3D Modeling by Scanning Physical Modifications

Conference Paper · August 2015

DOI: 10.1109/SIBGRAPI.2015.8

Ammar Hattab
Brown University
6 PUBLICATIONS **18** CITATIONS
SEE PROFILE

Gabriel Taubin
Brown University
231 PUBLICATIONS **11,415** CITATIONS
SEE PROFILE

3D Modeling by Scanning Physical Modifications

Ammar Hattab*, Gabriel Taubin†
School of Engineering
Brown University
Providence, USA
*Email: *ammar_hattab@brown.edu, †gabriel_taubin@brown.edu*

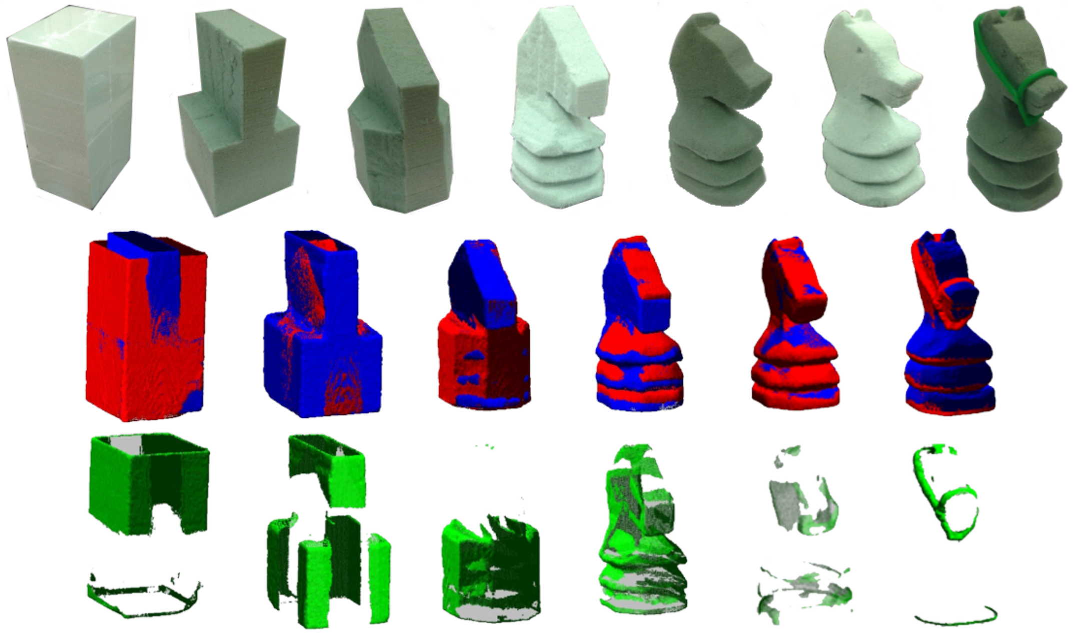


Figure 1. Carving Experiment; top row: steps of physical carving in foam, middle row: the scanned point clouds aligned to each other, the bottom row: the detected changes.

***Abstract*—3D shape design tends to be a long and tedious process, with the design of a detailed 3D part usually requiring multiple revisions. Fabricating physical prototypes using low cost 3D fabrication technologies at intermediate stages of the design process is now a common practice, which helps the designer discover errors, and to incrementally refine the design. Most often, implementing the required changes directly in the computer model, within the 3D modeling software, is more difficult and time consuming than modifying the physical model directly using hand cutting, caving and sculpting tools, power tools, or machine tools. When one of the two models is modified, the changes need to be transferred to the other model, a process we refer to as synchronization. Changes made to the computer model can be transferred to the physical model by 3d printing a new physical model. In this paper, we address the problem of synchronizing the computer model to changes made in the physical model by 3D scanning the modified physical model, automatically detecting the changes, and updating the computer model. The proposed process comprises algorithms to: 1) register each 3D scan with a previous 3D scan and/or with the 3D representation used by the 3D modeling software; 2) detect the changes (subtractive and/or additive); and 3) perform the changes on the 3D computer model.**

***Keywords*-3D registration, 3D reverse engineering; 3D modeling; tangible interface; human-computer interaction;**

I. Introduction

It usually takes a designer several hours to create a detailed 3D model using interactive 3D modeling software. The process may be long and tedious. Learning how to use a specific 3D modeling software package, with a specific graphical user interface, requires a significant amount of time. In addition, the designer needs to develop skills to manipulate and modify 3D models based on their 2D projections. As a result, many people leave this craft to experts.

Also designing a 3D model is an iterative process that could involve several steps, usually not well integrated with each other. For example, a designer could start by drawing 2D sketches. Then he could create a rough physical model using cardboard or some other material. Subsequently he could create a rough 3D model on the computer. At any time in the process he could use a rapid prototyping machine, such as a low cost 3D printer, to fabricate a physical version of the 3D model. The designer might continue working and refining his 3D model, going back and forth between these representations until he achieves his goal, and finishes the design of his 3D model. The computer representation of the resulting 3D model varies between different applications. It could be a polygon mesh, for example in some animations applications, or it could be a parameterized surface patch CAD model, as used in mechanical and industrial applications.

In this paper we only focus on the process of synchronizing the computer model to changes made to the physical model. We introduce a new method that allows the designer to move fluidly from the physical model (for example his 3D printed object, or his carved object) to the computer model. In our proposed process the physical modifications applied by the designer to the physical model are detected by 3D scanning the physical model and comparing the scan to the computer model. Then the changes are reflected in the computer model. The designer can apply further changes either to the computer model or to the physical model. Changes made to the computer model can be synchronized to the physical model by 3D printing a new physical model, see Figure 2.

For thousands of years, people have used various tools to modify physical objects. Lots of hand and power tools exist to perform operations such as cutting, sculpting, and carving. In this paper we want to exploit this fact. Motivated in part by the current Makers Movement [1] we argue that modifying the physical model using tools is much more intuitive for most designers than making changes to the computer model using software, and most designers already have the necessary skills to do so.

In addition, by recording the history of physical modifications, resulting from the sequence of 3D scans made after each set of modification, our method allows the user to *undo* changes, since he could go back to any previous step in the process by 3D printing any of the various instances of the computer model from the saved history.

II. Related Work

A. 3D Modeling

Several approaches have been proposed to enhance the 3D modeling process based on exploring different kinds of computer interfaces. Jankowski [2] summarizes enhancements proposed for traditional interfaces such as mouse and touch. Other works are based on virtual reality, tangible physical interfaces, or a combination of these two methods.

A virtual reality interface puts the user in a virtual or augmented reality environment, and allows him to use his hands to draw or modify the 3D models. Some studies use smart glasses, such as the commercial Meta glasses,

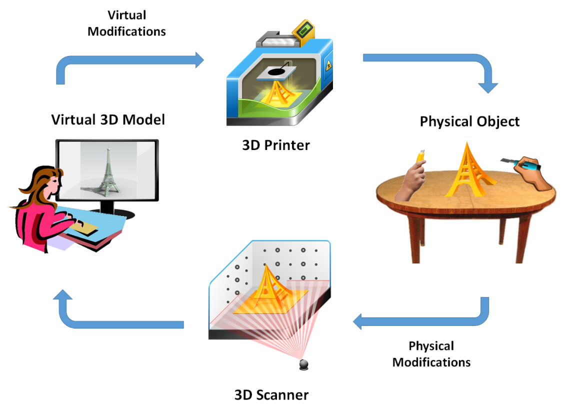


Figure 2. Virtual-Physical 3D Modeling Cycle

for 3D modeling (see Figure 9). Keefe [3] uses the CAVE environment to perform 3D painting. Weichel [4] and Verlinden [5] mix a virtual reality interface with tangible modeling. Several studies introduced new types of tangible interfaces for 3D modeling.

For example Song [6] uses a special pen to draw annotations on physical objects which are interpreted as cut and edit operations on the computer 3D model; Sheng [7] uses a deformable physical prop and camera-based motion tracking to perform virtual 3D sculpting; Shen [8] uses a physical curve that could be hand shaped to get a rotational symmetrical 3D model out of that curve; Huang [9] and Anabuki [10] use physical folding of polygons to shape the 3D model like an origami; Wibowo [11] uses a 3D interface for clothing design; Willis [12] builds interactive fabrication machines that allow hand-controlled 3D printing; and Reed [13] uses sensors inside clay for 3D modeling.

Since tangible interfaces provide real-time tactile and visual feedback to the user, allowing him to fine-tune the 3D shape he is creating using his hands, tangible interfaces are considered superior to virtual reality interfaces. The major problems with most existing tangible interfaces are: that they are not intuitive to use; most are very complex; hard to learn; require additional device, and finally, sometimes they only allow for the creation of limited types of 3D models (see Figure 9).

The goal of this paper is to provide a natural and easy way, yet accurate and fast, for the user to create 3D models using his cutting, sculpting and carving tools, and a process based on 3D scanning to reflect the changes in to the computer model.

B. The Use of 3D Scanning

The creation of product prototypes is critical to many industries. Huang [14] explains that, because of the convenience of using physical tools in the tangible modeling of object, people often choose to create their prototypes in the workshop rather than on the computer, using materials such

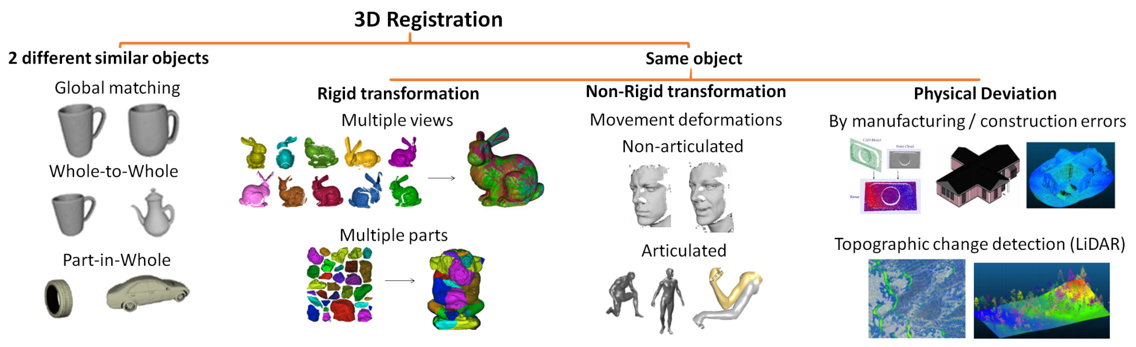


Figure 3. Some 3D registration applications

as clay or wood, as they do in the automobile design industry, and then using a 3D scanner and reverse engineering to create CAD model of their physical prototypes.

3D scanners generate point clouds as samples of the surface geometry of 3D objects. These point clouds usually contain gaps, are not accurate, and, depending on the technology and price of the 3D scanner, they may be noisy. To be able to use them for 3D modeling, several processing steps must be applied to convert them into 3D CAD models, as described for example by Wang [15]: smoothing/denoising; reconstructing triangular meshes; segmenting the meshes into surfaces; reconstructing solid features from the surfaces; and applying modeling operations to get the final reconstructed geometric model.

These steps are called 3D reverse engineering, and there are several commercial applications that help the user in performing these steps. Because of the complexity of the required reverse engineering steps, and depending on the nature of the data, which may be noisy or contain gaps, this process is usually hard and slow, produces many errors, and requires manual intervention.

The benefit of our proposed method is that we only need to reverse engineer the changes, not to repeat the whole reverse engineering process for the whole model. Also, since the changes are usually small, in many cases we don't have to follow the same complex reverse engineering steps to reflect them to the virtual model, but much simpler and faster operations are sufficient.

We believe that recent advances in 3D scanning technology will eventually allow for real-time capturing of the smaller physical changes as the user apply them, thus simplifying and speeding up our method, and resulting in a greater advantage for the user.

C. 3D Registration

The main component of our method is the 3D registration algorithm. Previous works on matching and aligning 3D models are described in the survey by Tam [16].

3D registration algorithms have many applications, summarized in Figure 3. The most common case, described for example by Huang [17], is the problem of creating complete unified 3D model from multiple partial 3D scans of the same object, captured from different viewpoints. Since these 3D scans describe parts of the same rigid object, the operation is called rigid registration, because only rigid body transformation are required to align them. As described by Allen [18], if the object deforms or moves in between 3D scans, then non-rigid registration needs to be performed. Sometimes 3D registration is required to match two different but similar objects. 3D registration is also used for checking the quality of manufactured products in industrial applications or building construction, as described by Tang [19] and Kahn [20], or to detect to topographical changes using LiDAR data, as described by Qin [21].

In all these applications, the 3D registration could be rigid or non-rigid, and it could be performed in several steps. For example, in the case of rigid registration, the algorithm usually starts by finding a rough initial alignment for the two models, called coarse registration, followed by an iterative refinement fine registration step, to determine the exact alignment.

Coarse registration is usually performed using constraints such as affine ratio (Aiger [22]), PCA (Liu [23]), RANSAC (Chen [24]); or by using features such as Spin Images (Johnson [25]), curvature or moments (Gal [26]), Integral Descriptors (Pottmann [27]), FFT, DCT Coefficients (Li [28]), or based on saliency (Digne [29]). Some other studies used bounded search in 3D space for the best initial alignment (Yang [30]).

The most common algorithm for performing fine 3D registration is the Iterative Closest Point (ICP) introduced by Besl [31], which has many variants, as summarized for example by Rusinkiewicz [32]. In general the algorithm alternates between establishing point to point correspondences between the two models, and finding the optimal alignment for those correspondences. The iterative process continues until the total alignment error falls below a specific threshold.

In the case of non-rigid registration, some algorithms are based on segmenting the 3D models; some use skeletons

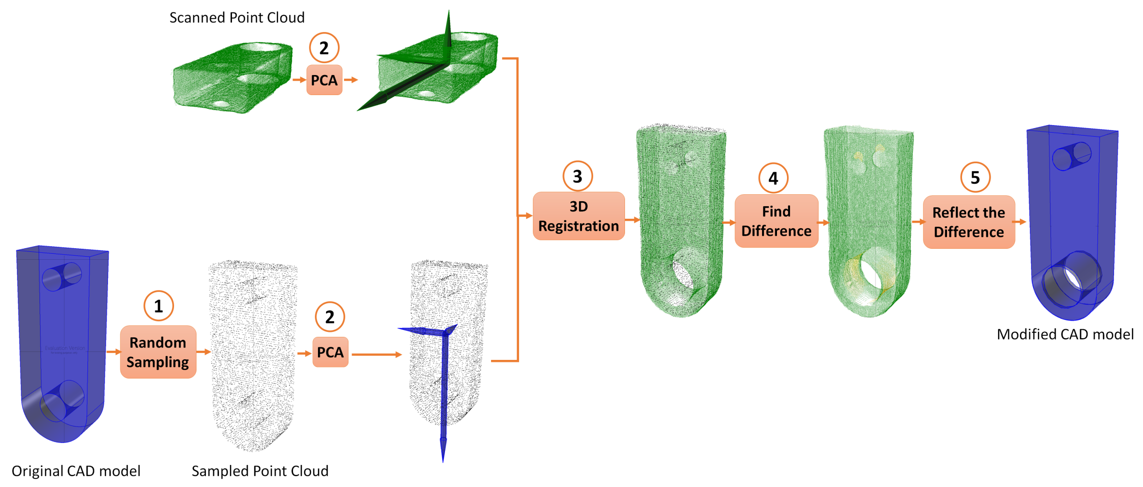


Figure 4. Method Steps: 1- Convert the 3D mesh or CAD model to point cloud. 2- Perform the principal component analysis for the two point clouds. 3- 3D registration. 4- Finding the difference. 5- Reflecting the changes to the 3D mesh or CAD model.

(Allen [18]) or bones and joints (Chang [33]) for articulated movements; and others are based on using a common template (Wand [34]). Yet other methods use registration in a different domain, such as the spectral domain (Jain [35]). We should also note that the same features mentioned above could be used for non-rigid registration. Common features used for non-rigid registration are the heat diffusion signature features (Bronstein [36]).

III. The Method

Our method starts with one point cloud corresponding to the 3D scan of the modified physical model, and either: 1) a point cloud corresponding to a previous scan; 2) a polygon mesh model; or 3) a CAD model. The steps are summarized in Figure 4, with the most important one being the 3D registration step. In this paper we focused on rigid deformations that only require rigid registration. Our method is based on the iterative closest point algorithm (ICP) [31], [32], where the algorithm iterates between finding the correspondences between the two models and finding the best alignment for those correspondences, until the total alignment error becomes smaller than a specific threshold.

We should notice that the point cloud is represented as a list of points, with each point having three coordinates X, Y and Z. A 3D polygon mesh is represented as a list of faces, a list of edges, and a list of vertices. And a CAD model is represented as a list of parametric NURBs surfaces, with each surface having a number of control points in the U and V directions (depending on the U and V degrees of the surface), a knot vector and a number of boundary curves (trim curves). So, to perform 3D registration between these different representations we have two options: 1) to directly optimize the distances from the points to the faces/surfaces; or 2) convert the 3D mesh or CAD model to a point cloud, and perform 3D registration between two point clouds. We tried the two options. The problem with the first option is that we need to find the distance between each point in the point cloud, and each face/surface in the polygon mesh or the CAD model, which may take very long time. Since we need to do this for each of ICP step, the overall algorithm may take a very long time to run, although the process could be sped up by using space partitioning data structures. The second option is much faster, but since sampling is involved, there is data loss which may result in accumulated errors, which could be problematic in some cases. Choosing the sampling rate carefully could help minimize the problem. Since in our experiments we did not observe much difference in the results, we decided to base our implementation on the second option, where the polygon mesh or the CAD model is densely sampled into a point cloud.

A. Step1: Convert the polygon mesh or the CAD model to point cloud

Algorithm 1 describes how we generate a point cloud from a CAD model by random sampling based on surface area. We use a similar algorithm to sample a polygon mesh. This algorithm starts with a specific sampling rate specified by the user, and then it performs a random uniform sampling on the CAD model surfaces, to get a point cloud with approximately uniform number of samples per unit area.

B. Step2: Perform principal component analysis on the two point clouds

The iterative closest point algorithm (ICP) could get stuck in a local minimum if it is started from a bad initial alignment. Our method determines the initial alignment by aligning the principal axes of the two models. To find the principal axes we perform principal component analysis using the singular value decomposition algorithm (SVD) on the matrices formed by the point clouds. The singular

Algorithm 1 CAD Model Random Sampling

```
Require: Sampling rate $s > 0$
  initialize resulting point cloud $R$
  for each Parametric surface $S$ in CAD model do
    Calculate the surface area $A$ = CalculateArea($A$)
    Calculate the number of random points to generate $n$ = GetNumberOfPointsForSurface($A$, $s$)
    for $i = 0$ to $n$ do
      Get a random value $u$ from $U$ domain
      Get a random value $v$ from $V$ domain
      Find the 3D point $P(X, Y, Z)$ = GetPoint($S$, $u$, $v$)
      if $P$ inside surface boundaries (trim curves) then
        Add $P$ to $R$
      else
        Repeat the same steps to find another point
      end if
    end for
  end for
  return resulting point cloud $R$
```

vectors define the principal axes. The result of matching similar singular values determines how to map each of the three axes from one point cloud to the other. Using this map we calculate the rotation component of rigid body transformation that we use to determine the initial alignment. The translation component of the transformation is computed as the difference between the centroids of the two point clouds. In fact we use multiple initial transformations. To generate more initial alignments we just flip the direction of one principal axis, calculate the transformation of the flipped axes, and add the result to the list of initial alignments to try. Then we start the ICP procedure using each of these initial alignments, and we choose the one that gives the least error.

C. Step3: 3D registration

After we generate the list of initial alignments, we use a variant of the iterative closest point (ICP) for fine 3D registration. Algorithm 2 shows the 3D registration steps. In summary, we first randomly select points from both point clouds. Then for each initial alignment in the list, the ICP loop repeats the following two steps until the total error drops below a user specified threshold:

1) Using a kd-tree structure, find the closest point in the second point cloud for each point in the first point cloud.
2) Determine the best least-squares rigid body alignment for the two point clouds, using for example the method described by Arun [37], and align the two point clouds.

D. Step4: Finding the difference

After the two point clouds are correctly aligned, we update the kd-trees. For each point in one of the point clouds, we use the kd-tree to find its closest point in the other point cloud. Then for each pair of point correspondences we calculate the Euclidean distance between the two points. If the distance between the two matching points is higher than a specific threshold relative to the point cloud diameter, then we label pair of points as belonging to the changes. Then we combine these labeled pairs into regions, and color code these regions to show them to the user.

Algorithm 2 3D Registration

```
Require: point clouds sampling rate $s$
  Calculate centroids $c1$ , $c2$ of input point clouds $p1$ , $p2$
  Translate each point cloud to the origin:
  $p1 = p1 - c1$
  $p2 = p2 - c2$
  Sample each point cloud:
  $p1$ = RandomSampling($p1$, $s$)
  $p2$ = RandomSampling($p2$, $s$)
  Build kd-tree $t2$ for the second point cloud
  Perform PCA for both point clouds $PCA1, PCA2$
  Generate a list of initial alignments $IAs$
  for each initial alignment $A$ in $IAs$ do
    Apply alignment $A$ on $p1$
    Start ICP iterations:
    while Step error below threshold do
      Calculate correspondences $nc$ using kd-tree $t2$
      Calculate best alignment:
      Calculate $H = \sum_{i=1}^{nc} pi1 * pi2$ {pi1 and pi2: 3D points of the correspondence }
      Find SVD of H: $H = UAV^t$
      Calculate $R = VU^t$
      Calculate $T = p2 - Rp1$
      Apply alignment $p1$ =ApplyAlignment($R$, $T$)
      Calculate step error
    end while
    Calculate Total error $E$
    if Total error $E <$ minimum error $Emin$ then
      Set minimum error $Emin = E$
      Set minimum transformation $RTmin = (R, T)$
    end if
  end for
  return $RTmin(R, T)$
```

E. Step5: Reflecting the changes to the CAD model

In this step we have the difference regions which comprise pairs of points, with the first point sampled from a face/surface of the original polygon mesh or CAD model, and the second point belonging to the point cloud which resulted from 3D scanning the modified physical model. Using these pairs we need to modify the original 3D polygon mesh or 3D CAD model. We need to add, remove and/or deform faces or surface patches from the computer model.

In the case of a polygon mesh, we follow these steps:

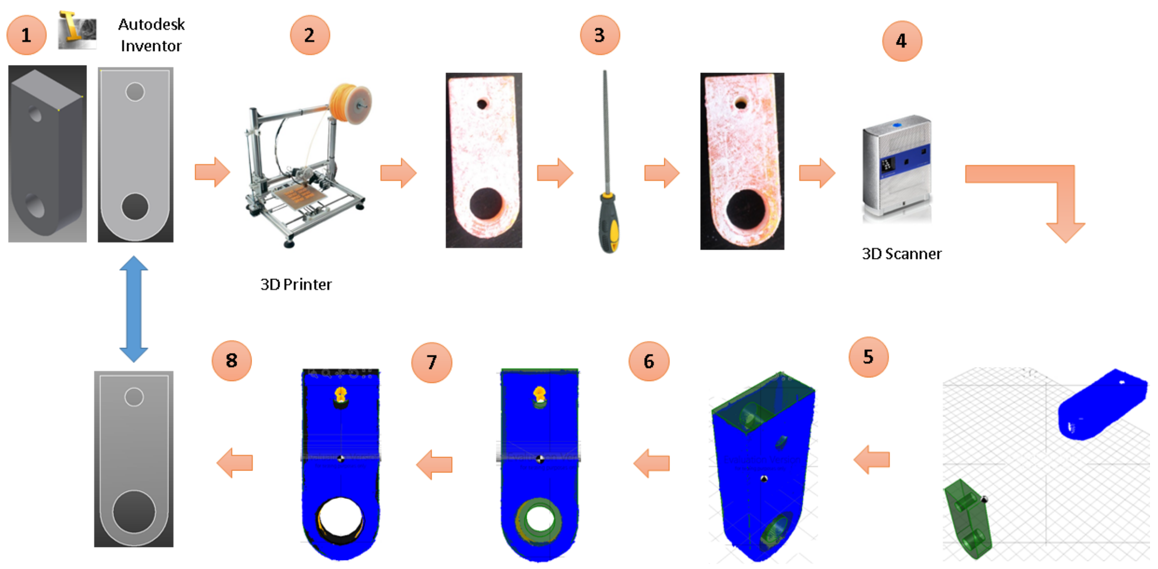


Figure 5. 3D printing experiment steps: 1- Draw the model in Autodesk inventor 2- Print the model using 3D printer 3- Using round file to expand the hole in the physical object 4- Scan the object using 3D scanner 5- 3D registration between the scan point cloud and the CAD model 6- Find and visualize the difference region (shown in yellow) 7- Reflect the changes by expanding the same region in the CAD model 8- Export the modified CAD model and import it back to Autodesk Inventor

1) Find all faces from the original polygon mesh that have sample points in the difference region, and remove them from the resulting polygon mesh. These faces represent old surface patches which had been removed in the physical model if the modification is subtractive, or covered by the new material, if the modification is additive.
2) Using the ball pivoting algorithm, as described by Bernardini [38], we reconstruct a triangular mesh patch for each difference region, using the second point of each pair in the difference region. These are the points coming from the 3D scanning point cloud, which represent new faces or surface patches to be added to the computer model.
3) Add the resulting triangular mesh patches to the original polygon mesh model.

In the case of CAD models with parametric surface patches we need to perform 3D reverse engineering of these regions. Changes to the physical model might result in new surface patches which need to be added to the model, or in existing surface patches which need to be modified. In this paper we focus on the simple case when the changes only modify existing surface patches. To understand the nature of the change, we associate these changed regions with CAD surfaces (assign each point in the changed region to its closest surface in the CAD model), and then for each surface calculate the histogram of distances from points to the surface. If the histogram shows a high peak at a specific distance (see Figure 6), it means that a large number of points are displaced by that distance, we need to offset the surface by that distance in the direction of points displacement.

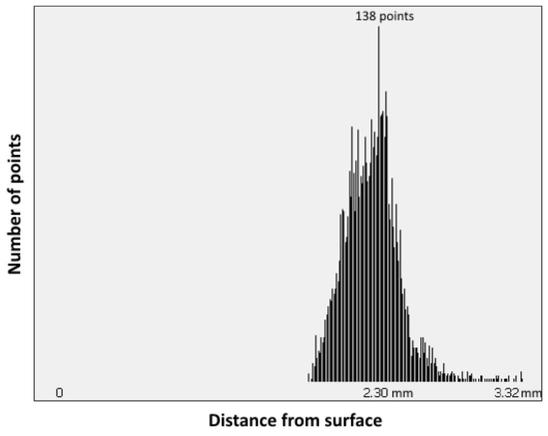


Figure 6. Distances Histogram (from each point to the surface)

IV. Experiments

A. 3D Printing Experiment

The goal of this experiment is to show a real life example of our proposed method that shows the advantage of reflecting the changes made to the physical model onto the computer model. The experiment shows a common scenario that would happen to most people who design their 3D models and 3D print them. Because people design their models on a virtual world they have no sense or feeling on how they will look like actually when they are printed, even with accurate measurements 3D printed parts sometimes require manual modifications (cutting or filing) to satisfy their goals, or to fit in their place. And since 3D printing takes quite a long time, it's not feasible to print the same part multiple times. But people prefer to modify the printed part to make it fit, and if they need to keep the modifications, they have to do it again on the virtual CAD model. One of the benefit of our method is that we help them save time by automatically reflecting these change to the CAD model.

Also in this experiment we used simple changes (expanding a cylindrical hole in the object) to show that we don't have to apply the complex reverse engineering steps in every case, but with very simple operations we can detect and reflect some types of changes.

In this experiment the goal is to design a holder for a cylindrical laser pointer. After printing the part, we discover that the laser pointer doesn't fit into the hole. We use a round file to physically expand the hole until the laser pointer fits in place. Then we use our method to reflect the physical change that we made using the round file, onto the original CAD model. The experiment steps are summarized in Figure 5:

- We measured the laser pointer dimensions.
- We drew the model in Autodesk inventor, dimensioning the hole to hold the laser pointer, and we exported the model in ".step" 3D solid format.
- We 3D printed the model using the "Velleman K8200" 3D printer. Here we discovered that the laser pointer didn't fit into the printed part hole.
- Using a round file to expand the cylindrical hole in the printed part until the laser pointer fits inside it.
- We scanned the modified object using a "NextEngine" 3D scanner.
- We performed the proposed 3D registration procedure to align the scanned point cloud to the CAD model.
- We detected and visualized the difference regions. Here we notice that the difference regions contained the actual changed region "modified physically by hand", as well as some noisy regions resulting from the 3D scanning process.
- We reflected the changes onto the CAD model. Here for each surface patch we calculated the histogram of distances from different points to that surface. Then we checked each histogram to find whether there was any high peak at a specific distance. In this experiment we found a high peak of 2.3 mm, as shown in figure Figure 6, in the cylindrical surface representing the hole. So, we needed to expand the cylindrical hole surface by adding 2.3 mm to its radius since the change direction is pointing to the central axis for all points.
- Finally, we exported the modified CAD model in ".step" 3D solid format and imported it back to Autodesk Inventor.

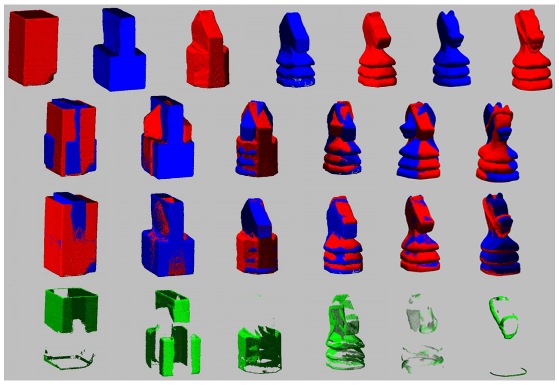
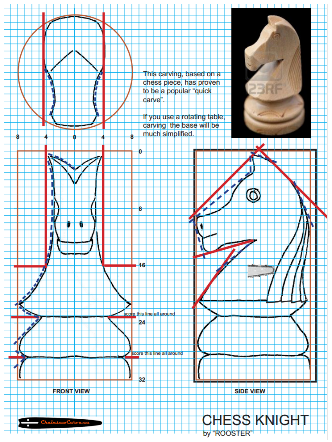


Figure 7. Carving Experiment Results: first row: scanned point clouds, second row: point clouds before registration, third row: point clouds after registration, fourth row: the difference region.


This carving, based on a chess piece, has proven to be a popular "quick carve".
If you use a rotating table, carving the base will be much simplified.
8
4
0
4
8
0
8
16
24
32
score this line all around
score this line all around
FRONT VIEW
SIDE VIEW
ChainsawCarve.ca
CHESS KNIGHT
by "ROOSTER"
Figure 8. Carving Guide

B. Foam Carving Experiment

The goal of this experiment was to test the 3D registration process, and the process of detecting changes through a multi-step carving operation. Here we carved a Chess knight piece in foam, as shown in Figure 1. We started by following the carving guide shown in Figure 8. Then, using simple carving and cutting tools we incrementally carved the foam object, and after each carving step we used a MakerBot Digitizer 3D scanner to capture a full 3D scan of the object. To test the performance of the method on addition operations, we added a horse handle to the object in the last step. In Figure 7 we can visualize the registration results for all steps, and the difference regions, whether they were added or removed.

V. Results and Discussion

By only using a 3D scanner to capture the shape of the modified physical object, we allow the user to use his skills with cutting, sculpting and carving tools to create his 3D models. Compared this process with previous expensive tangible interfaces which are also limited to certain applications (see Figure 9 the image to the left). At the same time we are allowing the user to get natural tangible feedback of what he is doing, which in our opinion is better than using augmented reality interfaces (see Figure 9 the image to the right).

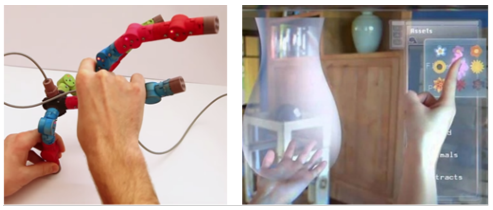


Figure 9. Previous 3D modeling techniques

The 3D printing experiment shows a possible useful direct application of our method in the 3D printing field. It also shows that we could use simple operations to reflect the changes to the original model in some cases, while we cannot avoid using reverse engineering operations to find new surfaces in other cases.

The foam carving experiment shows that our 3D registration algorithm works well even if initial alignments of the two models are in opposite direction (see the last two steps in Figure 7). The reason for this good behavior is that the method searches for the best initial alignment by trying different alignment candidates produced by the PCA analysis.

Getting good 3D scans was challenging. We could avoid the noise coming from the 3D scanner by taking a 3D scan before and after the modifications, and finding the difference between the two scans. This process will cancel out the scanning errors.

VI. Conclusion

The method described in this paper enables the user to start with a physical object like a 3D printed object, and physically modify it, with the system reflecting the modifications onto the computer 3D model automatically. The user is not required to manually perform the modification on the computer 3d model in the modeling software as well. The proposed algorithm allows for the physical and the computer model to stay synchronized independently of whether changes are performed to the physical or to the computer model.

In terms of future work, to complete the design process loop, we need to work on transferring changes in the other direction, from the computer model to the physical model. Here we need to develop technologies to speed up the fabrication and/or modification of physical models from computer models.

We should notice here that despite their advantages and flexibility, 3D printers and other digital fabrication machines are far too slow for an interactive and fluid 3D shape design process. The same could be said about 3D scanners. Also the materials used for 3D printing normally are very hard to modify. But with the right selection of the printing materials and with more advances in 3D printing/scanning our method will find more applications and become more practical.

Acknowledgment

References

[1] D. Dougherty, "The maker movement," *innovations*, vol. 7, no. 3, pp. 11–14, 2012.

[2] J. Jankowski and M. Hachet, "A survey of interaction techniques for interactive 3d environments," in *Eurographics 2013-STAR*, 2013.

[3] D. F. Keefe, D. A. Feliz, T. Moscovich, D. H. Laidlaw, and J. J. LaViola Jr, "Cavepainting: a fully immersive 3d artistic medium and interactive experience," in *Proceedings of the 2001 symposium on Interactive 3D graphics*. ACM, 2001, pp. 85–93.

[4] C. Weichel, M. Lau, D. Kim, N. Villar, and H. W. Gellersen, "Mixfab: a mixed-reality environment for personal fabrication," in *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 2014, pp. 3855–3864.

[5] J. Verlinden, A. Kooijman, E. Edelenbos, and C. Go, "Investigation on the use of illuminated clay in automotive styling," in *6th International Conference on Computer-Aided Industrial Design and Conceptual Design (CAID&CD), Delft, NETHERLANDS*, 2005, pp. 514–519.

[6] H. Song, F. Guimbretière, C. Hu, and H. Lipson, "Modelcraft: capturing freehand annotations and edits on physical 3d models," in *Proceedings of the 19th annual ACM symposium on User interface software and technology*. ACM, 2006, pp. 13–22.

[7] J. Sheng, R. Balakrishnan, and K. Singh, "An interface for virtual 3d sculpting via physical proxy." in *GRAPHITE*, vol. 6, 2006, pp. 213–220.

[8] Y. Shen, K. Dou, and J. Gu, "Rocumodel: an iterative tangible modeling system," in *Proceedings of the 8th International Conference on Tangible, Embedded and Embodied Interaction*. ACM, 2014, pp. 73–76.

[9] Y. Huang and M. Eisenberg, "Easigami: virtual creation by physical folding," in *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*. ACM, 2012, pp. 41–48.

[10] M. Anabuki and H. Ishii, "Ar-jig: a handheld tangible user interface for modification of 3d digital form via 2d physical curve," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 2007, pp. 55–66.

[11] A. Wibowo, D. Sakamoto, J. Mitani, and T. Igarashi, "Dressup: a 3d interface for clothing design with a physical mannequin," in *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*. ACM, 2012, pp. 99–102.

[12] K. D. Willis, C. Xu, K.-J. Wu, G. Levin, and M. D. Gross, "Interactive fabrication: new interfaces for digital fabrication," in *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*. ACM, 2011, pp. 69–72.

[13] M. Reed, "Prototyping digital clay as an active material," in *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*. ACM, 2009, pp. 339–342.

[14] J. Huang and C.-H. Menq, “Automatic cad model reconstruction from multiple point clouds for reverse engineering,” *Journal of Computing and Information Science in Engineering*, vol. 2, no. 3, pp. 160–170, 2002.

[15] J. Wang, D. Gu, Z. Yu, C. Tan, and L. Zhou, “A framework for 3d model reconstruction in reverse engineering,” *Computers & Industrial Engineering*, vol. 63, no. 4, pp. 1189–1200, 2012.

[16] G. K. Tam, Z.-Q. Cheng, Y.-K. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X.-F. Sun, and P. L. Rosin, “Registration of 3d point clouds and meshes: a survey from rigid to nonrigid,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 19, no. 7, pp. 1199–1217, 2013.

[17] Q.-X. Huang, S. Flöry, N. Gelfand, M. Hofer, and H. Pottmann, “Reassembling fractured objects by geometric matching,” *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, pp. 569–578, 2006.

[18] B. Allen, B. Curless, and Z. Popović, “Articulated body deformation from range scan data,” in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 612–619.

[19] P. Tang and S. H. Rasheed, “Simulation for characterizing a progressive registration algorithm aligning as-built 3d point clouds against as-designed models,” in *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World*. IEEE Press, 2013, pp. 3169–3180.

[20] S. Kahn, U. Bockholt, A. Kuijper, and D. W. Fellner, “Towards precise real-time 3d difference detection for industrial applications,” *Computers in Industry*, vol. 64, no. 9, pp. 1115–1128, 2013.

[21] R. Qin and A. Gruen, “3d change detection at street level using mobile laser scanning point clouds and terrestrial images,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 90, pp. 23–35, 2014.

[22] D. Aiger, N. J. Mitra, and D. Cohen-Or, “4-points congruent sets for robust pairwise surface registration,” in *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3. ACM, 2008, p. 85.

[23] Y.-S. Liu and K. Ramani, “Robust principal axes determination for point-based shapes using least median of squares,” *Computer-Aided Design*, vol. 41, no. 4, pp. 293–305, 2009.

[24] C.-S. Chen, Y.-P. Hung, and J.-B. Cheng, “Ransac-based darces: A new approach to fast automatic registration of partially overlapping range images,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 11, pp. 1229–1234, 1999.

[25] A. E. Johnson and M. Hebert, “Using spin images for efficient object recognition in cluttered 3d scenes,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 5, pp. 433–449, 1999.

[26] R. Gal and D. Cohen-Or, “Salient geometric features for partial shape matching and similarity,” *ACM Transactions on Graphics (TOG)*, vol. 25, no. 1, pp. 130–150, 2006.

[27] H. Pottmann, J. Wallner, Q.-X. Huang, and Y.-L. Yang, “Integral invariants for robust geometry processing,” *Computer Aided Geometric Design*, vol. 26, no. 1, pp. 37–60, 2009.

[28] X. Li and I. Guskov, “Multiscale features for approximate alignment of point-based surfaces.” in *Symposium on geometry processing*, vol. 2. Citeseer, 2005.

[29] J. Digne, J.-M. Morel, N. Audfray, and C. Mehdi-Souzani, “The level set tree on meshes,” in *Proc. 3DPVT*, vol. 2, 2010.

[30] J. Yang, H. Li, and Y. Jia, “Go-icp: solving 3d registration efficiently and globally optimally,” in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1457–1464.

[31] P. J. Besl and N. D. McKay, “Method for registration of 3-d shapes,” in *Robotics-DL tentative*. International Society for Optics and Photonics, 1992, pp. 586–606.

[32] S. Rusinkiewicz and M. Levoy, “Efficient variants of the icp algorithm,” in *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. IEEE, 2001, pp. 145–152.

[33] W. Chang and M. Zwicker, “Global registration of dynamic range scans for articulated model reconstruction,” *ACM Transactions on Graphics (TOG)*, vol. 30, no. 3, p. 26, 2011.

[34] M. Wand, B. Adams, M. Ovsjanikov, A. Berner, M. Bokeloh, P. Jenke, L. Guibas, H.-P. Seidel, and A. Schilling, “Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data,” *ACM Transactions on Graphics (TOG)*, vol. 28, no. 2, p. 15, 2009.

[35] V. Jain and H. Zhang, “Robust 3d shape correspondence in the spectral domain,” in *Shape Modeling and Applications, 2006. SMI 2006. IEEE International Conference on*. IEEE, 2006, pp. 19–19.

[36] M. M. Bronstein and I. Kokkinos, “Scale-invariant heat kernel signatures for non-rigid shape recognition,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1704–1711.

[37] K. S. Arun, T. S. Huang, and S. D. Blostein, “Least-squares fitting of two 3-d point sets,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 5, pp. 698–700, 1987.

[38] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, “The ball-pivoting algorithm for surface reconstruction,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 5, no. 4, pp. 349–359, 1999.