An Interactive Approach for Functional Prototype Recovery from a Single RGBD Image

Anonymous cvm submission

Paper ID 0041

Abstract

Inferring the functionality of an object from a single RGBD image is hard. The difficulties are two-fold: the lack of semantic information of the image object; and the missing data due to occlusion. In this paper, we present an interactive framework to recover the 3D functional prototype from a single RGBD image. Instead of precisely reconstructing the object geometry for the prototype, we focus more on recovering the object functionality along with their geometry. Essentially, our system allows users to scribble on the image to create initial rough proxies for the parts. Then after the user annotation of high-level relations among parts, our system automatically optimizes the detailed junction parameters (axis & position) and part geometry parameters (size & orientation & position) together. Such recovery of prototype enables a better understanding of the underlying image geometry and allows for further physical plausible manipulation. We demonstrate our framework on various indoor scene objects with simple or hybrid functions.

1. Introduction

That form ever follows function. This is the law.

Louis Sullivan

With the popularization of commercial RGBD cameras such as Microsoft's Kinect, people can easily acquire 3D geometry information for the RGB image. However, due to occlusion and noise, recovering meaningful 3D contents from single RGBD images remains one of the most challenging problems in computer vision and computer graphics research.

Over the past years, many researches have been devoted to recovering high-quality 3D information from RGBD images [9, 7]. Most of these approaches, starting either from a single image or multiple images, are dedicated to recovering the faithful 3D geometry of image objects, regardless of their semantic relations, underlying physical settings, or even functionality. In recent, researches have been developed to explore high-level structural information to facilitate 3D reconstruction [26, 19, 18]. For example, Shao *et al*. [18] leverage physical stability to hallucinate the interactions among images objects and obtain physically plausible reconstruction of objects in RGBD images. Such high-level semantic information plays an important role in constraining the underlying geometric structure.

Functionality is to the central of object design and understanding. Objects in man-made environments are often designed for one or multiple intended functionalities (Figure 1). *That form ever follows function* is the law of physical manufacturing [20]. In this paper, we develop an interactive system to recover functional prototypes from a single RGBD image. Our goal is to allow a novice user to be able to quickly lift the image objects into 3D using 3D prototypes, with just a small amount of high-level annotations of junction types and geometric/functional relations; and meanwhile explicitly explore and manipulate its function. We focus on prototypes with simple proxies (e.g. cuboids) representing parts as a means to alleviate the difficulties in precise 3D reconstruction which is a harder problem. And, by taking into consideration of physical functionality, we could gain a much more faithful interpolation of the underlying objects. The functional properties could further be used for applications such as in-context design and manipulation.

It is a challenging problem to infer object function just from user annotated junction types and geometric/functional relations. Our system should automatically

Figure 1. Objects in man-made environments are often designed for one or multiple intended functionalities.

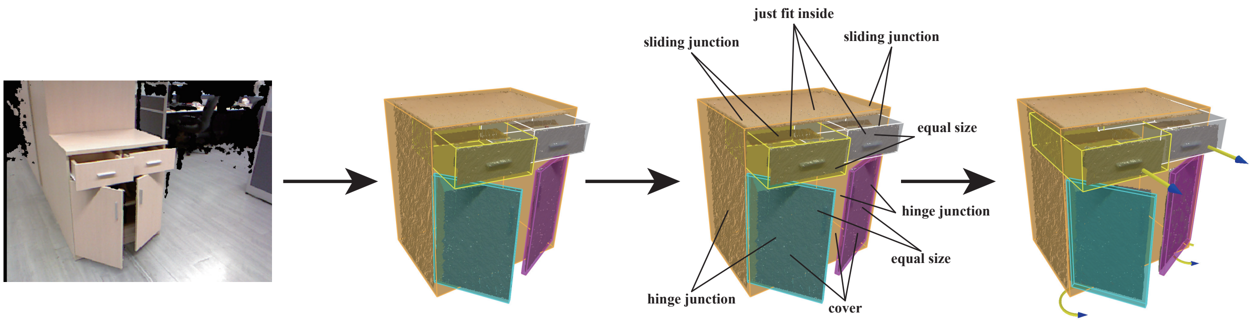

Figure 2. Algorithmic pipeline. Given the input RGBD image (left), our system generates initial proxy cuboids (middle-left) from the parts segmented by user with strokes or polygon tools. Then the user annotates a set of high-level relations among the proxies including junction types and geometric/functional relations (middle-right). Finally our system simultaneously optimizes the junction parameters (axis & position) and the part parameters (orientation, position and size) to get the functional prototype with parts moving as user expected (right).

optimize the detailed junction parameters (axis & position) in order to make the parts move correctly, whereas this task is typically done in CAD softwares by carefully adjusting the parameters by the user. Besides, initial proxies from user-segmented depth is rather rough with incorrect orientation and position, and would be much smaller than real size because of occlusion. Hence initial proxies often fail to satisfy the functional relations such as *A covers B*. Therefore our system should also optimize the proxy parameters (size & orientation & position), in order to make parts satisfy functional relations.

Our method starts with a single RGBD image. We let the user segment the image object into parts by scribbling on the image using simple strokes or polygons. Then each segmented part is assembled with a 3D proxies. We use simple cuboid in this paper [12]. Given the initial proxies, our system then allows the user to annotate the junction types and functional/geometric relations among parts. In a key stage, our algorithm simultaneously optimizes the detailed junction parameters (axis & position) and the proxy parameters (size & orientation & position). Finally, a functional prototype is produced with moving parts satisfying the user annotated relations.

We tested our system on a variety of man-made hybrid functional objects taken from various sources. Our results show that even with only a few user annotations, our algorithm is capable of faithfully inferring geometry along with the functional relations of the object parts. In summary, this paper makes the following contributions:

- identifying and characterizing the problem of integrating functionality into image-based reconstruction;
- simultaneous optimization of detailed junction and geometry parameters from user's high-level annotation of junction types and functional/geometric relations;
- developing an interactive tool for functional annotation, and testing in on a variety of indoor scene images and physical designs.

2. Related Work

Proxy-based analysis. There has been a significant amount of work that leverages proxies to understand objects or scenes. Li *et al.* [14] and Lafarge *et al.* [13] consider global relationships as constraints to optimize initial RANSAC-based proxies to produce structured outputs; similarly, Arikan *et al.* [1] use prior relations plus user annotations to create abstracted geometry. For scene analysis, a lot of approaches encode input scenes as collections of planes, boxes, cylinders, etc. and studying their spatial layout [5, 6, 8, 10, 11, 25]. Recently, proxies were commonly used for functionality analysis of a design. Umetani *et al.* [21] use physical stability and torque limits for guided furniture design in a modeling and synthesis setting. Shao *et al.* [17] create 3D proxy models from a set of concept sketches that depict a product from different viewpoints and with different configurations of moving parts. Koo *et al.* [12] annotate cuboids with high-level functional relationships to fabricate physical works-like prototypes. Different from these approaches, to our knowledge, we are the first to focus on the proxy-based functionality recovery from a single RGBD image, particularly recovering how the object works by jointly optimizing the part geometry along with functional relationships based on user annotations.

Constraint-based modeling. Our work is related to the constraint-based modeling research in the graphics and CAD communities. Similar graphics work involves automatically determining the relevant geometric relationships between parts for high-level editing and synthesis of 3D models [4, 22, 2, 26]. Previous mechanical engineering research introduces declarative methods for specify-

Figure 3. Initial proxy generation. The user is allowed to scribble strokes on the image (left), and based on the scribbles, depth-augmented GrabCut is applied to segment the input object to different parts (middle). Initial cuboids are then fitted to the corresponding points (right).

ing the relevant geometric constraints for a mechanical design [3, 24]. Some professional CAD softwares like AutoCAD and SolidWorks contain constraint-based modeling modules, but the users are required to manually adjust the low-level part/junction parameters to specify the relationships. In contrast, our system can automatically interpret the user annotated high level functionalities into the specific geometric constraints.

3D modeling from single RGBD images. Much effort has been devoted to obtaining high-quality geometry information from a single RGBD image [23, 7]. To recover structure information, Shen *et al*. [19] extract suitable model parts from a database, and compose them to form high quality models from one RGBD image. Shao *et al*. [18] adopt physical stability to recover unseen structures from a single RGBD image using cuboids. However, their techniques focus on creating static 3D geometry and structure, whereas our goal is to produce models with correctly moving parts.

3. Overview

As illustrated in Figure 2, given a single RGBD image, we first let the user scribble strokes over the image objects to cut out functional parts of the object. Those parts, being either a semantic component or an additive object, will finally take place in the function recovery. To segment the parts, we use a depth-augmented version of the GrabCut segmentation [15] similar to [18]. Optionally, if the color and depth are too similar which makes it difficult to separate the parts with GrabCut, we provide a polygon tool like PhotoShop to do segmentation (see in accompanying video). We assemble a set of proxies (cuboids in our case) to fit each individual part. We then let the user annotated the high-level relations among these cuboids. The relations consist of three categories: junction relations (e.g., hinge, sliding), functional relations (e.g., cover, fit inside, support, flush, connect with) [12], and geometric relations (e.g., equal size, symmetry).

Given the user annotated relations, in a key step, our method recovers the cuboid orientation, position and size along with the junction parameters using a joint optimization. We choose the joint optimization strategy because the cuboid parameters are always coupled with the junction parameters. That is, given a set of junctions, the cuboid geometry should change accordingly to satisfy the functional constraints.

The optimization is done using a two-stage sampling strategy. In the first stage, our algorithm samples possible cuboid edges as junction candidates [17] for the specified junction type. Given one set of possible junction candidates, the orientation of the cuboids can be aligned and the positions can be refined by adjusting the corresponding junction edges. We assume that the junction must be snapped to the nearest cuboid face and be parallel to the nearest cuboid edge (as in [12]).

With one set of adjusted junctions and cuboid orientation and position, our method further samples a set of possible candidate rest configurations for the cuboids. A rest configuration is a state where the object is in a *closed* state [12]. Because the cuboid size is not certified yet, the system does not known which state is the *closed* state. Thus we sample possible candidates for the rest configurations, as shown in Figure 6. For each possible rest configuration, we optimize the cuboid size parameters according to the user annotated functional/geometric relations as in [12]. Finally, the optimized cuboids which lead to the minimal difference against the initial point cloud are selected, and the best prototype with best junction and cuboid parameters is produced. We next describe the detailed algorithm.

4. Algorithm

Our method takes as input a RGBD image of a functional object. By *functional* we refer to objects those have particular moving parts, such as rotatable cover, slidable window, etc. Such objects are very commonly seen in our daily life, for instances, rolling chairs, foldable tables, printers, seesaw etc. In addition, such objects populate our man-made environments, especially indoor scenes.

Initial cuboids generation. Given the input RGBD image, our first task is to anchor the object functional parts. Automatically identify image object and object parts in RGBD images has been explored in recent methods, however, without any prior knowledge the performance is still not satisfactory for our purposes. We resort to an interactive solution. As in [18], we let the user to scribble on the image object to specify object parts. In particular, we allow the user to draw free strokes over parts to indicate a segment (part). We perform the depth-augmented GrabCut algorithm [18] to the underlying point cloud along with their pixel and adjacency information. Optionally, if the color and depth are too similar which makes it difficult to separate the parts with GrabCut, we provide a polygon tool like PhotoShop to do segmentation. We then run the Efficient RANSAC algorithm [16] on the selected points to generate

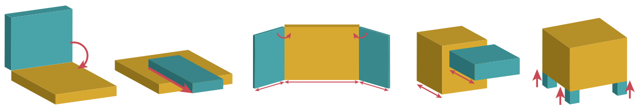

Figure 4. User annotated junction types and some typical functional relations. From left to right: hinge junction, sliding junction, exactly cover, just fit in, and support.

candidate planes. The largest of the planes is selected as the primary plane, and the second largest plane is made orthogonal to the primary one. We extract the initial cuboids determined by these orthogonal directions (third direction is the cross product of the two plane normals). Figure 3 illustrates the process of generating the initial cuboids. Note that the generated cuboids have erroneous orientations, positions and sizes. In the next steps, our goal is to simultaneously optimize these parameters along with the junction parameters so that the extracted cuboids form a prototype whose functionality closely follows the image object.

Relation annotation. Denote the set of initial cuboids as $(B_1, ..., B_N)$, in an important step, we let the user to annotated the high-level relations among cuboids. To this end, we define three categories of relations. Category I is the junction relations (types) (e.g., A has a hinge relation w.r.t. B), and Category II is functional relations (e.g., A covers B) and Category III is geometric relations (e.g., symmetry, equal size, etc.). To specify the Category I relation, the user selects a pair of cuboids and right click a button to indicate a junction type. The same interface is used for Category II and III.

To further classify the relations, we define two main types of junction relations, namely, hinge and sliding. For functional relations, similar to [12], we define the following function types: *A covers B*, *A fits inside B*, *A supports B*, *A is flush with B*, and *A connects with B*. For geometric relations, we mainly use 2 types: symmetry and equal size. These relations pose different geometric constraints on the following optimization stage and some relations might be dependent on each other. For example, if both A and C covers B, A is geometrically constrained w.r.t. B and C. Figure 4 shows the junction types and some typical types of functional relations. Note that unlike the method of [12], we do not need to explicitly specify the junction position and axis as well as cuboid orientations and positions, instead, we optimize these parameters in a joint manner.

Joint optimization of cuboids and junctions. We now detail our cuboid optimization algorithm. Our goal is to jointly optimize the cuboid orientation and their shape parameters (i.e., positions, sizes) as well as the detailed junction parameters according to the user annotated relations. The optimized cuboid configuration should deviate little from the input point cloud and move correctly as user expected. Essentially, given the input point cloud I and initial cuboids $\mathcal{B} = (B_1, ..., B_N)$, along with the user annotated junction types $\mathcal{J} = (J_1, ..., J_M)$, functional relations $\mathcal{F} = (F_1, ..., F_P)$ and geometric relations $\mathcal{G} = (G_1, ..., G_Q)$, we want to obtain the best junction parameters $\mathbf{\Theta}^* = (\Theta_1^*, ..., \Theta_M^*)$ for the junction types $\mathcal{J}$ along with the best cuboids $\mathcal{B}^* = (B_1^*, ..., B_N^*)$, satisfying the functional relations $\mathcal{F}$ and geometric relations $\mathcal{G}$. The formulation is defined as:

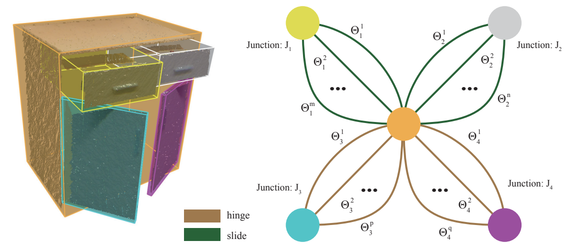

Figure 5. Junction configuration graph. Each cuboid corresponds to the node with the same color, while each annotated junction type corresponds to the multiple connections between nodes. One connection is associated with one candidate junction parameter.

$$\underset{\mathcal{B},\mathbf{\Theta}}{\operatorname{argmin}}\, E(\mathcal{B}, \mathbf{\Theta}, I) \text{ s.t. } \mathcal{B}, \mathbf{\Theta} \text{ satisfy } \mathcal{J}, \mathcal{F}, \mathcal{G}. \quad (1)$$

Here $E(\mathcal{B}, \mathbf{\Theta}, I)$ measures the deviation from the optimized cuboid configuration to the input point cloud, which is defined as

$$E(\mathcal{B}, \mathbf{\Theta}, I) = \sum_j \sum_k dist(B_j - p_j^k), \quad (2)$$

where $\sum_k dist(B_j - p_j^k)$ gives the deviation from cuboid B_j to its containing points p_j^k.

The challenge is how to wrap down the annotated relations to geometric constraints while retaining the cuboids' conformity with respect to the input point cloud. Since the annotated relations are high level specifications, this leads to large search space in the optimization due to the potential ambiguities raised from the loose annotations. Another challenge is that the cuboid parameters are highly coupled with the junction parameters. That is, given a set of junctions, the cuboid geometry should change accordingly to satisfy the functional constraints. Thus we cannot optimize the parameters locally and separately, but instead do it in a global manner. To solve the above challenges, we device a multi-stage optimization paradigm to first populate the solution space with a two-step sampling algorithm and then jointly optimize the cuboid parameters and junction parameters.

In the first stage, we sample the possible junction's parameters, i.e., axial position and orientation. Let us denote the set of junction types as $(J_1, ..., J_M)$, and the parameters we wish to estimate as $(\Theta_1, ..., \Theta_M)$. We start by building a junction configuration graph. For each cuboid we create a graph node and for each junction type J_i, we create multiple graph connections, with each connection associating

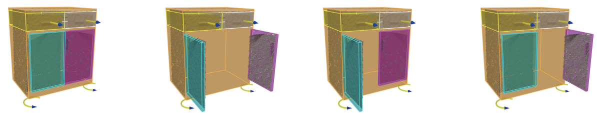

Figure 6. Possible rest poses for the hinge junctions. Since we don't know which face of the cabinet door should cover the cabinet, we rotate the hinge junctions to sample a set of rest configurations to guess possible covering faces. (Figure 6)

with a candidate parameter Θ_i^l for J_i. If A forms a hinge relation with B, each cuboid edge of A can be a candidate hinge axis. We choose only those cuboid edges which are closely attained to B. More specially, we only choose the edges parallel to the face if there is also a *cover* relation, and only choose the edges perpendicular to the face if there is a *fit inside* or *support* relation. This leads to a configuration graph where any traversal path of the graph represents a possible configuration of junctions. Figure 5 shows such a graph. Algorithm 1 gives the pseudo-code of this state.

Given the junction configuration graph, for each junction configuration we optimize the cuboids orientation, position and size based on annotated functional/geometric relations. The cuboid orientation and position is firstly adjusted based on the current candidate junction configuration, by adjusting the corresponding junction edges. We assume that the junction must be snapped to the nearest cuboid face and be parallel to the nearest cuboid edge (as in [12]). Then we optimize the cuboid size to satisfy the functional/geometric relations from current junction configuration. Note that the functional relations typically indicate the geometry of the cuboids satisfying certain constraints in a *closed* configuration (i.e., a rest configuration [12]. For an instance, if A covers B, this typically means that one face of A is rotated about the hinge junction to be in close agreement with a face of B (Figure 6). Since we don't know which face covers B, we enumerate through multiple possible cuboid faces to sample a set of rest configurations (Figure 6) and for each rest configuration we optimize the cuboid parameters. In specific, given a rest configuration of cuboids, we employ a similar optimization method of [12] to optimize the cuboid parameters $(B_1^*, ..., B_N^*)$. We then compute the optimization cost from Eq. (2). Finally, the configuration which leads to the least deviation from the point cloud is selected as the best configuration and the optimized cuboids are then computed. The overall algorithm is detailed in Algorithm 2.

5. Results

We used our system to recover functionality prototypes for 6 different objects (Figure 7). The first 4 examples (cabinet, drawer, firebox and chair) are real RGBD images captured with Microsoft Kinect, while the last 2 examples (toolbox and dining table) are synthetic depth data captured

Algorithm 1 Building Junction Configuration Graph

Input: N initial cuboids $(B_1, ..., B_N)$; M junctions $(J_1, ..., J_M)$ with unknown parameters $(\Theta_1, ..., \Theta_M)$;
Output: Multi-connection junction Graph $G := (V, E)$, where each connection e_i^j corresponds to a parameter Θ_i^j for J_i;

$G \leftarrow \varnothing$
for $i = 1$ to N **do**
  $V_i \leftarrow B_i$
end for
/*** Building multi-connections between nodes ***/
for $i = 1$ to M **do**
  $B_c \leftarrow$ child cuboid of J_i
  $B_p \leftarrow$ parent cuboid of J_i
  $l \leftarrow 1$
  /*** Test each edge of the child cuboid ***/
  for $j = 1$ to 12 **do**
    $E_j \leftarrow$ j-th edge of B_c
    $D_j \leftarrow$ direction of E_j
    $C_j \leftarrow$ center of E_j
    for $k = 1$ to 6 **do**
      $F_k \leftarrow$ k-th face of B_p
      $N_k \leftarrow$ normal of F_k
      if $dist(E_j, F_k) < \epsilon_d$ **and** $\text{abs}(dot(D_j, N_k)) < \epsilon_a$ **and** $\text{abs}((dot(D_j, N_k) - 1) < \epsilon_a$ **then**
        $\Theta_i^l \leftarrow (C_j, D_j)$ //set candidate parameter for the J_i
        $e_i^l \leftarrow \Theta_i^l$ //add a connection e_i^l
        $l \leftarrow l + 1$
      end if
    end for
  end for
end for

from existing 3D designs. Please check our submission video to see how the various parts move and fit together. Creating one functional prototype took 0.5-5 minutes for our experimental examples. The time for user interaction (segmenting points with strokes and specifying part relationships, plus the waiting time for the plane detection for initial cuboid generation) ranges from 27 seconds to 108 seconds, and the optimization time varies a lot from 1 second to 241 seconds, depending on the sampling space of junction parameters and rest poses. The experimental statistics are listed in Table 1.

As shown in Figure 2, though the geometry of our prototypes may appear simple, the relationships between the moving parts are often complex. Adjusting the geometry and relation parameters would be rather time consuming and labor consuming. Our system automatically infer the junction parameters (position & axis) and the geometry parameters (size & position & orientation) by jointly optimizing them together under the user annotated high-level constraints. All the desired part parameters and junction parameters are obtained in our experiment data. For example, in Figure 7 (1), our algorithm automatically place the

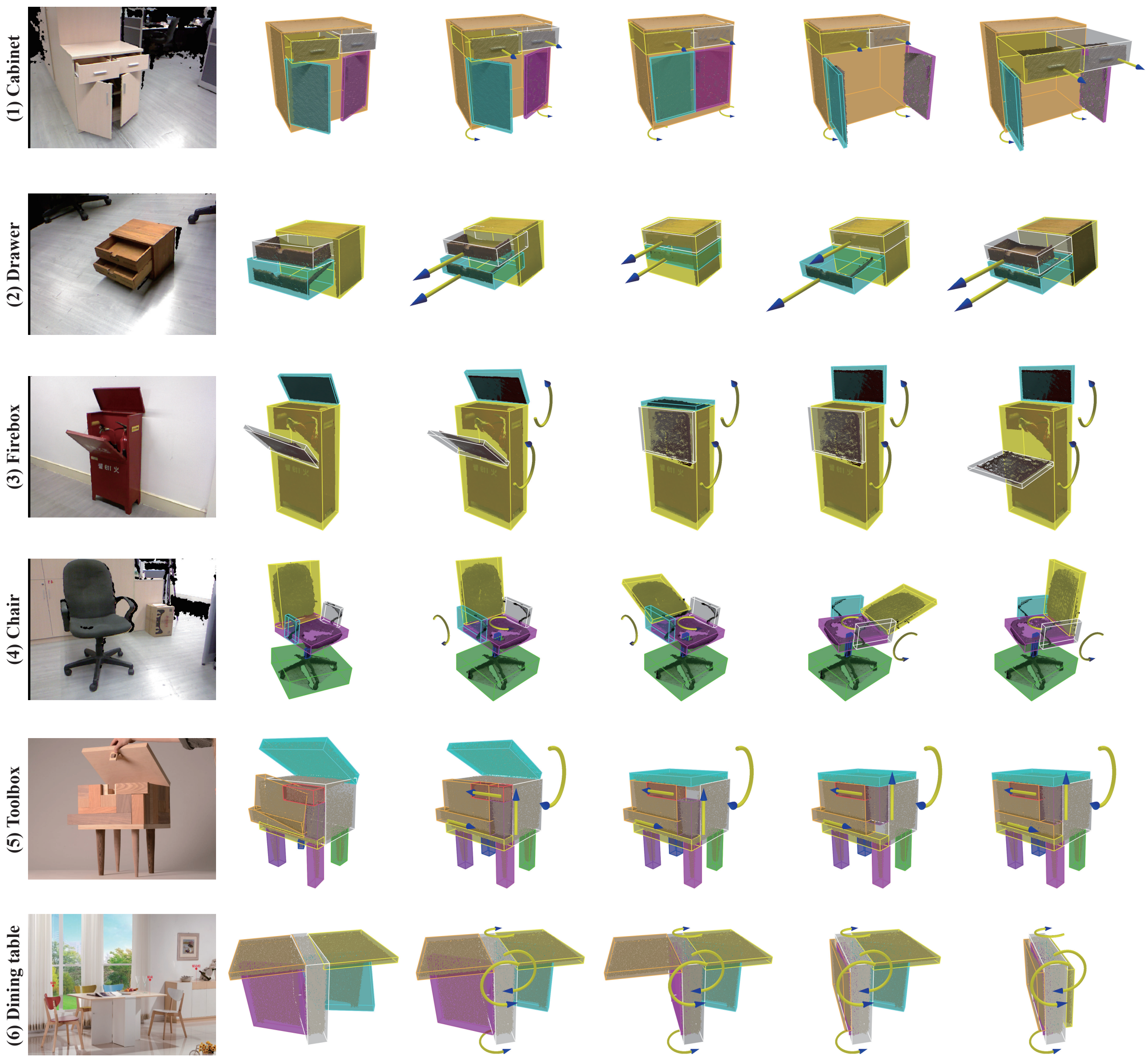

Figure 7. Experimental results. From left to right: the input RGBD image, initial cuboids, optimized cuboids and junctions, and how parts move and fit after the optimization (3 configurations).

hinge junctions to the correct edges of the cabinet doors, and adjust their orientations accordingly by aligning the hinge junctions onto the nearest cabinet face and make them parallel to the nearest cabinet edges. The size of the doors are also optimized to be equal size and cover the cabinet. The drawers in Figure 7 (1) and (2) obtain the desired orientation by aligning their sliding junctions with the cabinet, and the size is optimized to just fit inside the cabinet and be equal. In Figure 7 (3), the top cap and the front door are both optimized to just cover to the boundary of the firebox. For the chair example (Figure 7 (4)), due to occlusion, the initial cuboids for the leg and the armrest have smaller size than real, but our algorithm successfully extend the leg to support the seat, and extend the armrests to connect with the back. Similarly, the occluded leg in Figure 7 (5) is extended to support the box and has the same size as other legs. In Figure 7 (6), the orientation and the size of the two doors are optimized to support the table top, and the orientation of the top is optimized to be horizontal.

User study. To better evaluate whether our approach can recover correct functional prototypes, we showed our sys-

Algorithm 2 Optimizing cuboids and junctions

Input: input point cloud I; N initial cuboids $\mathcal{B} = (B_1, ..., B_N)$; junction configuration graph G; functional relations $\mathcal{F}$; geometric relations $\mathcal{G}$

Output: N optimized cuboids $\mathcal{B}^* = (B_1^*, ..., B_N^*)$; M optimized junction parameters $\mathbf{\Theta}^* = (\Theta_1^*, ..., \Theta_M^*)$;

/*** Sampling candidate junction parameters from G and accordingly optimizing the cuboid orientation, position and size ***/
$err \leftarrow INF$ //deviation from cuboids to input point cloud
while 1 **do**
  Gather an connection combination $(e_1^k, ..., e_M^l)$ from G
  if no more connection combination **then**
    break
  end if
  Create junctions with parameters $\mathbf{\Theta}' = (\Theta_1^k, ..., \Theta_M^l)$ from $(e_i^k, ..., e_M^l)$
  adjust the cuboid position and orientation by snapping the junction edge
  /*** Calculating possible angles for rest configurations ***/
  for $i = 1$ to M **do**
    calculate candidate angles $(\alpha_i^1, ..., \alpha_i^w)$ to parallelize parent and child
  end for
  /*** Sampling possible rest configurations ***/
  while 1 **do**
    Gather an angle combination $(\alpha_1^u, ..., \alpha_M^v)$
    if no more angle combination **then**
      break
    end if
    Transform to rest configuration with $(\alpha_1^u, ..., \alpha_M^v)$
    Optimizing the cuboid size satisfying $\mathcal{F}$ and $\mathcal{G}$ to get an solution $\mathcal{B}' = (B_1', ..., B_N')$
    if $E(\mathcal{B}', \mathbf{\Theta}', I) < err$ **then**
      $err \leftarrow E(\mathcal{B}', \mathbf{\Theta}', I)$
      $(B_1^*, ..., B_N^*) \leftarrow (B_1', ..., B_N')$
      $(\Theta_1^*, ..., \Theta_M*) \leftarrow (\Theta_1^k, ..., \Theta_M^l)$
    end if
  end while
end while

tem to 20 students. 5 of them are undergraduate majoring in computer science, and another 4 students are master candidates in industry design. The rest ones are 8 master candidates and 3 PhD candidates in computer science. We showed them the captured RGBD images and asked them to imagine how the objects work. Then these students used our system to add annotations to the pre-generated initial cuboids based on their imagination. All the students reported that our system successfully recovers the functional prototypes with the parts moving as they expected. Besides, the optimized part geometry also satisfies their imagination. One exception is that 6 students said they imagined the hinge junction on the cabinet door (Figure 7 (1)) was exactly on the boundary edge of the cabinet, while our optimization did not consider it as the best configuration.

Model	Hinge	Slide	Fxn	Geom	Int. Time (s)	Opt. Time (s)
Cabinet	2	2	4	2	62	18
Drawer	0	2	2	1	29	1
Fire box	2	0	2	0	27	16
Chair	2	0	3	0	90	2
Tool box	1	3	6	0	108	3
Dining table	4	0	6	2	55	241

Table 1. Statistics for recovered functional prototypes.

Comparison with real objects and 3D design models. We also check the recovered prototypes with the captured real objects and 3D design models. As illustrated in the top 2 rows in Figure 8, the generated prototypes have similar functionality as the real objects, and they can move parts to generate almost the same configurations as the real ones. Besides, the optimized simple cuboids can approximate the real geometry well, with almost the same size, orientation and position. We also compare our recovered prototypes with the 3D design models whose junctions are added and adjusted manually in Autodesk 3ds Max (bottom row in Figure 8). We can see our recovered prototype from user's high-level annotation has very similar functionality as the manually designed model.

6. Conclusions

In this work, we present a novel approach to recover functional prototypes from user's high-level annotations on relationships. By providing the junction types and other functional/geometric relations, the junction parameters and part geometry parameters are jointly optimized. With such interface, we allow users to focus on the functional goals of the target object rather than working on low-level geometry and junction parameters. Our results demonstrate that our system can generate functional models with a small number of user annotations. In the user study, the recovered prototypes work correctly as the users expected. The comparison with the real objects and 3D design models also prove the feasibility of our system.

Limitations and future work. The main limitation of our approach is that we use cuboids as proxies to approximate the part geometry. While compositions of cuboids are sufficient for the understanding of functionality of many products, users often like higher fidelity geometry to better understand the geometry and relationships. Similarly, the restricted set of junction types is another limitation. In the future, we will add other primitives for part proxy, such as cylinder and sphere. We also plan to integrate more junction types between parts, like ball junctions and simple mechanical units. Current optimization framework may need to be modified to handle more geometry and junctions. Another future direction is to consider other high-level func-

Figure 8. Top two rows: comparison result with the captured real data; bottom row: comparison result with the 3D design model. We can see that our system can faithfully recover the functionality as user expected.

tional constraints among parts. Exploring more high-level relationships would help the further exploration of the functionality as well as the geometric properties.

References

[1] M. Arikan, M. Schwärzler, S. Flöry, M. Wimmer, and S. Maierhofer. O-snap: Optimization-based snapping for modeling architecture. *ACM TOG*, 32(1):6:1–6:15, 2013. 2

[2] M. Bokeloh, M. Wand, H.-P. Seidel, and V. Koltun. An algebraic model for parameterized shape editing. *ACM Trans. Graph.*, 31(4):78:1–78:10, July 2012. 2

[3] M. Daniel and M. Lucas. Towards declarative geometric modelling in mechanics. In P. Chedmail, J.-C. Bocquet, and D. Dornfeld, editors, *Integrated Design and Manufacturing in Mechanical Engineering*, pages 427–436. Springer Netherlands, 1997. 3

[4] R. Gal, O. Sorkine, N. J. Mitra, and D. Cohen-Or. iwires: An analyze-and-edit approach to shape manipulation. *ACM Transactions on Graphics (Siggraph)*, 28(3):#33, 1–10, 2009. 2

[5] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*, 2010. 2

[6] A. Gupta, M. Hebert, T. Kanade, and D. M. Blei. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1288–1296. Curran Associates, Inc., 2010. 2

[7] Y. Han, J.-Y. Lee, and I. S. Kweon. High quality shape from a single rgb-d image under uncalibrated natural illumination. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, pages 1617–1624, Washington, DC, USA, 2013. IEEE Computer Society. 1, 3

[8] E. Hartley, B. Kermgard, D. Fried, J. Bowdish, L. D. Pero, and K. Barnard. Bayesian geometric modeling of indoor scenes. *IEEE CVPR*, pages 2719–2726, 2012. 2

[9] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568, New York, NY, USA, 2011. ACM. 1

[10] Z. Jia, A. Gallagher, A. Saxena, and T. Chen. 3d-based reasoning with blocks, support, and stability. In *IEEE CVPR*, pages 1–8, 2013. 2

[11] H. Jiang and J. Xiao. A linear approach to matching cuboids in rgbd images. In *IEEE CVPR*, 2013. 2

[12] B. Koo, W. Li, J. Yao, M. Agrawala, and N. J. Mitra. Creating works-like prototypes of mechanical objects. *ACM Transactions on Graphics (Special issue of SIGGRAPH Asia 2014)*, 2014. 2, 3, 4, 5

[13] F. Lafarge and P. Alliez. Surface reconstruction through point set structuring. In *Proc. of Eurographics*, Girona, Spain, 2013. 2

[14] Y. Li, X. Wu, Y. Chrysanthou, A. Sharf, D. Cohen-Or, and N. J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM SIGGRAPH*, 30(4), 2011. 2

[15] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM SIGGRAPH*, 23(3):309–314, 2004. 3

[16] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, June 2007. 3

[17] T. Shao, W. Li, K. Zhou, W. Xu, B. Guo, and N. J. Mitra. Interpreting concept sketches. *ACM Transactions on Graphics*, 32(4), 2013. 2, 3

[18] T. Shao, A. Monszpart, Y. Zheng, B. Koo, W. Xu, K. Zhou, and N. J. Mitra. Imagining the unseen: Stability-based cuboid arrangements for scene understanding. *ACM Trans. Graph.*, 33(6):209:1–209:11, Nov. 2014. 1, 3

[19] C.-H. Shen, H. Fu, K. Chen, and S.-M. Hu. Structure recovery by part assembly. *ACM Trans. Graph.*, 31(6):180:1–180:11, Nov. 2012. 1, 3

[20] L. Sullivan. The tall office building artistically considered. *Lippincott's Magazine*, 57, 1896. 1

[21] N. Umetani, T. Igarashi, and N. J. Mitra. Guided exploration of physically valid shapes for furniture design. *ACM SIGGRAPH*, 31(4):86:1–86:11, 2012. 2

[22] W. Xu, J. Wang, K. Yin, K. Zhou, M. van de Panne, F. Chen, and B. Guo. Joint-aware manipulation of deformable models. *ACM Trans. Graph.*, 28(3):35:1–35:9, July 2009. 2

[23] H. Yu. Edge-preserving photometric stereo via depth fusion. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 2472–2479, Washington, DC, USA, 2012. IEEE Computer Society. 3

[24] P.-A. Yvars. Using constraint satisfaction for designing mechanical systems. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 2(3):161–167, 2008. 3

[25] B. Zheng, Y. Zhao, J. C. Yu, K. Ikeuchi, and S.-C. Zhu. Beyond point clouds: Scene understanding by reasoning geometry and physics. In *IEEE CVPR*, 2013. 2

[26] Y. Zheng, X. Chen, M.-M. Cheng, K. Zhou, S.-M. Hu, and N. J. Mitra. Interactive images: Cuboid proxies for smart image manipulation. *ACM SIGGRAPH*, 31(4):99:1–99:11, 2012. 1, 2