maximize the resemblance between $M$ and $G$ due to the limited number of rendered views used in the retrieval stage, and interactively fix any wrong correspondences. We formulate the optimization as a quadratic energy minimization problem that can be efficiently solved by sparse linear system solvers.

4.1 3D Plane Generation

Generating plane proxies of meshes in accordance with human perception is not a trivial problem on its own. To reduce the overall complexity, we adopt a greedy scheme to find candidate planes for each sketch part on its corresponding model part. Let $C = \{ (v^i_g, v^i_m) \}$ denote the part correspondence set between $G$ and $M$, where $v^i_m \subset M$ is a subset of model part meshes corresponding to $v^i_g$. We first perform PCA on $v^i_m$ to form a candidate plane set $P^i = \{ p^i_k \}$, and then use a greedy rule to select an appropriate plane $\overrightarrow{p^i}$ from $P^i$.

The plane set $P^i$ is generated as follows. Let $\{ e^i_1, e^i_2, e^i_3 \}$ denote the principal axes of $v^i_m$ sorted by their eigenvalues in a decreasing order after PCA and $v_{nm}$ be the view direction to $M$. A common plane suggested by PCA is the plane spanned by $e^i_1$ and $e^i_2$, with the normal direction same as $e^i_3$, which is added to $P^i$ in the first place. There are also two cases to be considered: (i) if $e^i_1$ dominates, i.e., $\overrightarrow{v^i_m}$ is similar to a cylinder, we add an extra view dependent plane with a normal $e^i_1 \times (e^i_1 \times v_{nm})$ to $P^i$; (ii) if there are no significant dominant axes, three planes, each with normal direction $e^i_1$, $e^i_2$, or $e^i_3$, are added to $P^i$. We also consider reflective symmetry if it exists in $M$. For case (i), if $\overrightarrow{v^i_m}$ and $\overrightarrow{v_{nm}}$ have reflective symmetry, planes that pass through them and are of reflective symmetry will constitute $P^i$ and $P^i$ instead. For case (ii), whether $\overrightarrow{v_{nm}}$ shows reflective symmetry or not, $P^i$ consists of planes that are parallel or perpendicular to the symmetry plane.

To select an appropriate $\overrightarrow{p^i}$ if $|p^i| > 1$, the greedy rule that we use is to choose a plane from $P^i$ that has the minimum angular difference between its normal and $v_{nm}$, since a large angular difference increases the foreshortening perception bias and reduces the plausibility of back-projection. Users can also interactively change it to a more suitable plane in $P^i$ if necessary, but this seldom happens for the examples shown in Fig. 4.

4.2 Reconstruction

With a plane selected from $P^i$ for $v^i_g$, naive back-projection of $v^i_g$, however, usually produces undesired disconnections and depth discontinuity between parts (Fig. 1-(f)) due to imprecise drawing and foreshortening perception bias. So, a subsequent optimization process is indispensable. To reconstruct the sketch $G$, we formulate a quadratic energy function based on the Laplacian mesh deformation framework [Sorkine et al. 2004]. Let $\Gamma$ denote the pointwise connections annotated by the user. We individually triangulate $v^i_g$ using its stroke points and its connection points in $\Gamma$. Note that we support two types of connections as shown in Fig. 1-(b): the X mark is the border connector and the star mark is the interior connector. We back-project the triangulated mesh of $v^i_g$ onto $\overrightarrow{p^i}$ in the camera space under the orthogonal view to get initial 3D positions (depth) of points in $\overrightarrow{v^i_g}$, which is defined as the 3D counterpart of $v^i_g$. We represent the back-projected mesh as $\Delta^i = (Q^i, F^i)$, where $Q^i$ is the set of mesh vertices in $\mathbb{R}^3$ and $F^i$ describes the vertex connectivity of $Q^i$. To deal with sketch parts that have no corresponding model parts, we divide all $\Delta^i$ into two classes: Parts in $\Delta_1$ have corresponding model parts and extracted planes, while parts in $\Delta_2$ do not (e.g., the dashed line in Fig. 4-(1)). For $\Delta^i \in \Delta_2$, we temporarily assign zero depth to it by back-projecting the triangulated mesh of $v^i_g$ onto the $xy$-plane in the camera space to get $Q^i$, and we expect the final positions of $Q^i$, i.e., the depth of each point, will be determined and smoothly interpolated through its connections to $\Delta_1$. To find the final plausible 3D positions of points in $\overrightarrow{v^i_g}$, we perform Laplacian mesh deformations to all $\Delta^i$’s while satisfying several constraints. Let $\overrightarrow{C^i}$ denote the unknown vertex positions of...
Our final energy function $E_{\text{md}}$ and $E_{\text{sd}}$. To keep the original shape as much as possible, we employ the traditional differential coordinates to define the mesh deformation cost $E_{\text{md}}$ [Sorkine et al. 2004] using $Q^i$ and $Q^j$, and the stroke deformation cost $E_{\text{sd}}$ [Nealen et al. 2007] using $v^i$ and $v^j$ for all $\Delta^i$’s.

Deformation cost $E_{\text{md}}$ and $E_{\text{sd}}$. To keep the original shape as much as possible, we employ the traditional differential coordinates to define the mesh deformation cost $E_{\text{md}}$ [Sorkine et al. 2004] using $Q^i$ and $Q^j$, and the stroke deformation cost $E_{\text{sd}}$ [Nealen et al. 2007] using $v^i$ and $v^j$ for all $\Delta^i$’s.

Proximity cost $E_{\text{pr}}$. Given a pointwise connection $\Gamma$ where points $u$ and $w$ come from different sketch parts, we enforce such a relationship in a proximity cost $E_{\text{pr}}$ by penalizing the Euclidean distance between the unknown 3D positions of $u$ and $w$.

Plane cost $E_{\text{pl}}$. Since we presume that $v^j$ envisioned by the user is hosted on a plane, we adopt a plane cost $E_{\text{pl}}$ to restrain $v^j$ from being non-planar. $E_{\text{pl}}$ is defined as the sum of point-to-plane distance of each point in $Q^j$.

Projection cost $E_{\text{proj}}$. To keep the resemblance between the user input and the 2D projection of the reconstructed result, we use a projection cost $E_{\text{proj}}$ to capture the $x$ and $y$ differences between points in $v^j$ and $v^j$.

Our final energy function $E_{\text{3d}}$ is a weighted average of $E_{\text{md}}, E_{\text{sd}}, E_{\text{pr}}, E_{\text{pl}}$ and $E_{\text{proj}}$, and it can be efficiently minimized using sparse matrix solvers. To get more visually pleasing results, if reflective symmetry exists in $M$, we also perform another symmetrization step, where point correspondences between symmetric sketch parts are computed by the iterative Closest Point (ICP) algorithm, while satisfying the constraints in $E_{\text{3d}}$ as well.

Figure 4: Reconstruction results. From left to right: User sketch, retrieved model and 3D reconstructed sketch under two different views. Black color in model (1) indicates no corresponding sketch parts and the dashed-line part of sketch (1) has no corresponding model parts.

5 Results and Discussions

To evaluate our proposed retrieval and reconstruction methods, we collected five categories of man-made objects to form the database, which contains 72 chairs, 54 tables, 45 airplanes, 20 glasses and 20 lamps. We implemented our system in C++ but our code is not fully hosted on a plane, we adopt a plane cost $E_{\text{pl}}$ to restrain $v^j$ from being non-planar. $E_{\text{pl}}$ is defined as the sum of point-to-plane distance of each point in $Q^j$.

Our final energy function $E_{\text{3d}}$ is a weighted average of $E_{\text{md}}, E_{\text{sd}}, E_{\text{pr}}, E_{\text{pl}}$ and $E_{\text{proj}}$, and it can be efficiently minimized using sparse matrix solvers. To get more visually pleasing results, if reflective symmetry exists in $M$, we also perform another symmetrization step, where point correspondences between symmetric sketch parts are computed by the iterative Closest Point (ICP) algorithm, while satisfying the constraints in $E_{\text{3d}}$ as well.

The ability of our system to handle different kinds of man-made ob-
ject drawings is limited by the richness of models in the database. However, if we increase the number of models, it will possibly ham-
per the running time performance of our structure-oriented retrieval approach. Thus a possible future work could be studying how to re-
duce the number of models in the database in terms of plane proxy arrangements extracted from models (e.g., by using a part-assembly approach [Shen et al. 2012]). Besides, due to our assumptions about sketch parts, we cannot deal with non-planar strokes. However, even with the above constraints, the results show that our model-driven sketch reconstruction with structure-oriented retrieval is still a viable solution to create interesting 3D forms of sketches.

Acknowledgements

This work was partially supported by grants from the Research Grants Council of HKSAR, China (No. HKUST16209514, HKUST16201315, CityU113513, CityU11204014 and CityU11237116), NSFC (No. 61502153), and the NSF of Hunan Province, China (No. 2016JJ3031).

References


