

Differential 3D Scanning

Ammar Hattab, Ian Gonsher, Daniel Moreno, and Gabriel Taubin ■ Brown University

Design is an iterative process, an essential feature of which is the ability for the designer to fluently translate abstract ideas into concrete form. To do this effectively, designers use various strategies to represent their ideas, ranging from sketches and low-resolution models to prototypes and CAD models. Each iteration is an opportunity to ask a question about design features, and these different modes of representations give designers the toolset they need to understand, prototype, and critique their solutions to any given design problem.

The rise of digital fabrication has revolutionized this process. Yet, at the same time, many designers bemoan the loss of hands-on, craft-based approaches to prototyping. Is something important lost when you cannot touch the thing you are making? Using your hands to give something form is a different cognitive process than drawing something on a screen. We must make assumptions about form and function when a model is more abstract than when it emerges from a direct engagement with materials.

By combining digital and physical modeling in different iterations, we can employ the best of the two worlds. We believe that new design tools can help designers more easily translate the abstract into the concrete, and more fluently move from the physical to the digital and back to the physical. This is how ideas become real.

A digital fabrication device like a 3D printer can facilitate moving from the digital to the physical, while moving from the physical to the digital requires using a 3D scanning device. Such devices work by making a copy of an entire 3D model. The designer then usually applies only a few changes to the object's 3D shape in each design iteration. Therefore, we argue that we don't need to operate on the whole 3D model, only on the relevant changes in each iteration. When one of the two models is modified, only the changes need to be

transferred to the other model, a process we refer to as *synchronization*.

Previous work showed how we can transfer changes made in a digital model to a physical model using a 3D printer and a milling machine.¹ In this article, we show **how to use a 3D scanner to synchronize the computer model to reflect changes made in the physical model**. Errors may be introduced and can accumulate during the proposed 3D design process by digital fabrication, 3D scanning devices, and surface reconstruction algorithms (see Figure 1). By only transferring the changes we can restrict those errors to the few regions of the 3D model that changed in the design iteration, and we reuse the rest of the digital model from the previous iteration.

There are many digital fabrication and 3D scanning technologies, each with its own problems and limitations. To use these devices for the design process, as we propose here, we need them to provide a fast and user-friendly experience, so designers can focus on their work and let the algorithm do the rest. And to encourage more designers to adopt this method, we want this process to be more intuitive than digital modifications on a 3D modeling software. Thus, it is important to select the right type of 3D scanning and digital fabrication devices and the right type of material that is easy to modify physically and easy to scan. Nevertheless, our method can handle the output of different types of devices with the right selection of parameters.

Completing Michelangelo's Sculpture

Before introducing our method details, we first show a design example using one of Michelangelo's statues.

Differential 3D scanning can detect the differences between a scanned model (point cloud) and a reference model (polygon mesh or CAD model) and then reflect those changes in the reference model. This can save designers time by reconstructing only the small changed regions rather than the entire object.

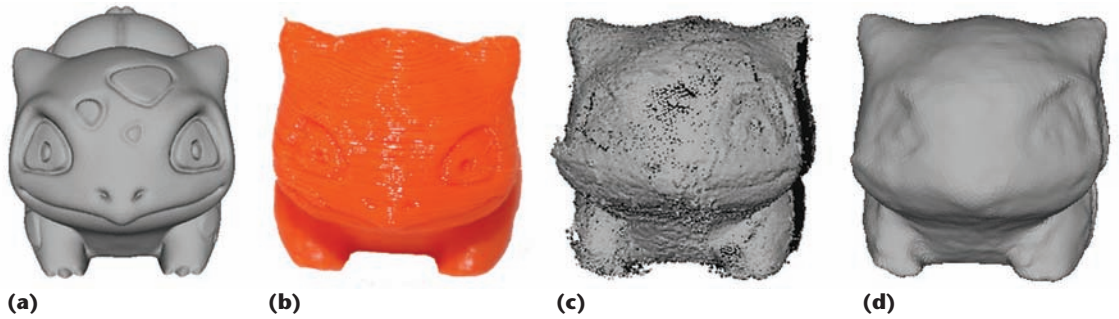


Figure 1. Error sources in the proposed 3D design process: (a) high-resolution reference 3D model, (b) 3D printed copy (which loses all details that are smaller than the 3D printer resolution), (c) 3D scanned copy (with several errors because plastic is shiny and hard to scan), and (d) 3D reconstructed model with important details smoothed.

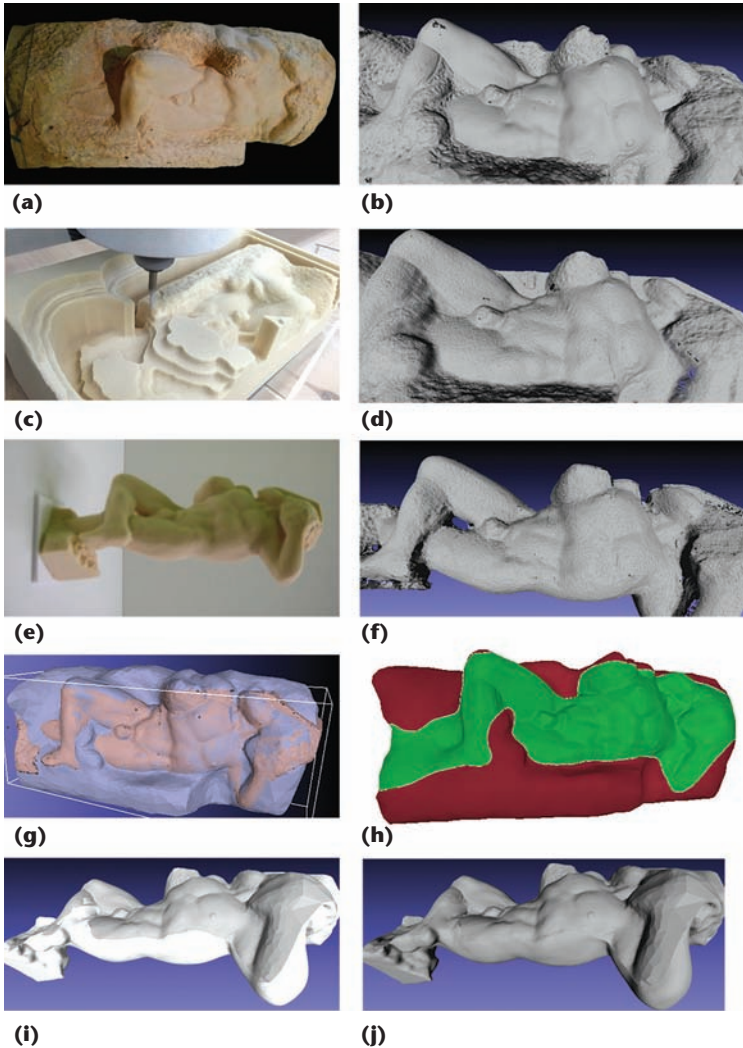


Figure 2. Steps in completing Michelangelo's unfinished work experiment: (a) Michelangelo's original "Awakening Slave" statue in Florence; (b) high-resolution scan; (c) physical copy made with a carving machine; (d) scan after carving; (e) designer manual carving to finish Michelangelo's work; (f) scan after manual carving; (g) aligning the two models using ICP; (h) reference model segmented into unchanged region in green and changed region in red, separated by the smooth boundary curves; (i) changed region deleted and replaced by the reconstructed changed region of the point cloud; and (j) regions merged.

Any design process is only as effective as the tools that allow the creator's imagination to find form in material. Perhaps no better example exists in the history of western art than Michelangelo for translating the clarity of the artist's vision into stone. In a famous quote attributed to the artist, Michelangelo describes this process with a single sentence: "Every block of stone has a statue inside it, and it is the task of the sculptor to discover it."

This quote was our inspiration for investigating how these design tools might be developed into creative strategies, which might be applied to the study of art history and, more generally, the making of art. We asked whether it might be possible to finish an unfinished masterpiece by taking a scan of an original, one of Michelangelo's "Awakening Slave" (intended for the tomb of Julius II), and make a physical model, which could be completed by hand. To do this, we needed to develop a process and the tools that would allow us to move fluently between the virtual and physical models. We needed to be able to both manipulate the model on the screen and manipulate the material by hand.

Michelangelo did not have the benefit of milling machines, 3D printing, CAD, or 3D scans. What he achieved, he did so with his hands and eyes. The tools we developed augment the creator's hands and eyes in order to speculate what could have been and to give these speculations physical form. The process we've developed for this project allowed us to speculate about what these missing pieces could have looked like in the mind of the artist and to modify the materials accordingly.

First, we obtained a high-quality scanned model of the statue from the Digital Michelangelo Project² that contains approximately 2 million vertices. Then, we used a milling machine to carve this model in foam. Next, a designer used sculpting tools to physically carve and finish the unfinished parts of the model. Lastly, we used a 3D scanner to scan the modified model, and we applied our method to get the final virtual model. Figure 2 illustrates all the experiment steps.

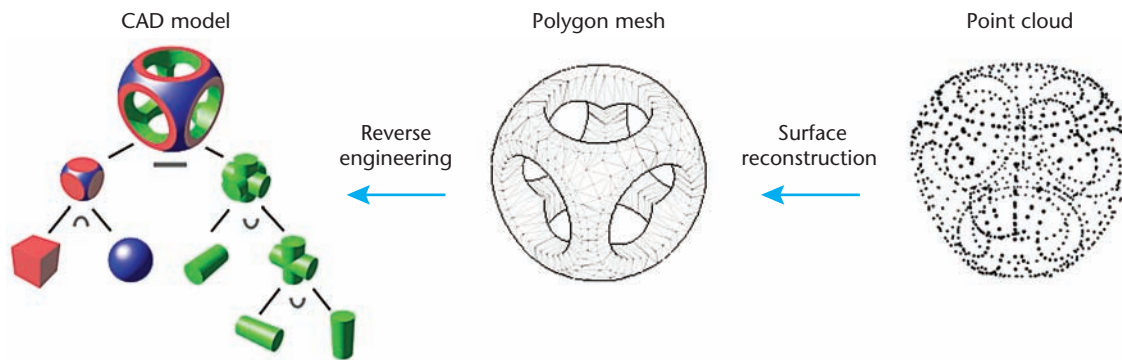


Figure 3. 3D digital representations. The point cloud is a collection of 3D points resulting from 3D scanning. The polygon mesh is a collection of vertices (3D points) connected by edges to form faces (used for 3D printing). CAD models are based on ideal mathematical formulations, such as a tree of binary Boolean operators applied to simple geometry objects.

Our method kept the original high-resolution parts of the virtual model that were carved by Michelangelo, updated only the parts that were modified by the designer, and then merged the two together. If we had tried to reconstruct the whole model’s surface, we would have lost many details from the original high-resolution parts (Figure 2b) because 3D reconstruction tends to smooth out the result.

Method Overview

Our method starts with a point cloud that contains the physical changes and a reference model. The designer could use 3D modeling software to manually apply the changes to the reference model. For example, MeshLab is a popular open source software program that could be used for 3D alignment and surface reconstruction. Then the designer could use mesh-editing software such as 3D Studio Max to identify the differences between the two models by applying a Boolean difference operation. The problem is that the two models match everywhere, except for a few regions. That means that many faces will be incident to each other, which is problematic when applying the Boolean operation. That approach will also detect all small deviations resulting from fabrication and scanning errors and mistakenly consider them as real changes.

Instead of this manual work, we propose an automated method that automatically aligns the two models and then finds and reflects the changes. The input to our method is a point cloud resulting from scanning the modified physical model and the 3D model of the previous iteration, which we refer to as the *reference model*.

The digital representation of the reference 3D model varies between different applications (see Figure 3). It could be a polygon mesh (for example, in some animations applications), or it could be a CAD model as used in mechanical and industrial applications. A polygon mesh is collection of verti-

ces (3D points) connected by edges to form faces, whereas CAD models are based on ideal mathematical formulations. There are several representations for CAD models, like parametric surface patches (NURBS) or constructive solid geometry (CSG). CSG represents the model using a tree, where leafs are the simple geometry objects (sphere, cylinder, cube, and so on), and the links are the binary Boolean operators applied to them (union, intersection, difference). We used CSG as the representation for CAD models because of its simplicity.

The method for transferring the physical changes to the reference model consists of three steps:

1. Align the two models.
2. Find the changes.
3. Reflect the changes.

The first two steps are similar for polygon mesh and CAD models, but the third step requires surface reconstruction of the changes in the case of polygon mesh and reverse-engineering of the changes in the case of CAD models.

Polygon Mesh

For polygon meshes, we propose an algorithm that segments the model into changed and unchanged regions with a smooth boundary curves that separates them. Then, it uses these boundary curves in the surface reconstruction of the changed regions in the point cloud and merges them to the mesh (see Figure 4).

3D Registration

The physical object could be placed in an arbitrary position and orientation after modifications, and it needs to be aligned with the original virtual model in order to perform accurate comparison to find the changes. This process is called *3D registration*. The inputs are a point cloud that resulted from scanning, and a polygon mesh that represents

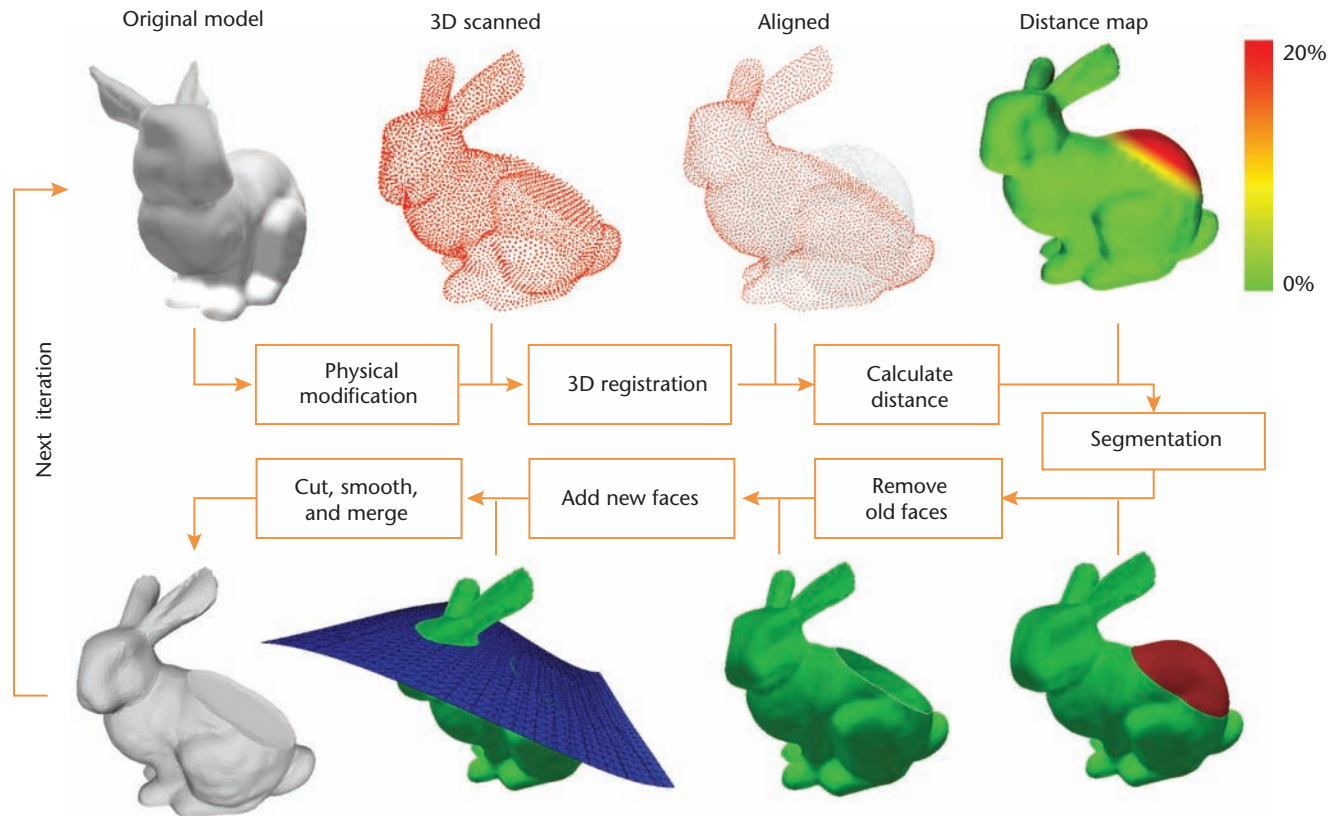


Figure 4. Steps for polygon-mesh construction: Physical modification is an example of design iteration, where the original model is fabricated, physically modified (cut), and scanned back, resulting in a point cloud with changes. 3D registration aligns the two models. Distance calculation detects change by measuring the distance between the two models. The color map shows the distance as a percentage of the diameter of the bounding box of the model. Segmentation segments the model into changed region (red) and unchanged region (green). Changed region faces are removed. The scanned point cloud is also segmented into changed and unchanged regions. Here we reconstruct the changed region only (blue) while using the boundary curves as strong constraints. Then we cut the reconstructed surface using the boundary curves, and we smooth the result. The updated merged model could be used as an input for the next iteration.

the reference model. In our method, we used the popular Iterative Closest Point (ICP) method.³ Our method determines the initial alignment by aligning the principal axes of the two models.

Finding the Changes

The translation from the virtual to the physical and back is not always without some loss of detail. Each device has a specific resolution, and all details in the original model smaller than this resolution will be lost (see Figure 1). For example, the drilling bit thickness on milling machines determines the resulting resolution, and the extruder nozzle diameter on FDM 3D printers controls the resolution. The same thing happens with 3D scanning devices.

Besides this systematic deviation from the reference model, the digital fabrication and 3D scanning devices will also introduce and accumulate noise errors. In some cases, whole regions of the model might deviate, possibly due to gaps in 3D scanning.

It is important for our method to distinguish between those deviations and the actual physical changes applied by the designer. To do so, the applied physical changes must be larger than the largest resolution of the used devices so they can be detected.

Then, when we detect changes, we must use a threshold that is larger than the largest resolution of the used devices, with some tolerance for the noise level, and smaller than the smallest physical change. Knowing the specifications of the used devices could help automate the selection of the threshold distance, which should be the same for the same settings.

The virtual and physical models will match everywhere except in few regions. Each of the changed regions has a part in one model and another corresponding part in the second model, and the size of those two parts might differ between the two models. For example, a small region in the reference model could be elongated in the point cloud. Establishing the correspondence between

the changed regions in the two models might be hard (see Figure 5). Thus, we decided to separate the process in two steps:

1. Find the changed regions in the point cloud. Here, we project each point in the point cloud onto the reference model (polygon mesh) and calculate the Euclidean distance between the point and the projection point. If that distance is larger than a threshold, we mark this point as a change.
2. Find the changed regions in the reference model (polygon mesh). To do that, we operate on the reference model vertices. For each vertex, we estimate the signed distance from this vertex to the point cloud using the nonconvex hull (NCH) surface algorithm.⁴ If this signed distance is larger than a threshold, we mark this vertex as a change.

We allow the user to control the threshold distance T , and anything above that is considered a change: $|d| > T$. The user can specify this threshold as a percentage of the 3D model's diameter or as an absolute value in the 3D model's units, whatever it be (see Figure 6). We also give the user the ability to ignore specific regions that might result from noise.

Finding the Boundary Curves

In the previous step, we calculated a signed distance field for the reference model's vertices that represents the distance to the point cloud. For the unchanged regions, the distance between the two models should be close to zero. In the changed regions, this distance is positive or negative, depending whether the change is to the inside or outside of the model. We need to find the isocontour where this distance field goes from zero to negative or positive.

To extract those boundary contours, we follow these steps:

1. Classify each vertex in the polygon mesh as changed or unchanged using the distance threshold.
2. Find the edges of the mesh that has one vertex marked as change and the other marked as unchanged.
3. For each such edge, extract the isovortex, where the boundary curve intersects the edge. We do that by using a sort of binary search. In each recursive step, we evaluate the signed distance from the edge's center to the surface, and depending on the result, we discard half

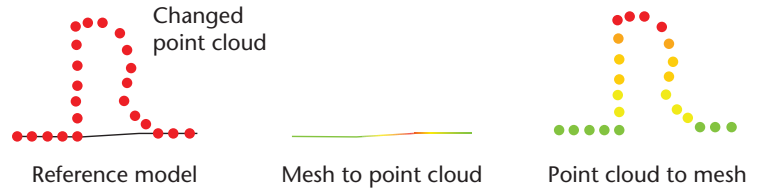


Figure 5. Two direction distance. The middle image shows the distance from the mesh to the point cloud, and the right shows the distance from the point cloud to the mesh.

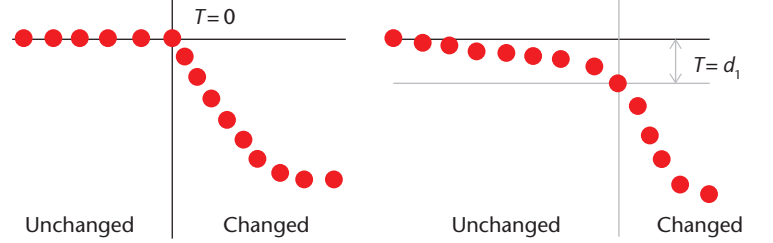


Figure 6. User-specified threshold distance T . Anything above that threshold is considered the new model (red point cloud) changed from the original surface (black line): (a) $T = 0$ indicates a clear change and (b) the user-specified threshold indicates a smooth gradual change.

of the edge and call the function again using the other half. We do that recursively until the two vertices of the edge approach each other.

4. Connect those isovortexes to form one connected contour for each changed region that separates that region from the rest.

Fixing the Curve Displacement

Depending on the noise and the specified threshold distance, the resulting contour might not be smooth and might not align with the actual boundary (see Figure 7). To fix this problem, we run an optimization algorithm where we restrict the isocontour vertices to stay on the surface while we minimize the distance between those vertices and the nearest K points of the changed region in the point cloud. At the same time, we minimize the distance between the two vertices of every isocontour edge to keep it smooth. We run gradient descent to minimize the following energy function for every contour vertex x until we get the desired result:

$$E(x) = \lambda_1 \sum_{(ij)} \|x_i - x_j\|^2 + \lambda_2 \sum_i \sum_j^K \|x_i - p_j\|^2$$

$$p_j \in \{\text{Nearest Points } (x_i, K)\}$$

The first term is a sum over the isocontour edges that contain x , and the second term is a sum over K nearest changed points to x . We restrict the

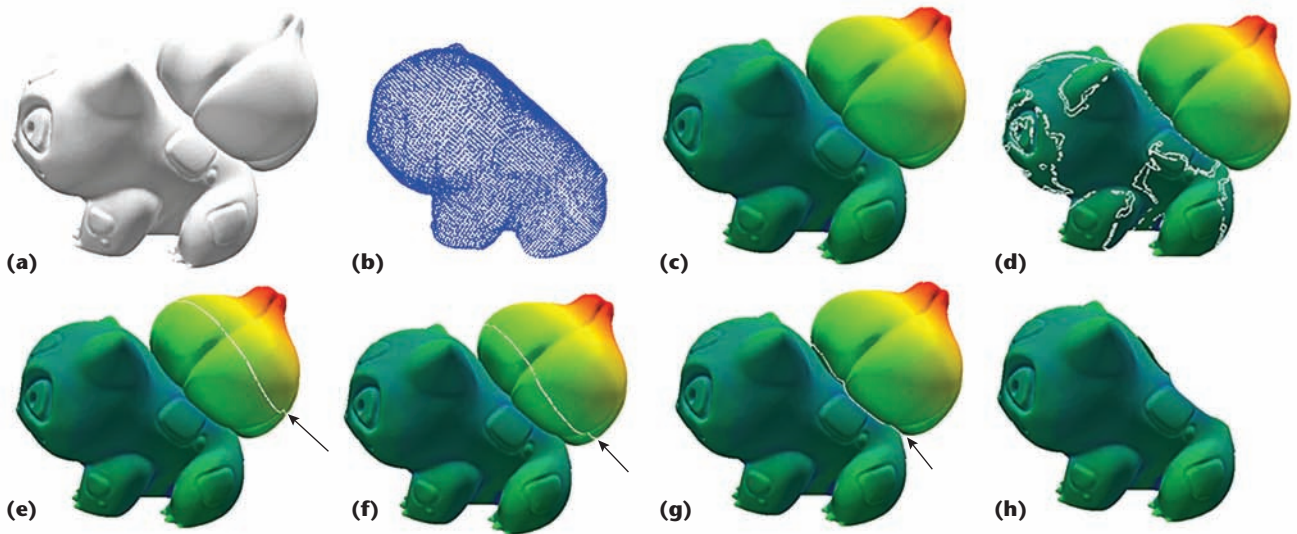


Figure 7. Optimization procedure to fix the boundary curve displacement: (a) original reference model, (b) scanned model of the 3D printed part that was cut by a cutting tool, (c) color map of the signed distance field between the two models, (d) using zero threshold (zero level set) will result in detecting many small changed regions (due to noise and errors), (e) using high threshold will result in one displaced boundary curve, (f) running the optimization algorithm will drag the boundary curve toward the right position, (g) the corrected position of the boundary curve, and (h) the final model cut by the boundary curve.

isocontour vertices to the surface by projecting them onto the mesh after each optimization step. Although the contour vertices must stay on the surface, the contour edges do not. To fix this problem, we project the contour edges on the mesh.

The result is a smooth contour that approximates the real boundary between the changed and unchanged regions. We cut the mesh along this contour because we need to get rid of the old changed regions of the mesh and replace them with the new reconstructed regions from the point cloud.

Now we have every vertex classified as changed or not changed and the contour boundary that separates the changed and the unchanged regions. To obtain a complete mesh segmentation, we also need to segment the faces. To do that, we find incident faces for each vertex—except for the vertices along the contour boundary—and we mark the faces as changed if the vertex is changed, and vice versa. After the segmentation, we delete the faces marked as changed from the mesh and then replace them with the new reconstructed regions from the point cloud.

Reflecting the Changes

To reflect the changes, we need to apply some sort of Boolean operations (intersection, union, subtraction, and difference) to remove and add new the changes to the original model.

Several previous works applied Boolean operations to 3D polygon meshes.⁵ In our algorithm, the changes happen on a point cloud instead, so we need to reconstruct the surface to get a polygon mesh. All previous studies reconstructed the

surface of the whole point cloud before applying Boolean operations, without taking advantage of the reference model.

We propose a surface reconstruction method that only needs to reconstruct the changed regions of the point cloud and takes into consideration the boundary curves that segment the model into changed and unchanged regions. Thus, our surface reconstruction algorithm takes as an input a group of points (the unchanged regions) and a boundary curve that surrounds them.

A few studies have handled surface reconstruction for open surfaces with boundaries. For example, Juncong Lin and his colleagues used the shape boundaries to compose multiple shapes.⁶ Also, the peel algorithm builds a Delaunay mesh using the sampled points while handling the boundary curves,⁷ but it suffers from robustness problems.

The input to this step is a list of contour lines that were extracted in the last step and a list of changed points from the point cloud. We also need to reconstruct those changed points, using contour lines as boundary constraints to the surface reconstruction problem. There are several methods for surface reconstruction, but implicit volumetric methods work well, especially in the presence of noise and holes in the model. For this reason, we tried two implicit methods, Poisson⁸ and SSD (smooth signed distance),⁹ while adding the boundary curves as constraints. Constraints could be added in a hard or soft way. We choose to apply soft constraints by sampling the contour curves and adding those sample points to changed regions of the point cloud. To do so, we follow these steps:

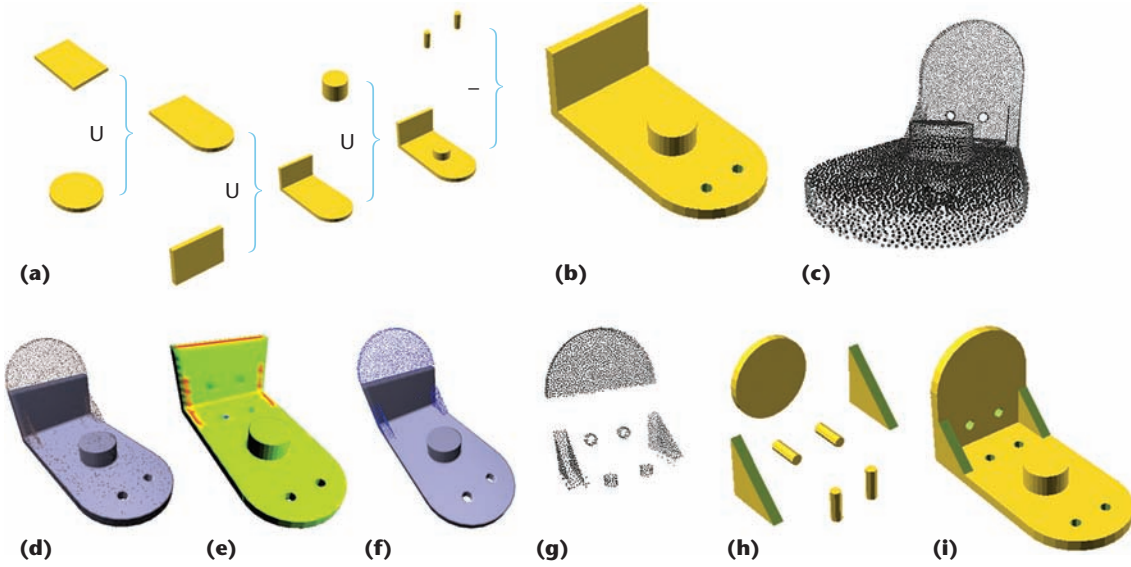


Figure 8. Steps for reverse-engineering CAD model changes: (a) CSG tree where different Boolean operations are applied to simple geometric shapes, (b) resulting model, (c) a 3D scan of the modified object (holes drilled and support added), (d) the two models aligned, (e) a color map shows the signed distance between the two models, (f) changed regions in the point cloud extracted, (g) CHANGED regions separated, (h) the result of fitting planes and cylinders to each changed region (these resulting shapes are added to the original CSG tree with simple union or difference operators), and (i) the updated CAD model.

1. Construct a volumetric grid (a regular grid or an octree) that contains both the changed points and the contour lines. For each cell, it must have at most one contour vertex.
2. Reconstruct the surface while using samples of the contour lines as point constraints. (By simply adding those samples to the changed regions of the point cloud, the more samples we add, the more we pull the surface toward them.) Then, the reconstructed surface will pass near the boundary curves, but it will not stop at them, so we need to cut it.
3. Cut the reconstructed surface by the planes defined by the normals of the contour lines, and get rid of the remaining part.
4. Snap the vertices along the cut to the contour line vertices, and consolidate the vertices into ones.
5. Minimize an energy function that smoothes the reconstructed surface (first three terms) while better approximating the changed points (last term).

$$\begin{aligned}
 E(x, n) = & \lambda_1 \sum_{(ij)} \|x_i - x_j\|^2 + \lambda_2 \sum_{(fg)} \|n_f - n_g\|^2 \\
 & + \lambda_3 \sum_{(ij-f)} (n_f^t (x_i - x_j))^2 \\
 & + \lambda_4 \sum_f \sum_{i \in f} \sum_{k \notin \phi_f} (n_f^t (p_f - x_i))^2
 \end{aligned}$$

If we let the vertex positions be constant, then the energy function becomes quadratic in the nor-

mals, and vice versa. Thus, we need to run a few iterations of the gradient decent with a constant vertex position and then a few iterations with a constant normal. Vertices on the contour are projected back to the contour after each smoothing step to keep them on the contour.

CAD Model and Reverse Engineering

For some applications, the changes might need to be reflected in a CAD model (see Figure 8). Retrieving the CAD model from a point cloud or a polygon mesh is a complex reverse-engineering operation. The advantage of our approach is that we only need to reverse-engineer a few regions and not the whole object.

The goal of reverse-engineering is retrieve the designer's original intent, which is usually represented using ideal shapes (such as a cylinder or sphere). To do that, we can search for those shapes in the point cloud, for example, using RANSAC (random sample consensus).¹⁰ The result will be better, however, if we can segment the point cloud and fit these shapes directly to different segments. In general, most reverse-engineering algorithms have two main components:

- **Segmentation:** Segment the model into surface patches (planes, spheres, quadratic surfaces, and so on, usually based on surface curvature).
- **Fitting:** Directly fit geometric objects (planes, quadratic surfaces, and free-form surfaces) to different surface patches.

Segmentation is the hard step, and the result depends on the sensitivity parameter. In most cases, the user must select some regions to merge manually. RANSAC-based segmentation could result in many false detected shapes, and it could miss existing shapes.

In our case, we do our own segmentation and extract boundary isocurves, which makes things simpler. We also restrict the problem such that the change is always of one type (one change at a time, so the user can, for example, only make one cut to the object or drill one hole at a time). That way we don't need any further segmentation. This solves most of the problems that prevent automatic reverse-engineering, but it also restricts the application of our method to only one change at a time.

The problem then reduces to finding the best-fitting plane, quadratic surface, or free-form surface to the changed points. To do that, we applied an algorithm from earlier work¹⁰ to the group of changed points and the sampled points from the boundary isocurves. We found that adding these sampled points greatly enhances the result of fitting.

The result of the fitting step is a group of shapes (spheres, cylinders, cones, and so on). To simplify the operation, we used CSG to design the reference CAD models; we simply add the shapes that result from fitting to the CSG tree using either a subtraction or addition operation (see Figure 8).

Limitations

When compared with digital modeling, physical modifications are more intuitive for many designers. For thousands of years, people have used various tools to modify physical objects, including hand and power tools that perform operations such as cutting, sculpting, and carving.

Furthermore, many designers already integrate 3D scanning devices with their physical sculpting work. For example, the car industry still uses clay modeling to build a full-sized car model. Then they use a 3D scanner to digitize the designed model. Oftentimes, designers need to design around or fit their designs to existing objects. For example, the medical field uses 3D scanners to create digital models of body parts and teeth to make perfectly fitted prosthetics and dentures. Such devices could also be used to design fixes or extensions to existing broken and old parts or to scan parts of a larger model. For example, automobile customizers use 3D scanning to scan the car parts they want to customize, ensuring that the customization piece will fit seamlessly.

More recently, with the invention of Cx5 sculptable 3D printing material,¹¹ 3D printers are also being employed as the starting point for the physical sculpting process. Cx5 is perfect for this approach because it can be easily modified and scanned (since it is fully opaque). Designers start with digital modeling (or 3D scanning) and then 3D print their models using the Cx5 sculptable material. With these materials, they can hand-sculpt the finest details onto their 3D prints.

In this article, we build on these existing approaches that combine physical design with digital fabrication and 3D scanning devices. The real determination of whether designers will adopt our approach is how easily that use these devices in their work compared with a digital software. We expect our approach to be more easily adopted by designers who already use 3D scanning and digital fabrication devices in their work. And as the use of 3D scanning and printing technologies becomes more widespread, we expect designers to become familiar with them and be more likely to adopt our approach.

Another limitation is that it is difficult to 3D scan shiny and translucent objects, which account for most 3D printed plastic parts. There are opaque 3D printing materials, however, that could be used to overcome this problem.

Also 3D scanning and digital fabrication (such as 3D printers) are generally slow, which limits the number of design iterations using them. For 3D scanners to get an accurate model of an object, we need to scan it from different viewpoints to cover all occluded areas, which takes even more time. For instance, in our experiments, the scanner missed several drilled holes. One way to address this is by using a hierarchical approach to scanning: we first make a fast low-resolution scan of a object, and then once we find the changed regions, we use a slower high-resolution scanning technology to accurately scan them.

The fluent translation from the virtual to the physical and from the physical to the virtual has many implications in the near future. By giving artists, art historians, designers, engineers, and many other creative professionals better tools to navigate between these spaces, new possibilities will emerge.

Moving forward, we intend to build on the work we describe here. We believe we can further refine these processes and work with partners in the application fields to design tools that can help them in their work.

Acknowledgments

This work was partially supported by a Brown Fellowship and by US National Science Foundation grant IIP-1500249.

References

1. A. Teibrich et al., "Patching Physical Objects," *Proc. 28th Ann. ACM Symp. User Interface Software & Technology*, 2015, pp. 83–91.
2. M. Levoy et al., "The Digital Michelangelo Project: 3D Scanning of Large Statues," *Proc. 27th Ann. Conf. Computer Graphics and Interactive Techniques*, 2000, pp. 131–144.
3. K.S. Arun, T.S. Huang, and S.D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 5, no. 5, 1987, pp. 698–700.
4. G. Taubin, "Non-convex Hull Surfaces," *SIGGRAPH Asia 2013 Technical Briefs*, 2013, article no. 2.
5. D. Pavić, M. Campen, and L. Kobbelt, "Hybrid Booleans," *Computer Graphics Forum*, vol. 29, no. 1, 2010, pp. 75–87.
6. J. Lin et al., "Mesh Composition on Models with Arbitrary Boundary Topology," *IEEE Trans. Visualization and Computer Graphics*, vol. 14, no. 3, 2008, pp. 653–665.
7. T.K. Dey et al., "Isotopic Reconstruction of Surfaces with Boundaries," *Computer Graphics Forum*, vol. 28, no. 5, 2009, pp. 1371–1382.
8. M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson Surface Reconstruction," *Proc. 4th Eurographics Symp. Geometry Processing*, vol. 7, 2006, pp. 661–670.
9. G. Taubin, "Smooth Signed Distance Surface Reconstruction and Applications," *Iberoamerican Congress on Pattern Recognition*, 2012.
10. R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for Point-Cloud Shape Detection," *Computer Graphics Forum*, vol. 26, no. 2, 2007, pp. 214–226.
11. A. Beane, "Cx5 Sculptable Filament Kickstarter Video," YouTube, 24 Aug. 2016; www.youtube.com/watch?v=7xqPNYedCTY.

Ammar Hattab is a PhD student at Brown University. His research interests include the integration of 3D scanning and printing, geometry processing, and computer vision. Hattab has an MS in computer engineering from Brown University. Contact him at ammar_hattab@brown.edu.

Ian Gonsher is an assistant professor of practice in the School of Engineering and Department of Computer Science at Brown University. His research interests include developing new strategies for design and enriching the creative process. Gonsher has an MFA in furniture design from the

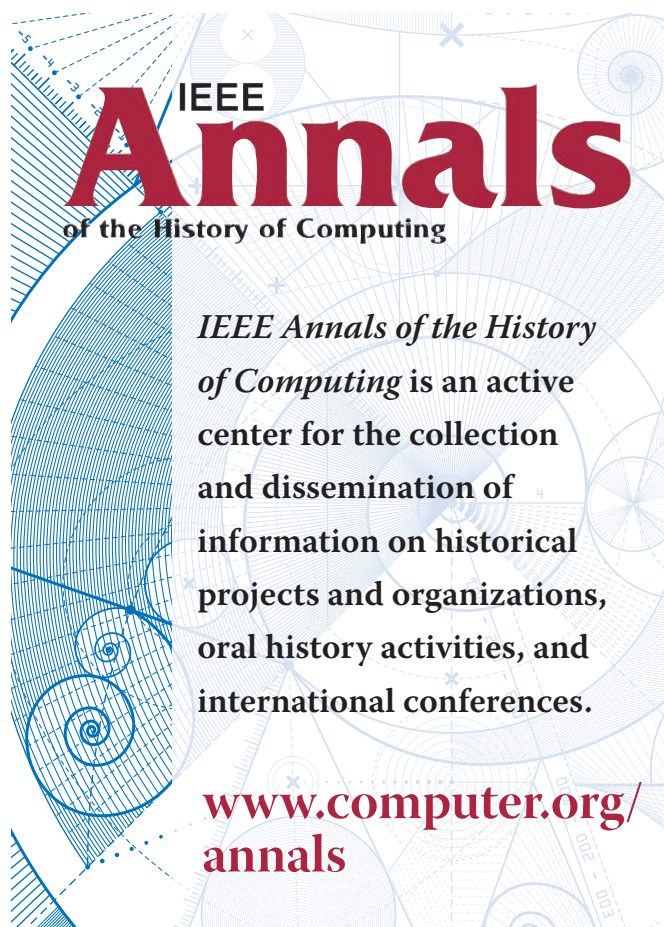
Rhode Island School of Design. Contact him at ian_gonsher@brown.edu.

Daniel Moreno recently graduated from Brown University. His research interests include precision 3D scanning for metrology applications, structured light and phase shifting algorithms, and digital geometry processing. Moreno has a PhD in computer engineering from Brown University. Contact him at daniel_moreno@brown.edu.

Gabriel Taubin is professor of engineering and computer science at Brown University. His research interests include applied computational geometry, computer graphics, geometric modeling, 3D photography, and computer vision. Taubin has a PhD in electrical engineering from Brown University. He is a former editor in chief of IEEE Computer Graphics and Applications and a member of the Geometric Models editorial board. He is also a fellow of IEEE, a member of the Fulbright Specialist Roaster, and a Fulbright Specialist grantee and was named IBM Master Inventor. Contact him at gabriel_taubin@brown.edu.

myCS

Read your subscriptions through the myCS publications portal at <http://mycs.computer.org>.

The graphic features a background of blue technical drawings, including a spiral, a gear, and various geometric shapes. Overlaid on this is the text "IEEE Annals of the History of Computing" in a large, bold, red serif font. Below this, in a smaller black serif font, is the text "IEEE Annals of the History of Computing is an active center for the collection and dissemination of information on historical projects and organizations, oral history activities, and international conferences." At the bottom right, the website "www.computer.org/annals" is written in a red serif font.

IEEE
Annals
of the History of Computing

IEEE Annals of the History of Computing is an active center for the collection and dissemination of information on historical projects and organizations, oral history activities, and international conferences.

www.computer.org/annals