GemSketch: Interactive Image-Guided Geometry Extraction from Point Clouds

Mehran Maghoumi[1], Joseph J. LaViola Jr.[1], Karthik Desingh[2] and Odest Chadwicke Jenkins[2]

***Abstract*— We introduce an interactive system for extracting the geometries of generalized cylinders and cuboids from single- or multiple-view point clouds. Our proposed method is intuitive and only requires the object's silhouettes to be traced by the user. Leveraging the user's perceptual understanding of what an object looks like, our proposed method is capable of extracting accurate models, even in the presence of occlusion, clutter or incomplete point cloud data, while preserving the original object's details and scale. We demonstrate the merits of our proposed method through a set of experiments on a public RGB-D dataset. We extracted 16 objects from the dataset using at most two views of each object. Our extracted models represent a high degree of visual similarity to the original objects. Further, we achieved a mean normalized Hausdorff distance of 5.66% when comparing our extracted models with the dataset's ground truths.**

I. Introduction

Recent advancements in the field of robotic research are propelled by 3D sensors and their ability to provide accurate measurements which are in turn used to geometrically model the objects or the environment surrounding a robot. Methods that use such models to aid perception fall under the category of model based generative approaches [1], [2], [3], [4]. These approaches use mesh models that are either synthetically designed using CAD tools or created by scanning real scenes. Further, object geometries are becoming a necessary prior information for grasping and manipulation [5]. Often times, object geometries are required to perform reliable grasping and manipulation actions on the object. Designing suitable models from scratch is a challenging endeavor.

Although automatic object scanning methods which typically rely on real-time surface reconstruction [6], [7], have gained interest in recent years, they are not without their own problems: their outputs may contain noise, holes and other imperfections which are usually caused by hardware limitations or inherent object or surface properties (transparencies, reflections, *etc*.). Concurrently, a variety of methods that leverage the human's cognitive ability in the task of 3D geometry extraction have been proposed in the literature [8], [9], [10], [11]. Inspired by the success of these human-in-the-loop methods, this paper is an effort towards reliable geometry extraction that can be used in model based generative algorithms geared towards robot manipulation.

In this paper, we introduce GemSketch: a human-in-the-loop system that can extract the 3D models of generalized cuboids and generalized cylinders from single- or multi-view point cloud data. By tracing the outlines of an object of interest with a mouse or a stylus on the 2D image of the object, GemSketch can extract complete mesh models of objects from point clouds, while preserving object scales as well as their details. Armed with the perceptual understanding of humans, GemSketch facilitates robust and reliable extraction, even in the presence of occlusion, and clutter as well as noisy or incomplete point clouds (see Figure 1). Through a series of experiments we demonstrate the power and the accuracy of GemSketch in 3D geometry extraction. Our results demonstrate that with usually one and at most two point cloud views, GemSketch is a viable tool for extracting accurate 3D geometries.

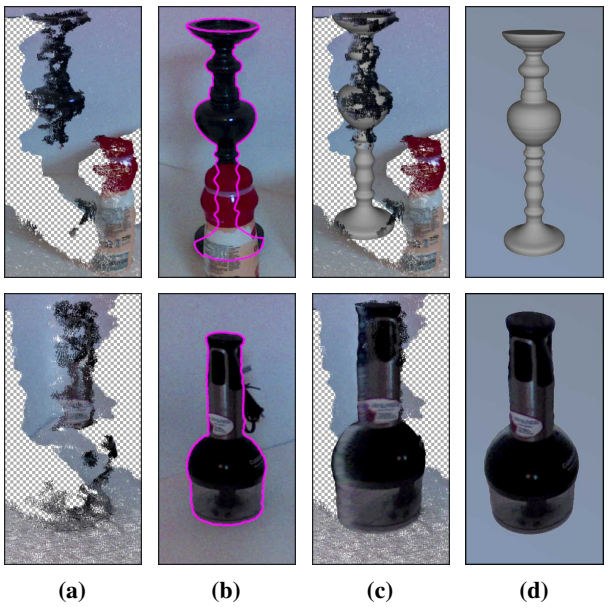


Fig. 1: Geometry extraction of and occluded candlestick (top) and a transparent grinder (bottom) using GemSketch. (a) Sideways single-view point cloud data. (b) User-traced silhouettes on the RGB image (users may use their intuition to trace the occluded object parts) (c) 3D mesh superimposed on the point cloud. (d) Fully extracted at-scale texture-mapped 3D model. The grinder in (d) is texture mapped. Checkered pattern indicates missing point cloud data. Note that in the final result, small deviations from the user input are expected due to smoothing.

Contributions. First, we introduce an interactive geometry extraction system designed to work with noisy and unreliable 3D point cloud data. Second, we develop a minimalistic, yet intuitive user interface that enables our computational methods to overcome limitations caused by occlusions, clutter and other imperfections. The last contribution is the evaluation of our proposed approach on a public RGB-D dataset.

[1]Department of Computer Science, University of Central Florida, Orlando, FL 32816 USA
[mehran|jjl]@cs.ucf.edu
[2]Department of Computer Science and Engineering, University of Michigan, Ann Arbor, MI 48109 USA
[kdesingh|ocj]@umich.edu

II. Related Work

Automatic Shape Extraction or Completion. Schnabel *et al*. [12] proposed an efficient RANSAC-based method for automatically extracting geometries that have analytical representations (*e.g*. planes, cylinders, tori, *etc*.) from point clouds. Their method is fast but may require post processing to merge together multiple partial-shapes that originally belonged to a larger shape. Kim *et al*. [13] proposed a method that can extract the full geometry of a target shape given topological information (such as placement of subparts) about the shape. The difference between these works and ours is that we utilize interactive interfaces to complement the computational power of computers and benefit from the perceptual understanding of the human operator. It is worth mentioning that automatic methods have been progressing rapidly in recent years and have been shown to demonstrate great success in shape [14] or point cloud completion [15]. However, these works mostly focus on plausible completion of the 3D models without specifically paying attention to preserving the original object's details or geometry.

Deep Learning. Recently, deep networks have gained a lot of interest due to their power in aiding robotic manipulation [5], 3D shape recognition [16], [17] and reconstruction [18], [19]. Zhaoliang *et al*. [18] proposed a method for reconstructing 3D objects using their 2D sketches. Their approach works by encoding the multi-view input sketches into comapct representations learned using a deep encoder network, and decoding the obtained representations into depth an normal maps from multiple viewpoints. The final 3D model is obtained by the fusion of these depth and normal maps. Although very promising, deep learning-based methods require a lot of training data that may not be readily available and as such differ in scope from the current work.

Interactive Modeling. Interactive modeling using sketches dates back to [20] and [21]. More recently, Gingold *et al*. [22] proposed a framework for creating 3D shapes using annotated sketches or images. The user would place a group of predefined annotation elements (*e.g*. shapes, geometric relationships, *etc*.) on an image or a freehand drawing of a target object to build a 3D model. Similarly, Shtof *et al*. [23] model a 3D shape using its constituent sub-parts through dragging and dropping predefined 3D primitives (eg. spheres, cylinders, *etc*.) on each sub-part. Dragged shapes are snapped to the silhouettes using geosemantic relations and are optimized to produce the best fit shape. Xu *et al*. [24] proposed a system for modeling mechanisms (such as a piston-engine) using sketches on multiview images. Their modeled structures were capable of simulating the behavior of their physical counterpart. The common element of all these existing work is the use of sketches as the underlying data to guide and aid the creation of 3D shapes. In some of these works, the shapes may not necessarily exist as physical objects. Whereas in this work, we focus on extracting existing objects for which we have 3D point cloud data.

Image-guided Modeling. In addition to sketching, modeling 3D shapes and geometries can also be guided by images or videos. Lau *et al*. [25] introduced a framework for creating customized complementary objects that may not exist in the real-world. Such creation is guided by an image of a similar object and aims to facilitate model designing. Conversely, the current work focuses on extracting the geometry of an object. Zou *et al*. [26] leverage the planar structure of piecewise planar objects to extract complete geometries. Given a rough sketch of an object of interest, the planar structure of object parts are exploited to accurately model the object as well as its occluded parts. In comparison, our work supports more generalized primitives. van den Hengel *et al*. [27] introduced VideoTrace, a system which pairs user interaction with sparse point cloud obtained from multiview video frames to model 3D objects. The current work differs from VideoTrace in that the presence of multiple views of an object is a necessity for the latter, while the former can still function with single view data.

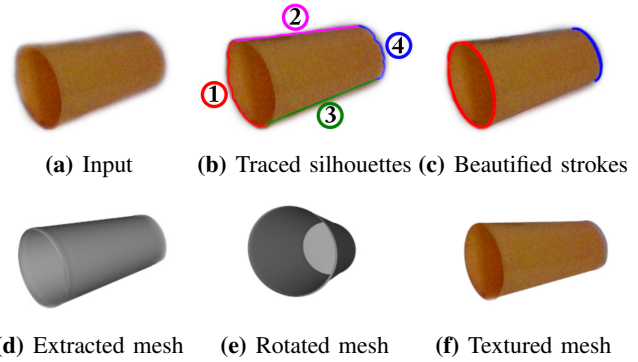
Our work is most inspired by 3-Sweep [8] that presents a tool for modeling generalized cylinders and cuboids using sweeping interactions on photos. Two sweeps determine the object's base shape (a disk or a rectangle) while the third sweep determines the object's height extent. The 3D shape and the pose of the base is estimated using its 2D projection and is copied along the extent direction of the third sweep to form the complete 3D shape. More complex objects are modeled by modeling their sub-parts individually. GemSketch differs from 3-Sweep and extends it in a number of ways. GemSketch can benefit from multiple views as well as the availability of 3D data in order to minimize the need for resolving pose ambiguities. We will demonstrate later that our method can model objects even in the presence of clutter and occlusion. We further discuss the differences of our work and that of Chen's *et al*. [8] in Section V.

III. Interactive Geometry Extraction

A. Overview

Our geometry extraction system can take a single input or a series of inputs. If only a single view of an object of interest is available, the input to our system is a triplet of a point cloud, an RGB image and camera calibration values for the RGB image. If multiple views are available, we additionally require the extrinsic transformation of the camera between each two consecutive views. To perform geometry extraction, the user traces the outlines of the object of interest on the RGB image using 4 disjoint strokes, one for each side of the object as shown in Figure 2b. Users may use as many views as they want to perform sketching. Drawn sketches can be transferred between views, since we can estimate the homography between the scenes [28] (see the accompanying video[1]). The system optionally provides a *magnetic selection tool* that snaps the selection path to the nearest object outlines. If the object is partially occluded, users are free to complete the traces using their intuition of how the occluded object parts look (see Figure 1b). When tracing is done, the system creates a texture-mapped 3D mesh of the object that was traced. In total, our geometry extraction is performed in 4 steps: preprocessing, 3D profile extraction,

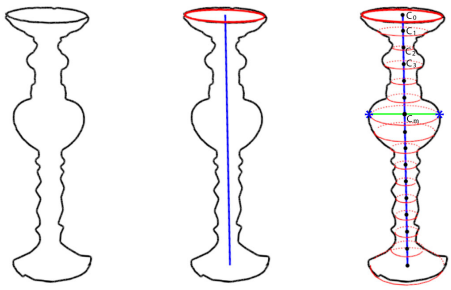
[1]The project page is located at:
`http://www.eecs.ucf.edu/isuelab/research/GemSketch/`



(a) Input **(b)** Traced silhouettes **(c)** Beautified strokes

(d) Extracted mesh **(e)** Rotated mesh **(f)** Textured mesh

Fig. 2: Tracing the outlines of a cup to extract its 3D mesh. The user can specify the mesh to be hollow.



(a) **(b)** **(c)**

Fig. 3: Modeling a generalized cylinder. (a) User-traced silhouettes. (b) Extracted object's profile and spine. (c) Propagated and scale-adjusted profiles at regular intervals along the spine. Each C_i denotes the translated center of the copied profile along the spine. Copied profiles are scaled such that they meet the user's strokes at the points denoted by asterisk.

object axis extraction and finally, profile propagation. We next describe these steps in detail.

B. Terminology

Before delving deeper into the geometry extraction pipeline, we define some of the terminology we frequently use throughout the text.

Shape Profile. We refer to the 3D disk or rectangular shape at either end of the object as the shape profile. This shape determines whether the target object's shape is a generalized cylinder or a generalized cuboid.

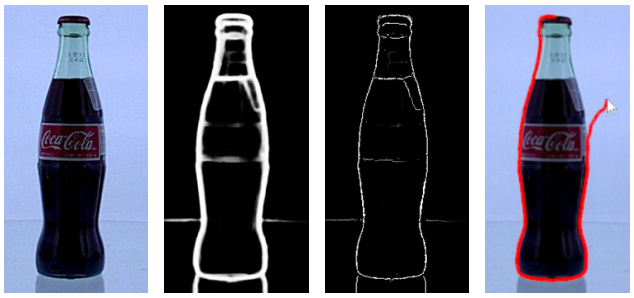
Profile Plane. The plane that contains the 3D profile shape is called the profile plane.

Profile Stroke. The stroke that is used to extract the 3D shape profile is called the profile stroke. This stroke is usually drawn at one end of the object (strokes 1 and 4 in Figure 2b).

Side Strokes. We refer to strokes 2 and 3 in Figure 2b as side strokes. These strokes usually span the object's dominant extent when tracing the silhouettes.

Spine. We refer to the object's main axis as the 3D spine. This line or curve spans through the extents of the object and determines whether the object is a straight or a bent shape.

Profile Propagation. We call the process of creating copies of the shape profile along the spine while accounting for normal and scale changes profile propagation (see Figure 3c).



(a) Input color image **(b)** Detected edges **(c)** Thinned edges **(d)** Selection path

Fig. 4: Proprocessing steps performed on the RGB image necessary for the magnetic selection tool. Edges of the input image (a) are detected using [Xie and Tu 2015] as shown in (b) and are thinned to produce the edges in (c).

Object Plane. The object plane is a plane that is orthogonal to the plane of the profile, is fully visible from the camera's viewpoint, and with respect to which the object is symmetric.

C. Preprocessing of Input Data

The input triplet to our system can be obtained from commodity depth sensors (such as Microsoft Kinect or Intel RealSense cameras). We assume that the point cloud is already transformed into the coordinate frame of the color camera to simplify our computations. In the case of multiple-view point clouds, the camera must be tracked too and the extrinsic transformations of the camera between consecutive views are needed. Although state-of-the-art registration methods [19] can be utilized for this purpose, we found the iterative closest point (ICP) [29] method to be sufficient for obtaining such transformations in case of small camera movements.

We should note that the point clouds obtained using commodity depth cameras contain inaccuracies and highly undersampled surfaces due to inherent object properties (see Figure 1a). As such, one important contribution of our system is using robust algorithms that can work around such issues. Consequently, in our algorithms we prefer fitting and estimation over the direct use of point cloud's points.

Our system performs a series of preprocessing steps on the input data. To accelerate ray intersection queries with the point cloud, we populate an octree using the point cloud's points. The color image is undistorted and filtered for noise using non-local means denoising [30]. Afterwards, existing contours of the color image are found using the method proposed in [31] and are thinned using non-maximal suppression [32] as shown in Figure 4c.

To aid users in tracing object outlines, we provide a magnetic selection tool that is based on Intelligent Scissors [33] (see Figure 4d). This tool snaps the active selection path to the nearest image contour. In our system, users can force this tool to draw straight lines by holding a modifier key on the keyboard, which is helpful for tracing objects that have straight silhouettes such as boxes. At the preprocessing stage, we precompute the necessary graphs for this tool to work (see [33] for more details).

Finally, users specify the object's silhouettes through stylus or mouse interactions. Because accurate stroke segmentation of complex trajectories is still an open issue, we opt for a

more reliable approach to avoid false positives. Specifically, a user draws four strokes, two of which specify the top and bottom of the target object, and two more that span the two sides (see Figure 2b). In our testing, we found this minimum number of strokes to be expressiveness enough to define all shapes of interest, regardless of occlusion.

D. Profile Extraction

The user starts the extraction process by sketching at one end of the object, whichever they deem more suitable for sketching (either stroke 1 or 4 in Figure 2b). This first stroke determines the shape of the profile of the object as well as the final 3D shape and is the cornerstone of the rest of the algorithm. If the shape of the first stroke is an arc or an ellipse (either complete or partial), the next step of the pipeline will treat the object as a generalized cylinder, otherwise the object will be treated as a generalized cuboid. The recognition of the first stroke is performed using a fast recognizer that supports gesture customization [34] so that a wide variety of shape profile sketches can be recognized.

Once the profile's 2D shape is determined, its 3D shape is obtained using the 3D point cloud's points. The goal is to fit a 3D circle or rectangle to the point cloud's points that correspond to the image points under the user's sketches (circular disks can be converted to elliptical disks if necessary). Further, the projection of this 3D shape must match the user's sketches as best as possible. We satisfy these requirements in four steps.

Noise Removal and Beautification. We first smooth all the user's strokes using Gaussian smoothing to eliminate noise and jitters. Also, we beautify the profile stroke by fitting the appropriate 2D shape to the user's input (see Figure 2c). We have found this beautification step to be instrumental for the subsequent steps of our pipeline. Beautification of arcs and ellipses is done using ElliFit [35], whereas beautification of polylines is done similar to [36]: sharp corner features are determined using IStraw [37] and are connected with straight lines to form the polyline. In case sketches are performed for more than one view, beautification is performed for sketches in all views.

Obtaining 3D Points. Next, we obtain the 3D points corresponding to the 2D sketched profile. This is done by backprojecting beautified stroke points to 3D rays using the camera's inverse projection matrix. We perform intersection tests between these rays and the point cloud's octree. In cases where the rays do not intersect 3D points directly, we perform a closest-point-to-line look up among the points in the octree voxel that the 3D ray intersects. To avoid selecting noisy points, outliers and points belonging to nearby objects, we filter all the obtained 3D points by performing a simple density-based clustering method similar to DBSCAN [38] and the points in the largest cluster are used for 3D fitting.

Multi-view Cases. In case of multiple-view point clouds, we use the method above to obtain and filter the 3D points per each view. We then combine all the obtained points together into the coordinate frame of one of the views using the known transformations between all the point cloud views.

Fitting a 3D Shape. The next step is to fit a 3D disk or a 3D rectangle to the 3D points obtained in the previous step. We use RANSAC for fitting 3D disks as we have experimentally found it to be the simplest and the most robust method when dealing with noisy or incomplete point clouds. To fit a 3D rectangle, we first fit a 3D plane to the 3D points using RANSAC. We then project the 3D points onto the obtained plane and find their minimum area bounding rectangle on the plane using the rotating calipers [39] algorithm. This rectangle is converted to a 3D rectangle by expressing it in terms of its 3D center and extents with respect to the fitted plane.

Pose Optimization. At this point, the fitted 3D shape may have a slightly wrong pose or scale. In our experience, this happens more frequently when only one view of the object is available. To correct this, we formulate and solve an optimization problem in which the goal is to match the 2D projection of the 3D profile to the beautified user strokes. This can be thought of as finding a rotation transformation and a scaling coefficient that minimizes the distance between the n points on the circumference of the 3D shape and the 3D rays obtained by backprojecting the corresponding 2D points on the beautified strokes. To make the formulation of the optimization problem simpler, the 3D rays obtained via backprojection can be represented using *Plücker lines* as detailed in [28]. Therefore, we can formulate the following optimization system:

$$\text{minimize } E = \sum_{i=1}^{n} d\big(\ (s\ .\ R_{Schmidt})\ P_i\ ,\ \ L_i\ \big)^2 \qquad (1)$$

where s is the scaling coefficient, $R_{Schmidt}$ is the rotation vector expressed in the formulation of [40], P_i is the ith 3D points sampled on the circumference of the modeled 3D profile. The formulation of Equation 1 using Plücker lines instead of reprojection error simplifies the computation of the Jacobian matrix. In Equation 1, L_i is the Plücker line obtained by backprojecting the ith 2D point on the beautified stroke corresponding to the ith 3D point. Each L_i is of form $L = (\vec{u}, \vec{m})$ where $\vec{u}$ is a unit vector and $\vec{m}$ is the moment vector. Finally, $d\big(P,\ L\big)$ is the distance between the 3D point P and the Plücker line L. This distance can be computed as [41]:

$$d\big(P,\ L\big) = ||P \times \vec{u}\ -\ \vec{m}||_2 \qquad (2)$$

where $\times$ is the cross product. Equation 1 is a non-linear unconstrained sum of squares minimization problem which we solve using the Levenberg-Marquardt method [42]. The optimization is very fast and is done at interactive rates.

E. Object Spine Extraction

The next step of the pipeline is to extract the object's spine in 3D using the user's sketches. For simplicity we assume that the object's 3D spine is planar from the camera's viewpoint. As a result, the spine can be either a line or a curve, depending on whether the object is a straight or a bent shape. We recover this line or curve in two steps. We first recover its 2D projection using the user traced silhouettes. We then use this 2D projection and the extracted 3D profile to fully recover the spine in 3D.

To extract the spine's 2D projection, the side strokes (strokes 2 and 3 shown in Figure 2b) are used. The challenge in the extraction of the spine using the side strokes is that these strokes may be made up of complex curves, but the object's spine may be a line, or a simple curve. As a result, we use the following procedure to robustly estimate the projection. We fit Bézier curves to each of the side strokes using a method proposed by Schneider [43]. Then, we average the two curves by first sampling the same number of equidistant points on each curve and finding the average of each pair of these points, where each point belongs to one of the parametric curves. Next, the center points of the 2D shape that was previously fitted to the beautified strokes of 1 and 4 in Figure 2b are added to these averaged points. We then fit another Bézier curve to the set of points that we just obtained and treat the resulting curve as the 2D projection of the object's spine[2]. We optionally perform a simple line test similar to [36], and if the test passes, replace the 2D curve with the line that passes through the curve's endpoints.

The reason for using Bézier curves in the above procedure is to remove potential noisy points from the input, as well as to obtain the more expressive parametric representation. Also, the benefit of our curve averaging method is that the resulting points will be less sensitive to finer changes in the object's outlines. We have found sampling 10 points on each curve during averaging to produce satisfactory results.

Once the 2D projection of the 3D axis is determined, we may fully recover the 3D spine itself by backprojecting the obtained 2D points into 3D rays using the camera calibration matrix. The intersection point of these rays with the object plane will be the 3D spine points.

F. Profile Propagation and Mesh Creation

The final step of the pipeline is to clone the obtained 3D profile along the direction of the spine, while accounting for the changes in the scale of the profile based on the changes of the object's outlines, as shown in Figure 3c. We may use the 3D spine itself to orient the normal of each copied profile. The challenge is determining the correct scale for each copied profile, since point cloud's points may be noisy or missing altogether.

To work around these issues, we use the user's strokes to determine the correct scale for each copied profile. Once each copy of the profile is created at the correct position, we keep this center position fixed and then compute its 2D projection and scale this 2D projection to meet the outlines of the side strokes drawn by the user at points signified by asterisks in Figure 3c. We may then backproject these points into 3D rays, compute their intersections with the object plane and use the 3D intersection points to determine the correct scaling factor for the copied profile. In our implementation, we densely propagate the profiles such that the distance between the 2D projections of two consecutive profile centers is 1 pixel. We found this to result in a more detailed 3D model.

The final mesh can be trivially created by uniformly sampling points on the circumference of all the 3D profiles and triangulating those points. Optionally, the user may specify one end or both ends of the created mesh to be hollow. This will result in a more natural looking 3D model for shapes such as a cup (see Figure 2e). Further, the mesh may optionally be textured by projecting the visible vertices onto the image and assigning texture values based on RGB pixels. Given that the object's plane is known, we may mirror the object's front texture to the back, in case no view of the back side of the object is available.

[2]Note that perspective projection may affect the alignment between object's spine and its image. In practice, we found this effect to be minimal

IV. Experimental Results

We performed a series of experiments with our tool to examine its capabilities. When performing geometry extraction and object modeling, a question of interest is how similar modeled objects are to their physical counterpart. To answer this, we performed a series of extraction comparisons using the BigBIRD dataset [44]. BigBIRD is a 3D dataset of object instances that contains RGB-D images and reconstructed mesh of 125 objects from 5 different views taken from 120 poses. Although there are other RGB-D datasets available (refer to [45] for a comprehensive list), BigBIRD has a few desirable properties. Namely, it provides accurate intrinsic and extrinsic calibration values, multiple views of each object in different poses as well as ground truth mesh of each object.

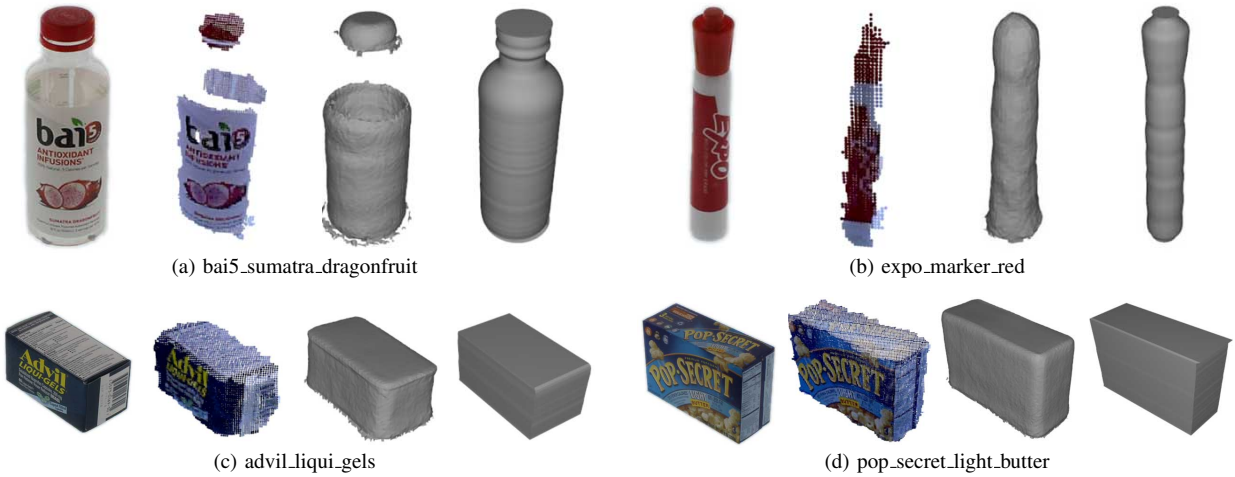
Most objects in BigBIRD are generalized cylinders and cuboids. We carefully selected 16 objects from the dataset which we believe represent a variety of shapes, sizes and surface materials. We extracted all selected objects using our system. For a demo, see the accompanying video. Figure 5 shows a subset of the results of extraction (see the supplementary material for the full set of extraction results). We did not texture-map the meshes in order to be able to better demonstrate the fine details in the extracted model.

To determine the similarity of our extracted meshes compared to the physical objects, we use the ground truth mesh models available in BigBIRD. To compare our extracted models with those meshes, we use Hausdorff distance. Prior to computing the Hausdorff distance between our extracted model and the ground truth, we use the information provided in BigBIRD to transform the two models to a common coordinate system. We next refine the alignment of the two models using ICP, and record the final reported error which is the average distance of the points in one point set to their nearest neighbor in the other point set. Afterwards, we compute and record the diagonal of the oriented bounding box of the combination of the two models. We then compute the Hausdorff distance values between the two models and divide the obtained distance value by the length of the bounding box diagonal that was computed in the previous step.

We also recorded the number of views that were necessary to successfully model each object. Table I summarizes the results of our findings. The minimum and maximum obtained ICP error values were 3.89 and 29.69 millimeters respectively. Also, the minimum and maximum obtained Hausdorff distance values were 3.40% and 7.39% respectively.

V. Discussion

We first examine our results from a qualitative point of view. As shown in Figure 5, our system is able to extract objects



(a) bai5_sumatra_dragonfruit (b) expo_marker_red

(c) advil_liqui_gels (d) pop_secret_light_butter

Fig. 5: Sample object extractions from the BigBIRD dataset. In each object group, the first column is the RGB image of the target object. The second column is the raw point cloud of the object. The third columns is the ground truth mesh as provided by the BigBIRD dataset. The fourth column is the object modeled using our system. Refer to the accompanying video and the supplementary material for a full set of our extractions.

Object Name	Shape	ICP Error (mm)	Hausdorff Distance (%)	Num. Views	Object Name	Shape	ICP Error (mm)	Hausdorff Distance (%)	Num. Views
3m_high_tack_spray_adhesive	◯	6.20	3.40	1	expo_marker_red	◯	3.89	6.95	2
advil_liqui_gels	■	8.04	6.50	1	haagen_dazs_butter_pecan	◯	6.21	4.69	2
bai5_sumatra_dragonfruit	◯	10.62	5.16	2	hunts_sauce	◯	6.56	5.85	1
cheez_it_white_cheddar	■	8.78	5.94	1	krylon_crystal_clear	◯	6.25	4.87	1
cholula_chipotle_hot_sauce	◯	5.82	4.84	2	pop_secret_light_butter	■	9.42	6.37	1
coca_cola_glass_bottle	◯	29.69	7.39	2	red_bull	◯	6.10	4.57	1
coffee_mate_french_vanilla	◯	5.59	4.63	2	red_cup	◯	4.25	7.34	1
dove_beauty_cream_bar	■	6.67	6.67	1	v8_fusion_peach_mango	◯	6.48	5.40	1
					Mean		8.16	5.66	1.31
					Standard Deviation		5.81	1.11	0.46

TABLE I: Quantitative comparison of our extracted models versus the ground truth meshes in the BigBIRD dataset. In the *shape* column, ◯ and ■ indicate generalized cylinders and generalized cuboids, respectively. *ICP Error* is reported as the average distance between the point sets after ICP alignment in millimeters. All values in *Hausdorff Distance* column represent Hausdorff distances scaled by the length of the diagonal of the bounding boxes and converted to percentage values (refer to Section IV for more details). *Num.Views* is the number of different point cloud views used for extracting a particular object.

while preserving their details, such as the small protrusions near the cap of the bottle in Figure 5a. Also, transparent surfaces (Figure 5a) or undersampled areas (Figure 5b) do not pose any problems for us, since the input from the user has the power of filling in the missing details.

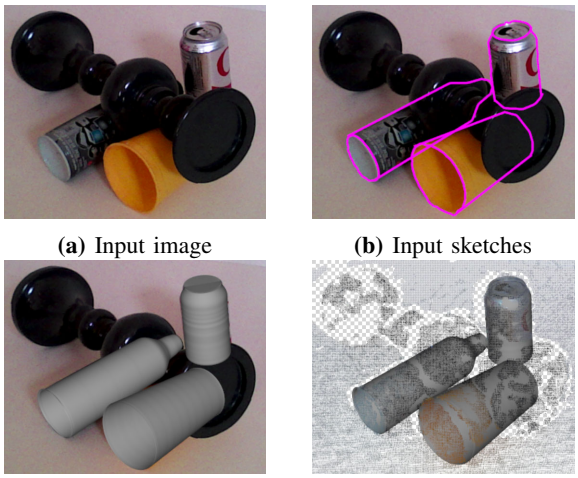
Visual similarity between our extracted meshes and the original objects is much more pronounced compared to that of the ground truth meshes. Moreover, all extracted meshes are created at the scale of the original object which makes our proposed approach suitable for tasks involving object localization in real-world (*e.g.* robotic manipulation [1]).

Examining our results from a quantitative point of view shows that our extracted meshes have high similarity values when compared to the ground truth meshes. Most ICP error values reported in Table I are small indicating that our extracted meshes were successfully aligned with the ground truth meshes. Two objects had relatively higher ICP errors compared to other objects, namely *coca_cola_glass_bottle* and *bai5_sumatra_dragonfruit* which we posit is the result of missing data points on the ground truth mesh due to surface transparencies. The Hausdorff distance values reported in Table I are generally small, indicating that our extracted models are similar to the ground truth meshes. This observation is further corroborated by the small standard deviation values reported in Table I.

Even though many views for each object was available, all results shown in Table I were obtained using at most 2 views. These results are notable considering that the ground truth meshes in BigBIRD were obtained using merged point clouds of 600 views of each object (5 views each containing 120 object poses).

Figure 6 demonstrates the power of GemSketch when dealing cluttered scenes and excessive occlusions. Even though a single view of the scene was available, GemSketch has been able to accurately model all occluded objects

Comparison to related work. As discussed in Section II our method shares similarities with 3-Sweep [8]. However, GemSketch differs from 3-Sweep and extends it in a number of ways. Most importantly, GemSketch is much easier to implement due to the use of point clouds. The existence of 3D



(a) Input image **(b)** Input sketches

(c) Models superimposed on the image **(d)** Models superimposed on the point cloud

Fig. 6: Extracting occluded objects from a cluttered scene. Our proposed method is capable of extracting objects even in the presence of clutter.

data minimizes the need for resolving pose ambiguities of the extracted object through complex optimization formulations. Furthermore, GemSketch supports input from multiple views. Another benefit of using 3D point cloud data is that the extracted objects will be at the scale of their physical counterparts. This permits the application of GemSketch in 3D printing or tasks involving real-world object localization [1].

Although GemSketch requires slightly more user interaction, such interactions permits supporting affordances that are not available in competitive methods. Affordances such as extracting occluded objects even in the presence of clutter (Figure 6) open doors to new possibilities and applications of GemSketch. With additional user engagement, we have been able to mitigate the reliance on automatically detected contours which was one of the limitations of 3-Sweep. Highly textured surface areas pose problems for edge detectors, whereas human input can mitigate ambiguities in such scenarios. Finally, the user can specify the objects to be hollow and GemSketch can extract a hollow mesh (such as the cup in Figure 2e).

Limitations. Our work has several limitations. We currently support generalized cylinders and cuboids, although adding support for other generalized shapes (such as spheres) is possible. Despite the power of generalized cuboids and cylinders [10], [46], they may not be suitable for objects with irregular or asymmetric shapes. Supporting such objects would require novel algorithms.

Viewpoint selection is an important aspect of 3D modeling systems [47]. Unsuitable viewpoints may inhibit the modeling power of any modeling system and GemSketch is no exception. Also, GemSketch is unable to model objects if their profile is occluded. However, the support for multiple views in GemSketch mitigates the effects of unsuitable viewpoints and grants the user the freedom to choose suitable extraction viewpoints.

Finally, extraction of objects under occlusion relies on the user's intuition. In some cases, such intuition could potentially be different from reality, which would affect the accuracy and the correctness of our extracted models. Regardless of these limitations, we believe that the strength of GemSketch lies in its ability to work with noisy and inaccurate point cloud data and extract complete geometries even if a lot of surface points are missing.

VI. Conclusion and Future Work

We introduced GemSketch, an interactive system for extracting 3D geometries from point clouds. Leveraging a human-in-the-loop and an image-guided interaction modality, GemSketch can accurately extract 3D meshes of generalized cylinders and generalized cuboids from point clouds while adhering to the original object's scale and preserving its details. Furthermore, GemSketch can benefit from the availability of multiple views of a single object to perform more robust extractions. GemSketch is capable of working with noisy data and extract objects in the presence of occlusion or clutter.

To demonstrate the merits of our proposed approach, we performed a series experiments. Our experiments were comprised of the extraction of 16 objects from the BigBIRD dataset [44]. Our results indicated a high degree of visual as well as quantitative similarity between the extracted objects and their ground truths.

In the future, we plan to address some of the limitations we discussed previously. Specifically, we wish to add support for more geometric primitives to be able to extract more complex objects. Improving user interaction is an important step towards creating robust human-in-the-loop frameworks for robotic manipulation [48]. As such, we hope to streamline the process of sketching objects by incorporating learning algorithms into the pipeline. A potential learning algorithm would learn from previously extracted objects in order to reduce the amount of user interaction or provide recommendations about camera viewpoints that will make sketching easier and faster. Finally, using GemSketch's ability of modeling object textures, resulting models can be used for generating synthetic scenes. This will benefit the training of neural networks [5] that require large amounts of textured/visual data to learn object models.

Acknowledgment

This work is supported by NSF Award IIS-1638060. We also thank the ISUE lab members at UCF for their support as well as the anonymous reviewers for their helpful feedback.

References

[1] Z. Sui, O. C. Jenkins, and K. Desingh, "Axiomatic particle filtering for goal-directed robotic manipulation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 4429–4436.

[2] K. Desingh, O. C. Jenkins, L. Reveret, and Z. Sui, "Physically plausible scene estimation for manipulation in clutter," in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*. IEEE, 2016, pp. 1073–1080.

[3] V. Narayanan and M. Likhachev, "Discriminatively-guided deliberative perception for pose estimation of multiple 3d object instances." in *Robotics: Science and Systems*, 2016.

[4] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[5] Z. Sui, Z. Zhou, Z. Zeng, and O. C. Jenkins, "Sum: Sequential scene understanding and manipulation," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017.

[6] M. Kowalski, J. Naruniec, and M. Daniluk, "Livescan3d: A fast and inexpensive 3d data acquisition system for multiple kinect v2 sensors," in *2015 International Conference on 3D Vision*, Oct 2015, pp. 318–325.

[7] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, 2011, pp. 127–136.

[8] T. Chen, Z. Zhu, A. Shamir, S.-M. Hu, and D. Cohen-Or, "3sweep: Extracting editable objects from a single photo," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 195:1–195:10, Nov. 2013.

[9] A. Kushal and S. M. Seitz, "Single view reconstruction of piecewise swept surfaces," in *2013 International Conference on 3D Vision - 3DV 2013*, June 2013, pp. 239–246.

[10] L. Nan, A. Sharf, H. Zhang, D. Cohen-Or, and B. Chen, "Smartboxes for interactive urban reconstruction," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 93:1–93:10, Jul. 2010.

[11] S. Tsang, R. Balakrishnan, K. Singh, and A. Ranjan, "A suggestive interface for image guided 3d sketching," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '04. New York, NY, USA: ACM, 2004, pp. 591–598.

[12] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," in *Computer graphics forum*, vol. 26, no. 2. Wiley Online Library, 2007, pp. 214–226.

[13] Y. M. Kim, J. Cho, and S. C. Ahn, "3d modeling from photos given topological information," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 9, pp. 2070–2081, Sept 2016.

[14] D. Li, T. Shao, H. Wu, and K. Zhou, "Shape completion from a single rgbd image," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2016.

[15] M. Firman, O. Mac Aodha, S. Julier, and G. J. Brostow, "Structured prediction of unobserved voxels from a single depth image," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[16] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proc. ICCV*, 2015.

[17] F. M. Carlucci, P. Russo, and B. Caputo, "A deep representation for depth images from synthetic data," in *Robotics and Automation, 2017. ICRA 2017. IEEE International Conference on*. IEEE, 2017.

[18] Z. Lun, M. Gadelha, E. Kalogerakis, S. Maji, and R. Wang, "3d shape reconstruction from sketches via multi-view convolutional networks," in *Proceedings of the International Conference on 3D Vision (3DV)*, 2017.

[19] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3dmatch: Learning local geometric descriptors from rgb-d reconstructions," in *CVPR*, 2017.

[20] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes, "Sketch: An interface for sketching 3d scenes," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 163–170.

[21] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: A sketching interface for 3d freeform design," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 409–416.

[22] Y. Gingold, T. Igarashi, and D. Zorin, "Structured annotations for 2d-to-3d modeling," in *ACM SIGGRAPH Asia 2009 Papers*, ser. SIGGRAPH Asia '09. New York, NY, USA: ACM, 2009, pp. 148:1–148:9.

[23] A. Shtof, A. Agathos, Y. Gingold, A. Shamir, and D. Cohen-Or, "Geosemantic snapping for sketch-based modeling," *Computer Graphics Forum*, vol. 32, no. 2, pp. 245–253, 2013, proceedings of Eurographics 2013.

[24] M. Xu, M. Li, W. Xu, Z. Deng, Y. Yang, and K. Zhou, "Interactive mechanism modeling from multi-view images," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 236:1–236:13, Nov. 2016.

[25] M. Lau, G. Saul, J. Mitani, and T. Igarashi, "Modeling-in-context: User design of complementary objects with a single photo," in *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, ser. SBIM '10. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2010, pp. 17–24.

[26] C. Zou, X. Peng, H. Lv, S. Chen, H. Fu, and J. Liu, "Sketch-based 3-d modeling for piecewise planar objects in single images," *Computers & Graphics*, vol. 46, pp. 130 – 137, 2015, shape Modeling International 2014.

[27] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr, "Videotrace: Rapid interactive scene modelling from video," in *ACM SIGGRAPH 2007 Papers*, ser. SIGGRAPH '07. New York, NY, USA: ACM, 2007.

[28] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003.

[29] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb 1992.

[30] A. Buades, B. Coll, and J.-M. Morel, "Non-local means denoising," *Image Processing On Line*, vol. 1, pp. 208–212, 2011.

[31] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of IEEE International Conference on Computer Vision*, 2015.

[32] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, Nov 1986.

[33] E. N. Mortensen and W. A. Barrett, "Intelligent scissors for image composition," in *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '95. New York, NY, USA: ACM, 1995, pp. 191–198.

[34] E. M. Taranta II, A. Samiei, M. Maghoumi, P. Khaloo, C. R. Pittman, and J. J. LaViola Jr., "Jackknife: A reliable recognizer with few samples and many modalities," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17. ACM, 2017, pp. 5850–5861.

[35] D. K. Prasad, M. K. Leung, and C. Quek, "Ellifit: An unconstrained, non-iterative, least squares based geometric ellipse fitting method," *Pattern Recognition*, vol. 46, no. 5, pp. 1449 – 1465, 2013.

[36] B. Paulson and T. Hammond, "Paleosketch: Accurate primitive sketch recognition and beautification," in *Proceedings of the 13th International Conference on Intelligent User Interfaces*, ser. IUI '08. New York, NY, USA: ACM, 2008, pp. 1–10.

[37] Y. Xiong and J. J. LaViola, Jr., "Revisiting shortstraw: Improving corner finding in sketch-based interfaces," in *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, ser. SBIM '09. New York, NY, USA: ACM, 2009, pp. 101–108.

[38] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[39] G. T. Toussaint, "Solving geometric problems with the rotating calipers," in *Proc. IEEE Melecon*, vol. 83, 1983, p. A10.

[40] J. Schmidt and H. Niemann, "Using quaternions for parametrizing 3-d rotations in unconstrained nonlinear optimization," in *Proceedings of the Vision Modeling and Visualization Conference 2001*, ser. VMV '01. Aka GmbH, 2001, pp. 399–406.

[41] T. Brox, B. Rosenhahn, J. Gall, and D. Cremers, "Combined region and motion-based 3d tracking of rigid and articulated objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 402–415, March 2010.

[42] J. J. Moré, *The Levenberg-Marquardt algorithm: Implementation and theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, pp. 105–116.

[43] P. J. Schneider, "Graphics gems," A. S. Glassner, Ed. San Diego, CA, USA: Academic Press Professional, Inc., 1990, ch. An Algorithm for Automatically Fitting Digitized Curves, pp. 612–626.

[44] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3d database of object instances," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 509–516.

[45] M. Firman, "List of rgbd datasets," http://www0.cs.ucl.ac.uk/staff/M.Firman/RGBDdatasets/, 2017, accessed: July 2017.

[46] Q. Zeng, W. Chen, H. Wang, C. Tu, D. Cohen-Or, D. Lischinski, and B. Chen, "Hallucinating stereoscopy from a single image," *Comput. Graph. Forum*, vol. 34, no. 2, pp. 1–12, May 2015.

[47] C. Li, H. Lee, D. Zhang, and H. Jiang, "Sketch-based 3d modeling by aligning outlines of an image," *Journal of Computational Design and Engineering*, vol. 3, no. 3, pp. 286 – 294, 2016.

[48] K. Desingh, M. Maghoumi, O. C. Jenkins, J. J. LaViola Jr., and L. Reveret, "Object manipulation in cluttered scenes informed by physics and sketching," in *RSS 2016 Workshop: Geometry and Beyond - Representations, Physics and Scene Understanding for Robotics*, University of Michigan, Ann Arbor, MI, 2016.