Data-driven segmentation and labeling of freehand sketches

Huang, Zhe; FU, Hongbo; LAU, Rynson W H

Published in:
ACM Transactions on Graphics

Published: 19/11/2014

Document Version:
Post-print, also known as Accepted Author Manuscript, Peer-reviewed or Author Final version

License:
Unspecified

Publication record in CityU Scholars:
Go to record

Published version (DOI):
10.1145/2661229.2661280

Publication details:

Citing this paper
Please note that where the full-text provided on CityU Scholars is the Post-print version (also known as Accepted Author Manuscript, Peer-reviewed or Author Final version), it may differ from the Final Published version. When citing, ensure that you check and use the publisher's definitive version for pagination and other details.

General rights
Copyright for the publications made accessible via the CityU Scholars portal is retained by the author(s) and/or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights. Users may not further distribute the material or use it for any profit-making activity or commercial gain.

Publisher permission
Permission for previously published items are in accordance with publisher’s copyright policies sourced from the SHERPA RoMEO database. Links to full text versions (either Published or Post-print) are only available if corresponding publishers allow open access.

Take down policy
Contact lbscholars@cityu.edu.hk if you believe that this document breaches copyright and provide us with details. We will remove access to the work immediately and investigate your claim.

Download date: 12/03/2019
Data-driven Segmentation and Labeling of Freehand Sketches

Zhe Huang  Hongbo Fu  Rynson W.H. Lau
City University of Hong Kong

Abstract

We present a data-driven approach to derive part-level segmentation and labeling of free-hand sketches, which depict single objects with multiple parts. Our method performs segmentation and labeling simultaneously, by inferring a structure that best fits the input sketch, through selecting and connecting 3D components in the database. The problem is formulated using Mixed Integer Programming, which optimizes over both the local fitness of the selected components and the global plausibility of the connected structure. Evaluations show that our algorithm is significantly better than the straightforward approaches based on direct retrieval or part assembly, and can effectively handle challenging variations in the sketch.

CR Categories:  I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation;

Keywords: Sketch Interpretation, Segmentation, Labeling

1 Introduction

A human being can easily understand a freehand sketch like those in Fig. 1(a). However, it is still challenging for a computer to interpret and recognize them. To ease ambiguity resolution, existing works typically require manual segmentation (e.g., [Xu et al. 2013]) or labeling (e.g., [Chen et al. 2009]) of sketches. While the manually specified information can certainly benefit sketch understanding, the resulting interfaces place various restrictions on how the user draws a sketch, e.g., on the drawing order of strokes or parts, thus disallowing the same freedom provided by pencil and paper.

We explore a new data-driven approach to semantic interpretation of freehand drawings. We focus on sketches of single 3D objects, without any semantic segmentation information as input, and aim at deriving semantically meaningful part-level structures (Fig. 1(d)). This leads to solve two challenging interdependent problems: segmenting input possibly sloppy sketches (Section 6.1) into semantically meaningful components (Sketch Segmentation), and recognizing the categories of individual components (Sketch Recognition).

To address the significantly increased ambiguity level, we propose to use a repository of 3D template models composed of semantically segmented components with labels, akin to visual memory of the human vision system, and solve these interdependent problems by heavily exploiting not only local shapes of individual parts but also their relative geometric configurations exhibited in the shape repository. Thus, our key problem is essentially to assemble repository parts properly with respect to input sketches, with minimal user intervention. We formulate the problem of identifying a globally optimal assembly as a Mixed Integer Programming problem.

To evaluate the performance of the proposed technique, we compare it with straightforward solutions based on direct retrieval [Xu et al. 2011] and part assembly [Shen et al. 2012], in terms of the ability to register correct semantics on freehand drawings made by several users. We have tested them on ten categories of objects and show great potential of our segmentation and labeling results for applications like sketch-based 3D modeling (Section 6.2).

2 Related Work

Existing works on sketch interpretation typically analyze an input sketch as a whole, without attempting to understand its component-level structures. Reviewing a large body of such existing works is beyond the scope of this paper. We refer the interested reader to insightful discussions on sketch-based 3D modeling (e.g., constructive [Rivers et al. 2010] or template-based [Kraevoy et al. 2009; Olsen et al. 2009]), sketch recognition (e.g., [Eitz et al. 2012a]), sketch-based shape retrieval (e.g., [Eitz et al. 2012b]).

Being able to segment sketches, at the part or object level, greatly benefit applications like sketch-based object modeling [Lee and Funkhouser 2008; Xie et al. 2013] and scene modeling [Shin and Igarashi 2007; Xu et al. 2013]. Such segmentations are typically achieved by design (e.g., manually drawing individual logical components in different layers, or drawing components one by one in a specific order), which however is often an error-prone process even
for experienced artists and thus defeats the main motivation behind sketch-based interfaces [Noris et al. 2012]. Having per-segment labels definitely improves the retrieval performance, but at the cost of manual specification of per-segment labels [Chen et al. 2009].

Sketch segmentation and recognition have been extensively studied for understanding hand-drawn 2D diagrams (see [LaViola et al. 2006] for an insightful survey), e.g., electric circuits, algorithm flowcharts, and various graphical models. However, unlike our input sketches (Fig. 2 first row), which are rough projections of 3D objects and thus often involve parts overlapped to each other in the image space, such 2D diagrams typically contain isolated, non-overlapping symbols that are linked together [Gennari et al. 2005].

Therefore, much simpler segmentation methods, e.g., based on low-level geometric descriptions like lines and arcs [Sezgin et al. 2001; Gennari et al. 2005], already suffice, but are inapplicable to our problems. Domain-specific knowledge is then often employed in their recognition steps to resolve ambiguities [Gennari et al. 2005].

The recent work by Sun et al. [2012] is probably the most relevant to ours. Their work also aims at semantic segmentation of freehand sketches, with the help of a million of clip art images as a knowledge base. Their solution is based on a greedy merging strategy and thus heavily depends on the drawing sequence of strokes. Instead, our work does not enforce such artificial restrictions on the user and needs to tackle the increased ambiguity level.

Simultaneous segmentation and recognition of images (e.g., [He et al. 2004; Shotton et al. 2006]) has recently been an active topic. The very recent effort has been devoted to segmentation and recognition of depth images (e.g., [Ren et al. 2012; Shao et al. 2012]), acquired by low-cost depth cameras like Microsoft Kinect. A similar problem has been studied in the context of segmentation and labeling of 3D meshes [Kalogerakis et al. 2010]. Due to the lack of region-based information, which is essential to the success of image/mesh segmentation, we take a fully geometric approach, instead of attempting to adapt machine-learning approaches used in most of the previous works, to our problem.

Our technique takes a part-assembly approach to sketch segmentation and recognition. Assembly-based 3D modeling was pioneered by Funkhouser et al. [2004] and has been extended for data-driven modeling suggestions [Chaudhuri et al. 2011] and open-ended 3D modeling [Kalogerakis et al. 2012; Xu et al. 2012]. Sketch-based interfaces have also been explored for object or scene compositions [Shin and Igarashi 2007; Lee and Funkhouser 2008; Xie et al. 2013], which, however, require manual sketch segmentation, as discussed previously. Shen et al. [2012] present a part-assembly approach for recovering 3D geometry and structure from raw scans acquired by RGBD cameras. Their approach is not designed for explicit segmentation of input raw scans.

### 3 Straightforward Solutions

Our goal is to take a freehand sketch as input and assign each point in the sketch with a label from a pre-defined set of labels, with the help of a repository of segmented and labeled 3D models belonging to the same class as the object depicted by the sketch. It is possible to adapt some previous solutions originally designed for other tasks to our problem, as discussed below.

**Direct retrieval approach.** We first discuss a two-step approach based on direct shape retrieval (Fig. 3). First, taking the input sketch as a query, we perform sketch-based shape retrieval [Eitz et al. 2012b], which returns the best-fit model from the shape repository and a specific viewpoint. Second, we transfer the labels from the model to the sketch. This can be achieved by model-driven labeled segmentation [Xu et al. 2011], which is originally designed for labeled segmentation of a photographed object. This direct retrieval approach is effective only if there exists a repository model that is globally similar to the input sketch. However, it is questionable whether such a desired model is in a repository of moderate size.

In addition, freehand sketches have inherent distortions. Therefore even if such a model indeed exists, it would be difficult to find a desired global camera projection due to the differences in pose and proportion between the sketched object and the 3D model.

**Part-assembly approach.** We adapt the part-assembly approach originally designed for structure recovery from raw 3D scans [Shen et al. 2012] to our problem, followed by our segmentation and labeling step described in Section 4.4. We disregard all depth-related components and use the input sketch as an edge map fed into the original algorithm. The original approach does not explicitly solve the segmentation problem but assumes that after being globally aligned with an input sketch given a specific viewpoint, each repository model provides a global context for its individual parts and thus to some extent a fuzzy segmentation map. However, this might over-constrain the search space of candidate parts (Fig. 4 (c)).

In addition, this approach also over-constrain the number of candidate parts for each category (only top $K$ best-matched candidates are kept) in order to limit the combinatorial search space of the structure composition step. Consequently, there might exist no suitable candidates for certain parts, as illustrated in Fig. 4.

**4 Methodology**

To address the shortcomings of the straightforward solutions we design a new part-assembly approach specifically geared towards sketch interpretation. Our approach first matches individual segmented components in our repository to the entire domain of the sketch, without constraining the search in local windows as in [Shen...
et al. 2012], and get local interpretations of the sketch (Section 4.2). There might exist multiple valid local interpretations for different parts of the sketch. We thus perform global sketch interpretation by finding a globally optimal combination of local sketch interpretations. We formulate our global interpretation problem as a Mixed Integer Programming (MIP) problem (Section 4.3), allowing us to handle many more local interpretations than the straightforward part-assembly approach. Finally, we find a labeled segmentation of the sketch guided by the optimal global sketch interpretation (Section 4.4). The above steps are shown in Fig. 5(c–e).

Our approach and the straightforward solutions require the estimation of the 3D viewpoint of the sketch. This problem is challenging on its own. In the following discussions, we first assume that such a viewpoint is already known for an input sketch (Fig. 5(b)). We will discuss how to interactively or automatically determine the viewpoint of a given sketch and how they influence the performance of sketch segmentation and labeling in Section 5.

4.1 Preprocessing

Shape repository. Our algorithm requires a repository of 3D models that are of the same class as the input sketch. For example, a repository containing only chair models should be used for a sketched chair. Each model is pre-segmented into semantic parts with labels, such as “leg” and “seat” for the chair class, and is well aligned, with consistent upright and front-facing orientation.

Input sketch. A Wacom tablet was used for sketch acquisition. We intend to place minimal constraints on how a user draws a sketch. For example, the user does not need to indicate if she has finished drawing a single component, or to complete one component before moving on to draw the next (implicitly required by [Sun et al. 2012]). There are also no restrictions on the number of strokes in each sketched component. Different components (e.g., chair back and legs) can even share the same strokes. However, our current algorithm does require an input of cleanly defined strokes (Fig. 2), without over-sketching or hatching effects.

The online drawing process naturally gives us a stroke-based representation. A sketch typically consists of tens of strokes. Each stroke is represented as a time-ordered sequence of points, with each point containing a 2D coordinate and a time stamp. Our current algorithm does not consider additional information, such as pressure and pen orientation, offered by pen-based input devices. We break each stroke into segments at high-curvature points or junction points, including X-junctions and T-junctions. This is necessary since occasionally a stroke may cover multiple components (e.g., Fig. 2(1)(a)), while it is less likely for a stroke segment to do so (Fig. 2(1)(b)). This step is analogous to over-segmentation, which is often used as a preprocessing step in image segmentation.

Fig. 2(bottom) shows four input sketches with stroke segments, which form the basic unit for many operations in our algorithm.

4.2 Local Interpretation

We examine how individual segmented components in the repository locally interpret the sketch. A good local interpretation, means that there is a good match between a repository component and a certain part of the sketch, under a specific view angle. As such, a local interpretation is defined as a 3D component with a fixed 2D location on the image plane, where it fits the sketch locally.

Given a global viewpoint (manually specified or automatically derived), we match each component of each repository model (Fig. 6(a)) against every part in the sketch. The matching quality is measured by the pixel-wise convolution on the sketch and the projected contour (on the image plane) of the component. For simplicity, we use the silhouette of a 3D model to get its line drawing representation in our current implementation. All the points corresponding to the local maxima of the convolution scores are identified as the candidate locations for the component’s local interpretations (Fig. 6(b)).

Each component often has many candidate locations. We perform the following steps to reduce the number of interpretations and prune interpretations that are very likely false. First, candidate locations are clustered using agglomerative clustering (2D clusters in our implementation) and only the candidate location with the highest matching score from each cluster is kept as the filtered candidate locations (shown as red crosses in Fig. 6(b)). Second, to accommodate for the inaccuracy in the given global viewpoint and/or geometry distortions of the input sketch, we perform the Iterative Closest Point (ICP) algorithm at each new candidate location to map each point of the component contour to the sketch (Fig. 6(c)). Since each stroke segment is a single unit in our algorithm, it is considered as interpreted (or covered) by a component if more than half of its pixels have correspondences to the component contour. All filtered candidate locations with low curve matching scores, which measure the proportion of the component contour having corresponding points on the sketch, are pruned. To compute the curve matching score, two points on the sketch and the component are matched, if their distance is smaller than 10% of the diagonal length of the component’s bounding box and the angle between their tangents is less than 30 degrees.

After all local interpretations of all components in a category (e.g., all kinds of seats of a chair) are found, we cluster them to further reduce redundancy as there might be many equally good local interpretations, arising from similar components covering almost the same set of stroke segments (Fig. 6(d)). Agglomerative clustering is again used here, where the distance between local interpretations is measured by the difference of their covered stroke segments. More specifically, let \( P_i \) and \( P_j \) be the sets of points in the stroke segments covered by local interpretations \( C_i \) and \( C_j \), respectively. The distance \( d(C_i, C_j) \) is computed as \( |P_i \cap P_j|/|P_i \cup P_j| \). After clustering, the local interpretation with the highest matching score in each cluster is retained (Fig. 6(e)). We keep about 30 components for each category.

4.3 Global Interpretation

Local sketch interpretation has very limited ability to resolve ambiguities. Due to the nature of local matching, many of the selected local interpretations are still false even after pruning. It is expected that only a few combinations of such local interpretations globally match the input sketch and have the involved repository compo-
nents well connected. Let $C = \{c_i\}$ denote the set of all valid local interpretations from all repository components, and $P$ a candidate global interpretation, represented as a combination of local sketch interpretations, i.e., $P \subseteq C$. We formulate the problem of identifying the best global interpretation of the sketch as a Mixed Integer Programming (MIP) problem.

### 4.3.1 Decision Variables

Taking individual local interpretations as nodes and their connections as edges, our task is to select an optimal subgraph from a complete graph formed by every pair of local interpretations. Finding the optimal subgraph can be achieved by making a series of decisions, which are represented by binary variables, often known as decision variables. We introduce three sets of decision variables for our MIP problem, as illustrated in Fig. 7.

**Node selection variables.** Our primary goal is to identify the best set of local interpretations. For brevity, we refer to each local interpretation $c_i \in C$ as a node. We associate each node $c_i$ ($i = 1, \ldots, |C|$) with a variable $x_i \in \{0, 1\}$, where $x_i = 1$ if $c_i \in P$ (i.e., $c_i$ is included into a global interpretation $P$) and $x_i = 0$ otherwise. Here $|C|$ is the cardinality of $C$.

**Edge selection variables.** The key to resolve the ambiguity in the sketch is to employ contextual information of individual components and their geometric relationships exhibited in the repository. To model such pairwise relationships, we associate every pair of nodes with an edge variable $y_{i,j}$ ($i, j = 1, \ldots, |C|$), where $y_{i,j} = 1$ if $c_i$ and $c_j$ are determined as connected.

**Segment assignment variable.** We need to determine which stroke segment is explained by which local interpretation. We thus associate every pair of stroke segment and local interpretation with a decision variable $v_{i,j}$ ($i = 1, \ldots, |G|$; $j = 1, \ldots, |C|$), where $G = \{g_i\}$ is the set of stroke segments in the preprocessed sketch (Section 4.1) and $|G|$ is its cardinality. $v_{i,j} = 1$ if and only if stroke segment $g_i$ is explained by local interpretation $c_j$.

### 4.3.2 Objectives and Constraints

To identify the best global interpretation, we introduce a set of objectives and constraints, which will be optimized over the decision variables to get the optimal interpretation.

**Fitness.** We require each selected component to locally fit the sketch reasonably well. This is achieved by introducing a cost term $W_f = x_i(1 - f_i)$ for each node $c_i$. Here, $f_i = m_i t_i$ is the fitness for $c_i$ in terms of both shape and relative context. The term $m_i$ is the curve matching score. Specifically, $m_i = |c_i^w|/|c_i|$, where $c_i^w$ is the portion (yellow dashed lines in Fig. 8(a)) of $c_i$ with matched points on the sketch and $|c_i|$ is its length.

The term $t_i$ is a *context matching score*. Contextual features have been extensively used in computer vision [Galleguillos and Belongie 2010]. Context is often defined at the label level, but not between individual geometry instances. For example, a leg of a chair is more likely to appear underneath a seat. Ideally, we should measure the context matching score $t_i$, as the difference between the context of $c_i$ in $P$ and the observed context in the repository. This, however, would lead to a more complicated optimization problem. For simplicity, the context of $c_i$ is computed as a histogram of nearby labels (in the spirit of shape context [Belongie et al. 2002]), based on the label distribution around $c_i$ given by all other local interpretations. This computed context is compared to the context of the components (computed in a similar way) in the database with the same label as $c_i$, and the difference is recorded in $t_i$. In this way, $t_i$ can be pre-computed, after all the local interpretations are found (Section 4.2). Note that $t_i$ is defined in the image plane, since a global viewpoint is already specified.

**Coverage.** We should select enough local interpretations so that a significantly large portion of the sketch is covered (Section 4.2). In other words, most of the sketch points should be matched by the components in $P$. We define the coverage of the sketch as the total length of the covered stroke segments divided by the total stroke length of the sketch (see Fig. 8(b)). To maximize coverage, we penalize the proportion of the uncovered sketch by introducing a cost term: $W_c = 1 - \sum_{i,j} y_{i,j} \frac{|g_i|}{|G|}$, where $|g_i|$ is the length of the stroke segment $g_i$ and $|G|$ is the length of all stroke segments, i.e., the total stroke length of the sketch. $y_{i,j}$ is a binary indicator variable that captures the event of $g_i$ being covered, that is, $y_{i,j} = 1$ if and only if at least half of $g_i$ has correspondence on some component. $y_{i,j}$ is determined by the segment assignment variables via constraints (see the supplemental for details). An upper bound $\beta^c$ is set for $W_c$, i.e., $W_c \leq \beta^c$, to ensure that at least a significant amount of the sketch is covered.

**Connection.** The selected components should be well connected and form a complete structure. We thus impose a cost term that penalizes bad connections: $W_o = y_{i,j}(1 - a_{i,j})$, where $a_{i,j}$ measures the quality of the connection between components corresponding to $c_i$ and $c_j$. Such a cost takes effect when $c_i$ and $c_j$ are determined to be connected, i.e., $y_{i,j} = 1$. The connection quality $a_{i,j}$ measures how much movement $d$ is required to move $c_i$ to connect with $c_j$; the larger the movement, the lower the connection quality. Precisely, $a_{i,j} = e^{d^2/(2\sigma^2)}$ (see Fig. 8(c)), where $\sigma$ is set to 15% of the diagonal length of the sketch in our implementation. Let $l_i$ and $l_j$ denote the labels of components $c_i$ and $c_j$, respectively. We require $c_i$ and $c_j$ to be connected in a way like a pair of components with labels $l_i$ and $l_j$ in the repository. $d$ is thus defined as the minimum distance to move $c_i$ to connect with $c_j$ by enumerating the connections between all pairs of components with labels $l_i$ and $l_j$. Note that since the depth of the candidate components is unknown, the distance is computed in 2D. If a pair of components with the same labels as $c_i$ and $c_j$ are never connected to each other in the repository, or $a_{i,j}$ is below a certain threshold $\beta^o$, we consider that $c_i$ and $c_j$ are not connectable and add a hard constraint $y_{i,j} = 0$ to the optimization.

**Overlap.** We seek for a compact interpretation of the sketch, i.e., to select as few local interpretations as possible to explain for the sketch. This property is captured by our overlap cost term and constraint. The cost term $W_r$ penalizes a stroke segment $g_i$ being covered by more than one selected local interpretation. It is defined as $W_r = o_i||g_i||/||G||$. Here $o_i = \max (n - 1, 0)$ counts the number of times $g_i$ is being covered by the selected local interpretations. For a simple example in Fig. 8(d), $o_i = 1$ since $g_i$ is covered by both $c_1$ and $c_2$ and thus $n = 2$. Note that $o_i = 0$ when $g_i$ is covered by no more than one selected local interpretation. The definition of $o_i$ in linear form is given in the supplemental.

A hard constraint is also used to prevent significant overlap between $c_i$ and the union of other selected local interpretations $C_j = \{c_j\}$. The overlap ratio of $c_i$ is defined as $R_i = ||G_i||/||G||$, where $|G_i|$ is the total length of the stroke segments covered by $c_i$, and $|G|$ is the total length of the stroke segments covered by both $c_i$ and $C_j$. See an illustration in Fig. 8(d). We require that it does not exceed a threshold, i.e., $R_i \leq \beta^r$. More precisely, $G_i = \{g_k\}$ is
the set of stroke segments covered by \(c_i\), and \(|G_i| = \sum_k |g_k|\) is fixed with respect to \(c_i\). \(G_i' \subseteq G_i\) and \(G_i'\) changes dynamically during optimization. The length of stroke segments in \(G_i'\) is computed as \(|G_i'| = \sum_k r_{k,i} |g_k|\), where \(r_{k,i}\) is the stroke segment overlap indicator, i.e., \(r_{k,i} = 1\) if and only if \(g_k\) is overlapped. The definition of \(r_{k,i}\) in terms of linear constraint is given in the supplemental.

**Stroke Segment Grouping.** The sketch itself contains some useful information from which we can hypothesize that some stroke segments tend to share the same labels. We explore the following grouping criteria, as illustrated in Fig. 8(e):

- **Temporal distance:** When temporal information is available, we define the temporal distance between two segments as the difference between their associated time stamps. Two stroke segments are of a temporal group if their temporal distance is below a threshold. This criteria is not used if no temporal sketching information is available, e.g., offline sketches traced using some automatic tools, e.g., Adobe Illustrator Live Trace.

- **Continuity:** Stroke segments that come from a single stroke form a continuity group, since the user tends not to break the stroke when drawing the same component.

- **Parallelism:** Stroke segments that are parallel to each other and of similar length are in a parallel group. Such heuristic comes from the regularity in many man-made objects.

As the grouping criteria are based on some heuristics, we enforce them using a cost rather than a hard constraint. The cost encourages the stroke segments within a group to be assigned with the same label. Let \(G_i = \{g_k\}\) be a stroke segment group and \(l_j\) be a component label (e.g., seat, back, etc.), the group labeling inconsistency cost is defined as \(W^c = \sum_i b_{k,j} |g_k|\), where \(|g_k|\) is the length of the stroke segment \(g_k\), \(b_{k,j}\) is an inconsistent labeling indicator. It takes the value of 1 if stroke segment \(g_k\) should have been assigned to a component with label \(l_j\), but is not. To elaborate this, we say that \(g_k\) is assigned label \(l_j\) if it is assigned to a candidate component of type \(l_j\), and the group \(G_i\) is activated by label \(l_j\) if some of its stroke segments are assigned label \(l_j\). \(g_k\) is said to be inconsistently labeled when the group is activated by \(l_j\) but \(g_k\) is not assigned \(l_j\), in which case \(b_{k,j} = 1\). The definition of \(b_{k,j}\) in linear form is given in the supplemental.

**Stroke segment assignment.** As mentioned in Section 4.3.1, a stroke segment needs to be assigned to a selected local interpretation dynamically during optimization, and the assignment is subject to constraints. For a set of sketch segments \(G_i = \{g_k\}\) covered by \(c_i\), some of \(g_k\) may be ambiguous. A stroke segment \(g_k\) is said to be ambiguous if \(g_k\) and another stroke segment \(g_{k'}\) match to the same part of \(c_i\) (see Fig. 8(f)). If \(g_k\) is not ambiguous, it is simply assigned to \(c_i\), i.e., setting \(v_{k,i} = x_i\). Otherwise, a group of stroke segments that are all matched to \(c_i\) must have only one of the members assigned to \(c_i\). In linear form, such a constraint is formulated as \(\sum_k v_{k,i} = x_i\), where \(k = (1...K)\) for a group of mutually conflicting stroke segments \(\{g_k\}\).

**Objective function.** Let \(n_g\) denote the numbers of stroke segment groups and labels, respectively. Combining the above cost terms and constraints, the total cost of selecting a candidate global interpretation is then:

\[
E = \alpha_c W^c + \alpha_f \sum_{i=1}^{n_g} W_i^f + \alpha_s \sum_{i=1}^{n_g} \sum_{j=1}^{n_l} W_{i,j}^s
+ \alpha_o \sum_{i=1}^{n_g} W_i^o + \alpha_g \sum_{i=1}^{n_g} \sum_{j=1}^{n_l} W_{i,j}^g,
\]

subject to the following constraints:

- \(W^c \leq \beta^c\): the proportion of uncovered sketch should not exceed \(\beta^c\).
- \(W_{i,j}^s \leq \beta^s\): bad connections with a cost higher than \(\beta^s\) are excluded.
- \(R_i \leq \beta^o\): maximum overlap allowed for \(c_i\) is \(\beta^o\).
- \(v_{i,j} = x_j\): stroke segment \(g_i\) covered by \(c_i\) is assigned to \(c_j\) if \(g_i\) is not conflicting with others.
- \(\sum_k v_{k,j} = x_j\): only one of the mutually conflicting stroke segments \(g_k\) covered by \(c_j\) is assigned to \(c_j\).

In our implementation, \(\alpha_c = 100, \alpha_f = 10, \alpha_s = 20, \alpha_o = 15, \alpha_g = 5, \beta^c = 0.1, \beta^s = 0.9\), and \(\beta^o\) is set as the median of all \(1 - a_{i,j}\) in order to retain half of the connections. For its high speed the Gurobi optimizer is adopted to solve our above Mixed Integer Programming problem (Eq. 1).

### 4.4 Segmentation and Labeling

After obtaining a global interpretation of the sketch, the assignments for the stroke segments can be used to label the sketch. A naïve approach is to label a stroke segment according to the label of the component assigned to it. However, a stroke segment could be mistakenly assigned to multiple candidate components, and we need to determine the final label, by hopefully removing the incorrect assignments. We use graph cut to optimize the labeling, similar to [Noris et al. 2012]. More specifically, a graph is constructed by connecting each point on the sketch to the nearest \(k\) (\(k = 5\) in our implementation) points, where label \(l\) can be assigned to a point \(p\) with a cost \(A(p, l)\), and if a pair of neighboring points \(p_i\) and \(p_j\) are assigned different labels, they induce a cost \(B(p_i, p_j)\). \(A(p, l)\) is determined by matching score of \(p\) to the matched point on the candidate component with label \(l\), to which the stroke segment containing \(p\) is assigned. \(B(p_i, p_j)\) is a constant. (Refer to [Noris et al. 2012] for more algorithm details.) The optimization will smooth out the labeling on the sketch, removing many incorrect assignments.

### 5 Evaluation

In this section we evaluate first the effectiveness of our method by comparing it to the straightforward solutions (Section 3). For fair comparison, we kept the implementation of the compared solutions faithful to their original forms. For example, ICP was not
performed for these methods, since ICP does not always improve matching quality and might cause extra problems (e.g., fitting the seat into the back in Fig. 6(c)), which cannot be resolved by the competing solutions. We then evaluate the impact of individual terms in our optimization (Eq. 1).

**Timings.** We have implemented our method in MATLAB on a PC with a i7 3GHz CPU and 18GB RAM. The sketch and the components were rasterized into $320 \times 320$ images. On average, the local interpretation step took about 30 minutes and the global interpretation less than 10 minutes. In comparison, direct retrieval (DR) varies from less than a minute to more than 2 hours, in which case the algorithm is terminated and run again with fewer candidates (setting $K = 4$).

### 5.1 3D Viewpoint Estimation

We assume that our input sketches were produced under orthographic projection. In our experiments, we tested both automatic and interactive ways to determine the viewpoint of the sketch. To automatically estimate a viewpoint, we performed sketch-based retrieval [Etz et al. 2012a] using the input sketch as a query and simply adopted the 3D viewpoint associated with the returned globally best-matched model from our repository as the viewpoint of the sketch. Since the input sketch might be very different from any repository in terms of structure and/or shape, such an automatically found viewpoint might not always be accurate. We thus allowed the user to interactively rotate and scale the retrieved model to match the orientation and size of the sketched object (Fig. 5 (b)). Our main experimentation was based on interactively specified viewpoints, though the performance when using automatically estimated viewpoints was also evaluated (Section 5.3). For fair comparison, the same viewpoints were used in both our approach and the straightforward approaches.

### 5.2 Datasets

We have tested 10 classes of models: chair (68 models), table (57 models), airplane (59 models), bicycle (37 models), fourleg (20 models), lamp (20 models), vase (61 models), human (19 models), candelabrum (35 models) and rifle (25 models). See the supplemental for the thumbnails of these models. The first 4 sets of models were from [Shen et al. 2012]. Fourleg (i.e., quadruped), lamp, vase and candelabrum were from [Wang et al. 2012], human was from [Kalogerakis et al. 2010] and rifle was from Trimble 3D Warehouse. For vase, we picked only 61 models of the 300, for their relatively simple structures. A few models were added to different classes to enrich the ways that components connect to each other. For each class, we collected 30 sketches from 3 users (i.e., 10 from each user); one user was an experienced artist and two were not. On average, each sketch took about 3 minutes to draw using a Wacom tablet. The user randomly selected one photo from an image repository that we prepared (see below), and sketched the object by looking at the selected photo, i.e., by observational drawing instead of image tracing. Fig. 9 shows some example photo/sketch pairs.

When asked to draw an object, most of us often take examples from around us. Given a limited number of participants with little drawing skill, we thus did not expect that they were able to produce a great variety of sketches, which, however, were needed to evaluate the generality of our algorithm. Thus, we decided to provide reference images as inspiration to the participants. Otherwise, we would have to recruit a much larger group of participants. To construct our image repository, for each object class, we identified the search keywords using the class keyword (e.g., ‘chair’), and subclass keywords (e.g., “bar stool”, “beach chair”, “bath chair”) that were manually picked from Wikipedia (e.g., the “List of chairs” page). For each keyword, we downloaded the first 1,000 images returned by Google image search, and manually excluded those images (see Fig. 10 for some examples) that contain components very different from the components available from our database models. At the end, we obtained several thousands of images for each of the 10 object classes.

**Figure 9:** Reference photos and corresponding freehand sketches. Note that sketches were drawn with observational drawing instead of image tracing. The resulting sketches typically contain significant deviations, in terms of both shape and viewpoint, from the original reference images.

**Figure 10:** Example images excluded from the image repository.

![Example images excluded from the image repository.](image)

### 5.3 Labeling Accuracy

Each of our method and the straightforward solutions assigns a label to each pixel of the sketch. To evaluate their performance, we define two accuracy metrics:

(a) **Pixel-based accuracy (P-metric),** which is the fraction of pixels that are assigned with the correct labels.

(b) **Component-based accuracy (C-metric),** which is the number of correctly labeled components divided by the total number of components. A component is correctly labeled if 75% of its pixels are assigned the correct label.

The two metrics complement each other: the pixel-based accuracy has a bias towards large components (which take up more pixels), and component-based accuracy gives a bias to small components (which are usually of larger number).

Note that the ground truth labeling for a sketch, which is a complete label assignment of all pixels, is not unique, due to at least two reasons: 1) some pixels belong to a stroke shared by several components (e.g., chair seat and back usually have a shared stroke); 2) part of the sketch has multiple interpretations (e.g., a region bounded by strokes can be hollow or solid). Therefore, each sketch has multiple equally valid ground truth labelings (manually indicated by the authors), and the computed labeling is compared with all of them, returning the highest accuracy as result.

Table 1 shows the average labeling accuracy of the three methods for each class of objects. Using P-metric, on average, the labeling...
accuracy of our method was 13% higher than the direct retrieval method, and 9% higher than the one based on [Shen et al. 2012]. Statistical significance test (t-test with $p$-value $< 0.05$) shows that on average our method significantly outperformed the other two methods. For individual object classes, our method significantly outperformed direct retrieval in all but the vase class ($p = 0.85$), and [Shen et al. 2012] in all but the bicycle ($p = 0.15$), lamp ($p = 0.07$) and vase ($p = 0.05$) classes. Using $C$-metric, the advantage of our method was even more obvious, with the average accuracy being 20% higher than direct retrieval and 11% higher than [Shen et al. 2012]. Our method was significantly more accurate than direct retrieval in all classes, and [Shen et al. 2012] in all except the bicycle ($p = 0.43$), vase ($p = 0.18$) and candle-abrum ($p = 0.91$) classes. The advantage comes from the fact that our method produced less partial labelings, e.g., the table legs in Fig. 11(2(e)) and (2(f)), which were considered as incorrect using $C$-metric, but partially correct using $P$-metric. This is because the structure found by our method, particularly in terms of component occurrence, matched with the sketched object better than those produced by the other methods. Table 2 shows the number of winning cases of each method for labeling each sketch individually.

In general, our method had a greater advantage when a sketched object exhibited higher structural variations (e.g., chair and table). Such an advantage was less obvious for classes of objects with relatively fixed structures (e.g., bicycle and vase). However, our method still performed significantly better on fourleg and human, though they had almost no structural variations. This was mainly due to their pose variations. Because of the ICP matching step and the use of many more local interpretations (compared to [Shen et al. 2012]), it was more likely for our method to find the right components undergoing rotation and/or scaling transformations.

Table 4: Result of per-stroke labeling, computed with pixel-based accuracy (P) and component-based accuracy (C).
meet the minimum coverage hard constraint, leading to a missing component in the highlighted region. In 4(b), without the fitness constraint, poorly fitted components (extra wheels) were selected to cover the sketch. In 5(b), the highlighted part was fitted with components heavily overlapping with each other due to the lack of the overlap constraint. In 6(b), a seat was incorrectly fitted to the highlighted part. In contrast, the relevant stroke segments (highlighted) in 6(a) were grouped due to temporal distance, continuity and parallelism and were assigned a consistent label.

As shown in Table 5, turning on all factors were not always optimal, and some factors had slightly negative effects on certain classes of objects. This is because the various scores (e.g., matching, context, connection quality, etc.) used in the optimization are only estimations and inevitably contain errors. Some representative cases are shown in Fig. 13, illustrating how the inaccuracies affected the results. For example, 1(a) incorrectly selected a middle support (blue component) to cover the highlighted part of the sketch, which was prevented in 1(b) since the table top (cyan component) already covered the lower part of the sketch (highlighted) to avoid picking the middle support. Although neither of them was correct, 1(b) happened to get more stroke segments correctly labeled and had higher accuracy. Comparing 2(a) and 2(b), the latter selected several overlapping components to cover the tail of the airplane (highlighted) and to achieve higher accuracy, which was not possible for the former due to the overlap constraint. Comparing 3(a) and 3(b), the latter correctly identified the tail (highlighted), while the former could not, since the relevant stroke segments were wrongly grouped with the nearby stroke segments representing the leg due to parallelism.

Finding the optimal setting automatically for a given sketch is non-trivial. One possible way is to try turning off each factor and see which setting leads to the best inferred structure. To evaluate each output structure, one potential approach is to perform 3D reconstruction using the selected components (Section 6.2) and evaluate.
the assembled model against some learned priors such as the meta-representation [Fish et al. 2014]. A full investigation into this issue is left as a future work.

**Symmetry.** Man-made objects like chair, table and airplane often exhibit strong symmetry, where many of their components have symmetric counterparts. Utilizing such symmetry relationships may further constrain the solution space and improve the accuracy of the structure inference. We attempted to incorporate such a constraint, by searching for a symmetric counterpart of each local interpretation that had a symmetric component in the database model, and forced the paired local interpretations to be selected or rejected simultaneously in structure inference. In Table 6 we experimented with reflective symmetry and assumed that the sketched object had a symmetry plane identical to that of the model used for viewpoint estimation (Fig. 5(b)). It may be surprising to see that using such a symmetry constraint did not generally improve the labeling accuracy. This may be because the symmetric counterparts in the object class of the input sketch is known. Such a requirement could possibly be relaxed by first performing sketch recognition (e.g., using end points of a cylindrical component). Finally, the components are deformed to connect to each other, as similarly done in [Kalogerakis et al. 2012]. Fig. 14 shows a few 3D models constructed using the above method.

We have found that while the symmetry constraint (Section 5.4) is not very effective for improving the labeling accuracy, it is useful to guide part-assembly-based modeling here. We envision more exciting applications, like turning a 2D sketch directly to a 3D sketch. With our approach, it is possible to estimate a 3D position for each point in the freehand sketch. This makes it possible to recover a 3D sketch directly from a 2D sketch, enabling a promising application of 3D sketching with 2D input.

<table>
<thead>
<tr>
<th>Grouping</th>
<th>Overlap</th>
<th>Symmetry</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>sym. on</td>
<td>0.775 0.803 0.772 0.704 0.680</td>
<td>0.658 0.638 0.657 0.578 0.541</td>
<td></td>
</tr>
<tr>
<td>sym. off</td>
<td>0.765 0.824 0.791 0.719 0.727</td>
<td>0.631 0.660 0.690 0.631 0.567</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: The effect of the symmetry constraint. Note that we did not apply the symmetry constraint on bicycle and lamp as they are usually drawn from a side view, nor on fourleg and human due to their pose variations.

## 6 Discussions

### 6.1 Assumptions on Input Sketches

The input sketches are assumed to be clean and contain well-defined strokes. A clean sketch is one that does not contain significantly extra dots or strokes that do not belong to the depicted object. Such noise is commonly seen in scanned sketches, which if to be used as input, should first go through a noise filtering process. The unfiltered noise could lead to over-segmentation and thus extra components. This is not an issue for our tested sketches, since they were all collected via online drawing. A stroke is well-defined if it is free of over-sketching, that is, it is drawn as a single curve rather than a bunch of nearly overlapping curves. Over-sketching can be supported by using interfaces like [Pusch et al. 2007], which automatically combines overlapping strokes to a single stroke. Note that our current method can handle curves that are broken into segments drawn at different times. Lastly, we also assume that the object class of the input sketch is known. Such a requirement could possibly be relaxed by first performing sketch recognition (e.g., using [Eitz et al. 2012b]).

Although users need not sketch hidden lines, e.g., Fig. 2(3(a)), having hidden lines is generally useful, e.g., for identifying more accurate local interpretations or completely hidden components. The

---

**Figure 13:** Negative effects of some factors used in MIP.

**Figure 14:** Application in sketch-based modeling by part assembly.
solutions based on direct retrieval and part assembly. We are interested in improving the computational performance of our algorithm to enable interactive sketch-based modeling. It is possible to further improve the labeling accuracy, for example, by learning and employing the occurrence probability of components in the repository. It would also be interesting to extend our work for semantic understanding of sketched scenes [Xu et al. 2013], possibly enabling a more automatic sketch-based scene modeling application. We anticipate that extra depth and occlusion cues, which might be recovered from a sketch (e.g. using dashed hidden lines), would improve the labeling accuracy if they could be formulated as constraints in the global optimization step.

Acknowledgments

We thank the reviewers for their constructive comments. This work was partially supported by three grants from RGC of Hong Kong (Project No.: CityU 113513, CityU 11204014 and CityU 115112).

References


