Interpreting Concept Sketches

Tianjia Shao[1,2] Wilmot Li[3] Kun Zhou[4] Weiwei Xu[5] Baining Guo[6] Niloy J. Mitra[1]

[1]Univ. College London [2]Tsinghua Univ. [3]Adobe [4]Zhejiang Univ. [5]Hangzhou Normal Univ. [6]MSRA

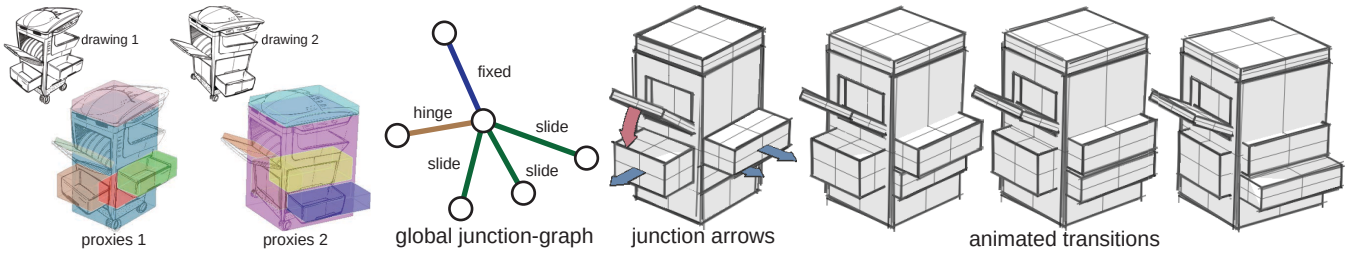


Figure 1: *We present a system to interpret concept sketches. Starting from input sketches (drawings 1 and 2) and rough geometry proxies, we automatically extract consistent proxy correspondence across the views and a global junction-graph encoding inter-proxy connections. The user can then interactively change view and/or manipulate junctions (based on arrow handles), or browse through animated transition sequences. The key observation is that consistent inter-part relations can be inferred even based on largely inconsistent geometry information.*

Abstract

Concept sketches are popularly used by designers to convey pose and function of products. Understanding such sketches, however, requires special skills to form a mental 3D representation of the product geometry by linking parts across the different sketches and imagining the intermediate object configurations. Hence, the sketches can remain inaccessible to many, especially non-designers. We present a system to facilitate easy interpretation and exploration of concept sketches. Starting from crudely specified incomplete geometry, often inconsistent across the different views, we propose a globally-coupled analysis to extract part correspondence and inter-part junction information that best explain the different sketch views. The user can then interactively explore the abstracted object to gain better understanding of the product functions. Our key technical contribution is performing shape analysis without access to any coherent 3D geometric model by reasoning in the space of inter-part relations. We evaluate our system on various concept sketches obtained from popular product design books and websites.

CR Categories: I.3.5 [Computer Graphics]: Three-Dimensional Graphics and Realism—Computational Geometry and Object Modeling

Keywords: concept sketch, product design, part relations, shape analysis, NPR

Links: DL PDF WEB VIDEO DATA CODE

1 Introduction

Creating *concept sketches* that show the form and function of potential designs is an essential part of the product design process. Such sketches typically represent the product from one or more viewpoints to convey important geometric features of the object and its constituent parts. Often, the function of a product involves moving or reconfigurable parts, and in this case, designers usually sketch all the relevant part configurations (e.g., the collapsed and expanded states of a folding chair). Concept sketches serve two primary purposes. Designers often gain a better understanding of the design space for a product by considering how the geometry and functional behavior of its constituent parts can vary, and sketching candidate designs can help reveal what types of variations are interesting or appropriate. More importantly, designers often use concept sketches to communicate ideas to their collaborators (product engineers, marketers, other designers within the same creative team, etc.) as well as external clients. For example, designers at IDEO often discuss ideas with clients on a weekly basis during a project, and concept sketches play an important role in such meetings.

However, interpreting the form and function of an object from static concept sketches can be difficult (see Figure 2). Constructing a mental representation of the product geometry requires viewers to first understand the spatial relationships between the viewpoints of multiple sketches and then establish correspondences between parts across the views. This is especially difficult for products with moving parts, where both the viewpoint and part configuration can change between two sketches. Furthermore, since the relative positions and orientations of parts can vary, viewers must interpret how parts are connected to each other and how they move between

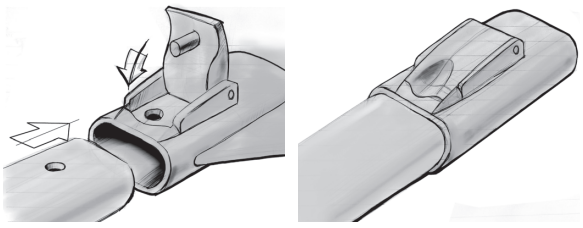


Figure 2: *Concept sketch illustrating stages of a lock mechanism.*

configurations. While some designers may have enough experience reading concept sketches to overcome these challenges, clients of design firms, engineers, and marketers often find it difficult to understand how a proposed design works from a set of sketches.

In this work, we present an interactive tool that helps designers create concept sketches that better convey the form and function of a product design. The input to our system is a concept sketch that includes drawings of the product in different configurations, possibly from multiple viewpoints. Using our tool, the designer then provides a few annotations to indicate the approximate geometry and contact relationships of important parts. Our system uses these annotations to create geometric *proxies* (e.g., cuboids, cylinders, etc.) for the object parts, and based on the relative positions and orientations of these proxies across all the sketches, we automatically infer a simple functional model that explains how the object configuration changes between each pair of drawings. Our system leverages this model to generate animated transitions that help viewers understand how the viewpoint and part configuration change between any two sketches by showing the relevant part proxies smoothly moving from one drawing to the next (see Figure 1). Designers can also use these transitions to create new sketches by scrubbing the animation to an intermediate pose, adjusting the viewpoint if necessary, and then sketching over the proxies. Finally, designers can scribble and add annotations in one sketch and have the system propagate the changes to the other sketches.

A key challenge in analyzing concept sketches is that parts are often drawn in a rough or incomplete manner, and their geometries are typically not consistent across multiple drawings. This makes it difficult to apply many existing computer vision approaches that assume images are projections of a single, fixed 3D representation (see Figure 3). One key insight of our work is that accurate, consistent part geometries are not necessary for producing a useful functional model of a product design from a set of concept sketches. Our analysis allows for variations in the size, shape, and orientation of part proxies across the input sketches, and focus on computing a globally consistent set of functional relationships between parts. In particular, we solve for a set of part junctions (e.g., fixed, hinge, slide, etc.) that explains the configurations of proxies across all sketches and defines how the proxies move from one configuration to another. This analysis also establishes correspondences between proxies across all the drawings.

The main contribution of this work is in demonstrating how we can instrument concept sketches with simple functional models to better convey important characteristics of product designs. We introduce appropriate geometric analysis techniques that automate this "rigging" process these functional models. In the examples presented in the paper, while is possible to manually rig the full motion, our goal is to automate the process. We demonstrate our system using a variety of concept sketches from product design books and websites. Initial feedback from professional designers indicates that our system simplifies the process of conveying and interpreting concept sketches, especially their functionality in terms of part movements.

Contributions. In summary, our main contributions include:

- an interactive system to facilitate interpretation of concept sketches of man-made objects;
- recovering relations and degrees of freedom of object parts from rough and inconsistent initial geometry; and
- introducing tools to help create variations of existing concept sketches by providing proxy-based context information, without requiring to commit to specific geometric details.

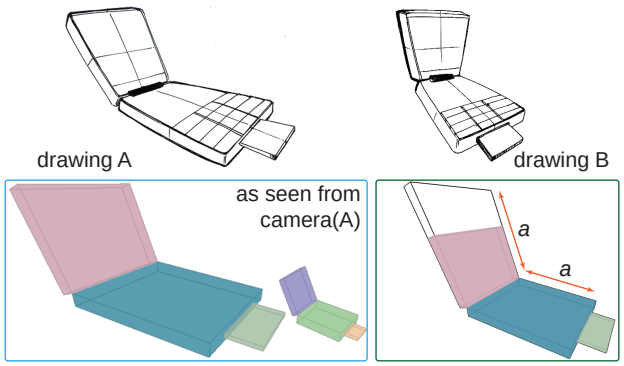


Figure 3: *Input drawings (top) being sketches rather than actual photographs of physical objects can have very different geometries. Based on user annotations we recover proxy geometry and camera models for the respective drawings. (Bottom-left) We observe a large discrepancy of size when both proxy sets are seen from the same camera; (bottom-right) even in the same view, part ratios can differ — making direct geometry-based shape analysis challenging.*

2 Related Work

Shape analysis. In geometric shape analysis, different methods have been proposed in the context of manmade objects, including symmetry detection [Mitra et al. 2012]; etc. The extracted information can then be used for intuitive model manipulation and creation. For example, manipulating man-made objects [Gal et al. 2009]; joint-aware deformations of humanoid objects [Xu et al. 2009]; or structure-aware manipulation in general [Mitra et al. 2013a]. In two related efforts, Mitra et al. [2013b] perform motion analysis of polygonal models of mechanical assemblies to understand and visualize how the assemblies work; while Zhu et al. [2011] analyze 2.5D sketches and perform hydraulic simulation to allow users to interactively edit and illustrate fluid flows.

Our goal is also to understand modes of working of man-made models. However, we neither have access to any consistent 3D geometry, nor can we rely on the accuracy of the sketches. Hence, the challenge is to robustly infer geometric relations with access to only sparse, imprecise, and possibly inconsistent input.

Thinking with sketches. Suwa and Tversky [2009] investigate how designers first consider general object configurations before committing to particular shape of geometries. They discuss the importance of extending sketches in time and space (i.e., in our context, changes in viewpoint and changes in object configurations) to better convey temporal events. Further, sketches are essential to communicate ideas and collaborate with others [Heiser et al. 2004]. However, for such a work-mode to be successful, sketches should be easy to understand and interpret across people with different skills and background (e.g., designer, engineers, clients).

Cuboid based scene understanding. In the context of man-made environments, researchers have demonstrated use of 3D cuboids for scene understanding from single images. This has led to interesting application in context-aware recognition and image-based modeling in computer vision and object-level image editing ([Gupta et al. 2011] and references therein). In order to improve fitting accuracy, Fidler et al. [2012] learn a statistical deformable cuboid model to improve precision. Zheng et al. [2012] propose an interactive algorithm for cuboid-based image understanding for indoor scene images via joint image- and screen-space optimization.

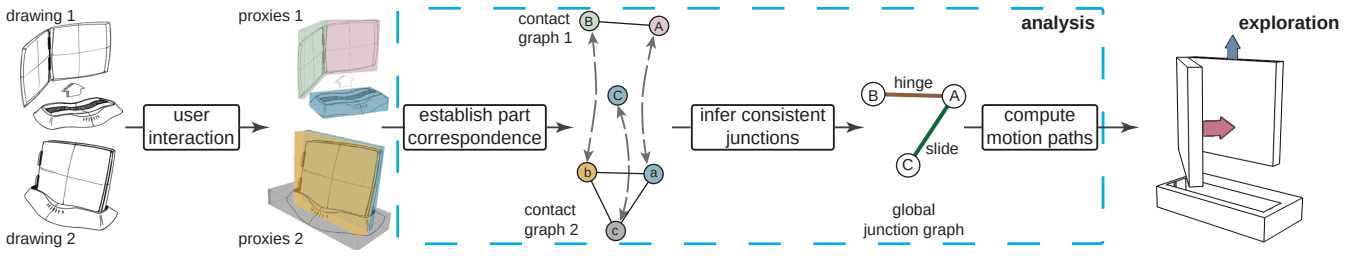


Figure 4: *Pipeline stages. After the user specifies rough proxies in each drawing (i.e., view), we establish correspondence across the parts, infer respective joint types and attributes, and recover valid motion paths to interpolate the poses. Although the junction types are globally consistent, each view has its own geometry and camera information, i.e., no consistent geometry information is stored across the views.*

In contrast, our focus is to interpret sketches of product designs, where unlike in photographs, there is often no real underlying 3D object. Specifically, geometry depicted in the different sketches can vary significantly, thus preventing extraction of a consistent 3D geometry across the views (see Figure 3).

Sketch-based modeling. Sketch-based modeling interprets user drawn 2D sketches to recover corresponding lines and curves in 3D space. Popular approaches include combining sketching with 3D modeling to produce precise manmade constraint-based models [Zeleznik et al. 1996]; creating 3D models from 2D sketches via extrusion [Igarashi et al. 1999]; inferring shapes from sketches based on cusps and T-junctions [Karpenko and Hughes 2006]. In other approaches, single view sketching interfaces have been proposed to create 3D polyhedrons ([Chen et al. 2008; Lee et al. 2008] and references therein) based on simple rules and optimization techniques to derive 3D positions for the drawn 2D vertices. More recently, ILoveSketch [Bae et al. 2008] supports sketching of curved edges by exploiting symmetry information or two-view epipolar geometry. In another notable, Schmidt et al. [2009] propose an analytic drawing method to lift 2D curved sketches to 3D curves using the geometry constraints derived from 3D scaffold guidelines.

Our goal is different. Instead of accurately modeling 3D geometry, we focus on the relations among parts, their connections, and how the object can change its configuration. Thus, sketching is only used to create crude per-view geometric primitives to ease subsequent structure analysis and visualization.

3 Characteristics of Concept Sketches

By consulting instructional books on design sketching [Haller and Cullen 2003; Steur and Eissen 2007] and analyzing many example sketches (e.g., by Michael DiTullo, Carl Liu, Steve McDonald; and websites *sketch-a-day*), we identified several characteristics of concept sketches that inform the design of our interactive system.

Rough geometry. In many sketches, designers approximate the form of object parts using simple shapes, such as cuboids, cylinders or spheres. Furthermore, sketches often include the constructions lines for these shapes as a way of emphasizing the underlying part geometry. Our system allows designers to quickly create approximate 3D representations of parts using simple annotations that often align with intersecting construction lines.

Consistent part relationships. Although part geometry is often specified in a rough manner, designers usually take care to accurately depict the spatial relationships between touching parts. For example, even though the top and bottom pieces of the clam shell phone design in Figure 3 are drawn as simple flat cuboids, the alignment of those cuboids clearly indicates how the parts hinge open and closed. Our analysis uses the relative alignment of part proxies to infer the types of joints that connect touching parts.

Multiple configurations and viewpoints. As mentioned earlier, concept sketches for objects with moving parts typically include several drawings of the object in different configurations. Some sketches show a series of step-by-step images that depict how parts move from one configuration to another. In many cases, designers will choose a different viewpoint for each object configuration. Our system jointly analyzes sketches of an object in multiple configurations to determine a combination of joint types and degrees of freedom that are consistent with all drawings.

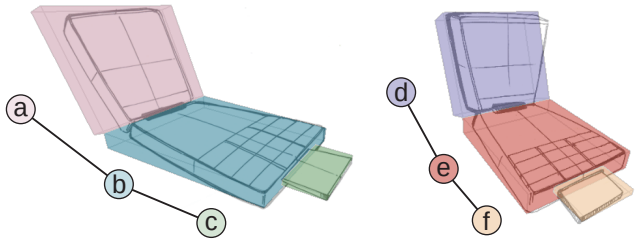
Simple shading. In addition to construction lines, many concept sketches use simple shadows to help convey part geometry and emphasize the spatial relationships between nearby parts. We render our part proxies with similar shading cues.

4 System Overview

In our interactive system (see Figure 4), the user loads a sketch image, indicates the *drawings*, and marks out rough proxy geometries for each drawing. We support cuboids, cylinders, and ruled surfaces as proxy types in our implementation. Note that the sketches are used only during the proxy creation phase, and not in the subsequent stages. The user indicates inter-part contact information in each view. Then, in the *analysis* phase (see Section 5), our system automatically establishes part correspondence across views, infer inter-part connection types (e.g., hinge, telescopic, slide, fixed) and their locations, and extract non-conflicting part movement sequences. The user can manually refine joint positions, if desired. (This mode was not used in the presented examples.) Subsequently, in the *exploration* phase (see Section 7), based on the extracted proxy relations, the user can interactively transition across the drawings, and use the proxies as context to create intermediate sketches and add text or scribble annotations (see supplementary).

5 Analyzing Concept Sketches

Overview. Starting from rough user provided proxy geometry for each drawing, our main goal is to perform the following: (i) establish correspondence across the proxies in the different views; (ii) infer consistent junction attributes for touching proxy pairs that explain all the drawings; and (iii) recover plausible motion paths connecting the different drawings. The goal is ambitious since the input sketches are rough, imprecise, and sometimes even incomplete. We make the key observation that although the sketches are insufficient to globally extract 3D geometry, we can still recover



$\phi(a) = 8.3, \phi(b) = 10.8, \ldots$
$\upsilon(a) = 1, \upsilon(b) = 2, \ldots$
$\eta(a,b) = 0.8, \ldots, \eta(d,e) = 0.7, \ldots$
$\mathbf{x}^L = [0.5\ 0\ 0\ 0\ 0.6\ 0\ 0.2\ 0\ 0.6]$
$\mathbf{x}^\star = [1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1]$

	(a,d)	(a,e)	(a,f)	(b,d)	(b,e)	(b,f)	(c,d)	(c,e)	(c,f)
(a,d)	.08	0	0	0	.61	0	0	0	.61
(a,e)	0	.01	0	.63	0	.33	0	0	0
(a,f)	0	0	.18	0	0	0	0	0	0
(b,d)	0	.63	0	.003	0	0	0	.20	0
(b,e)	.61	0	0	0	.002	0	.52	0	1
(b,f)	0	.33	0	0	0	.005	0	0	0
(c,d)	0	0	0	0	.52	0	0	0	0
(c,e)	0	0	0	.20	0	0	0	0	0
(c,f)	.61	0	0	0	1	0	0	0	.0001

Figure 5: *Starting from the contact-graphs of the drawings, we compute an affinity matrix M for pairwise node correspondences. Based on a spectral analysis of M, we extract a consistent bipartite correspondence assignment $\mathbf{x}^\star$ between the part proxies in the respective drawings (see text for details).*

globally consistent relations between the different parts of the object across the multiple views. Hence, we focus on *recovering relations* (e.g., types of joints) between the object parts, while using crude geometry proxies only to guide the analysis.

5.1 Establish part correspondence

Based on the user inputs, we construct *contact-graphs* $\{G^1, G^2, \ldots\}$ corresponding to each drawing. Specifically, for the i-th drawing, G^i is a graph, where node n^i_j denotes proxy p^i_j and edge e^i_{kl} appears if proxies p^i_k and p^i_l share a contact. Note that proxy geometries can *differ* across the different drawings. Hence, we cannot use existing methods to directly reconstruct consistent object geometry and camera poses from the sketches (see Figure 3).

Later, in Section 6, we explain how the user provides such initial proxy information.

We now bring the individual parts (i.e., contact-graph nodes) into correspondence. However, there are several challenges: (i) geometric information is rough and largely inconsistent across different drawings preventing direct part-level correspondence assignment; (ii) parts can be hidden in certain views; and (iii) junction types between parts in contact are unknown at this stage and thus cannot be used to validate candidate correspondence. Instead of solving the correspondence assignment at a node level, we simultaneously establish correspondence across *all* the nodes across drawing pairs, say (G^1, G^2), using the available contact information as context.

Suppose the contact graphs (G^1, G^2) have no hidden nodes (parts), i.e., they have the same number of nodes, say k. Our goal is to extract a bipartite match between the two set of nodes (c.f., [Leordeanu and Hebert 2005]). We first construct an affinity matrix $M_{k^2 \times k^2}$ between the nodes across the two views where, the columns denote all pairwise assignments of the form $n^1_i \rightarrow n^2_j$ for all $n^1_i \in G^1$ and $n^2_j \in G^2$. Say, column ij denoted as a_{ij} indicates $n^1_i \rightarrow n^2_j$. We use $\mathbf{x}$ to denote an assignment with $\mathbf{x}_i = 1$ (or 0) indicating that the i-th assignment is selected (or not).

The diagonal entries $M(a_{ij}, a_{ij})$ capture the similarity between nodes n^1_i and n^2_j across the two drawings. If their proxy types mismatch, we assign $M(a_{ij}, a_{ij}) \leftarrow 0$. Otherwise, $M(a_{ij}, a_{ij}) \leftarrow \exp(-|\phi(n^1_i) - \phi(n^2_j)| - |\upsilon(n^1_i) - \upsilon(n^2_j)|)$, where $\phi(.)$ denotes the ratio of two parameters of the corresponding proxy (e.g., for cuboids, ratio of longest to shortest edge; for cylinders, length to radius; for ruled surfaces, curve length to edge length) and $\upsilon(.)$ denotes the valence of the corresponding node in the respective contact graph. Note that we compare relative ratios as the absolute parameters across views for corresponding nodes can differ significantly. We found that adding a sketch-matching cost does not help, since the views have inconsistent shading and hatching lines.

The diagonal entries, by themselves, can easily lead to wrong assignments. For example, in Figure 5, a greedy assignment will select $a \rightarrow f$ (based on the maximum among $M(1,1)$, $M(2,2)$, $M(3,3)$). Instead, we consider the key non-diagonal entries. Any such entry $M(a_{ij}, a_{kl})$ captures the *joint* correspondence possibility of $n^1_i \rightarrow n^2_j$ and $n^1_k \rightarrow n^2_l$. If any of the corresponding proxy types contradict, or if the shortest hop-distance from the nodes in the respective contact-graphs mismatch, we set $M(a_{ij}, a_{kl}) \leftarrow 0$. Else, if the corresponding nodes in the contact-graphs are connected by single edges (i.e., 1 hop), we assign an affinity cost $M(a_{ij}, a_{kl}) \leftarrow \exp(-|\eta(n^1_i, n^1_k) - \eta(n^2_j, n^2_l)|)$ where, $\eta(x, y)$ denotes ratio of proxy parameters of parts x and y (adjacent to the common edge connecting proxies $x \leftrightarrow y$). Generalizing to nodes with connecting path lengths longer than one, we sum up the differences along the shortest path, i.e., $M(a_{ij}, a_{kl}) \leftarrow \exp(-\sum_{path} |\eta^1(.,.) - \eta^2(.,.)|)$ (with abuse of notation). Figure 5 shows an example to illustrate some of the terms in the matrix M.

Solving for a consistent node assignment then amounts to obtaining an assignment vector $\mathbf{x}$ such that

$$\underset{\mathbf{x}}{\operatorname{argmax}}\, \mathbf{x}^T M \mathbf{x} \quad \text{with} \quad \mathbf{x}_i \in \{0, 1\}. \tag{1}$$

Since M is symmetric with non-negative entries, we extract a solution based on the dominant eigenvector $\mathbf{x}^L$ of M by relaxing the requirement $\mathbf{x}_i \in \{0, 1\}$ and instead normalize using $\|\mathbf{x}\| = 1$. In order to convert the continuous solution to a binary assignment, we use a greedy strategy similar to that proposed by Leordeanu et al. [2005]. Specifically, at any stage, we pick the largest entry in $\mathbf{x}^L$ (i.e., set a node pair correspondence), remove all the elements from $\mathbf{x}^L$ that conflict with the current assignment(s), and then proceed to pick the next largest assignment among the remaining ones until all the nodes in G^1 have an unique assignment in G^2, say $\mathbf{x}^\star$. Finally, we take $s(1,2) := (\mathbf{x}^\star)^T M \mathbf{x}^\star$ as the score for node correspondence between G^1 and G^2 using the final assignment vector. Note that in the greedy relaxation, by removing conflicts with the current (partial) assignment, only bipartite matches are explored.

Handling hidden nodes. We can account for hidden nodes as long as the visible nodes have sufficient information to correctly resolve ambiguity. Suppose, for a drawing pair between the contact-graphs G^1 and G^2, there is a hidden node in G^2. We create a dummy node n^2_ϵ and construct an affinity matrix while making suitable updates. We set the diagonal terms in M involving n^2_ϵ to a fixed cost (0.1 in our tests). For non-diagonal terms $M(a_{ij}, a_{kl})$ involving only visible nodes, we compute affinity score as before (Note that we do not consider potential shortcuts via hidden nodes). For term like $M(a_{ij}, a_{k\epsilon})$, we set $M(a_{ij}, a_{k\epsilon}) = 0$ if n^1_i and n^2_j have different types; else we set $M(a_{ij}, a_{k\epsilon})$ to a constant cost (0.1 in our tests). We then extract a bipartite assignment as before.

This mode was used for the folding camera example (see Section 8 and Figure 11). In case of more complex situation, the approach can fail, in which case the user has to intervene.

Building a global graph. We compute the pairwise contact-graph node correspondence for all the pairs (G^i, G^j) along with their assignment scores $s(i,j)$. We construct a meta-graph with the graphs G^i as nodes and $\exp(-s(i,j))$ as edge costs to extract a consistent node assignment across the different views using its minimum spanning tree. A global contact-graph $G^\star$ is created where each group of corresponding nodes across the drawing gets assigned as a node. We add contact edges between any two nodes if a corresponding edge exists in any of the original contact-graphs.

5.2 Infer consistent junctions

Having extracted the global contact-graph, in a key stage, we assign junction type for each of the contact edges that best explains all the drawings. We handle the following commonly occurring junction types in our system (see Figure 6): fixed, hinge, slide (or telescopic) where each type has at most one degree-of-freedom (DOF). Specifically, *fixed* junction has no DOF; *hinge* has only one DOF about the axis of rotation; and *slide* also has only one DOF in the direction of sliding. Further, we observe that the axis of rotation or sliding is commonly related to symmetry information of the underlying proxies, e.g., plane of reflection for cuboids, or axis of rotation for cylinders (see also [Mitra et al. 2013b]). Note that if the same junction is explained equally well by a fixed joint and any other junction type, we give priority to the fixed joint.

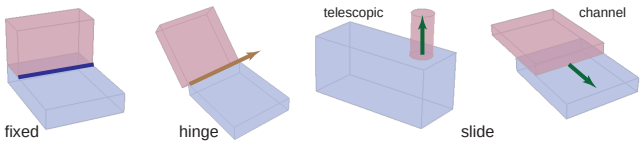


Figure 6: *Different types of junction types handled by our system. Fixed has no degree of freedom (DOF), while each of the others has 1 DOF. Telescopic and channel are both instances of slide joints and depend on the corresponding primitives types (i.e., cuboid vs. cylinder). If case of a tie, we give preference to fixed joints.*

A simple solution to determine junction types can be: for each pair of proxies in contact (i.e., edge in the global graph), we find different poses (i.e., views or drawings) with the known correspondence of the respective nodes and find the junction type that best explains *all* the views. However, there are challenges that makes such a greedy assignment unsuitable: (i) proxy and camera information in each view are only approximate resulting in even correct junction assignment getting a poor alignment score; and (ii) nodes and/or junctions can be hidden (hence corresponding junction edge information will be missing) resulting in no direct information for the corresponding contacts.

Our main observation is the following: Suppose proxies P_a, P_b, and P_c share contact such that P_a is in contact with P_b; P_b is in contact with P_c; but P_a and P_c are not in direct contact. Now, even if (junction of) P_b is hidden or missing in any drawing, information about the *relative* pose of P_a and P_c still provide valuable information about both joints $P_a \leftrightarrow P_b$ and $P_b \leftrightarrow P_c$. In other words, even if proxy P_b is missing, we can jointly reason about the invisible joints $P_a \leftrightarrow P_b$ and $P_b \leftrightarrow P_c$ based on the visible poses of P_a and P_c. Hence, we assign the junction types using a globally-coupled formulation.

Based on the extracted global contact-graph, we first build a multi-connection graph (see Figure 7) by explicitly listing all the types of connection edges. Specifically, we replace each contact edge $e_{ij} \in G^\star$ by a set of connection edges explicitly denoting all types

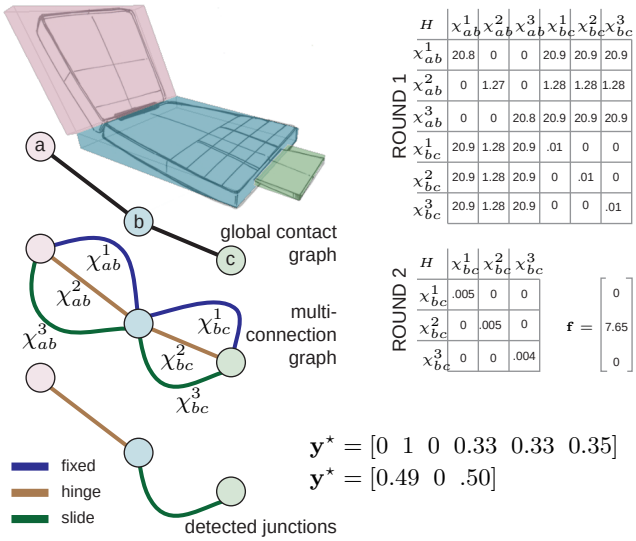


Figure 7: *Starting from the extracted node correspondence and the global-contact graph, we formulate a quadratic program to extract a consistent set of junctions to explain all the drawings (see Figure 5). The assignments happen in rounds, with $\mathbf{y}^\star$ showing the two rounds in this example. See text for details.*

of junction edges as $e_{ij} \rightarrow \{e_{ij}^1, \ldots e_{ij}^k\}$ where, for example, e_{ij}^1 denotes fixed junction, e_{ij}^2 denotes hinge junction, etc. We assign indicator variables of the form $\chi_{ij}^k \in \{0,1\}$ to denote if the edge type e_{ij}^k gets selected ($\chi_{ij}^k = 1$) or not ($\chi_{ij}^k = 0$). Since we expect any contact to have a unique junction type, we require

$$\sum_k \chi_{ij}^k = 1 \qquad \forall e_{ij} \in G^\star. \tag{2}$$

Now, suppose $E(\chi_{ij}^k)$ denotes the penalty of e_{ij} being classified as the k-th type junction; and $E(\chi_{ij}^k, \chi_{pq}^l)$ denotes the penalty of e_{ij} being classified as the k-th type junction AND e_{pq} being classified as the l-th type junction. (Later, we elaborate these terms.) Thus our problem amounts to finding assignments such that:

$$\underset{\{\chi_{ij}^k\}}{\operatorname{argmin}} \sum_{k, e_{ij} \in G^\star} E(\chi_{ij}^k) + \sum_{k,l; e_{ij}, e_{jq} \in G^\star} E(\chi_{ij}^k, \chi_{jq}^l) \tag{3}$$

subject to the linear constraints in Equation 2. We first explain the penalty terms.

Penalty terms. We calculate single edge penalty term $E(\chi_{ij}^k)$ between two proxies P_i and P_j sharing a direct contact (i.e., by e_{ij}). Since P_i and P_j appear across multiple views, we choose to compute the single edge term by aligning the relative orientation and position of P_i to P_j across all drawings (i.e., views) according to its junction type assignment (i.e., fixed, hinge, etc.). For example, with $k = 2$ we test how well a hinge joint explains the relative pose of P_i and P_j across all the drawings by optimizing over the 1DOF hinge movement. We formulate this as:

$$E(\chi_{ij}^k) := \min_\mu \sum_l \left(\|\mathbf{R}^l(\mu) - \mathbf{R}^0\|_2 + \|\mathbf{t}^l(\mu) - \mathbf{t}^0\|_2 \right) \tag{4}$$

where, $\mathbf{R}$ represents the relative orientation between P_i and P_j, $\mathbf{t}$ represents the relative translation of the respective proxy center (normalized using available correspondence information), l iterates over all the views containing P_i and P_j, and we arbitrarily fix one

view as the comparison basis (as view 0). The relative orientation is represented by a 3×3 matrix and its difference is calculated by Frobenius norm. The set of joint parameters μ varies for different joint types: for a hinge joint, its parameters contain the rotation axis and position; for a sliding joint, an axis and the slide parameter. Note that we assume that hinges are at proxy edges; while slide directions are aligned to proxy symmetry planes. All the terms are computed in the coordinate system of the 0-th drawing. For each junction type assignment in the single edge term, we optimize the objective function in Equation 4 as its final value.

The pair-edge penalty term $E(\chi_{ij}^k, \chi_{jq}^l)$ captures the penalty that junction e_{ij} gets assigned type-k AND junction e_{jq} gets assigned type-l. If node P_j is visible in all views, we set $E(\chi_{ij}^k, \chi_{jq}^l) \leftarrow E(\chi_{ij}^k) + E(\chi_{jq}^l)$. If P_j is hidden but P_i and P_q are visible in any l-th view, then we use inverse kinematics (IK) on the corresponding proxies in a view where all three are visible to guess the position of P_j in the l-th view using the visible P_i and P_k as the end effectors. Essentially, we use IK to 'move' the proxies across views to guess the position of any missing proxy. If the end proxies (i.e., P_i or P_q) are invisible, we simply set the corresponding penalty term to zero.

Optimizing Equation 3 then amounts to

$$\mathbf{y}^\star := \underset{\mathbf{y}}{\operatorname{argmin}} \frac{1}{2}\mathbf{y}^T H\mathbf{y} + \mathbf{f}^T\mathbf{y} \quad \text{s.t. } \mathbf{y}(i) \geq 0 \qquad \forall i. \tag{5}$$

subject to corresponding *linear* constraints in Equation 2 on $\mathbf{y}$. We relax the problem by allowing elements in $\mathbf{y}$ to take continuous values and solve the minimization using quadratic programming (QP) to obtain $\mathbf{y}^\star$ (using Matlab *quadprog* function). Note that in the first round $\mathbf{f} = \mathbf{0}$.

We convert the solution to a valid assignment as follows: The highest component, i.e., $\operatorname{argmax}_i \mathbf{y}^\star(i)$ is selected and the conflicting entries from the assignment vector are removed. We then update matrix H and $\mathbf{f}$ by setting the assigned element in $\mathbf{y}$ to 1, and thus reducing the size of $\mathbf{y}$ to the number of existing unknowns. We repeat the process until all the junctions are assigned. For example, in Figure 7, first $\mathbf{y}^\star(2)$ gets selected, i.e., a hinge junction is assigned between the pink and blue proxies. Then, we fix junction ab resulting in $\mathbf{y} \leftarrow [0\ 1\ 0\ \chi_{bc}^1\ \chi_{bc}^2\ \chi_{bc}^3]$ and update H and $\mathbf{f}$ as shown in the figure. In practice, since the number of proxies is small, the QP approximation can be replaced by an exhaustive enumeration of the possible assignments for a more exact result.

The extracted global assignment can still have loops in the final graph (e.g., in the case of the folding camera in Figure 11). We compute minimal spanning tree (MST) of the final graph to break any loops using the corresponding per-edge costs as edge weights. Thus, at the end of this stage, we have determined the type of each junction between different proxy pairs. Note that although each drawing has its own set of geometry parameters (i.e., proxy size) and camera, all the views have a consistent set of junction types assigned to the corresponding proxy pairs.

Determine joint parameters. We already recover the junction locations and the corresponding edges in the above formulation. Further, we obtain the junction parameters (e.g., rotation angle or magnitude for sliding) for each drawing, which are later used in the motion interpolation stage.

5.3 Compute motion paths

At this stage, we have global junction-graph $G^\star$ with corresponding nodes identified in each view and consistent junction types (and respective parameters) identified across the different drawings. Given any view pair, we look at the corresponding graphs G^i and G^j. We first test if linearly interpolating junction parameters (recall we have only 1 DOF moving junctions), usually one junction at time, is collision free. (We discretize the path in 10 steps.) Since the proxy sizes are different across views, we also linearly interpolate their respective sizes for intermediate poses weighted by relative distances from the end configurations. If we detect any collision, we further refine the interpolation space as follows. We assume that CSG, if any, have already been applied to the proxies (see Section 6).

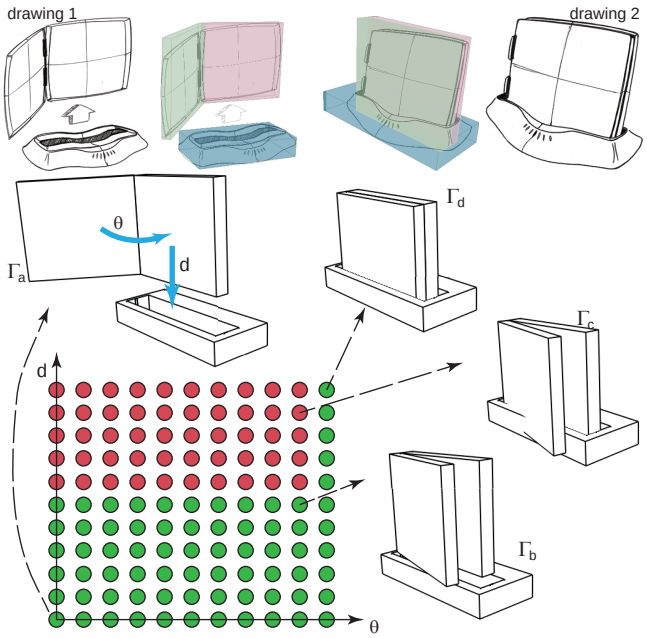


Figure 8: *We sample the configuration space to compute collision free motion paths. In this figure, red points denote self-collision, while green denotes collision-free states. Since directly interpolating junction parameters $\Gamma_a \rightarrow \Gamma_d$ goes through collision states (e.g., Γ^c), we interpolate a motion sequence using non-colliding states as $\Gamma_a \rightarrow \Gamma_b \rightarrow \Gamma_d$.*

Say, Γ^i and Γ^j denote the vector of junction parameters (e.g., rotation angles for hinge motion, distance traveled for sliding motion) for view graphs G^i and G^j (number of dimensions of Γ is equal to the number of junctions). We discretize the space of intermediate junction parameters (using 10 steps only along the components where Γ^i and Γ^j differ) and test for intersections among the proxies for the intermediate configurations. We then mark these configuration space points as *colliding* or *not-colliding*, and simply select a path (e.g., shortest path) in this configuration space going through only *non-colliding* states. In order to detect collision (or not), we use the approximate proxies. However, the proxies in different drawings have different sizes, and hence for any configuration Γ we interpolate the proxy dimensions from the end positions (drawings) weighted by $\|\Gamma^i - \Gamma\|$ and $\|\Gamma^j - \Gamma\|$, respectively. For example, in Figure 8, the proxy geometry of Γ_b is interpolated from Γ_a and Γ_d. We found this simple approximation along with basic motion planning to work well in our examples.

6 User interaction

The user starts by indicating the different drawings. Then, for each drawing, she over-sketches to provide rough proxy geometries (cuboids, cylinders, and ruled surfaces). For cuboids, the user

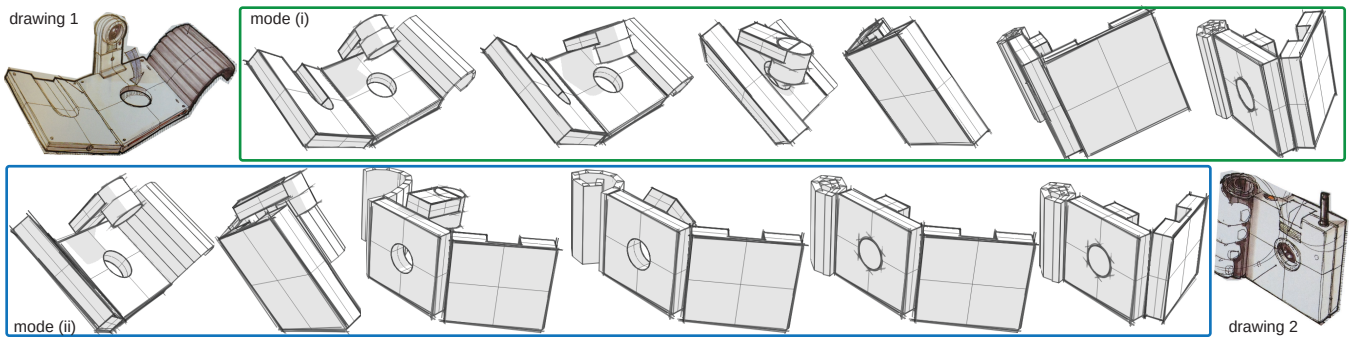


Figure 9: *Given two drawings, we support two modes to automatically generate animated transitions: (i) simultaneously interpolate camera, proxy geometry, and junction attributes; or (ii) first interpolate camera and proxy geometry, followed by interpolation of junction attributes.*

marks the corner points; for cylinders, a diameter line and an axis line; and for ruled-surfaces, a guiding curve and the swept edge. Note that the user implicitly specifies which parts (i.e., proxies) are in contact by indicating shared edges between proxy pairs (see supplementary video). We use the first cuboid's corners to calibrate the camera [Hartley and Zisserman 2006], and use the camera information to recover subsequent proxies. The user indicates when switching from one drawing to another, thus implicitly grouping the proxies for the respective drawings. While the geometries can be very rough, the key information is which proxies are in contact. In case of models with separated parts, the user can mark two proxies as being coplanar (e.g., in Figure 8, in drawing 1 the user indicated the blue and pink proxies to be coplanar). Thus, unlike typical sketch-based modeling systems, we allow proxy geometries to be *inconsistent* across multiple views. (Note that while it is conceivable to vectorize the input sketch and then try to extract proxy boundaries, we decided in favor of a simpler user-guided system as robustly classifying the sketch lines can be ill-posed due to shadows, shading, hatching lines, etc.)

Hidden node placement. After this initial per-drawing proxy creation, our system proceeds automatically to extract part correspondence across the views, and infer consistent joint information across the different views (Sections 5.1 and 5.2). Although we can automatically detect correspondence across (limited) hidden nodes and associated junctions, the user later has to assist in positioning the hidden proxy. Specifically, the system fills in the hidden part by transferring the corresponding proxy from another view and placing it in the relevant view with default joint attributes. The user then simply adjusts the part position (using up to 1 DOF freedom of the joint) and adjusts the proxy size (since the geometric attributes can be different). See the supplementary video for an example.

CSG operation. The user can use simple CSG operations to refine the proxies. Specifically, in any drawing, the user can select initial proxy pairs and apply simple CSG operations (e.g., subtract). We convert the CSG model to a 3D mesh, and transfer to the other drawings using anisotropic scaling to match original proxy sizes (since sizes can differ across the drawings). For example, in Figure 8, the user applied a CSG operation in the second drawing to refine the geometry of the base, which was then transferred over to the first drawing proxy.

7 Exploring Concept Sketches

In this section, we use the information extracted in the analysis stage to assist the user to transition across the drawings, interactively inspect the object by reposing it in novel configurations and views, and add annotations to the drawings — thus helping the user to create a mental representation of the product.

Animated transitions. The input drawings can be difficult to interpret as they can have large camera motion and pose changes making it challenging to mentally recreate the path. For example, the folding camera example in Figure 9 undergoes significant view change while the camera gets folded and the flash tucked in. In order to create better previews, we generate two modes of animated transitions: (i) we simultaneously interpolate camera, proxy geometry, and junction attributes; and (ii) we first interpolate camera and proxy geometry, and then interpolate the junction attributes. In case of large camera and object motions, we found the second mode to be more effective, but at the cost of more transition frames. The user can also directly prescribe a camera view and just interpolate the motion. The supplementary viewer shows the different modes.

Note that we center the drawings to prevent the proxies from moving across the screen during the animation, which can be distracting, making it difficult to follow the part motions. We linearly interpolate the proxy geometries (i.e., their size), while using quaternions to interpolate the rotation components. Thus, both for camera and the hinge joints, we directly interpolate the angles.

In order to provide context and depth, we render the proxies with simple jittered lines, add cross lines, and add simple shading and shadow as recommended in [Steur and Eissen 2007]. We assume the scene to be lit by a default directional light.

Creating novel poses. The user can also use our sketch viewer to run the animation, stop in between, reposition and repose the object. We provide motion arrows (pink for rotation, and blue for sliding) to indicate the available degrees of freedom (see Figure 10). They are placed using simple rules: (i) arrows are attached to the respective proxy faces and aligned to their symmetry axis/plane; (ii) arrow directions are selected to keep the arrows facing the viewer; and (iii) arrow sizes are determined based on size of corresponding proxies/junctions. For hinge motion, the respective junction edge is selected as the axis of rotation. While we do not account for pairwise arrow occlusion, we believe more advanced methods can be used [Mitra et al. 2013b].

Linked annotations. Our system supports two ways of annotating the drawings: (i) the user may add sketch lines in one drawing, which get associated to a proxy face, and transferred to the other drawings using available proxy correspondence. Since proxies can have different parameters, we use relative coordinates to perform the transfer; and (ii) the user can annotate proxy faces with text labels (e.g., material assignment, part label) and the system propagates the information to other drawings. For the second mode, we locally search (via sampling) the empty regions of the canvas around the tagged proxy face to decide on a placement location (see Figure 10-top). Finally, the user can over-sketch using the rendered proxies to provide context in order to create new drawings.

8 Results

We used our system to analyze concept sketches for eight different product designs (see Figure 11): a folding camera, cellphone, toolbox, printer, hairdryer, toaster oven, tablet, and lock. We asked a designer to create the cellphone and the tablet sketches, and the remaining artworks come from various books on product design [Olofsson and Sjolen 2005; Steur and Eissen 2007] and webpages (e.g., Carl Liu's sketches, sketch-a-day, etc.). The simplest sketch was the toaster oven, which included two drawings from (slightly) different viewpoints showing two parts connected by a single joint in two different configurations. The most complicated sketch was the folding camera, which had three drawings showing the object in several different poses. The number of parts in the designs themselves ranges from two (hairdryer and toaster) to six (printer). For the cellphone, toolbox, tablet and lock, in the respective first viewpoint, the user tagged two proxies as being coplanar since they do not share any direct contact edge. In addition to annotating the sketches to create part proxies, we used CSG operations to cut out the cavities for the camera lens and tablet base, etc. (see Table 1 and supplementary video). We also specified the size and position of one hidden proxy for the camera lens. No other user intervention was required to generate the functional models from the eight sketches.

Table 1: *Statistics for the different examples.*

model	#drawings	#parts	#junctions	user inter.
folding camera	3	5	4	hidden/CSG
cellphone	3	3	2	coplanar
toolbox	2	3	2	coplanar/CSG
printer	3	6	5	CSG
hairdryer	3	2	1	CSG
toaster oven	2	2	1	-
tablet	2	3	2	coplanar
lock	2	3	2	coplanar/CSG

For each sketch, we generated animated transitions between all of the individual subdrawings. Figure 11 shows snapshots from one animated transition for each product; for more transitions, please refer to our supplemental video and/or the executable of our sketch viewer. The transitions help clarify the relationships in several sketches. For example, Figure 11a makes it clear how the lens, screen, and flexible outer cover fold together as the camera collapses into its closed position. Furthermore, since the screen looks quite different in the two original sketches (we only see the yellow face of the screen in the collapsed sketch), the transition also helps establish the correspondences between parts. Even for relatively simple objects like the cellphone in Figure 11b, it may not be clear which end of the phone is closer to the camera (top or bottom?) in the final sketch. Here, the transition confirms that we are looking at the bottom of the phone and that the black slit depicts the memory card slot. For sketches with more moving parts, like the printer (Figure 11d), our functional model conveys how different parts move. In this case, we can quickly see that the two paper trays at the bottom slide out from the main compartment, while the panel at the back hinges open to allow access to paper jams.

To get some informal feedback on our system, we showed our interactive tool to three professional designers, including one current and one former IDEO employee. All the designers felt that our animated transitions would "facilitate communication" and help viewers interpret sketches to better understand the functional characteristics of the depicted design. They also agreed that our visualizations would likely be most useful for non-designers and non-engineers (e.g., marketers, clients). In terms of the authoring workflow, the designers felt that the amount of work required to annotate sketches was very reasonable, especially compared to the time and

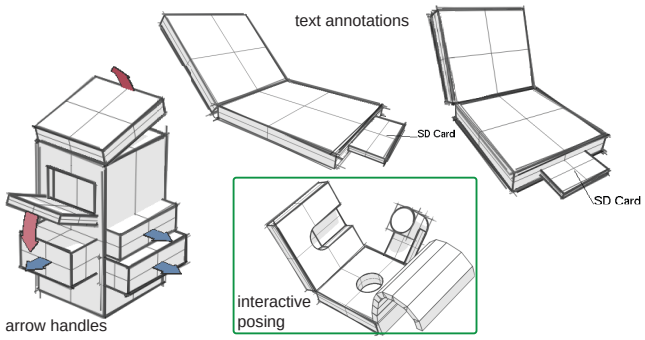


Figure 10: *In the exploration stage, our system supports: (i) coupled annotations, i.e., user marks in one drawing that then gets transferred and automatically placed based on the underlying proxy correspondence; (ii) arrow handle based changing of part poses; and (iii) interactive view and pose changes.*

effort required to create the original sketches. One designer said that our authoring workflow seemed "a million times faster than doing this [modeling] in SolidWorks," which is what he sometimes does when he wants to create a functional prototype to show others.

While the feedback was predominantly positive, there were a few suggestions for improvement as well. One designer wondered how effective our proxies would be for representing objects with more complex or organic shapes. For such shapes, it is possible that cuboids, cylinders and ruled surfaces could provide enough geometric information to convey the overall configuration of parts and how they move, but it would be interesting to validate this hypothesis with perceptual experiments. Another designer questioned whether the current system could be useful for creating functional models from extremely rough sketches during the early, brainstorming stages of the design process. Despite the fact that our annotation workflow is already quite straightforward, it is likely still too much overhead for this use case (e.g., ideally, annotating a 10-second sketch would take just a few additional seconds). Conducting perceptual experiments and further automating the annotation workflow are both interesting areas for future work. Note that in our current formulation we only handle junctions with up to 1 DOF, but in the future we will like to explore more junction types (e.g., combination of hinge and slide). This will possibly require revisiting the formulation in Equation 3.

We will further like to discuss the following limitations: (i) In certain situations, there can be insufficient information to assign an unique bipartite correspondence, but there can be multiple very comparable assignments. For example, in the tablet example, there was another assignment that got very close node correspondence scores, and just luckily the desired one was selected. In such situations, we believe allowing the user to select among the top assignment suggestions will be appropriate. (ii) By focusing on relations, we demonstrated robustness to relatively large variations in proxy geometries, our algorithm does break down when the proxies are very poor or the drawings are less structured (e.g., for napkin sketches). It remains unclear how such ambiguity can be tamed;. Finally, (iii) for complex models, as the part numbers increase leading to more hidden parts, our method can fail due to the ambiguity arising from multiple hidden nodes.

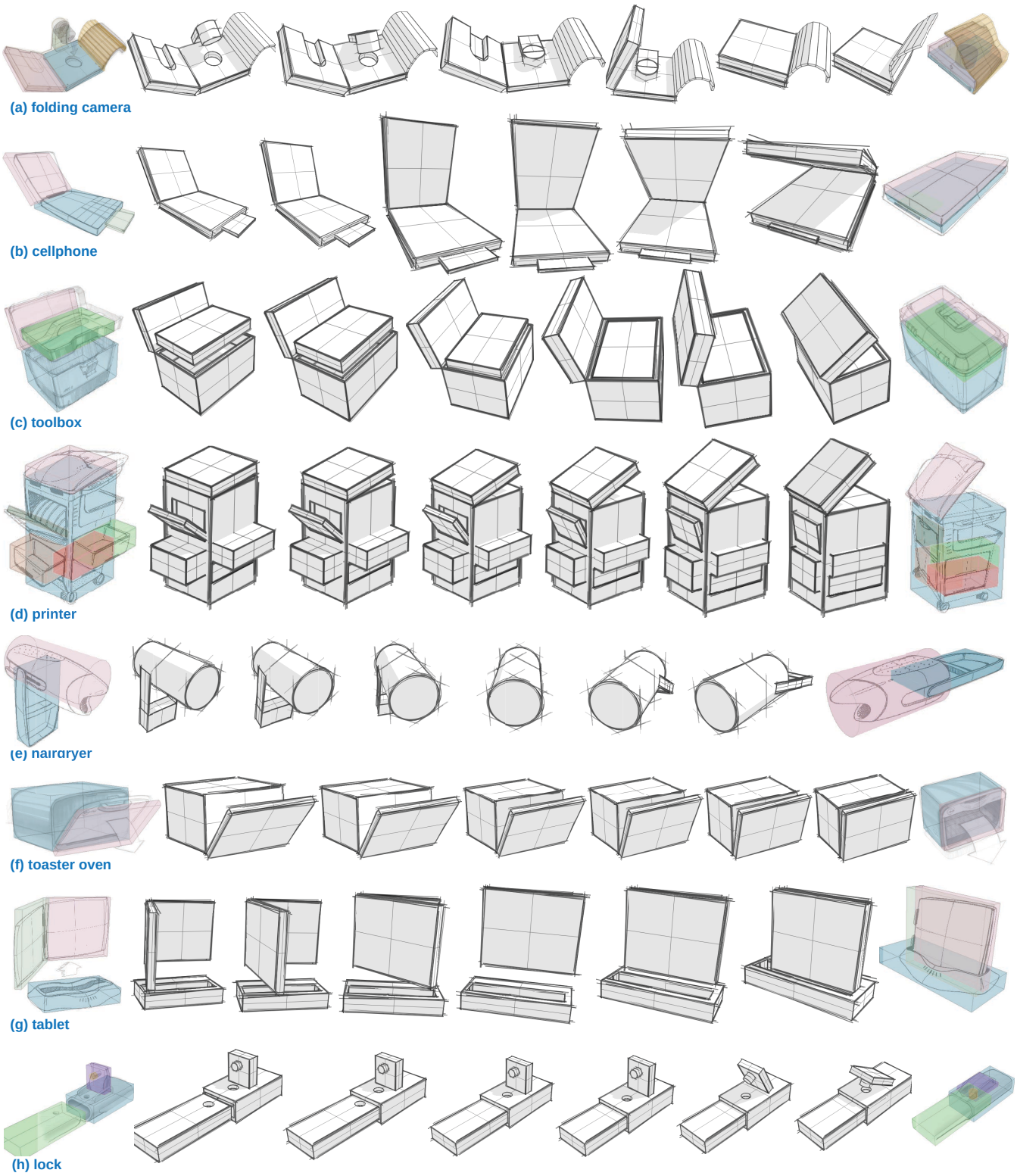


Figure 11: *Animated transitions created automatically produces by our system to facilitate interpretation of the input concept sketches. The initial sketches are used only for guidance while specifying the initial proxies (indicated by colored boxes).*

9 Conclusion and Future Work

We presented an interactive system to facilitate interpretation and exploration of concept sketches. Starting from rough user annotations on a set of drawings, our system builds a set of guiding proxy geometry, and then solves for a globally consistent joint assignment that links the different drawings. We proposed a novel analysis method to solve for such joint assignments by focusing mainly on inter-part relations, with geometry playing only a secondary role. We also presented simple exploration options that use the view specific geometry and camera information, linked by a global contact graph, and allows the user to interactively explore the concept sketches, thus allowing her to better understand the object functions, rather than specifics of its geometric form.

There are several interesting future directions to explore:

Constrained modeling. Man-made objects often have self-similarity and structures, with different parts coupled by non-local constraints [Gal et al. 2009]. For example, in Figure 11a, the two cuboid proxies have similar length and breadth in drawing #2; but do not satisfy the same constraints in the other drawing. In the future, we plan to explore how to link the drawings and also carry the constraints across them. Different drawings, however, can impose potentially conflicting constraints, thus preventing a direct adaptation of techniques from traditional constraint-based modeling.

Sketch-to-fab. In this work, we focused on generating intermediate sketches to facilitate interpretation of a product's functionality. From the extracted consistent global junction-graph, we can possibly extract a coherent 3D geometric model that satisfies the junction types, without necessarily matching the sketched drawings. Such a functional 3D model can then be adapted by the engineer to produce an easy to fabricate model (with joint types, part parameters, etc.), thus simplifying the transition to a prototype 3D model.

Integrated sketching. In the future, we plan to directly integrate our analysis framework into a smart sketching system so that the designer can use the proxy geometries to correctly sketch perspective views and part configurations, reminiscent of analytic drawing using scaffolds [Schmidt et al. 2009].

Acknowledgement. We are grateful to Xiaolong Lou for his help with sketching; Chen Li for his help with the rendering; Cristina Amati for video voiceover; Bongjin Koo, Hung-Ku Chu, and Tao Du for their comments on the initial draft of the paper; and the anonymous reviewers for their suggestions. We are thankful to Carl Liu for the folding camera; Spencer Nugent for the toolbox and toaster oven; and Danian Yang for the lock sketches. The project was partially supported by a China Scholarship Council Fund, Adobe, NSFC grants, 973 program of China, the Open Project Program of the State Key Lab of CAD & CG, and a Marie Curie CIG.

References

BAE, S.-H., BALAKRISHNAN, R., AND SINGH, K. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3D curve models. In *UIST*, 151–160.

CHEN, X., KANG, S. B., XU, Y.-Q., DORSEY, J., AND SHUM, H.-Y. 2008. Sketching reality: Realistic interpretation of architectural designs. *ACM TOG 27*, 2, 11:1–11:15.

FIDLER, S., DICKINSON, S., AND URTASUN, R. 2012. 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In *NIPS*, 620–628.

GAL, R., SORKINE, O., MITRA, N. J., AND COHEN-OR, D. 2009. iwires: an analyze-and-edit approach to shape manipulation. 33:1–33:10.

GUPTA, A., SATKIN, S., EFROS, A. A., AND HEBERT, M. 2011. From 3D scene geometry to human workspace. In *CVPR*, 1961–1968.

HALLER, L., AND CULLEN, C. D. 2003. *Design Secrets: Products 1 and 2: 50 Real-Life Product Design Projects Uncovered*. Rockport Publishers.

HARTLEY, A., AND ZISSERMAN, A. 2006. *Multiple view geometry in computer vision (2. Ed.)*. Cambridge University Press.

HEISER, J., TVERSKY, B., AND SILVERMAN, M. 2004. *Sketches for and from collaboration*. Sydney: Key Centre for Design.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: a sketching interface for 3d freeform design. In *Proc. of SIGGRAPH*, 409–416.

KARPENKO, O. A., AND HUGHES, J. F. 2006. SmoothSketch: 3D free-form shapes from complex sketches. *Proc. ACM SIGGRAPH 25*, 3.

LEE, S., FENG, D., AND GOOCH, B. 2008. Automatic construction of 3D models from architectural line drawings. In *Proc. I3D*, 123–130.

LEORDEANU, M., AND HEBERT, M. 2005. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 1482–1489.

MITRA, N. J., PAULY, M., WAND, M., AND CEYLAN, D. 2012. Symmetry in 3D geometry: Extraction and applications. In *EUROGRAPHICS State-of-the-art Report*.

MITRA, N. J., WAND, M., ZHANG, H., COHEN-OR, D., AND BOKELOH, M. 2013. Structure-aware shape processing. In *EUROGRAPHICS State-of-the-art Report*.

MITRA, N. J., YANG, Y.-L., YAN, D.-M., LI, W., AND AGRAWALA, M. 2013. Illustrating how mechanical assemblies work. *Comm. of the ACM (Research Highlight) 56*, 1, 106–114.

OLOFSSON, E., AND SJOLEN, K. 2005. *Design Sketching*. Keeos Design Books.

SCHMIDT, R., KHAN, A., SINGH, K., AND KURTENBACH, G. 2009. Analytic drawing of 3D scaffolds. *ACM TOG (SIGGRAPH Asia) 28*, 5, 149:1–149:10.

STEUR, R., AND EISSEN, K. 2007. *Sketching: Drawing techniques for product designers*. BIS Publishers.

SUWA, M., AND TVERSKY, B. 2009. *Thinking with sketches*. Oxford University Press.

XU, W., WANG, J., YIN, K., ZHOU, K., VAN DE PANNE, M., CHEN, F., AND GUO, B. 2009. Joint-aware manipulation of deformable models. *Proc. ACM SIGGRAPH 28*, 3, 35:1–35:9.

ZELEZNIK, R. C., HERNDON, K. P., AND HUGHES, J. F. 1996. SKETCH: an interface for sketching 3D scenes. In *Proc. ACM SIGGRAPH*.

ZHENG, Y., CHEN, X., CHENG, M.-M., ZHOU, K., HU, S.-M., AND MITRA, N. J. 2012. Interactive images: cuboid proxies for smart image manipulation. 99:1–99:11.

ZHU, B., IWATA, M., HARAGUCHI, R., ASHIHARA, T., UMETANI, N., IGARASHI, T., AND NAKAZAWA, K. 2011. Sketch-based dynamic illustration of fluid systems. In *ACM TOG (SIGGRAPH Asia)*, 134:1–134:8.