Sketch-Based Pipeline for Mass Customization

1 Introduction

The digital age in manufacturing is coupled with new output devices that allow mass customization and rapid manufacturing, revolutionizing the way how we design, develop, distribute, fabricate, and consume products. To open the new technology to a broad audience of users it is one of the central challenges to develop effective and easily accessible interfaces that support the customization of products. Especially, because it is difficult for most users to create their own content using a 3D modeling system.
We rather suggest a different workflow where the user starts from an already existing object that is retrieved from a large 3D shape collection and modifies it intuitively. Therefore it is necessary to not only have the possibility for direct shape modifications but also to search in a large collection of 3D shapes within one unified interface.
Sketching as a tool that supports ideation and gives the user the freedom to create images easy and fast [26], [22] provides an exceptional input tool for searching large image and 3D shape databases, as well as modeling 3D surfaces.
In this work we present, for the first time, a closed workflow from a user drawn 2D sketch to a 3D printed personalized object by taking manufacturing limitations into account. This is achieved by combining different sketch-based retrieval and modeling aspects, enabling the user to control the process with a sketch-based input metaphor.
In the development of the system we focus on two main aspects. First, we want to give non-expert users an easy-to-use interface for customized fabrication. Second, we want to ensure that each 3D model created with our system can be manufactured regardless the modifications made. Both aspects are truely challenging and still an active field of research. Amateur users often have limited drawing skills resulting in strongly simplified sketches that, for instance, do not take perspective into account. To this end, we use a retrieval system by Eitz *et al.* [11] that works sufficiently even with a simple set of strokes conveying rough visual ideas as commonly done in architectural or in product design. We also provide modification possibilities that require only a very limited prior knowledge about modeling tools and 3D surface manipulation very similar to Zimmermann *et al.* [31].
One central observation is that not every modification can be fabricated due to geometric errors introduced during the deformation process or due to 3D printing limitations. Hence, we present a novel approach to take fabrication limitations into account. Figure 1 shows the different components of our system that can be summarized as follows:

1. The process starts by drawing a simple binary sketch as outlines of a 3D shape. This sketch image is used to query a large database containing 2D line renderings of 3D models, where each rendering maps to the corresponding 3D model (see Section 3).

2. By selecting an object out of the result set of the search the system switches to the modeling step of our pipeline. The user is able to modify the object directly by drawing additional strokes outlining the new silhouette of the shape. Mesh deformation parameters are derived and the model is deformed according to the user strokes (see Section 4).

3. During the deformation process we have to make sure that the model is still printable. Therefore each deformation step triggers a 3D printing simulation that validates the user's modification. The deformation is only executed as long as the printing constraints are fulfilled. The final modified object is printed using a standard off-the-shelf 3D printer (see Section 5).

Note, as sketches are less suited to draw precise details our system also does not incorporate these features. We neither can distinguish between very fine detailed differences in the query sketches nor modify the models very precisely.
The system described in this paper is an extended version of the pipeline proposed by Hildebrand *et al.* [15].

2 Related Work

Line Drawings and Sketch-based Shape Retrieval To develop sketch-based interfaces it is necessary to understand how people create line drawings [6] and how they sketch objects [10]. Specifically Eitz *et al.* [11] report on a large-scale experiment which provides insight into how an average user of a 3D retrieval system

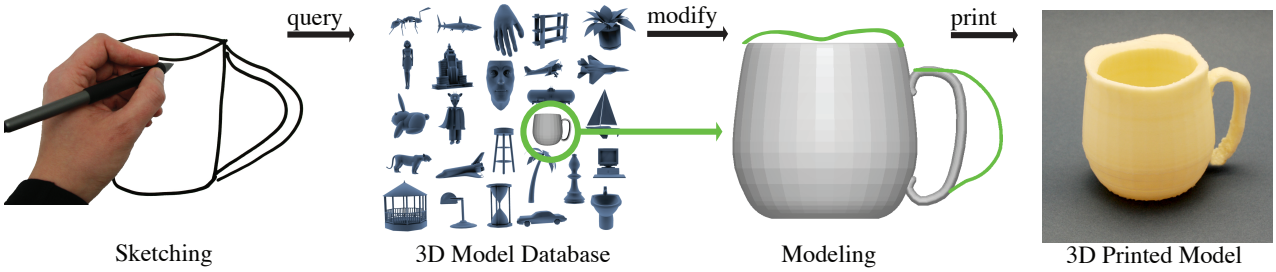

Figure 1: We present a sketch-based pipeline guiding the process of manufacturing. We start by sketch-based retrieval of a user sketch in a large 3D model database. We show the possibilities of sketch-based modeling to customize the results intuitively. Customized 3D shapes are manufactured using a 3D printer.

would sketch a query for such a system. Funkhouser [13], Yoon [30] and Eitz [11] recently proposed systems enabling the user to retrieve shapes from a large 3D model database. These ideas work on a set of descriptors that are extracted from the 3D models in a preprocessing step. However, Funkhouser and colleagues [13] rely on complete 3D shape descriptors whereas Eitz *et al.* [11] and Yoon *et al.* [30] extract the features on 2D images that are the renditions of a set of line renderings (see Section 3). Our shape retrieval process is bases strongly on the work of Eitz [11].

Sketch-based Shape Modeling There is a lot of work about the nature of sketching and how it can be embedded in modeling tools to enhance and simplify design work by Suwa *et al.* [26], Tversky *et al.* [27] and Buxton [4]. There is also a variety of sketch-based modeling systems. Recently, Eitz and colleagues [12] proposed a framework to synthesize novel images from an image collection relying on the sketch-based input metaphor. Modeling 3D shapes very intuitively using sketches was introduced by Igarashi *et al.* [17]. Follow up research was done by Bae *et al.* [2], Schmidt *et al.* [23] and Nealen *et al.* [21], [20]. Our approach mostly relies on the approach of Zimmermann *et al.* [31] who use already existing geometry that is modified using simple silhouette strokes.

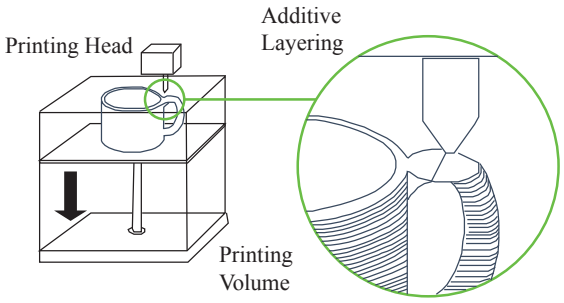

Figure 2: The 3D printing process is illustrated. The object is printed within a printing volume. A printing head generates the object layer-by-layer in an additive manner.

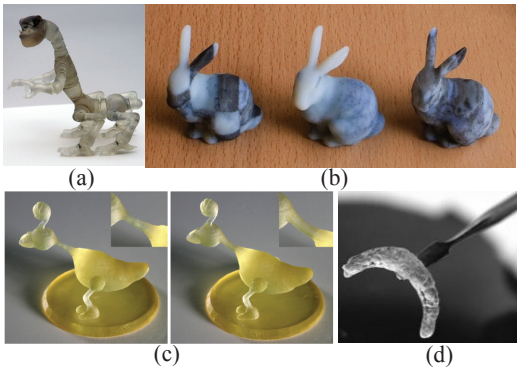

Figure 3: (a) 3D printing of articulated characters [1]. (b) Fabrication of subsurface scattering effects [14]. (c) Modification of geometry to support printablility and stability [25]. (d) 3D printing of tissue [5].

3D Printing and Digital Fabrication 3D printing is the process of fabricating solid objects from digital shapes. It is achieved by successively stacking layers of different shape. Recent publications also focus on further possibilities of this additive manufacturing technology. Hiller *et al.* [16], Cohen *et al.* [5], Lipson *et al.* [18] and Vilbrandt *et al.* [28] discuss the general properties of 3D printing, the challenges and possibilities. Baecher and colleagues [1] propose an automatic process from an input mesh to a fabricatable model that approximates the 3D kinematics of a virtual character by adding physical joints into the 3D printed model. Bickel *et al.* [3] acquire

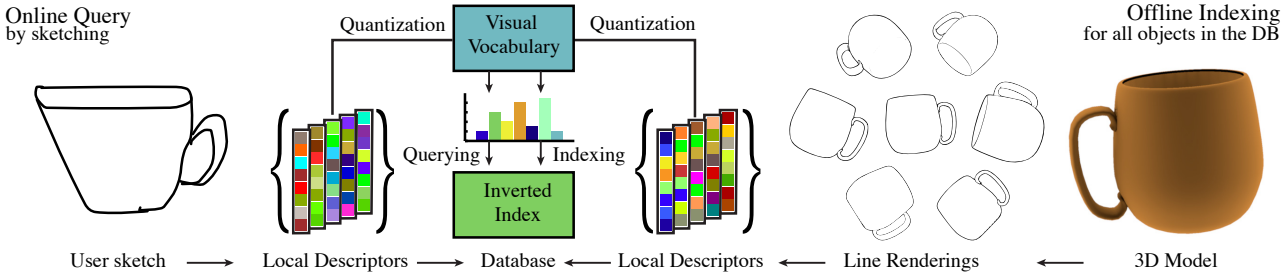

Figure 4: We present a sketch-based pipeline guiding the process of manufacturing. We start by sketch-based retrieval of a user sketch in a large 3D model database. We show the possibilities of sketch-based modeling to customize the results intuitively. Customized 3D shapes are manufactured using a 3D printer [11].

the deformation properties of a material and reproduce this behaviour using a multi-material 3D printer. Dong *et al.*and Hasan *et al.* [9],[14] show that the surface appearance and subsurface scattering effects of objects can be captured and also reproduced. One of the major challenges regards the printability of 3D shapes. Recently, Stava *et al.* [25] focused on the stability of the object and adjusted the geometry according to the manufacturing constraints. This closely relates to our proposed solution and could be additionally added as a post-process to our method.

3 Shape Retrieval

The basic idea behind the retrieval process is to compare the 3D models based on a histogram of features, as described in detail in Eitz *et al.* [11] [10]. The features are extracted in a pre-process over all models in the database. However, to match the 2D input user sketch with the 3D models we do not extract features in the 3D domain. We rather compare the 2D user input directly in the same domain. To this end, we generate 2D sketch-like drawings with non-photorealistic rendering algorithms [8] and render the 3D models with selected feature lines. We also use several virtual viewpoints to collect as many perspectives of the shape as possible. This turns the problem into a comparison of a single sketch image to several two-dimensional sketch-like renderings per object. Keep in mind that we do not compare the images directly but a set of features extracted from the images. Therefore we have to define a feature vector that describes the properties of the user sketch and the line renderings. We also rely on a specific data representation [24] to access the data fast. The complete feature extraction pre-process can be summarized in the following steps for each model in the database (see Figure 4 from right to middle):

View selection and rendering We render the 3D model from different view points uniformly distributed over the bounding sphere. We place a virtual camera at the each view point with the viewing direction to the center of the sphere and create line renderings using the view-dependent algorithm suggestive contours [8]. Suggestive contours convey the shape by drawing lines that are clearly visible on parts of the surface, where a true contour would first appear with a minimal change in viewpoint (see Figure 4).

Feature extraction Our goal is to compare image features, i.e. descriptors, that capture information about the user sketch and the line renderings. We extract the features locally at several positions as image patches over the image. We base our descriptor on gradient information, specifically we use a histogram of oriented gradients, as described in Eitz *et al.* [10]. We define the gradient as Gaussian derivatives of the image and store the magnitude of the gradient vector in the bins of a histogram. The histogram consists of spatial and orientational bins stacked into a single column vector. We refer the reader to Eitz *et al.* [10] about the details of the descriptor.

Visual vocabulary A common approach to query large databases is to use a bag-of-features representation [24]. The idea is to reduce the datasize by quantizing the feature vectors and build a so-called visual vocabulary using k-means clustering. Each cluster defines a visual word in feature space. To this end, we randomly sample one million local feature vectors as training data. The resulting cluster centroids form the visual words representing the local features in the same cluster. Counting the occurrences of visual words in a view

(line rendering) yields a histogram that is typically very sparse, as the number of distinct features occurring in a single view is usually much lower than the size of the vocabulary [11]. In order to achieve quick lookups during the query stage we store the resulting frequency histogram in a standard inverted index datastructure [29]. Once we have generated the visual vocabulary and the inverted index from it we can query the system using an input sketch. To retrieve a set of resulting 3D models that match the input sketch we require the following steps (see Figure 4 from left to middle):

Feature extraction At runtime, the user draws an input sketch and queries the retrieval pipeline. As before for the offline indexing we extract local feature vectors in the sketch image. The feature vector is quantized against the visual vocabulary and represented as a histogram of visual word occurences over the sketch.

Retrieval During the retrieval process we want to find similar line renderings (and their corresponding 3D models) to our input sketch. Two images can be defined as similar if their histograms and their corresponding high-dimensional feature vectors point into the same direction. Typically histograms are not compared using raw word counts but weights expressing the importance of the visual word. We use a common function called tf-idf (term frequency inverse document frequency) [29]. The system computes the closest matching visual words and returns a set of corresponding views that are mapped to the resulting 3D models as can be seen in Figure 5.

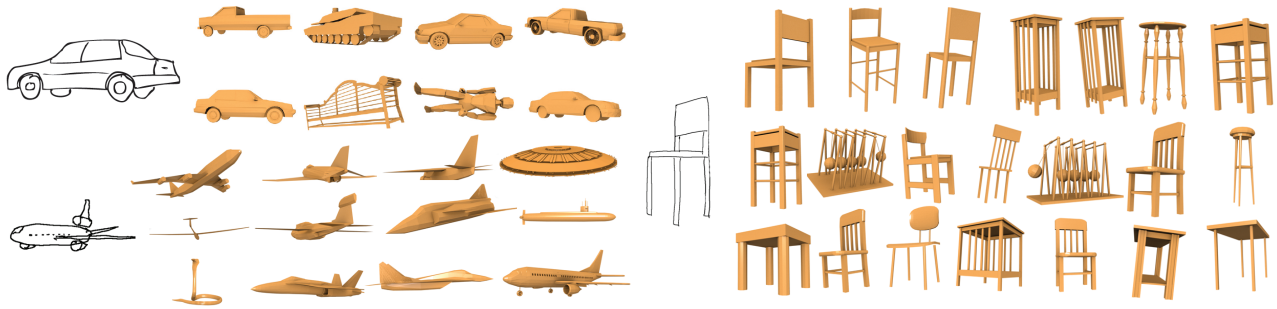

Figure 5: Results of the sketch-based retrieval method each with the user drawn sketch.

4 Shape Modeling

We continue the sketch-based input metaphor within the next step of our pipeline. At runtime, the user selects one of the matching retrieved 3D models which best fits his preference. We define a sketch-based modeling process very closely related to Zimmerman *et al.* [31]. We enable the user to modify the model by sketching new view-dependent silhouettes of the mesh. For each stroke drawn by the user a corresponding feature-preserving deformation is computed. The steps of the interaction can be comprised as follows (see Figure 6) [31]:

1. The user draws a stroke that suggests a deformation of the input model (Fig. 6(a)). To be able to deform the model we first identify a set of silhouette lines in the 2D rendering of the shape. These silhouette describe depth discontinuities in the rendering. We segment the extracted lines in image space and store them as a set of feature lines (Fig. 6(b)).

2. The stroke (target line) is matched against all feature lines to find the corresponding feature lines close to the stroke. Our goal is to to derive a subset of the silhouette that works as deformation handles in 3D (see Figure 6(b,c)).

3. Given a matching feature line we need to determine the corresponding mesh vertices and their transformed positions respectively. This is done by projecting the pixel positions of the feature line back in object space using the corresponding depth map. We derive the displacement that is necessary for the deformation from the relative vertex positions onto the target line (see Figure 6(d)).

4. We define region of interest that filters all vertices that will be influenced by the deformation process. We use a region growing algorithm based on the vertex handles (see Figure 6(e,f)). The derived deformation parameters are used as input to existing mesh deformation tools [21]. Figure 6(g,h) show the deformation result.

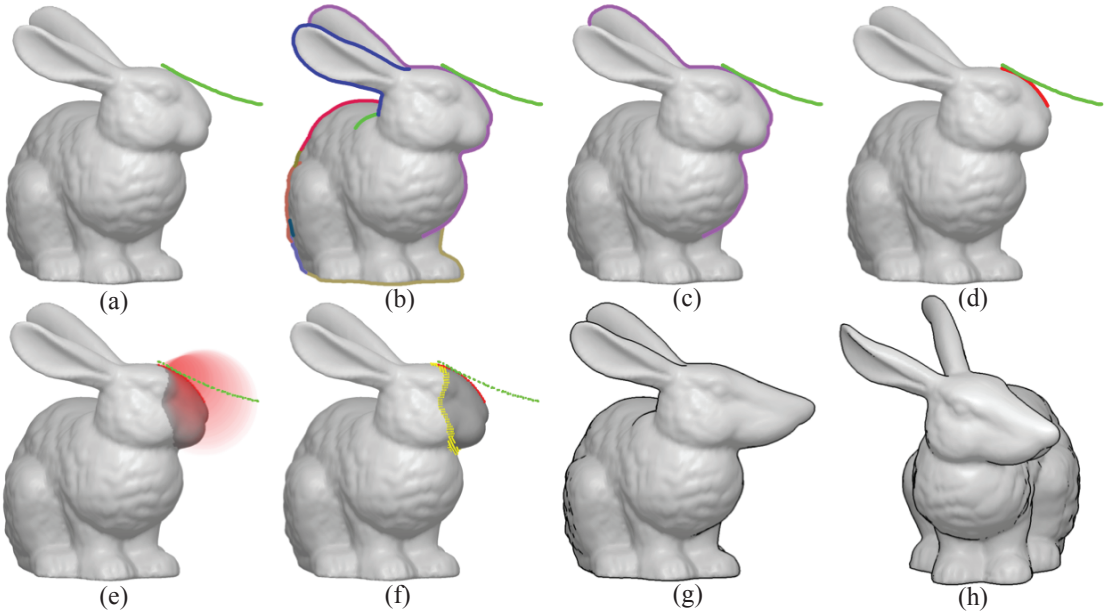

Figure 6: The example of the Stanford bunny shows the single steps of the modeling steps.©Zimmermann [31]

For further details please refer to Zimmermann and colleagues [31].

5 Fabrication

The objective of this work is to manufacture the resulting modified shape. An important observation, however, is that the object could potentially be deformed in a way that it cannot be fabricated with a 3D printer, so limitations in the fabrication process are an important consideration. Therefore we focus on three aspects that could arise during the object modification (see Fig. 7):

1. thin and fragile structures that introduce problems during printing or produces unstable results.
2. self-intersections and interpenetration of surface parts.
3. exceeding of the available printing volume.

Our goal is to guarantee the printablility regardless the modification. We ensure this by simulating the 3D printing process. Therefore it is necessary to understand its central operation. The object is sampled equidistantly along the 3D printing normal direction (see Figure 2). At each sample point a plane-geometry intersection is performed and a set of intersection slices (polygons) describing the outer and inner contours is generated. We simulate the production process by slicing the object and evaluating each slice geometry independently. If any of the intersection polygons resulting from the deformation violate the constraints above we backtrack the deformation linearly to the point the printing simulation will generate a valid slices.
Different to Stava *et al.* [25] we do not adapt the geometry directly to make sure the object is printable. We also do not take material behaviour into account. Our goal is to find a deformation between the original shape and the desired target shape that is as close as possible to the modification stroke drawn by the user. In other words, we propose an automatic correction of errors resulting from the flexible sketch-based interface metaphor. Figure 7 shows a 3D model sliced in a set of production layers. Each of these layers could be modified during the deformation process. Our pipeline evaluates the three cases mentioned above as follows:

Printing volume The user drawn stroke results in a target deformation that exceeds the printing volume. Our framework tries to re-position and fit the bounding box of the deformed model within the printing volume. If that fails we backtrace to the ideal deformation that still fits the printing volume. Another approach would be to segment the object into several parts similar to Luo *et al.* [19] and print them separately. (see Figure 7(b))

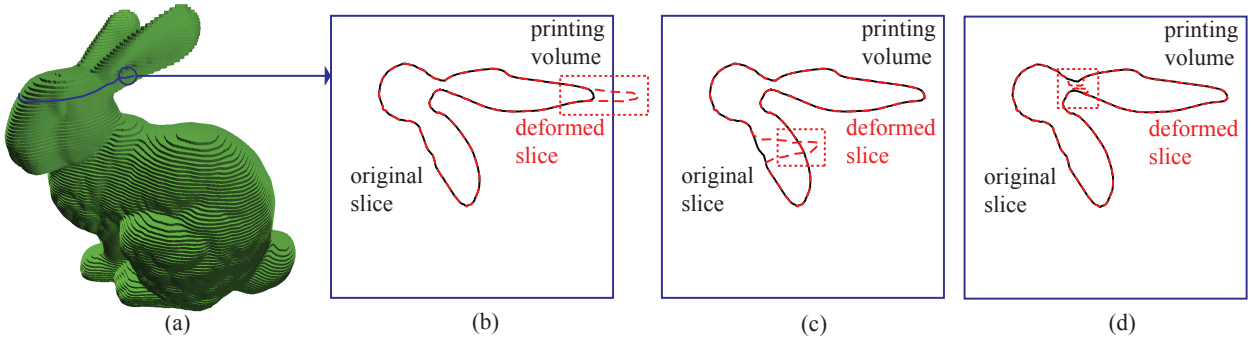

Figure 7: (a) The original model and one intersection slice. (b) The deformation exceeds the printing volume even after the object is recentered. (c) One of the resulting intersection polygons is sellf-intersecting and inside-outside of the shape becomes ambiguous. (d) The deformation introduces very thin structures that are difficult to print or are highly fragile.

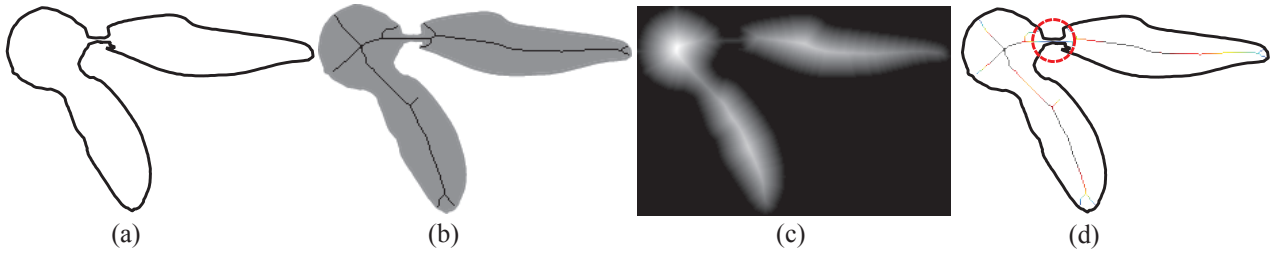

Figure 8: (a) The original contour (b) Floodfilled contour. A skeletonization using morphological operators is computed. (c) Distance transform over floodfilled binary image. (d) The distances are mapped to the skeleton pixels. Clusters of neighboring pixels are identified if they are below a threshold. If the cluster size of neighboring pixels exceed a length d the slice violates the deformation target.

Self-intersections Figure 7(c) shows a case where the slicing process after the deformation results in polygons that are self-intersecting and inside-outside of the shape becomes ambiguous. This also leads to significant printing problems. We check all resulting polygons on self-intersection using the Bentley-Ottmann algorithm [7] and solve the issue by backtracking the deformation until the model has valid slices.

Thin-features A deformation that introduces very thin structures (see Figure 7(d)) is difficult to print or produces results that are highly fragile. We identify this case in image space based on the idea of connected regions that lie very close to the polygon border but exceed an overall area or length. We start the thin structure test by rendering the polygon in printing direction into a texture and calculate skeleton pixels S using morphological image operators. We also compute a distance transform D storing the distance from any pixel within the slice polygon to its closest border pixels (see Fig. 8(a,b,c)). To identify regions in the polygon where the skeleton is very close to the polygon border we map the distances onto the skeleton by $S_D = S \cdot D$ and evaluate S_D to find connected regions on the skeleton pixels (see Figure 8(d)). To this end, we define two thresholds relating to the 3D printing parameters. We define τ as the maximum distance to the border (minimal printing resolution or width of thin structures) and d as the size threshold for connecting neighboring pixels that are below τ. We filter all pixels p in S_D with $p_i \in S_D < \tau$. All 8-connected neighboring pixels $p_i < \tau$ define a connected component C. The length $\|C\| > d$ identifies very thin components in the graph. We again solve this violated constraint by interpolating between source and target until constraints are valid.

6 Results and Discussion

We present an effective pipeline for sketch-based retrieval and modification of 3D models for the purpose of manufacturing. While this combined interface is novel in itself we further identify and propose solutions for three cases that introduce manufacturing problems during the sketch-based modelling process. User centered

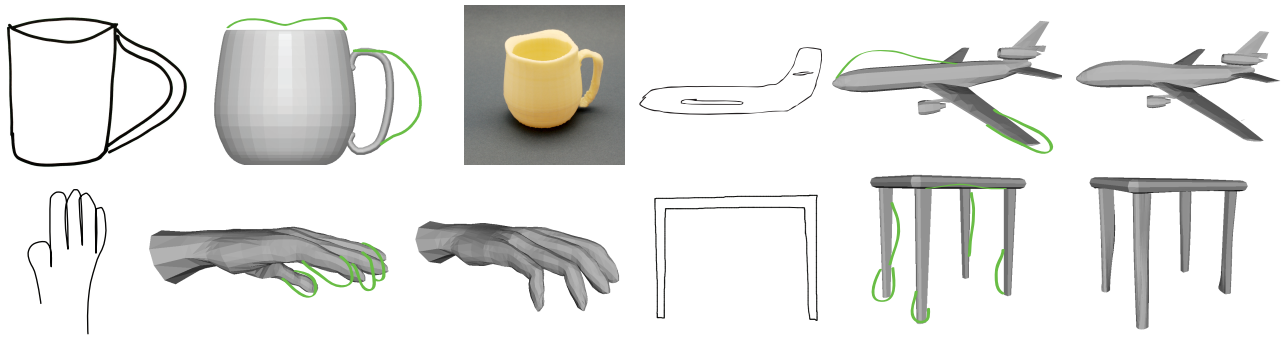

Figure 9: Results of our system. Left: The original user input sketch. Middle: The chosen search result including the modifying silhouette strokes. Right: Final object after the deformation process.

interfaces for mass-customization open up many possibilities for further research. Specifically interesting is to guarantee that the resulting 3D model is still printable. A deeper understanding of the production difficulties is necessary to identify more cases and improve the quality of the digital manufacturing pipeline. Furthermore a formal user study is necessary to evaluate the interface and incorporate feedback for improvements.
Figure 9 shows several examples created with our system. Our 3D model database consists of several thousand 3D models.

Bibliography

[1] M. Bächer, B. Bickel, D. James, and H. Pfister. Fabricating Articulated Characters from Skinned Meshes. *ACM Transactions on Graphics*, 31(3), 2012.

[2] S.-H. Bae, R. Balakrishnan, and K. Singh. Ilovesketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 151–160, New York, NY, USA, 2008. ACM.

[3] B. Bickel, M. Bächer, M. a. Otaduy, H. R. Lee, H. Pfister, M. Gross, and W. Matusik. Design and fabrication of materials with desired deformation behavior. *ACM Transactions on Graphics*, 29(4):1, 2010.

[4] B. Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[5] L. H. B. L. Cohen D. L., Malone E. 3D direct printing of heterogeneous tissue implants. *Tissue Engineering*, 12(4):1325—-1335, 2006.

[6] F. Cole, A. Golovinskiy, A. Limpaecher, H. S. Barros, A. Finkelstein, T. Funkhouser, and S. Rusinkiewicz. Where do people draw lines? *ACM Transactions on Graphics*, 27(3):1, 2008.

[7] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, second edition, 2000.

[8] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella. Suggestive contours for conveying shape. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 22(3):848–855, 2003.

[9] Y. Dong, J. Wang, F. Pellacini, X. Tong, and B. Guo. Fabricating spatially-varying subsurface scattering. *ACM Transactions on Graphics*, 29(4):1, July 2010.

[10] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *ACM Trans. Graph.*, 31(4):44:1—-44:10, 2012.

[11] M. Eitz, R. Richter, T. Boubekeur, K. Hildebrand, and M. Alexa. Sketch-based shape retrieval. *ACM Trans. Graph.*, 31(4):31:1—-31:10, 2012.

[12] M. Eitz, R. Richter, K. Hildebrand, T. Boubekeur, and M. Alexa. Photosketcher: Interactive Sketch-Based Image Synthesis. *IEEE Computer Graphics and Applications*, 31(6):56–66, 2011.

[13] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. . *ACM Transactions on Graphics*, 22(1):83–105, 2003.

[14] M. Hašan, M. Fuchs, W. Matusik, H. Pfister, and S. Rusinkiewicz. Physical reproduction of materials with specified subsurface scattering. *ACM Transactions on Graphics*, 29(4):1, 2010.

[15] K. Hildebrand and M. Alexa. Sketch-based pipeline for mass-customization. In *ACM SIGGRAPH 2013 Talk Program, to appear*, 2013.

[16] J. Hiller and H. Lipson. Design and analysis of digital materials for physical 3D voxel printing. *Rapid Prototyping Journal*, 15(2):137–149, 2009.

[17] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3D freeform design. In *Proc. of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 409–416, New York, NY, USA, 1999.

[18] H. Lipson and M. Kurman. *Fabricated: The New World of 3D Printing*. Wiley Press, 2012.

[19] L. Luo, I. Baran, S. Rusinkiewicz, and W. Matusik. Chopper: Partitioning models into 3D-printable parts. volume 31, Dec. 2012.

[20] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. FiberMesh: designing freeform surfaces with 3D curves. *ACM Trans. Graph.*, 26(3), 2007.

[21] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph.*, 24(3):1142–1147, July 2005.

[22] M. Pache. *Sketching for Conceptual Design: Empirical Results and Future Tools*. Verlag Dr. Hut, 2005.

[23] R. Schmidt, B. Wyvill, M. C. Sousa, and J. A. Jorge. ShapeShop: sketch-based solid modeling with

BlobTrees. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[24] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 1470–, Washington, DC, USA, 2003. IEEE Computer Society.

[25] O. Stava and N. Carr. Stress Relief : Improving Structural Strength of 3D Printable Objects. *ACM Transactions on Graphics*, 2012.

[26] M. Suwa and B. Tversky. What architects see in their sketches: implications for design tools. In *Conference Companion on Human Factors in Computing Systems*, CHI '96, pages 191–192, New York, NY, USA, 1996. ACM.

[27] B. Tversky and M. Suwa. Thinking with sketches. volume 1, pages 75–85. 2009.

[28] T. Vilbrandt, E. Malone, H. Lipson, and A. Pasko. Heterogeneous objects modelling and applications. chapter Universal, pages 259–284. Springer-Verlag, Berlin, Heidelberg, 2008.

[29] I. H. Witten, A. Moffat, and T. C. Bell. Managing gigabytes: Compressing and indexing documents and images - errata, 1996.

[30] S. M. Yoon, M. Scherer, T. Schreck, and A. Kuijper. Sketch-based 3d model retrieval using diffusion tensor fields of suggestive contours. pages 193–200, 2010.

[31] J. Zimmermann, A. Nealen, and M. Alexa. SilSketch: automated sketch-based editing of surface meshes. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, SBIM '07, pages 23–30, New York, NY, USA, 2007. ACM.