DOI:10.1145/3007175

Extracting 3D Objects from Photographs Using 3-Sweep

By Tao Chen, Zhe Zhu, Shi-Min Hu, Daniel Cohen-Or, and Ariel Shamir

Abstract

We introduce an interactive technique to extract and manipulate simple 3D shapes in a single photograph. Such extraction requires an understanding of the shape's components, their projections, and their relationships. These cognitive tasks are simple for humans, but particularly difficult for automatic algorithms. Thus, our approach combines the cognitive abilities of humans with the computational accuracy of the machine to create a simple modeling tool. In our interface, the human draws three strokes over the photograph to generate a 3D component that snaps to the outline of the shape. Each stroke defines one dimension of the component. Such human assistance implicitly segments a complex object into its components, and positions them in space. The computer reshapes the component to fit the image of the object in the photograph as well as to satisfy various inferred geometric constraints between components imposed by a global 3D structure. We show that this intelligent interactive modeling tool provides the means to create editable 3D parts quickly. Once the 3D object has been extracted, it can be quickly edited and placed back into photos or 3D scenes, permitting object-driven photo editing tasks which are impossible to perform in image-space.

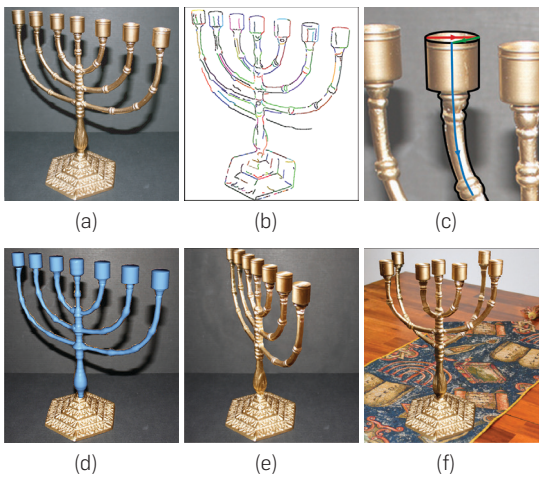
1. INTRODUCTION

Extracting three dimensional objects from a single photo is still a long way from reality given the current state of technology, since it involves numerous complex tasks: the target object must be separated from its background, and its 3D pose, shape, and structure should be recognized from its projection. These tasks are difficult, even ill-posed, since they require some degree of semantic understanding of the object. To alleviate this difficulty, complex 3D models can be partitioned into simpler parts that can be extracted from the photo. However, assembling parts into an object also requires further semantic understanding and is difficult to perform automatically. Moreover, having decomposed a 3D shape into parts, the relationships between these parts should also be understood and maintained in the final composition.

In this paper, we present an interactive technique to extract 3D man-made objects from a single photograph, leveraging the strengths of both humans and computers. Human perceptual abilities are used to partition, recognize, and position shape parts, using a very simple interface based on triplets of strokes, while the computer performs tasks which are computationally intensive or require accuracy. The final object model produced by our method includes its geometry and structure, as well as some of its semantics. This allows the extracted model to be readily available for intelligent editing, which maintains the shape's semantics (see Figure 1).

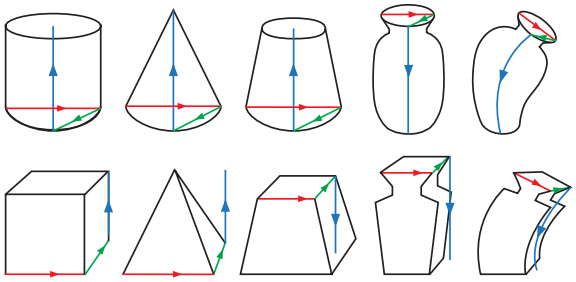
Our approach is based on the observation that many man-made objects can be decomposed into simpler parts that can be represented by generalized cylinders, cuboids, or similar primitives. A generalized cylinder is a cylindrical primitive shape where the central axis is a curve instead of a line, the shape's profile can be any 2D closed curve and not just a circle, and this shape can also change along the curve. In this work, we use just circular and cuboid profiles. The key contribution of our method is an interactive tool that guides and assists the user in the creation of a 3D editable object by defining its primitive parts. The tool is based on a rather simple modeling gesture we call *3-Sweep*. This gesture allows the user to explicitly define the three dimensions of a geometric primitive using three sweeps. The first two sweeps define the first and second dimension of a 2D profile and the third, usually longer, sweep is used to define the main curved axis of the primitive (see Figure 2).

Figure 1. 3-Sweep Object Extraction. (a) Input image. (b) Extracted edges. (c) 3-Sweep modeling of one component of the object. (d) The full extracted 3D model. (e) Changing the object viewpoint. (f) Editing the model by rotating each arm in a different direction, and pasting onto a new background. The base of the object is transferred by alpha matting and compositing.



The original version of this paper is entitled "3-Sweep: Extracting Editable Objects from a Single Photo" and was published in *ACM Transactions on Graphics*, Volume 32, Issue 6—*Proceedings of ACM SIGGRAPH Asia 2013*, November 2013 Article No. 195.

Figure 2. The 3-Sweep paradigm is used to define general cylinder and cuboid parts.



As the user sweeps the primitive, the program dynamically adjusts the progressive profile by sensing the pictorial context in the photograph and automatically snapping to it. Furthermore, relationships between the various primitive parts are automatically recognized and preserved by the program. Using several such 3-Sweep operations, the user can model 3D parts consistent with the object in the photograph, while the computer automatically maintains global constraints linking them to other primitives comprising the object. Using 3-Sweep technology, non-professionals can extract 3D objects from photographs. These objects can then be used to build a new 3D scene, or to alter the original image by changing the objects or their parts in 3D and pasting them back into the photo.

2. BACKGROUND

Modeling from images. Images have always been an important resource for 3D modeling. Many techniques use multiple images or videos to model 3D shapes and scenes.[20] However, our focus is on modeling objects from a single photograph. This task is challenging because of the inherent ambiguity of the mapping from a 3-dimensional world to a 2-dimensional image. To overcome this, methods use both constraints on the type of images that can be used—such as non-oblique views, and prior assumption on the possible geometry that can be extracted.

Fully automatic methods use assumption such as symmetry,[23] smoothness,[18] or the existence of a prior 3D model similar to the one in the photograph.[22] Some limit the geometry to planes or smooth surfaces, while others limit the application range (e.g., to architectural models[3, 10]). There are methods that automatically fit edges or regions in an image or sketch using optimization.[15] These are basically 2D methods, generating either 2D models or 3D models by extrusion. Our method can directly generate complex oblique shapes in 3D space such as the menorah in Figure 1 and the telescope in Figure 7. Automatic methods would fail on such examples since their prior assumptions are not met, or because they rely on region color or clear edges, which can be missing or occluded.

Other methods use a human-in-the-loop for modeling,[16] but usually require extensive manual interaction, and are more time consuming compared to the 3-Sweep approach. They either require a complete sketch of the object or tedious labeling before the machine takes control of an optimization process, while in 3-Sweep user guidance and automatic snapping are interlaced.

Sketch-based modeling. The task of 3D modeling from a single image is closely related to the reconstruction or definition of a 3D shape from a sketch.[17] The user either directly draws the curves of the object[1] or fits parts or primitives to a predefined sketch.[12] Our work was inspired by a tool for modeling simple 3D objects from sketches presented by Shtof et al.[19] In that work geo-semantic constraints were used between primitive parts to define their semantic and geometric relationships, and connect them to form the final object. However, their approach is geared towards sketches and uses a drag and drop interface for primitives that need to fit the sketch contours.

Computer aided design. The use of constraints in computer-aided design has been studied extensively and allows the definition of semantic information that relates different geometric parts in an object. Automatically inferring constraints from the object or its parts' geometry has been used for reverse engineering[5] and object deformation and editing.[11, 21] Similarly, sweep-based models have been defined and used since the beginning of the field.[9] While we cannot report all computer-aided design work aiming at modeling sweep-based primitives, to our knowledge, none of these methods have been applied to modeling from photographs, nor they have paired sweeping with snapping to image edges.

Object-based image editing. Apart from modeling a 3D object, 3-Sweep allows the application of object-based image editing operations, that previously required extensive user interaction[8] or massive data collection.[13, 14]

3. OVERVIEW

Our interactive modeling approach takes as input a single photo such as the one in Figure 1a. Our goal is to extract a 3D model whose projection exactly matches the object in the image. Using the *3-Sweep* modeling technique, the user constructs the whole object gradually from its parts. This means that the object is implicitly decomposed into simple parts, which are typically semantically meaningful. We define two types of primitives: both use a piecewise linear centerline to sweep a cross-section which is assumed to be perpendicular to the centerline at every location. One type of primitive uses a circular cross-section that can vary in radius along the sweep, and the other uses a rectangle cross-section that can change its aspect ratio along the sweep.

Such decomposition is both easy and intuitive for users, but provides the computer significant information for reconstructing a coherent 3D man-made object from its parts' projections. The parts are expected to have typical geometric relationships that can be exploited to guide the composition of the whole object. Although the user interacts with the given photo, she does not need to exactly fit the parts to the photo or connect them to each other. 3-Sweep automatically snaps primitive parts to object outlines created from edges, and connects them to previously defined 3D parts.

To create a single 3D primitive part on the given photo, the designer uses three strokes. The first two strokes define a 2D profile of the part and the third stroke defines its main axis, which is either straight or curved (see Figure 2). Defining the profile and sweeping the axis are simple operations since

they do not demand accuracy. The profile dimensions are guided by the object's outlines. While sweeping, the 3D extent of the part is also defined by snapping to these outlines. To compensate for perspective distortion, during this process, the camera's angle of view is estimated. Therefore, the part need only be sketched quickly and casually by the user. Figure 1c shows the result of sweeping along the tubes of the menorah (more examples could be seen in an online video at https://vimeo.com/148236679). We elaborate on the 3-Sweep operation in Section 4.

As more model parts are added, geometric relationships between them serve (i) to assist in disambiguating and defining the depth dimension and (ii) to optimize the positioning of the parts. These geometric relationships include parallelism, orthogonality, collinearity, and coplanarity. We use optimization to satisfy these geo-semantic constraints, while taking into account the snapping of the 3D geometry to the object's outlines and the user's sweeping input. A complete model with geo-semantic relationships is shown in Figure 1d. These geo-semantic relationships not only help define the 3D model, but become part of the 3D representation of the model, allowing smart editing of the 3D model later, as demonstrated in Figure 1f and in other figures in the paper.

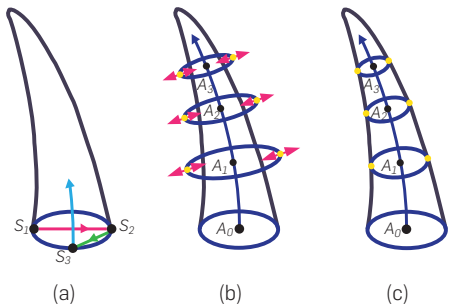
Our interface also supports several operations for more effective modeling. For example, the user can copy and paste similar parts to other positions on the image. Although many geo-semantic constraints are automatically inferred, the user can also manually specify constraints between selected parts, constrain the parts to have uniform or linearly changing radii etc.

4. SINGLE PRIMITIVE FITTING

In this section, we first describe the 3-Sweep technique for creating one generalized cylinder. Simpler primitives such as spheroids or simple cubes are also supported by direct modeling in our system.

Preprocessing. In a preprocessing stage, we extract image edges and build candidate object outlines. We adopt a method for hierarchical edge feature extraction based on spectral clustering.[2] We then apply a technique to link the detected edge pixels into continuous point sequences,[7] each shown in a different color in Figure 1b. An edge orientation computed over a 5×5 neighborhood is associated with each edge pixel.

Figure 3. Modeling a primitive by defining a 2D profile and sweeping it along the main axis of the object (a). The profile is copied along the centerline (b), and the copies are snapped to the image edges (c).



Profile definition. In the first stage, the user draws the 2D profile of the generalized cylinder, usually at one end of the shape. This is illustrated in Figure 3, where black curves are outlines detected in the input image. The task is to draw a 2D profile correctly oriented in 3D. This can be regarded as positioning a disk in 3D by drawing its projection in 2D. To simplify this task, we assume that the disk is a circle, thus reducing the number of unknown parameters. Later, the circular disk can be warped into an elliptical disk based on the 3D reconstruction. The drawing of a circular disk is accomplished by drawing two straight lines S_1S_2 and S_2S_3 over the image, as shown in Figure 3a (red and green arrows). The first line defines the major diameter of the disk, and the second line is then dragged to the end of the minor diameter. This forms an ellipse in image space that matches the projection of a circular disk: see the dark blue circle in Figure 3a. The depth of the disk is set to 0. The normal direction and radius of the disk are assigned according to the length and orientation of the two diameters of the elliptical projection.

Generalized cuboids are modeled in a similar way. The two strokes that define the profile of a cuboid follow the two edges of the base of the cuboid instead of the diameters of the disk, as shown by the red and green lines in the bottom row of Figure 2.

Sweeping. After completing the base profile, the user sweeps it along a curve that approximates the main axis of the 3D part. In general, this curve should be perpendicular to the profile of the 3D primitive (see blue arrow in Figure 3a). As the curve is drawn, copies of the profile are placed along the curve, and each of them is snapped to the object's outline.

While sweeping, the axis curve is sampled in 2D image space at a uniform spacing of five pixels to produce 3D sample points $A_0, \ldots, A_N$, that lie on one plane. At each sampled point A_i, a copy of the profile is created, centered around the curve. Its normal is aligned with the orientation of the curve at A_i, and its diameter is adjusted so that it's projection on the image will fit the object's outline. Together, the adjusted copies of the profile in 3D form a discrete set of slices along the generalized cylinder, as shown in Figure 3c.

At each point A_i, we first copy the profile from A_{i-1} and translate it to A_i. We then rotate it to take into account the bending of the curve. We then consider the two ends of the major axis (yellow points in Figure 3b) of the profile, denoted by p_i^0, p_i^1. For each contour point $p_i^j, j \in [0, 1]$ we cast a 2D ray from point A_i along the major axis, seeking an intersection with the object outline. Finding the correct intersection is somewhat challenging as the image may contain many edges in the vicinity of the new profile. The closest edge is not necessarily the correct one, for example, when hitting occluding edges. In other cases, the correct outline may be missing altogether. To deal with these cases, we first limit the search for an intersection to a fixed range, which limits the major axis of adjacent profiles not to vary by more than 20% in length. Secondly, we search for an intersecting edge that is not collinear to the ray (creates an angle larger than $\pi/4$). Although this method cannot guarantee that it will find the correct intersections, the

subsequent profile propagation step can tolerate a limited number of missing intersections.

When an intersection point is found, we snap the contour point p_i^j to it. If both contour points of the profile are snapped, we adjust the location of A_i to lie at their midpoint. If only one side is successfully snapped, we mirror the length of this side to the other side and move the other contour point respectively. Lastly, if neither contour points is snapped, the size of the previous profile is retained.

Post-processing. The above modeling steps closely follow user gestures, especially when modeling the profile. This provides more intelligent understanding of the shape but it is less accurate. Therefore, after modeling each primitive, we apply a post-snapping stage to better fit the primitive to the image as well as to correct the view. We search for small transformations (±10% of primitive size) and changes of vertical angle of view (±10°) that create a better fit of the primitive's projection to the edge curves it was snapped to in the editing process.

In many cases, the modeled object type has special properties that can be used as priors to constrain the modeling. For example, if we know that a given part has a straight spine, we can constrain the sweep to progress along a straight line. Similarly, we can constrain the sweep to preserve a constant or linearly changing profile radius. In this case, the detected radii are averaged or fitted to a line along the sweep. We can also constrain the profile to be a square or a circle. In fact, a single primitive can contain segments with different constraints: it can start with a straight axis and then bend, or use a constant radius only in a specific part. Such constraints are extremely helpful when the edge detection provides poor results.

To further assist in modeling interaction, we also provide a copy and paste tool. The user can drag a selected part that is already snapped over to a new location in the image and snap it again in the new position. While copying, the user can rotate, scale, or flip the part.

5. COMPOSITE OBJECT CONSTRUCTION

The technique described above generates parts that fit the object outlines. The positions of these parts in 3D are still ambiguous and inaccurate. However, the assumption is that these parts are components of a coherent man-made object, and semantic geometric relationships exist among them. Constraining the shape to satisfy such relationships allows creation of meaningful models.

Since each component has many degrees of freedom, direct global optimization of the positions of parts while considering their geo-semantic relationships is computationally intensive and vulnerable to trapping in local minima. In our setting, the modeled components are also constrained to agree with the outlines of the object in the image. These constraints can significantly reduce the degrees of freedom for each part, reducing the dimensionality of the optimization space and avoiding local minima. In the following discussion, we describe how we simplify the general positioning problem and ensure that geo-semantic constraints are satisfied among the 3-swept parts.

The key idea is that if the projection of a part is fixed, its position and orientation can be determined by only one or two depth values. We first describe the method for simple parts that can be modeled by a single parameter, namely parts which were modeled using a *straight* axis. General cylinders and cuboids with curved axes will be later approximated using two arbitrarily connected straight axis primitives at the start and end of the whole part.

Determining straight shapes. The position and orientation of a generalized cylinder i with a straight-axis can be determined by two points we call *anchors*, $C_{i,1}$ and $C_{i,2}$, on its main axis (see Figure 4). Similarly, a cuboid part can be represented by six anchors $\mathbf{C}_{i,j}\, j \in [1, 6]$ positioned at the center of each face. Every opposite pair of anchors defines one main axis of the cuboid. Even though four anchors are enough to fix the position and orientation of a cuboid, we use six to simplify attaching various geo-semantic constraints from other parts to each side of the cuboid.

We define a local 3D orthogonal coordinate system for each part using the three strokes defined by the user for the three dimensions of the part. First, we define the origin of the coordinate system of part i at a reference point $\mathbf{R}_i$ on the part's projection. For a cuboid part, we pick the point connecting the first and second user strokes, and for a cylinder we pick the point connecting the second and third strokes. Due to the internal orthogonality of the straight part, the profile of the part is perpendicular to the main axis. Therefore, we can use the endpoints of the user's strokes (after snapping them to the image edges) to define three points that together with $\mathbf{R}_i$ create an orthogonal system (red points and lines in Figure 5). Note that this coordinate system is defined

Figure 4. Three examples where we infer geo-semantic constraints from primitives where such relationships "almost" hold: Collinear axes (left), Parallel axes (top right), and Perpendicular axes (bottom right).

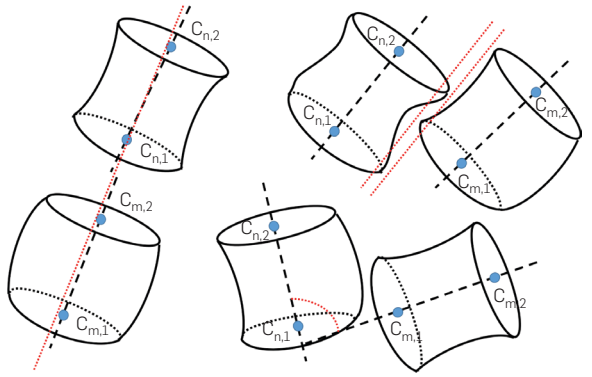
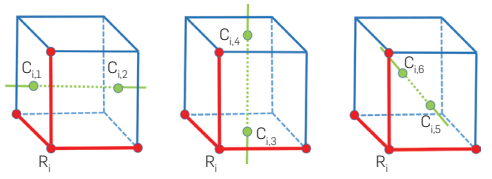


Figure 5. Determining coordinates $\mathbf{C}_{i,j}$ for axis endpoints of a cuboid from the depth value z_i of the reference point $\mathbf{R}_i$.



in camera coordinates. The x and y values of the end points are determined by the projection and their depth values can be found as a function of z_i, the z value of $\mathbf{R}_i$, by using three orthogonality constraint equations.

Next, the positions of the anchor points $\mathbf{C}_{i,j}$ in world coordinates are defined using the local orthogonal axes, giving the structure of part i. Since the local axes depend only on the depth value z_i of the point $\mathbf{R}_i$, we can parameterize the positions of $\mathbf{C}_{i,j}$ as a function of z_i: $\mathbf{C}_{i,j} = \mathbf{F}_{i,j}(z_i)$: the position and orientation of the whole part become a function of a single unknown z_i. $\mathbf{F}_{i,j}$ has the form $F_{i,j}(z_i) = b/(a(z_i + v))$ for each coordinate component, where a depends only on the x and y coordinates of the endpoints of the local axes, and b, v are decided by perspective parameters. They are different for each axis endpoint and for each coordinate component (see Ref.[6] for the full definition).

Defining geo-semantic constraints. We use the anchor points to define the geo-semantic relationships between parts. Specifically, we support six types of constraints: parallelism, orthogonality, and collinearity of primitive axes, overlapping endpoints of axes, coplanar axis endpoints, and coplanar axes. During modeling, for each type, we test whether a pair of components is close to satisfying one of the above geo-semantic constraints. If so, we add this constraint to the definition of the object (see Figure 4). For example, for two cylinders with indices m and n, if the angle between vector $(\mathbf{C}_{m,1} - \mathbf{C}_{m,2})$ and $(\mathbf{C}_{n,1} - \mathbf{C}_{n,2})$ is less than 15°, we add a parallelism constraint $(\mathbf{C}_{m,1} - \mathbf{C}_{m,2}) \times (\mathbf{C}_{n,1} - \mathbf{C}_{n,2}) = 0$ to the system of constraints. Similarly if any three of the four anchors of two cylinders form a triangle containing an angle larger than 170°, we add a collinear axes constraint: $(\mathbf{C}_1 - \mathbf{C}_2) \times (\mathbf{C}_1 - \mathbf{C}_3) = 0$. Internal constraints such as orthogonality and concentricity of a cuboid's axes are also added to the definition, and some editing operations such as copying and pasting parts (see Section 6) can impose same-size constraints. Finally, we also allow the user to manually enforce or revoke any constraint for selected primitive parts.

Establishing an objective function. Suppose we have found p geo-semantic constraints G_k for a set of n components. Together with the objective function for fitting the image outline, we define the following optimization system:

$$\text{minimize } E = \sum_{i=1}^{n} w_i \left(\sum_{j=1}^{mi} ||\, \mathrm{C}_{i,j} - \mathrm{F}_{i,j}(\mathrm{z}_i) \,||^2 \right), \qquad (1)$$

$$\text{subject to} \quad G_k(\mathbf{C}_{1,1}, \ldots, \mathbf{C}_{n,m_n}), \quad k = 1, \ldots, p, \qquad (2)$$

where m_i is the number of axes of the ith primitive part. We add weights w_i proportional to the radius of the base profile of each part and the length of its axis. Larger parts have more impact on the solution since typically larger parts are modeled more accurately. Intuitively, the first equation tries to fit projection of the part's geometry ($\mathbf{C}_{i,j}$) to the image outline and the user's gestures, while the second set of equations imposes the geo-semantic constraints.

Two-step solution. Solving for $\mathbf{C}_{i,j}$ and z_i together is a nonlinear non-convex optimization problem with nonlinear constraints. Directly solving such a system without becoming trapped in a local minimum is very difficult. Hence, we decompose the solution into a two step procedure. The first step tries to find a good initial position for all parts at once, by changing only their depths (governed by z_i) to meet the geo-semantic constraints. In the second step, the full system is solved, allowing the shapes of the parts ($\mathbf{C}_{i,j}$) to change as well.

In the first step, we modify the soft constraint in Equation (1) to a hard one, and replace $\mathbf{C}_{i,j}$ by $F_{i,j}(z_i)$ in all equations. This means that Equation (1) is trivially true and we are left with just the constraints in Equation (2). In effect, this means we fix the projection and find the optimal z_i meeting the geo-semantic constraints. This reduces the number of variables to n (z_i, $1 \leq i \leq n$) and changes Equation (2) into a potentially over-determined system, where each equation only contains two different variables. We find the least squares solution $\bar{z}_i$ by using the conjugate gradient method, with all z_i values initialized to 0.

This first step provides a good initialization to find the optimal solution for $\mathbf{C}_{i,j}$, which should be close to $\mathbf{F}_{i,j}(\bar{z}_i)$, requiring only small inconsistencies to be fixed with the geo-semantic constraints. Hence, in the second step, we carry out full optimization of Equation (1) with the set of constraints in Equation (2) by an augmented Lagrangian method. Both steps are fast, and we are able to avoid local minima by the initialization of the first step. This also permits optimization to be carried out at interactive speed (see examples at https://vimeo.com/148236679). Note that the nonlinearity of $\mathbf{F}_{i,j}(\,)$ arises due to the assumption of a perspective projection. However, we can linearly approximate this projection since we assume the change in z_i is small. This further increases the speed and stability of our solution.

Curved shapes. To handle parts with a non-straight axis, we first simplify the problem by assuming that the general axis lies in a plane. We define a non-straight part as a blend of two straight-axis sub-parts, placed at the two ends of the part. The position of each of these sub-parts is determined by a single depth value in the optimization above. The blend between these parts for the whole part is defined by connecting the two subparts with a piecewise linear curve where the axis points are determined when constraining the profile while snapping it during the user's sweep. More details can be found in Ref.[6]

Texturing. Once the object has been modeled, we can map the texture from the image onto the object, as shown in Figure 6. By projecting a vertex of the mesh to the image plane, we can get the 2D coordinates of the vertex in the image, which are then used as texture coordinates to map the corresponding part of the image onto the model. Since there is no information regarding the back of the object, we simply use a symmetry assumption and mirror the front texture to the back. In each half of the model (front and back), we assign the same texture coordinate for the two vertices that are mirrored symmetrically about the central plane. On the two sides of the object (left and right), there are vertices whose normal is perpendicular, or almost perpendicular, to the image plane. To deal with such vertices, we treat the texture associated with them as holes, and use the image completion technique[4] to fill them.

6. EXPERIMENTAL RESULTS

We implemented the 3-Sweep interactive technique in C++ inside a simple modeling system. The system provides an

Figure 6. Modeling different objects: a table (a), a lamp (b), a monument (c), a samovar (d) and a menorah (e). Top: input photos. Middle: extracted 3D models (blue) are rotated and repositioned. Bottom: modified objects inserted into the same or a new environment, with their textures.

(a) (b) (c) (d) (e)

outline view for 3-Sweep interaction, a solid model view, and a texture view for checking the model and for image editing. The user can choose between cuboid, cylinder, and sphere primitives using a button or key shortcut. The system also provides conventional menu selection, view control and deformation tools. The 3-Sweep technique has been tested and evaluated on a large number of photos as we demonstrate in this section and in the online video (see https://vimeo.com/148236679). As shown in the video, most of the examples were modeled in a few minutes or less. The modeling process is intuitive and fluent, and can be used by unskilled people following very little training. Editing and repositioning an object after modeling requires an effort similar to using other parametric editing techniques.

6.1. Modeling from single image and editing

In the following examples, we show how the acquired 3D textured-model allows semantic image editing. Before editing, the image of the 3D model is cut out from the photo, leaving a black hole which is filled using a standard image completion technique.[4]

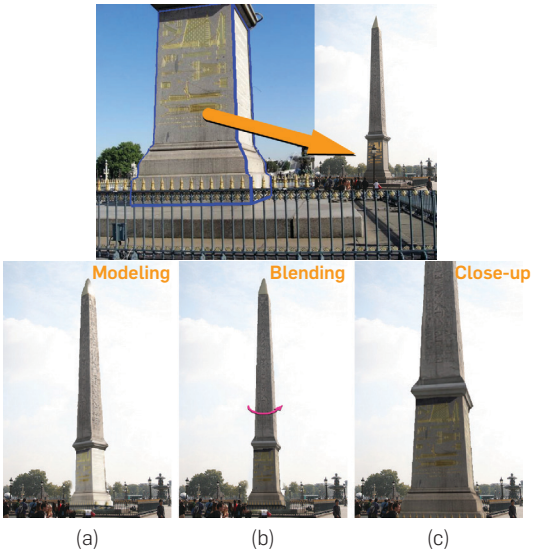
Figure 1e demonstrates the modeling of a menorah, and then rotating its arms to a different angles preserving the inter-part relationships of the object. Note that all candle holders have the same size, but due to the oblique view, their sizes appear different in the photo. During modeling, we modeled one candle holder and copied it to fit all other holders in the image, while requiring that they all lie on the same plane and that their 3D sizes be the same. This efficiently recovered the true 3D position and shape of each part.

Figure 6 shows several modeling results. In the top row we show the input photos, in the middle row we show the extracted and repositioned 3D models, and in the third row, they are inserted with their textures into the same or a new environment. The rightmost column shows the modeling and repositioning of three objects in one complex photo. Note that the menorah has been rotated, and translated on the ground plane.

Figure 7 shows three examples of modeling and editing at the part-level, where some parts of the objects (highlighted in gold) are replicated, copied, and optionally rotated to enhance and enrich the shape. At top left is a tap, whose handle is augmented to be four-sided, and also rotated. The whole tap is also copied and attached to the other side of the wall. The top right shows a street lamp with duplicated lamps moved to a lower position and rotated. The whole lamp pole is also copied to other positions on the street. The bottom row shows different editing operations carried out on a telescope after modeling it. Note that different scaling factors have been applied to the different telescope parts.

Figure 7. Top: modeling and replicating parts for image editing. Orange parts are replicated or deformed. Bottom: editing a telescope. The leftmost images are the original photos. Note that different parts have been scaled differently.

Figure 8. Modeling the Obelisk in Paris from two photos. Top: the base of the Obelisk is modeled from a closer view which captures more details. Bottom: (a) The partial 3D model is transported to a more distant view (in which part of the base is occluded). (b) A rotated textured Obelisk; the texture of the transported part is blended into the region it occupied. (c) Details of the base are visible in the close-up of the new view.



In Figure 8, we show a case where two input photos are used to create one model of an object: the Obelisk in Paris. Firstly, the base of the Obelisk is modeled from a close up view in (a), allowing more detail to be captured. The partial 3D model is then moved to another photo where the entire Obelisk is visible, but the base is occluded. Similar to a copy and paste procedure, the user positions the extracted base inside the image, and it snaps to the image contours in (b). The user then continues the modeling process with other parts. The texture of the transported part is blended to match the shading of the region in the new image, to maintain consistency: see the rotated view (c). Details of the base can be seen in the close up view (d) of the final model of the Obelisk.

In Figure 9, we show a photograph with a collection of objects that were modeled and copied from other photos. The online video (https://vimeo.com/148236679) shows the process of modeling and editing these objects. The modeling and editing time for each example is shown in Table 1, as well as the number of manually provided geo-semantic constraints. Objects in oblique views typically need more manual constraints, most of which designate coplanar axes, which are difficult to infer automatically.

6.2. Comparison to sketch based modeling

As discussed in Section 2, our method shares some similarities with the one in Shtof et al.,[19] which models objects from sketches. We previously discussed the main differences between the methods. Their system is based on sketches rather than photographs, which makes it easier to assume that parts have sufficient bounding curves around them. It relies on labeling, using a drag and drop metaphor for choosing and positioning the primitives before snapping with the sketches. We make a comparison based on their sketch inputs (see Figure 10), since their method cannot handle the examples presented in this paper. Comparing the modeling time shows that sketch labeling and drag and drop snapping steps are significantly less efficient and less intuitive compared to our 3-Sweep method. Our modeling time (60s on average) is significantly lower than the time they report for their technique (180s on average).

6.3. Limitations

Our work has several limitations. First, many shapes cannot be decomposed into generalized cylinders and cuboids, and cannot be modeled using our framework (e.g., the base of the menorah in Figure 1). It would be desirable to extend the types of primitives which can be modeled using similar principles. 3-Sweep also relies on the fact that the object modeled

Figure 9. A rendered scene using source images from the top image strip.

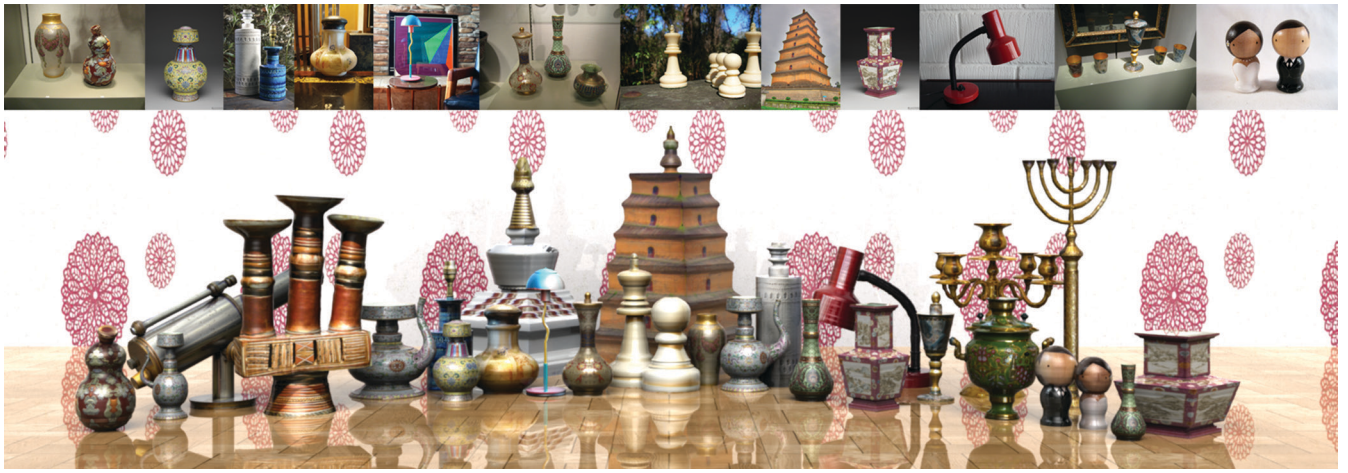
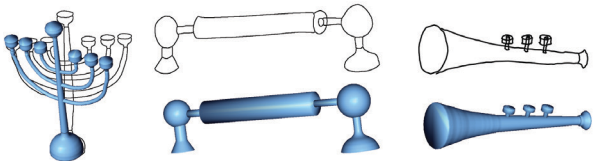


Table 1. Modeling + Editing times (in seconds), and the number of manually provided geo-semantic constraints (added or removed) for each example.

Figure	1			6			8		7			9	
Example	Menorah	(a)	(b)	(c)	(d/e)	(f)	Obelisk	Tap	Lamp	Telescope	Trumpet	Handle	Horn
Time (s)	80 + 25	75 + 15	20	35	30	65 + 35	20	30 + 25	40 + 50	100 + 30	80	30	60
Constraints	2	4	2	1	1	1	0	2	1	2	1	1	1

Figure 10. Modeling from sketches. Input sketches are taken from Ref.[19]



includes typical relationships among parts such as symmetry, parallelism, and collinearity. More free-form objects, or parts, that differ from this would need to be positioned by hand.

Even for a generalized cylinder there is sometimes ambiguity. We assume that the profiles of the cylinder are uniformly scaled and do not rotate around the main axis. This assumption is not always satisfied, as demonstrated in Figure 11b. We further assume that the main axis is mostly visible and parallel to the viewing plane. Using a perspective assumption, we can handle a small amount of skew, but not a large one.

The snapping algorithm relies on good detection of object edges and assumes the main part of the object is not occluded. Objects that are too small, such as the cross in Figure 6e, or objects that have fuzzy edges such as the example in Figure 11c are hard to model accurately. The photographs themselves often have some distortions from an ideal perspective projection (see Figure 11a), where corrections should be applied before modeling. Lastly, our editing assumes a simple illumination model without shadows. Relighting and shadow computations are not currently supported by our system.

Figure 11. Failures. (a) Due to perspective projection, the table legs cannot be snapped to the image under a parallelism constraint. (b) Due to the assumption of uniformly scaling profiles, the bottom of the toothpaste tube is not flattened. (c) Snapping fails due to an ill-defined edge caused by the shadow cast by the bottle.

(a) (b) (c)

7. CONCLUSION

We have presented an interactive technique which can model 3D man-made objects from a single photograph by combining the cognitive ability of humans with the computational accuracy of computers. The 3-Sweep technique is designed to allow extracting an editable model from a single image. The range of objects that our technique can support are objects that consist of simple parts, without much occlusion. As we demonstrated, this range is surprising large to be useful for interactive modeling—our tests show that our method can model a large variety of man-made objects in photographs, as well as objects in sketches. The modeled objects can be edited in a semantically meaningful way, both in the original image, or for use in composing new images. In the future, we hope to extend the range of primitives, and to allow modeling of the

freer shapes of natural objects. We also wish to add symmetry and smoothness constraints on the shapes. 3-Sweep could also be extended to allow modeling from multi-view images or video, without the help of depth data. The applications demonstrated mainly show editing and manipulation of geometry, but the recovered 3D models and surface normals can be used for re-lighting and material editing. C

References

1. Andre, A., Saito, S. Single-view sketch based modeling. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling (SBIM '11)* (2011). ACM, New York, NY, USA, 133–140.
2. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J. Contour detection and hierarchical image segmentation. *IEEE Trans. Patt. Anal. Mach. Intell. 33*, 5 (2011), 898–916.
3. Arikan, M., Schwärzler, M., Flöry, S., Wimmer, M., Maierhofer, S. O-snap: Optimization-based snapping for modeling architecture. *ACM Trans. Graph. (TOG) 32*, 1 (2013), 6.
4. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.-TOG 28*, 3 (2009), 24.
5. Benko, P., Kós, G., Várady, T., Andor, L., Martin, R. Constrained fitting in reverse engineering. *Comput. Aided Geom. Des. 19*, 3 (2002), 173–205.
6. Chen, T., Zhu, Z., Shamir, A., Hu, S.-M., Cohen-Or, D. 3sweep: Extracting editable objects from a single photo. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 1–195.
7. Cheng, M. Curve structure extraction for cartoon images. In *Proceedings of the 5th Joint Conference on Harmonious Human Machine Environment* (2009), 13–25.
8. Cheng, M., Zhang, F., Mitra, N., Huang, X., Hu, S. Repfinder: Finding approximately repeated scene elements for image editing. *ACM Trans. Graph. (TOG) 29*, 4 (2010), 83.
9. Choi, B., Lee, C. Sweep surfaces modelling via coordinate transformation and blending. *Comput.-Aided Des. 22*, 2 (1990), 87–96.
10. Debevec, P.E., Taylor, C.J., Malik, J. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)* (1996). ACM, New York, NY, USA, 11–20.
11. Gal, R., Sorkine, O., Mitra, N., Cohen-Or, D. iwires: An analyze-and-edit approach to shape manipulation. *ACM Trans. Graph. (TOG) 28* (2009) 33.
12. Gingold, Y., Igarashi, T., Zorin, D. Structured annotations for 2d-to-3d modeling. *ACM Trans. Graph. (TOG) 28* (2009) 148.
13. Goldberg, C., Chen, T., Zhang, F., Shamir, A., Hu, S. Data-driven object manipulation in images. *Comput. Graph. Forum 31* (2012) 265–274.
14. Lalonde, J., Hoiem, D., Efros, A., Rother, C., Winn, J., Criminisi, A. Photo clip art. *ACM Trans. Graph. (TOG) 26* (2007) 3.
15. Mille, J., Boné, R., Cohen, L.D. Region-based 2d deformable generalized cylinder for narrow structures segmentation. In *Proceedings of Computer Vision–ECCV 2008: 10th European Conference on Computer Vision*, Forsyth, D., Torr, P. and Zisserman, A. (eds) (Marseille, France, October 12-18, Part II, 2008). Springer, Berlin, Heidelberg, 392–404.
16. Oh, B., Chen, M., Dorsey, J., Durand, F. Image-based modeling and photo editing. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)* (2001). ACM, New York, NY, USA, 433–442.
17. Olsen, L., Samavati, F., Sousa, M., Jorge, J. Sketch-based modeling: A survey. *Comput. Graph. 33*, 1 (2009), 85–103.
18. Oswald, M.R., Toppe, E., Cremers, D. Fast and globally optimal single view reconstruction of curved objects. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012). IEEE, 534–541.
19. Shtof, A., Agathos, A., Gingold, Y., Shamir, A., Cohen-Or, D. Geosemantic snapping for sketch-based modeling. Volume 32. In *Eurographics* (2013), 245–253.
20. Snavely, N. Scene reconstruction and visualization from internet photo collections: A survey. *IPSJ Trans. Comput. Vision Appl. 3* (2011), 44–66.
21. Xu, K., Zhang, H., Cohen-Or, D., Chen, B. Fit and diverse: Set evolution for inspiring 3d shape galleries. *ACM Trans. Graph. (TOG) 31*, 4 (2012), 57.
22. Xu, K., Zheng, H., Zhang, H., Cohen-Or, D., Liu, L., Xiong, Y. Photo-inspired model-driven 3d object modeling. *ACM Trans. Graph. (TOG) 30* (2011) 80.
23. Xue, T., Liu, J., Tang, X. Symmetric piecewise planar object reconstruction from a single image. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2011). IEEE, 2577–2584.

Tao Chen and Daniel Cohen-Or (taochen@ee.columbia.edu) Tel Aviv & Tsinghua University, Israel & Tsinghua University, Beijing, China.

Zhe Zhu and Shi-Min Hu (ajex1988@gmail.com, shimin@tsinghua.edu.cn) Tsinghua University, Beijing, China.

Ariel Shamir (arik@idc.ac.il) The Interdisciplinary Center, Herzliya, Israel.