EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2008)
C. Alvarado and M.- P. Cani (Editors)

Sketch-Based Search and Composition of 3D Models

Jeehyung Lee and Thomas Funkhouser

Princeton University

Abstract

There is growing interest in developing tools with which novice users can create detailed 3D models of their own designs. The most popular approaches to this problem include sketch-based interfaces and part-composition systems. The sketch-based modeling systems provide natural interfaces for creating 3D models from 2D sketches, but are generally limited to creating simple geometric models. The part-composition systems provide tools for combining parts extracted from a database of 3D models, and thus can generate very detailed 3D models, but usually with much higher overhead and expertise required by the user for manipulating 3D geometry interactively. In this paper, we introduce a new modeling method that overcomes these limitations by combining both approaches – we introduce a modeling system for parts composition with a sketching interface. The system allows the user to find a part in a database and composite it into a model with a single sketch. This approach combines the benefits of both approaches – i.e., it allows creation of highly detailed models/scenes (as details come from parts in the database), while 2D sketched strokes provide all the information for part selection and composition (no 3D manipulation is required, in general). To enable this modeling method, we investigate an algorithm for 3D shape search with 2D sketch as a shape query and a part placement algorithm which automatically orients, translates, scales, and attaches a new part into a modeling scene by taking the user sketch as a hint. User experiences with our prototype system show that novice users can create interesting and detailed models with our system.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Interaction techniques]: sketch-based modeling.

1. Introduction

The development of easy-to-use systems for creation of 3D models is one of the most important research areas in modern computer graphics. While demand for 3D models has been greatly increasing over the last few years, 3D surface modeling remains difficult for most people, as commercial modeling systems generally require technical expertise and artistic skill.

Many efforts have been made to simplify the process of 3D modeling. One major approach is to provide sketch based interfaces [ZPF96, IMT99] which allow users to draw 2D sketches and infer simple 3D shapes from those sketches. Another popular approach is to provide tools for composition of parts from existing 3D models [FKS*04, SCoL*04]. While these approaches have greatly simplified the 3D modeling process, there are also limitations. Sketch based systems are natural and easy to learn, but limited to designing simple shapes. Part composition systems can construct detailed, complex models, but often require cumbersome geometric operations to construct similarity queries, place retrieved parts in the desired coordinate frames, and fuse parts together.

In this paper, we describe a sketch-based modeling system for composition of 3D parts. Unlike most previous systems, sketched strokes are interpreted both as similarity queries into a database of 3D parts *and* as alignment cues for placement and composition of parts into a scene (Figure 1). Specifically, for each modeling operation: (1) The user first sketches an outline of the part he/she wants to include in the scene at the place where it should be loaded. (2) The system takes the sketch as a shape query, performs a shape search on a given parts database, and shows the user the search result. (3) The user then selects a part he likes best from the search result. Finally, (4) The system determines the selected part's geometrical relationship with the rest of the modeling scene by taking the user sketch as a hint and automatically orients, translates, scales, and attaches the part into the modeling scene.

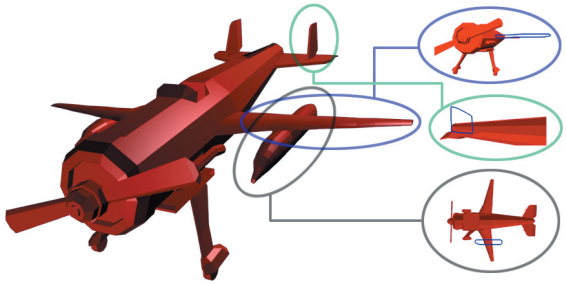

Figure 1: *Modeling example using our prototype system*

Our system combines the advantages of sketching with those of composition. On the one hand, the system is easy to learn and intuitive to use (simply sketch an outline of what you want, where you want it). On the other hand, it supports creation of detailed, complex models (all the details of models used are included in the final model). This synergy significantly lowers the technical threshold to create interesting 3D models. It is suitable for designing new instances of most man-made objects (vehicles, furniture, etc.), many types of natural objects (animals, people, etc), and scenes composed of multiple objects (e.g., office scenes). As such, we believe that it is useful for a wide variety of applications ranging from rapid prototyping for industrial designers to educational purposes for children.

2. Related Work

Our modeling system combines concepts of sketch-based modeling, content-based retrieval, and 3D part composition. Here we review previous work that our system builds upon within these categories.

Sketch-Based Modeling: Our system is largely inspired by the recent development of sketch-based modeling systems and the trend towards building modeling tools for novices. The motivating idea we share with previous work is to provide a simple and intuitive sketching method for creating 3D models. However, the interpretation and use of 2D sketches is quite different in our system than it is in previous ones. Some systems interpret sketched strokes as parameters for construction of geometric primitives [KQW06, YSv05, ZPF96]. Others interpret sketched input as contours [KH05, KH06] or displacements [BCCD04] that imply shape. Some use sketched outlines to infer the shape of a complete, closed 3D surface [IMT99, IH03, NISA07], while other systems use them as guidelines for local mesh deformation [ZNA07, NSACO05]. In contrast, our system interprets sketched lines as queries and placement guidelines for parts of 3D models retrieved from a database. The work most relevant to ours is Magic Canvas [SI07] system which uses sketched lines in the same way to place 3D models in a scene with limited orientations. However, this paper extends that work for design of general objects as well as scenes with larger degrees of freedom.

Sketch-Based Retrieval: Our work utilizes many ideas previously proposed for retrieval of 3D models from a database based on similarity to 2D sketches drawn by a user. Several such systems exist, varying in the number and type of sketches expected for each query, and how sketches are matched to 3D models [FMK*03, HR06, Lof00, JPR05]. For example, the Princeton Search Engine expects outline sketches for up to three views and matches them to precomputed sketches for thirteen views of a 3D model using Fourier descriptors of rendered images [FMK*03]. The Purdue search engine matches 2D views using 2.5D spherical harmonic descriptors, Fourier descriptors of boundary contours, and Zernike moments of rendered images [HR07]. SR-3DEditor matches sketches to outline contours with turning functions and places retrieved models into the same working space [AS05]. However, these systems only allow retrieval of existing 3D models and provide no special features for composition of parts into new models.

3D Parts Composition: Our work is also inspired by recent work towards building 3D models by composition of parts [FKS*04, KJS07, LJW06, ONI06, SBSC06]. The general philosophy is to gather a large database of 3D models and then to provide methods to cut and composite them to create new, interesting models. For example, Shuffler finds a consistent segmentation of parts across several examples within a class of objects and provides an easy interface for exchange of corresponding parts [KJS07]. Modeling by Example [FKS*04] provides interactive methods for cutting 3D models into parts, searching a database for similar parts, aligning parts, and attaching them across open boundaries. However, these systems do not provide good easy ways to construct queries for new parts. In Modeling by Example, for example, the user must construct a coarse 3D model out of boxes to represent the shape for a query into the database. Our main contribution is to augment such systems with ideas from sketch-based querying and modeling.

3. Overview

Our system is based on Modeling by Example (MBE) [FKS*04]. It supports all the features of the original system, including intelligent scissoring of 3D meshes, 3D shape similarity search for retrieving 3D meshes, and stitching/blending for composition of 3D meshes. It augments those features with a new sketch-based interface and new algorithms to make search, placement, and composition easier and more automatic.

Since the main design goal of our system is to provide a sketch-based modeling interface that can be learned by a novice user within minutes, we ask the user to understand only one type of modeling command entered with a single 2D stroke: "find a part in the database whose outline has a shape similar to my stroke and place it onto a surface of the current scene so that it matches the position, size, and orientation of the stroke as best as possible." (Figure 1).

Of course, supporting such a command is challenging, as the type, placement, and attachment of a new part must all be inferred from a single 2D sketch. This is a grossly under-constrained problem, and thus we must provide optimization algorithms to find a solution that best matches the user's input. Since these algorithms are not always perfect, we also must provide simple selection and undo commands to revise and reverse the proposed solutions quickly and easily.

In our system, processing of each sketch-based modeling command proceeds as shown in Figure 2. When the user draws an editing stroke, the system feeds the 2D contour as a shape-based query to 3D search engine which compares the shape of the contour to the outlines of all parts in the database using 2D image to 3D shape matching algorithm (Section 4). The system generates a ranked list of the best matching parts and displays them as a list from which the user selects a part. Then, the user's 2D sketch, the selected part, and the current scene are fed into an optimization algorithm that produces a best transformation for the part so that its outline fits the user's stroke while connecting as best as possible to the existing surfaces of the current scene (Section 5). If the part has a open boundary contour that could be attached topologically to the current scene, a hole is cut into the meshes for the scene, and the meshes are stitched and blended together across the resulting boundary contours (Section 6).

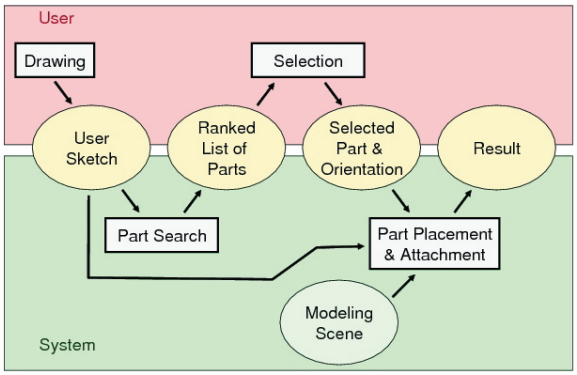

Figure 2: *Steps performed for each modeling operation.*

In the following sections, we describe our implementations for the main steps of this process: (1) retrieving parts from a database to match a single 2D sketch, (2) placing retrieved parts into a scene to match constraints provided by the 2D sketches and 3D surfaces, and (3) topologically attaching disjoint parts while blending across the seams.

4. Part Search

The first step is to retrieve parts from a database of 3D polygonal models matching a single sketch by a novice user. Our assumption is that the user draws an approximation to the boundary contour of a desired 3D part as seen from the current view. Thus, our goal is to search the database to find a part such that the projected boundary contour of the part for some view best matches the user's stroke in screen-space.

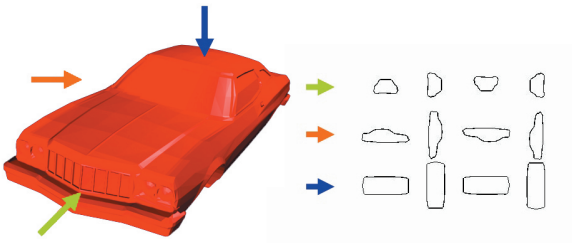

Figure 3: *Example boundary contour images of a car body.*

Like many prior systems [FMK*03,HR06,Lof00,JPR05], we leverage preprocessing of the 3D model database to simplify and accelerate the matching process. First, we extract individual parts from 3D models using automatic mesh segmentation algorithms, scene graph decompositions, and/or interactive segmentation tools [FKS*04]. Second, we align the parts in a canonical 3D coordinate system, normalizing for translation, scale, and rotation using continuous PCA methods (as in [FKS*04]). Third, we generate multiple images for each part containing outline contours for different orthogonal view directions [FMK*03]. Finally, we compute a shape descriptor for each image that can be matched rapidly to compute a similarity measure to any 2D sketch drawn by a user. The shape descriptors are indexed and tagged by part ID and view direction.

Out system works with arbitrary sets of views. However, in early experiments with our system, we have found that novice users tend to sketch outlines for axis-aligned views of parts, even when viewing a scene from an oblique view. Thus, we have chosen to index 24 orthographic views of each 3D part, corresponding to combinations of 6 possible axis-aligned "towards" vectors and 4 possible axis-aligned "up" vectors. For every view, V, for every part, P, the polygonal mesh is first centered and scaled isotropically such that its bounding box fills the image horizontally or vertically. Then, the polygons are rendered in black on a white background, and edges are detected with a Canny edge detector to produce a *boundary contour image*, $B(P,V)$ (Figure 3). Then, we compute the squared distance transform of that image to facilitate later computation of image similarities. The stored result is a pair of images for every view of every part containing a boundary contour and its squared distance transform. These two images form the shape descriptor for every view of every part and are stored in an indexable database.

For each user stroke, we aim to find the part(s) and view(s) whose boundary contour image(s) is most similar to the user's stroke. To satisfy such a similarity query, several distance measures are possible, including ones based on Turn-

ing Functions, Fourier Descriptors, etc. For this project, we have chosen to define the similarity between images based on the sum of squared distances between closest non-zero pixels [FKS*04]. That is, for every pixel in the user's stroke, the distance measure adds the square of the distance to the closest non-zero pixel in the boundary contour image, plus vice versa. This distance measure is nice for our problem because: 1) it approximates the amount of work required to deform non-zero pixels of one image into the other, 2) it works for images of any topology (the boundary contour of a part may have many disconnected components), and 3) it can be computed and indexed efficiently using the shape descriptors described in the previous paragraph. Specifically, if squared distance transform images, $SD(A)$ and $SD(B)$ are available for two images A and B, then the dissimilarity measure can be computed with a dot product:

$$d = Dot(A, SD(B)) + Dot(SD(A), B)$$

We use this dissimilarity measure to produce a list of all precomputed boundary contour images sorted by similarity to the user sketch. The system then presents the list as a list of thumbnail images where each thumbnail corresponding to each boundary contour image $B(P,V)$ is a rendering of the part P from the view V. The user can select the thumbnail he likes best, indicating the part in the thumbnail should be added to the scene with the orientation in the thumbnail.

5. Part Placement

The second step is to determine the similarity transformation with which the selected part should be added to the scene. The goals for this step are three-fold: (1) the screen-space outline of the transformed part should match the user's stroke, (2) the part should not be occluded or penetrated by any part of the scene, and (3) the surface of the part should be in stable contact (3 contact points) with the rest of the scene.

Of these three goals, we consider the stroke a hint (since most user drawings are approximate), while depth-order and contact with the scene are constraints to be satisfied, if possible. Thus, we take a two-step approach to computing candidate part transformations, where an initial guess for the transformation is generated based on the user's stroke, which is then optimized in a second step to enforce stable contact with the current modeling scene, if possible.

In the first step, we compute a similarity transform for which the projected outline of the part on the screen best matches the user's stroke. For this step, we utilize the orientation derived from the thumbnail image that user chose to determine the rotation. To determine the translation and scale, we constrain the center of mass of the part to the center of mass of the user's stroke (a ray through the 3D scene), and we constrain the bounding area of the part's projection onto the screen to match that of the user's stroke [AS05]. These guidelines provide six constraints (three for rotation, two for translation, and one for scale) on the seven parameters of a similarity transformation. The final degree of freedom (translational depth) is initially chosen so that the part centroid matches the average depth of polygons in the scene (Figure 4a).

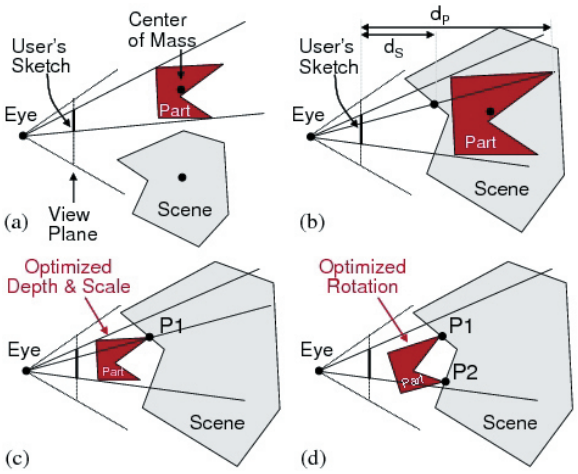

Figure 4: *(a-b) The part (red) is initially placed so that its size, orientation, center of mass in screen-space match the user's stroke and the depth of its center of mass matches average depth of the scene (gray). (c) If the part overlaps the scene in screen-space, it is moved back to a new position with a single contact point (P1). (d) Then, the part is rotated around P1 to produce a stable contact at P2 between the part and the scene.*

If the screen-space projections of the part and current modeling scene overlap (as in Figure 4b), we optimize the part's depth and scale so that it sits in front of and touching the scene without penetration, while ensuring that it maintains its screen-space projection (Figures 4c and 5b). To perform this optimization, we must find the first point of contact, $P1$, between the back side of the part and the front side of the scene as the part is translated away from the eye while being simultaneously scaled to maintain its screen-space projection (such that all points of the scene travel along rays from the eye). We perform this search in discrete screen space using the depth buffer of the GPU. We first render the current scene without the part and store the closest depth value $d_s(x,y)$ of every pixel (x,y). Then, we reset the depth buffer, invert the depth test function such that higher depth values overwrite lower ones, render the part, and store the furthest depth values $d_p(x,y)$. Then, we search through pixels of these depth buffers to find the pixel (x,y) with the minimal ratio of $d_s(x,y)$ to $d_p(x,y)$, and use it to determine the scale $(d_s(x,y)/d_p(x,y))$ around the eye point to ensure that: (1) the part's screen-space projection is preserved, (2) the part is in front of the scene, (3) the part and scene are in precise contact at least at one point, $P1$, and (4)

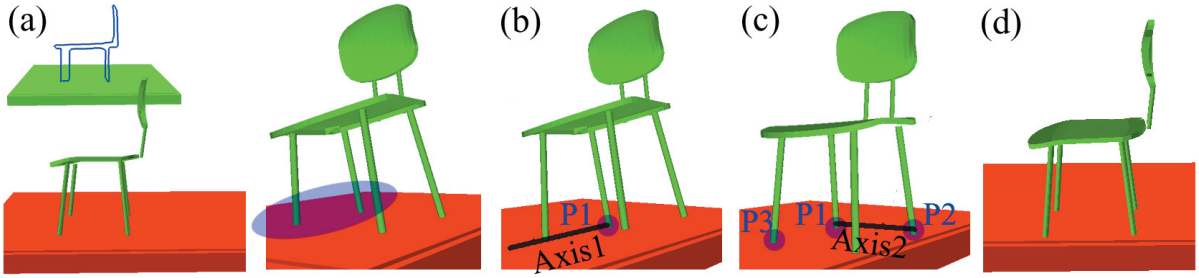

Figure 5: *(a) The chair is initially placed so that it's size, orientation, center of mass match the user's stroke and its depth matches average depth of the scene. The chair overlaps with other object in the scene (blue area). (b) The chair is moved to a new position with a single contact point(P1). (c) The part placement is optimized to find at least two more points of contact between the part and scene(P2 and P3). (d) As a result, all legs are touching the floor.*

no point on the surface of the part penetrates the surface of the scene.

The resulting transformation provides an optimal match between the part's outline and the user's sketch in screen space. However, it usually places the part in only one point contact with the scene (at $P1$). During early tests, we have found that users generally prefer part placements with more stable contact with the scene, even if the placement causes the part to rotate slightly such that the outline no longer matches the drawn sketch optimally (i.e., the users care more about the scene composition than the exact details of their sketches, especially since they tend to draw orthographic views of parts, even when they are adding the part to a scene viewed in oblique perspective). Thus, during a second step, we optimize the part transformation further, applying the minimal rotation around $P1$ for which more points of contact are found between the part and the scene (Figure 5). To do this, we sequentially check for such a rotation around three orthogonal axes through point $P1$: (1) $N \times V$, (2) $N \times V \times V$, and (3) V, where N is the normal to the surface of the scene at point $P1$ (Figure 5b-c). For each of these three axes in sequence, we check rotations up to M degrees ($M = 45$) in increments of m degrees ($m = 5$) in least-to-most order interleaving positive and negative directions, checking for part-surface penetrations using the z-buffer at each rotation (as described in the previous paragraph). We select the minimal rotation before a part-scene penetration is found with this procedure, and use the location of the penetration to determine a second point of contact $P2$ (Figures 4d and 5c). If no second point of contact is found (within M degrees for all three axes of rotation), the part is left at the transformation with one point of contact. Otherwise, we perform a similar rotational search around the axis supporting $P1$ and $P2$ to find a third point of contact. We apply the found rotations to providing stable placement of the part into the scene (Figure 5d).

While this rotational optimization may cause the part's projection onto the screen to deviate from the user's stroke, it is almost always what the user intended. This is a key feature, not a bug. The optimization allows the user to draw strokes approximately (without worrying about drawing precisely in perspective from an oblique view), and the system automatically transforms the part to integrate stably with the scene. Thus, it is simple to put a chair on a floor with all 4 legs touching (Figure 5d), put a wheel on the back of a truck (top row of Figure 6), or put a vase on a table (bottom row of Figure 6) from a wide range of views.

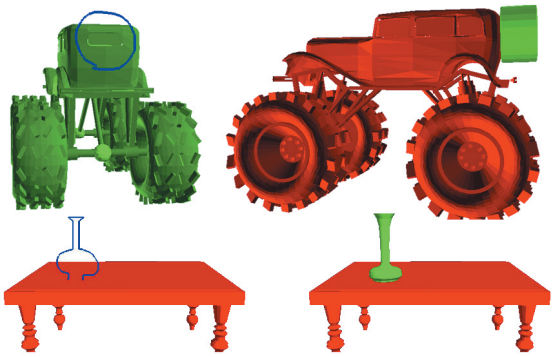

Figure 6: *Example optimized part placements.*

6. Part Attachment

The last step is to attach the part topologically to the scene. This operation is performed only when the following two conditions are satisfied: (1) at least three points of contact are found between the part and the scene, (2) the part has a hole in its mesh within a small distance of the plane containing those three points of contact. If any of these two conditions is not satisfied (e.g., a mug on a table), the part is simply added to the scene without topological attachment, which is usually what the user intended.

The operations for topological attachment proceed for each hole H_P in the mesh of the part M_P as follows. If three points of contact are found during part placement, we use them to establish a contact plane, CP. We then trace the boundary of the hole on the mesh of the part and check

whether every vertex on the boundary is within a distance threshold d_A of CP and whether the plane of best fit through the boundary has a normal within a_A degrees of CP's (the default values for a_A and d_A are 45 degrees and 0.1 times the distance from the furthest point on the part to C_P in our system). If so, the part's hole(H_P) is considered suitable for topological attachment to the scene. Similar operations are used to determine whether the scene has a corresponding hole, H_S.

In the case where this process finds a hole for the part but not the scene, the system cuts a hole H_S in the scene mesh M_S automatically. To do this, it first finds the face F_S in the scene mesh that is closest to the centroid of the boundary of the part's hole, H_P (Figure 7b). It then removes F_S from M_S and grows a hole H_S using a depth-first search of M_S. The search stops when a face is found whose projection onto CP lies entirely outside the projection of H_P onto CP, or when the face normal is back-facing with respect to CP. Note that this process makes a hole H_S that is usually one ring of faces larger than that of the part, which is well-suited for topological stitching followed by geometric blending (Figure 7c). H_S could be much larger than H_P when the tessellation of M_S is coarse, but we believe this can be overcome by subdividing M_S in the future work.

Once two corresponding holes are available, one in the part, and another in the scene, the system uses the method described in [FKS*04] to attach them. Briefly, a new set of triangles is created to connect the boundary vertices of the corresponding holes. Then, the neighborhood of the new triangles is blurred by successive averaging of vertex positions to smooth the joint between part and scene. This process produces a mesh that is locally water-tight and smooth along the attachment (Figure 7).

7. Results

We have implemented the methods described in the previous sections and deployed them in an interactive sketch-based modeling system for 3D mesh retrieval and composition. This section describes early experiences with this system, particularly for novice users. All processing and experiments were performed on a laptop computer with a 1.66GHz CPU and 1GB of memory using a single set of parameter settings.

The system's database is built upon the 907 3D models provided with the Princeton Shape Benchmark [SMKF04]. The models were all segmented manually into parts and then processed to construct boundary contour images and shape descriptors for all 24 views of all parts. This processing took 3 hours of off-line computer time. Using this database, we conducted two informal studies to investigate how easy our sketch-based modeling system is to learn and to use. During these experiments, each search of the database took the system less than 2 seconds, while part placement and attachment took about 3 seconds, on average.

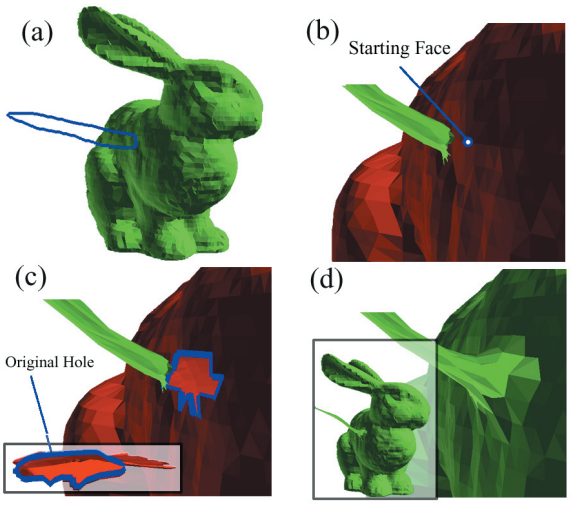

Figure 7: *Attaching a wing to a rabbit. (a) User draws a wing. (b) The wing is placed and the closest face on the surface to the centroid of the wing's hole is found. (c) New hole is created. (d) Two holes are topologically attached with a water-tight and smooth junction*

During the first study, we selected three novice users with no prior 3D modeling experience. Each user was trained for 10-20 minutes and then asked to design a 3D model matching a scene provided in a photograph. This study is intended to investigate how closely a user can create a model of an object provided as a target. The photographs of target objects (inset in upper left) and resulting 3D models are shown in Figure 8.

The indoor scene was designed by a non-CS undergraduate without experiences in 3D modeling, but with some exposure to 3D Poser. It took him about 10 minutes to design the room. He spent most of time trying to figure out good "view" to draw things so that they will have desired orientation as they are added to the scene, ultimately deciding upon the top-view for most operations.

The winged unicorn model was designed by a different non-CS undergraduate without any experiences in 3D modeling. It took him 5 minutes to create it. Wings and horn were successfully added/attached in a single try. After creating a plausible rendition in a minute or so, he spent further time trying to re-draw wings at different positions to achieve more attractive results.

The chair model was designed by a third non-CS undergraduate student without any experience in 3D modeling. It took him about 10 minutes to design the chair. He started by retrieving a frame and added the seat and the back to the frame in one or two tries. However, he had trouble adding two legs in a consistent manner. After trying several times for the second leg, he decided to draw it approximately and translate it further using direct manipulation controls provided with the system.

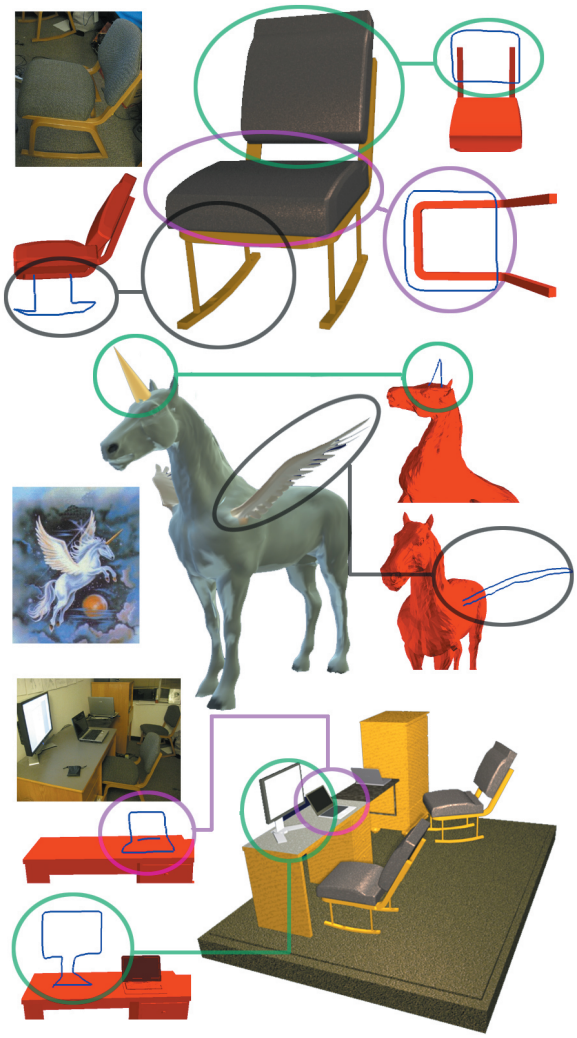

Figure 8: *3D models created by novice users in 5-10 minutes after 10-20 minutes of instructions. The target provided to the user is shown above-and-left of each model.*

During informal surveys, we found that all three users found the design process "intuitive" and generally were satisfied with the search results and part placements provided automatically by the system. However, they also found that it is hard to perform certain operations in our system. For example, when adding legs to both side of the chair as in Figure 8, it is hard to add them in exactly same sizes and in exactly symmetric manner, as sketches tend to be casual. Of course, further capabilities to enforce symmetric edits could be added to the system, but features of this type are beyond the scope of this paper.

During a second study, we selected two different novice users to create a model of their own design using the system. This study was intended to investigate the suitability of the system for conceptual design. Figure 9 shows the results, each of which was made in 15-20 minutes.

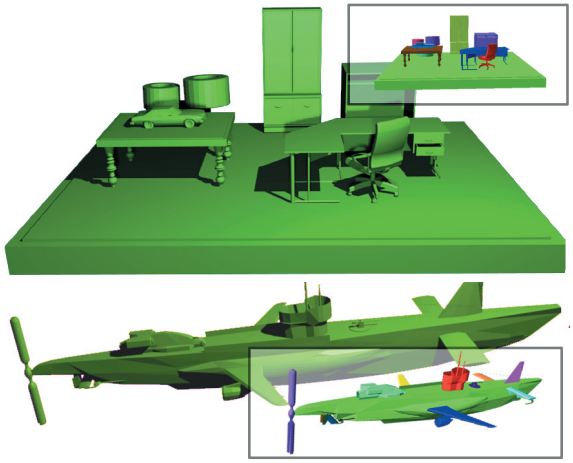

Figure 9: *3D models designed and created by novice users in 15-20 minutes. Inset images are colored by part.*

In general, these users found the system easy to learn and useful for creating new models. Even when a user had only an abstract idea of what he wanted to add to a scene, he could always draw a shape, and pick a part that he found interesting from query results. Moreover, as each step of composition requires only a single sketch, and the system supports infinite undo's, the user could try out his ideas with little burden. This suggests that our approach not only lowers the technical threshold to 3D modeling, but also that of artistic creation as well.

8. Conclusion & Future Work

In this paper, we investigate a new type of modeling system for creating a new 3D object or scene with a sketching interface. The system overcomes limitations of both popular approaches in easy 3D modeling, sketch based modeling and modeling by composition. User experiences with our prototype system suggest that the system is indeed easy to learn and can create interesting and detailed 3D models.

This paper provides only a small, first step towards the goal of providing advanced modeling to novice users. While we do believe that our system is easier to learn and faster to use than any other system capable of producing models with the same detail, there still are many limitations and areas for future improvement. One possible area of future work is to investigate ways to allow composition of parts while maintaining global constraints (e.g., symmetries), as discussed in previous section. Another possible research topic is to investigate sketch-based methods that support not only search and composition, but also deformation. For example, currently in our system, the user cannot find and use an arm that's bent if there exist only straight arms in the database. In the future, it seems interesting to combine ideas on sketch-based composition of this paper with complementary ideas on sketch-based deformation

(e.g., [NSACO05,OSSJ05,YZX*04,ZNA07]). Investigating other synergies between sketch-based modeling and sketch-based retrieval seems like a fruitful area for future research.

Acknowledgements

We thank Don Kim for assistance with preparing figures and the National Science Foundation (CNS-0406415, IIS-0612231, and CCF-0702672) and Google for funding.

References

[AS05] ANDREOU I., SGOUROS N.: Shape-based retrieval of 3d models in scene synthesis. 3124–3129.

[BCCD04] BOURGUIGNON D., CHAINE R., CANI M.-P., DRETTAKIS G.: Relief: A modeling by drawing tool. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling* (2004).

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. In *Proceedings of SIGGRAPH 2004* (2004).

[FMK*03] FUNKHOUSER T., MIN P., KAZHDAN M., CHEN J., HALDERMAN A., DOBKIN D., JACOBS D.: A search engine for 3d models. In *ACM Transactions on Graphics* (2003), vol. 22, pp. 83–105.

[HR06] HOU S., RAMANI K.: Sketch-based 3d engineering part class browsing and retrieval. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling* (2006).

[HR07] HOU S., RAMANI K.: Classifier combination for sketch-based 3d part retrieval. *Journal of Computers & Graphics 31* (2007).

[IH03] IGARASHI T., HUGHES J. F.: Smooth meshes for sketch-based freeform modeling. In *ACM Symposium on Interactive 3D Graphics* (2003), pp. 139–142.

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3d freeform design. In *Proceedings of SIGGRAPH 1999* (1999), pp. 409–416.

[JPR05] JIANTAO PU K. L., RAMANI K.: A 2d sketch-based user interface for 3d cad model retrieval. *Computer-Aided Design & Applications 2*, 6 (2005), 717–725.

[KH05] KARPENKO O., HUGHES J. F.: Inferring 3d free-form shapes from contour drawings. In *SIGGRAPH Technical Sketches* (2005).

[KH06] KARPENKO O., HUGHES J. F.: Smoothsketch: 3d free-form shapes from complex sketches. *ACM Transactions on Graphics (SIGGRAPH 2006)* (2006).

[KJS07] KRAEVOY V., JULIUS D., SHEFFER A.: Shuffler: Modeling with interchangeable parts. In *Visual Computer Journal 2007* (2007).

[KQW06] KU D. C., QIN S. F., , WRIGHT D. K.: A sketching interface for 3d modeling of polyhedrons. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling* (2006).

[LJW06] LIN J., JIN X., , WANG C.: Sketch based mesh fusion. *Advances in Computer Graphics 4035* (2006), 90–101.

[Lof00] LOFFLER J.: Content-based retrieval of 3d models in distributed web databases by visual shape information. *iv 00* (2000), 82.

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fibermesh: Designing freeform surfaces with 3d curves. In *Proceedings of SIGGRAPH 2007* (2007).

[NSACO05] NEALEN A., SORKINE O., ALEXA M., COHEN-OR D.: A sketch-based interface for detail-preserving mesh editing. In *Proceedings of SIGGRAPH 2005* (2005), pp. 1142–1147.

[ONI06] OWADA S., NIELSEN F., IGARASHI T.: Copy-paste synthesis of 3d geometry with repetitive patterns. *Lecture Notes in Computer Science 4073* (2006), 184–193.

[OSSJ05] OLSEN L., SAMAVATI F., SOUSA M., JORGE J.: Sketch-based mesh augmentation. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2005).

[SBSC06] SHARF A., BLUMENKRANTS M., SHAMIR A., COHEN-OR D.: SnapPaste: an interactive technique for easy mesh composition. 835–844.

[SCoL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., ROESSL C., SEIDEL H. P.: Laplacian surface editing. In *In Proc. Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2004), pp. 179–188.

[SI07] SHIN H., IAGARASHI T.: Magic canvas: Interactive design of a 3-d scene prototype from freehand sketches. In *Proceedings of Graphics Interface 2007* (2007).

[SMKF04] SHILANE P., MIN P., KAZHDAN M., FUNKHOUSER T.: The Princeton Shape Benchmark. In *Shape Modeling International* (June 2004), pp. 167–178.

[YSv05] YANG C., SHARON D., VAN DE PANNE M.: Sketch-based modeling of parameterized objects. *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling* (2005).

[YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.-Y.: Mesh editing with poisson-based gradient field manipulation. In *ACM Transactions on Graphics* (2004), vol. 23.

[ZNA07] ZIMMERMANN J., NEALEN A., ALEXA M.: Silsketch: Automated sketch-based editing of surface meshes. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling* (2007).

[ZPF96] ZELENIK R. C., P.HERNDON K., F.HUGHES J.: Sketch: An interface for sketching 3d scenes. In *Proceedings of SIGGRAPH 1996* (1996).