An Evaluation of Interactive and Automated Next Best View Methods in 3D Scanning

Grigori D. Pintilie[1] and Wolfgang Stuerzlinger[2]

[1]York University, Toronto, Canada, gregdp@csail.mit.edu
[2]York University, Toronto, Canada, wolfgang@cse.yorku.ca

ABSTRACT

Scanning of objects to produce 3D models is becoming more commonplace as the required hardware is becoming less expensive and more widely attainable. The process involves obtaining scans of multiple views so as to create a complete 3D model. This is typically a user-driven process, and an analysis of the difficulty of this task, use of automation, visualization methods, and their effect on the final result has not yet been thoroughly investigated. In this paper we report on a user study related to 3D scanning in which the user is asked to use a simulated, somewhat simplified 3D scanner with a simple user interface. Our investigation focuses on scanning the complete surface of an object with decent sampling density, but does not take all sampling issues, such as reflections, into account. We evaluate different visualization methods, which aim to help the user complete or improve scans, and compare the results obtained by participants to those obtained using an automated approach. The results show that users can easily obtain complete scans or improve existing scans using this simple interface, and that different visualization methods are more or less equally effective; moreover, user performance is on par with automated scanning methods.

Keywords: human factors, humanizing engineering design, reverse engineering.
DOI: 10.3722/cadaps.2013.279-291

1 INTRODUCTION

Scanning of objects to produce 3D models has become popular in fields such as art preservation, design, virtual reality, and many others. The process is primarily driven by the acquisition of 3D points that lie on the surface of an object. Common methods use lasers, structured light, or stereo-vision. Since from any one given view only part of the object is visible, a single scan can only produce

an incomplete 3D model. Thus the object or scanner has to be moved and rescanned from different views in order to obtain a complete model.

When multiple scans are acquired, they are typically aligned, or registered, so that the data from all the scans can be combined into a single model. Automated methods such as iterative closest point registration can be used [3]. However, user-guidance is also typically needed, since the point sets from different scans usually have only a subset of points in common, and hence automatic alignment is not always reliable. Alternatively, a robotic system can be used to move the scanner or the object, so that registration can be initialized using the known relative motion between the two [4]. Here we focus on such a scenario, in which the user task becomes to plan new views to be scanned in order to further complete or improve the scanned model. We envision that this can be done completely through a graphical user interface. The user changes the view within the interface, and initiates the acquisition of new scans from multiple views in order to completely scan the object. This paper focuses only on scanning each part of the object at least once, i.e., sampling each surface with adequate density, but ignores other issues such as unreliable data due to oblique scan angles or reflections.

While the technical aspects of 3D scanning are fairly well understood, such a user interface has not yet been investigated in a controlled study. It is thus still unclear how difficult or easy this task is, in particular with respect to obtaining a complete and accurate 3D model. This paper explores the task of scanning as both a user-driven process and also an automated process. To facilitate evaluation, we use a simulated scanning process, in which a single scan can be performed much faster than with typical real scanners. Alignment and measurement errors are also avoided, and the scanned model can be compared to the virtual model being scanned to measure performance. This enables us to analyze the effect of different visualization methods on the task of obtaining a complete and accurate 3D model, and also to compare user-driven process with an automated scanning procedure.

1.1 Related Work

Methods for scanning objects are well documented in the literature [9]. Furthermore, automated methods for scanning have also been proposed: two main approaches are to either exhaustively scan all views, or to automatically compute good new views. In the first approach, an exhaustive scan is attempted by moving the object, for example using a robotic platform. The exhaustiveness of the search depends strongly on the number of degrees of freedom allowed. For example, most platforms simply rotate about a single axis, which is not sufficient for most objects. This exhaustive approach can be unnecessarily slow, and may produce redundant data.

Another approach for automated scanning is to compute what the next view should be in order to minimize redundancy, or so as to uncover as much of the un-scanned object as possible. This problem has been referred to as the *next best view* problem [10][11]. Such methods can also be considered to perform a form of view planning, with the aim being to reduce the total number of scans [14]. Such approaches commonly maintain a partition of space dividing what has been seen and what has not, and the next view is planned so as to uncover as much of the unseen space as possible.

Reconstruction of surfaces from point sets, which is an important aspect of producing 3D models from scanning, has also been extensively researched. Methods include the use of Voronoi methods [2] and point set surfaces [1]. Such methods typically perform poorly when points are noisy or sparse. For rendering purposes alone, splatting can be used instead [13]. We use an octree to quickly store and combine multiple scans, and a simplified splatting-based method for real-time interactive display.

In this paper we build on prior scanning-related work by exploring whether the task of acquiring a complete scan or improving an existing scan is difficult from a novice-user perspective. As such, we conducted a user study, with participants that are not too familiar with 3D modeling or scanning tools

and limited the complexity by using a somewhat simplified, simulated scanning environment. In particular, we focus on the task of scanning the complete surface of an object with decent sampling density, but ignore some sampling issues, such as reflections. We explore whether different visualization methods can aid users in the scanning process, and how an automated approach based on view planning methods performs compares to user-directed scanning.

2 METHODS

2.1 Reconstruction of Models from Simulated 3D Scans

We simulate the process of scanning using existing 3D models of objects. Rays cast from the virtual scanner are intersected with the geometry to produce scan points (Figure 1). Intersection points are stored as 3D points or *scan points*. At each scan point, a surface normal is computed, through an eigen decomposition of the covariance matrix built from other nearby scan points. A check against the ray direction is performed, and the normal is flipped if it points away from the scanner (such a point would not be visible to the scanner when scanning opaque objects, an assumption we make here).

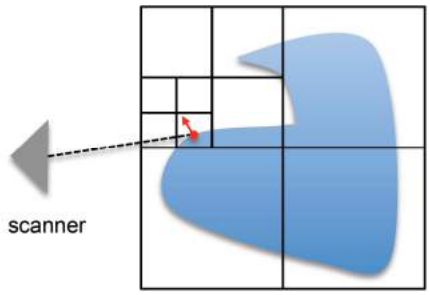


Fig. 1: Simulation of a scanner using ray-casting. The cast ray (dashed line) is intersected with the object (blue solid). The point where it hits the object is added as a point into an octree (here shown as a quadtree in 2D), which has a pre-set maximum depth. The normal for a scan point is computed from neighboring points.

2.1.1 Storage and combination of scan points

Scan points are stored in an octree, which has a maximum pre-specified depth (here we use 7). Once the octree is initialized to enclose the object being scanned, it is sub-divided dynamically to the given depth during the scanning procedure, and new scan points are added to cells at this level. After each scan is complete, the points and normals in each cell are averaged, to create a single average point and normal. This scheme acts to reduce the number of scan points through local averaging in the neighborhood of the cell. The depth of the octree thus dictates the overall sampling density desired: the deeper the octree, the higher the resulting sampling density. One advantage of using such a method is that it becomes very easy and efficient to combine points from different scans. Due to averaging, noise and spurious points can be easily suppressed.

However, such averaging does not preserve sharp features. To preserve sharp features, samples within each cell can be clustered and split into multiple representative points and normals, e.g. similar to [6]. However, this can be a time-intensive process when considering the hundreds of thousands of samples that can result from even modest-resolution scans. Hence we did not perform feature preservation for this user study, as system responsiveness and high speed of the scanning were considered more important. This speed advantage also allows us to test multiple conditions per participant. To minimize the effect of artifacts in the reconstruction due to averaging, we focus here

only on relatively smooth objects with few sharp features. Since the average point and normal computed in each cell will be used to draw the surface using splats, we will henceforth refer to these points as *splat points*.

2.1.2 Rendering scanned objects

To render the scanned object, the octree is traversed, and a disc is drawn for each cell containing one or more scan points, with its center at the average position, and aligned so that its central axis points in the direction of the normal. This rendering technique is commonly known as splatting [13]. To maintain a high rendering frame rate and thus interactivity, we do not correct for perspective or use alpha blending to correct for jaggedness near sharp features, which would require multiple passes. To minimize the effects of such inaccuracies in the rendering of scanned objects, we focus here mostly on the use of smooth objects with minimal sharp features. An example splatted surface is shown in Figure 4.

2.1.3 Computation of unseen areas

During the scanning process, the cast rays are intersected with the octree. Rays that hit the object terminate on the intersection point, while rays that do not terminate at infinity. All rays are also represented as frusta that intersect the edges of the pixel through which the ray was cast. These frusta are intersected with the octree, subdividing to the same level in which the scan points where added. Leaf nodes are all initialized as unseen. During scanning, leaf nodes that are found to intersect a ray frustum are changed to seen. After each scan, leaf nodes that are empty and belong to the same parent node are merged, so as to conserve memory and reduce further computation time. When intersecting a ray frustum with the octree, the intersection proceeds from the highest node towards the leaf nodes. Only nodes that intersect a frustum are further explored for that frustum, and only if the node is not yet labeled as seen. When the entire object is visible in the entire view frustum, the entire view frustum is also intersected with the octree, and leaf nodes outside the view frustum are labeled as seen. This is so that corners of the octree outside the view frustum, which do not contain any part of the object, are not shown as unseed areas. An example of the unseen area after a single scan is shown in Figure 4.

2.1.4 Scan score

Since we are using a simulated scanning environment, we can compute a scan score, which is proportional to how much of the virtual object has been scanned. For this a set of points is first randomly placed on the geometric surface of the object, and a relaxation procedure moves them to distribute them as evenly as possible. The relaxation (which is only done once in pre-computation) finds all other points near to a given point, and pushes those points away with a force proportional to their distance (a spring-like effect). The number of points distributed on the surface is chosen so that a given density of points will be reached on all parts of the surface once the relaxation process converges. The scan score is computed by finding all the points on the surface within a small distance of each splat point in the octree. The points on the surface that meet this criterion are flagged as scanned, and all other points are flagged as unscanned. The scan score is then simply the ratio of the two, i.e. $Nscanned/Nunscanned$. For real objects with non-zero surface $Nunscanned$ will always be larger than zero, and thus the scan score will range from 0 for a completely unscanned object to 1 for a completely scanned object.

2.1.5 Computing densities of splat points

Local densities of scan points are a good indicator of the quality of the scanned model [12]. We use splat points to estimate density rather than the actual scan points. This is for two main reasons: 1) there are fewer splat points and hence the computation is much faster, and 2) the desired density of

the scan (directed by the depth of the octree) is reflected by considering splat points rather than the actual scan points. Hence the highest density at a splat points will reflect this desired density of the scan, and not the highest density of actual scan points acquired. The density at each splat point $\vec{p}$ is estimated using a Gaussian kernel:

$$density(\vec{p}) = \frac{1}{N}\sum_{\vec{r}} \exp\left(\frac{\|\vec{r}-\vec{p}\|^2}{2\sigma^2}\right) \tag{2.1}$$

In the above equation, $\vec{r}$ is a splat point within a distance D of $\vec{p}$, and σ is a scale factor. Here we take the scale factor to be 5 times the size of a leaf octree cell. The distance D is 4 times this value, so that the contribution to the density of points further away than this distance becomes insignificant. N is the number of points within distance D.

The density value can be visualized directly on the surface of the reconstructed model; an example is shown in Figure 4. The highest and lowest densities are stored during the computation of all density values at each splat point. The color of a splat is computed from its density value using a linear scale, such that the color of the splat with the lowest density is red, and the color of the splat with the highest density is blue. When drawing a view to compute next good views as described below, the color is instead a grey value ranging from 0 for the lowest density to 1 for the highest density.

2.1.6 Searching for good next views

After each scan, using the model acquired so far, an automated approach is used to find a next view, which would further complete or improve the scanned model. To find such views, the reconstructed model is rotated exhaustively through two degrees of freedom, and a view score is computed in each orientation. Unseen areas are rendered as a black box enclosing each unseen cell of the octree. Lighting is turned off, so that pixels in unseen areas have a color of 0. Splats are colored using a grey value ranging from 0 (lowest density) to 1 (highest density), also without lighting calculations. Thus when more unseen areas or areas with low density are visible from the viewpoint, the resulting view score is higher. The view score is computed as follows:

$$view\_score = \frac{1}{N}\sum_{pixel=1}^{N_{Pixels}} 1 - C_{pixel} \tag{2.2}$$

Above, C_{pixel} is the color of the pixel, and ranges from 0 to 1. N_{pixel} is the total number of pixels with colors that are not 1. The background is colored white (pixel values of 1), so this normalizes the view score to the total non-background surface area visible in the view.

After the rotation part of the search, the top 20 view scores and corresponding views are further improved by Monte Carlo search. The position and rotation are randomly changed, and the move is accepted with a probability proportional to the increase in the view score.

Such automated approaches have been posed to try to find a next best view [10][11]. However, without knowing the complete 3D model beforehand, attempting to find the best view does not seem feasible. Even when the geometry is known beforehand, a next best view could potentially be searched for, although even this goal is NP-complete through equivalence to art gallery problem [8][16]. For this reason we here adopt the term next good view for views found by the automated search procedure.

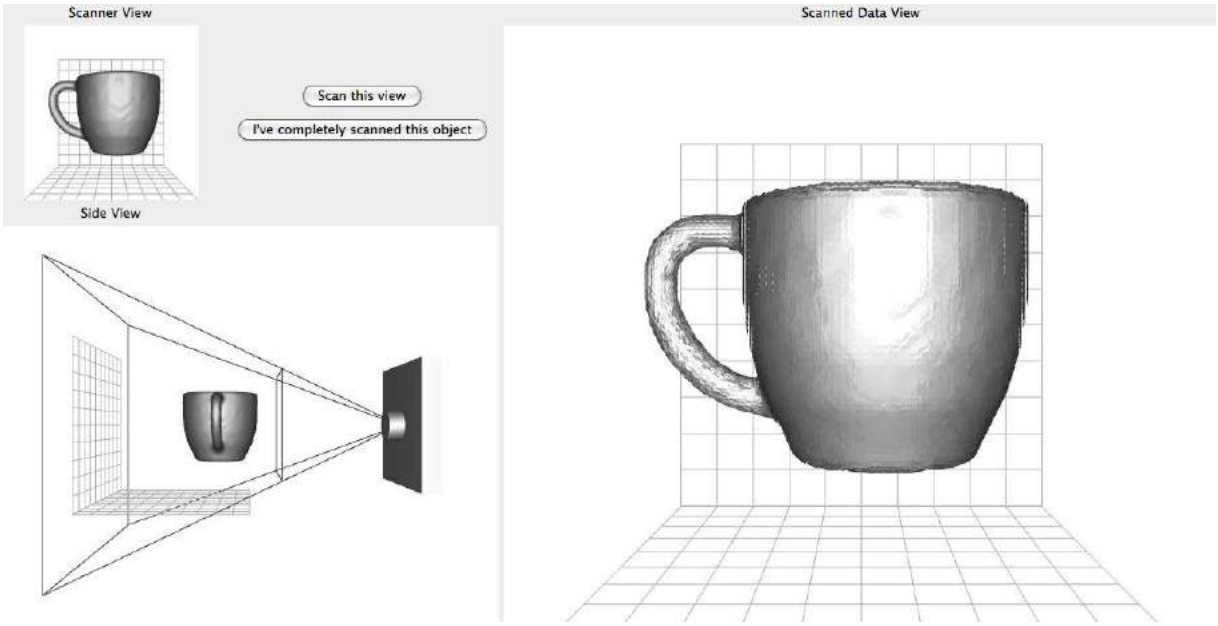


Fig. 2: The scanning user interface. Three views are shown, including the scanner view, which is what the scanner sees, a side view showing a simple representation of the scanner, its view frustum, and the object placed in front of it, and a scanned data view where the splatted surface is shown.

2.2 The Scanning User Interface

The user interface used for the simulated scanning system is shown in Figure 2. Three views are shown: a scanner view, a side view, and view of the scanned data. *Scanner view*: in this view, the object is rendered using a solid, grey surface. This is a representation of what the scanner sees of the object. *Side view*: in this view, the scanner is drawn using a simple representation (a box and a cylinder representing the lens), and the visible frustum is also drawn using straight lines. This view helps the user to visualize the object position relative to the scanner. *Scanned data view*: In this view, the current 3D model of the scanned object is drawn, using the splatting technique previously described.

The user interacts with the scanned data view using the mouse. Since a first scan of the object is always performed at the start, a surface is always visible in this view. The object can be rotated with the left mouse button, moved side to side with the right mouse button, or moved back and forward using the middle mouse button or scroll wheel. The visual feedback consists of a sphere and 3 coordinate axes for rotation, which follows the Arcball method [15], and arrows for the other to operations (Figure 3).

2.3 Visualization Methods for Acquisition of Complete Models (Part A)

In the first part (part A) of the user study, we sought to evaluate how different visualization methods compare in helping the user obtain complete 3D models of an object. A first scan was automatically made with the object centered in the middle of the view frustum, in an initial orientation, which was the same for each participant. The user was then asked to change the view and perform further scans. To do a scan, they were asked to press the *Scan this view* button on the user interface (Figure 2). When they felt they completely scanned the object, they were asked to press the **ò***I've completely scanned this object"* button. They were instructed to try to do as few scans as possible

in order to completely scan the object. We tested 4 different visualization methods in part A, as described below and illustrated in Figure 4.

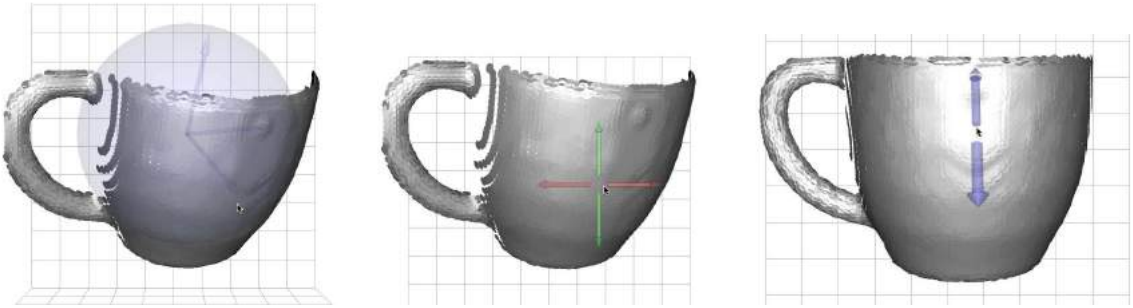


Fig. 3: The user can rotate and move the object in the scanned data view using the mouse. The left mouse button rotates the view (left). The right mouse button moves the object left, right, up and down (middle), and the middle mouse button or wheel moves it forward and back. When the object is rotated or moved in the scanned data view, the transform of the object is also propagated to the scanner view and the side view.

After they were done scanning all the objects, the participants were asked to give each visualization a rank reflecting how easy/difficult it was to scan the object using that visualization. They were also asked to give free-form comments about how the visualization methods compared to each other.

The *grey surface* display simply draws the surface of the scanned 3D model using grey splats. The splats are sized in proportion to the diagonal distance across a leaf cell in the octree, so that the surface appears solid in areas where adjacent cells contain scan points. Each splat was coloured using the same grey color, and a single-light source was used to light the scene. Rendering was done with openGL. The *coloured surface* display draws splats as in the grey surface display, however the splats are coloured to represent the local density of scan points around each splat. The colour varies from blue to red, with blue corresponding to the highest density found amongst the splats, and red corresponding to the lowest density. In the *unseen areas* display, the areas not yet seen by any scan is drawn using black cubes. The octree is traversed starting with the top node, and when a node that is unseen is reached, a black cube is drawn spanning the dimensions of that node. In the *suggested view arrows* display, next good views are computed as described above, and the top 5 views are drawn using green arrows pointing in the direction of the view. When the user moves the mouse pointer over the arrow, it is selected, and this is indicated by it turning red. The user can then click on it, in which case the view is moved to this position through gradual interpolation.

2.4 Visualization methods for improving scan quality (part B)

In the second part of the study (part B), we sought to evaluate how different visualization methods influence the improvement of an existing scan. An initial scan of the object is created automatically by taking surface points evenly distributed over the surface, adding them to the octree as scan points, and computing splat points as before. The depth of the octree was chosen to be one level lower than the level at which new scan points taken by the user were inserted (6). Thus, as the user performed more scans, the quality of the surface could be seen to improve, mainly as a result of less averaging of scan points, and the fact that splats become smaller. When computing a completion score, only splat points at the deeper level (7) are used, so that splats from the higher level which are not at the desired scan density do not contribute to the completion score.

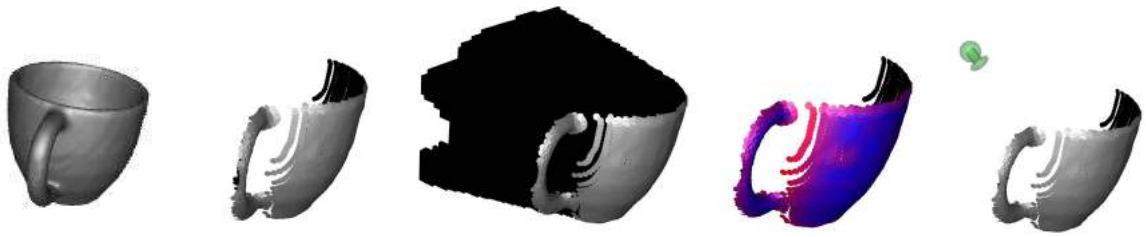


Fig. 4: The object to be scanned, a mug, is shown on the left. The remaining 4 images, from left to right, show the different visualization methods: grey surface, unseen areas, coloured surface, and suggested view arrows.

Four different visual cues were used, as described below. In each trial, an initial scan improving the quality was performed automatically for an object in an initial orientation (again, same for each object across participants). The participants were asked to change the view and to perform further scans, as before, until the quality was improved everywhere on the object. Four different visualization methods were tested, as described below and illustrated in Figure 5.

The *grey surface* again consisted of only splats drawn at splat points. In lower quality scans, the splats are larger, since they are sized proportional to the size of the larger cells at one level higher in the octree. As new scans are made, the splats are computed at one level deeper, and hence become smaller. The *coloured surface* was again drawn with a gradual change between blue for splats at lower density, to red for splats at higher density. Since the splats in the starting scan are at a higher level in the octree, they also have lower densities than splats one level deeper. In the *surface with holes* display, the splats at both levels involved are drawn at with a size proportional to the deepest level of the octree. Hence the splats at one level higher, since they are now smaller, tend to produce an incomplete surface, even when splats exist in adjacent cells. The splats in the deeper level form a more closed surface, since their size more closely matches the size of the cells at the deeper level. *Suggested view arrows* were again drawn, using views computed with the automated approach. In this computation, the views found tend to be focused on parts of the surface scanned at lower quality, since the density there is lower.

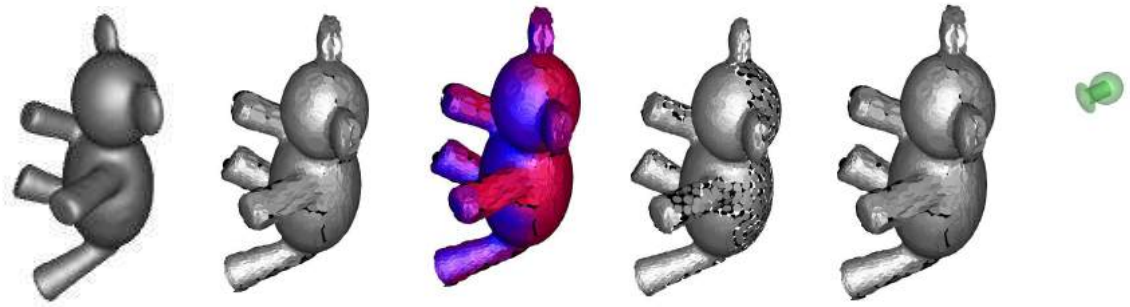


Fig. 5: Visualizations used for improving the quality of an initial scan. The object to be scanned is shown on the left. The remaining four visualizations, from left to right, are: grey surface, colored surface, surface with holes, and grey surface with suggested view arrows.

2.5 User Study

For each part of the study, all four different visualizations are tested. To avoid learning effects, the order of the four visualization methods was presented based on a 4x4 Latin square [7]. We recruited 8 participants and each row of the Latin square was used twice. The participants were between 19 and 35 years of age, with half male and half female. None of the participants declared themselves to be color-blind. Only one had experience with 3D graphics, 4 of them declared spending less than 1 hour a week playing games, and the rest declared 3-6 hours per week of game playing.

The objects to be scanned were obtained from the mesh segmentation benchmark [5]. The meshes are all watertight and the selected ones did not have significant sharp features. In a pilot study, we realized that due to differences between these objects, direct comparison of scan scores for different conditions and objects could be difficult. For example, the scan scores computed even at the first scan can be different for each object. While one could use only a single object, one can expect significant learning effects between conditions in this case. To address this issue, we used different objects for each visualization per participant, so that averaging across objects allows relative comparisons.

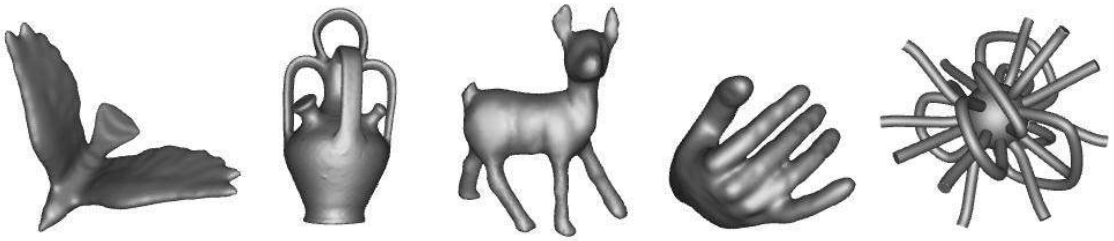


Fig. 6: The objects used in the user study. The first four from the left are the normal objects while the last on the right is the randomly occluding spherical object (ROSO).

To further alleviate issues due to potential differences in objects, we also created a more complex randomly occluding spherical object (ROSO), also shown in figure 6. Due to the visual complexity of this object, we don't expect learning to have a large effect between visualizations. The ROSO is designed to be spherical, such that a similar surface area is visible from any viewpoint. It was randomly generated by extruding tubes from a sphere, with either end of a tube at a certain radius away from the sphere, or looped back to contact the sphere in a different location. We used the same ROSO with each visualization in both part A and part B.

Half the participants were asked to do part A first, and half did part B first, also to counter learning effects. In each part, the four normal objects were scanned first, using the same visualization throughout the scanning of each object. The order of the displays was selected based on a 4x4 Latin square. The order of the objects was chosen so that across the 8 subjects, each visualization was used with each object (thus allowing averaging of scan scores for each visualization across all 4 objects). Once the 4 normal objects were scanned, the participant scanned the ROSO 4 times. Each time, a different visualization was used, again with the order picked from the 4x4 latin square.

2.6 Automated Scanning

We also performed automated scanning of all the objects as a comparison point for the user guided scanning process. This method initially scans the object with the same view used in the study for each object, computes the next good views and uses the view with the highest view score for the next scan. This process is terminated when the scan score becomes better than 0.99. The automated process was run on each of the normal objects as well as for the ROSO object, terminating in each case. This

process and the user study were run on a laptop with 15ð screen, and a 2.8 GHz Intel core 2 duo processor. The average time per scan is around 7s. The automated procedure usually terminates within 6-10 scans, thus taking roughly 1min in total.

3 RESULTS AND DISCUSSION

3.1 Scan Scores after Each Scan and Participant Rankings

The average completion scores after each scan are plotted in Figure 7. In both parts, all 8 participants performed at least 5 scans before pressing the *I've completely scanned this object* button when scanning normal objects, and all 8 participants performed at least 7 scans when scanning ROSOs. They all achieved a scan score of at least .98.

For normal objects, the standard deviations were quite high, as expected, due to the substantial differences between the objects used. Still, it can be seen that amongst the participant scores, the colored surfaces are on average slightly higher than for other displays. This resonates well with user comments: several participants commented that the colored display was the easiest to use. It is also consistent with the ratings of each display (Figure 8(a)), in which the colored surface was ranked slightly above on average than the other displays.

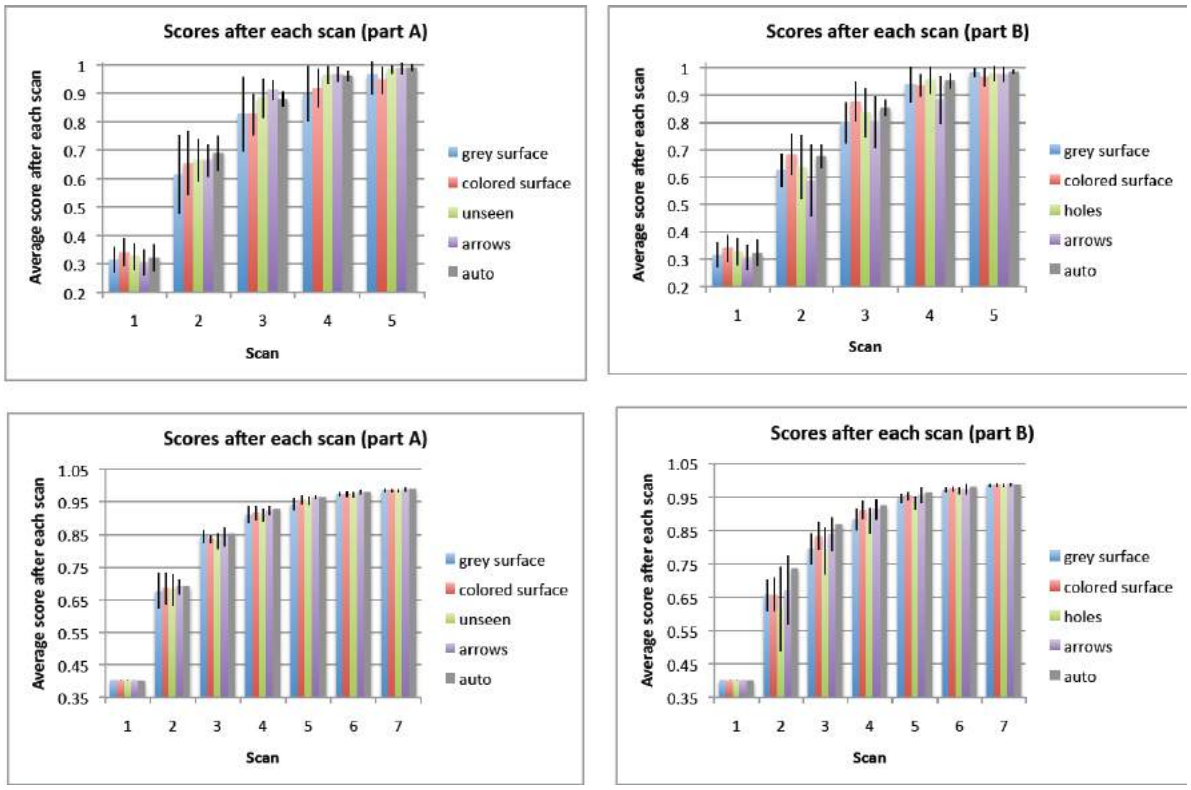
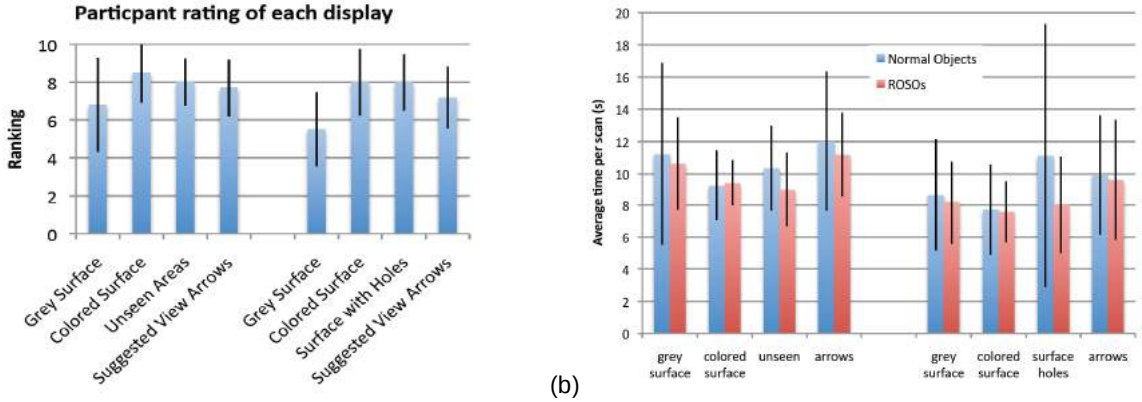


Fig. 7: Average scan scores after each scan are plotted for normal objects (top) and for ROSOs (bottom), for both parts A and B of the study. Standard deviations are shown with black lines on each bar.

The scan scores for the ROSOs have significantly smaller standard deviations. However, just as with scores for normal objects, there is no significant difference in scan scores for different visualizations. This seems to indicate that all visualizations are adequately effective, and it is not hard for participants to decide on a good next scan view using any of them. The standard deviations were higher for scores in part B compared to scores in part A. This may be because it can be a bit harder to determine a good next scan to improve the surface quality. However, again, the differences were not statistically significant, hence all visualization methods were adequately effective in this task as well.

In participant rankings of each display, the grey surface display was ranked the lowest on average. Several comments were that the unseen area was also very useful and intuitive. Several comments were that suggested view arrows were also useful, though their ranking was not the highest. Several comments were made that they could be confusing at times, and increasingly so as more scans were done, when there tended to be more arrows shown. Logs shown that every participant used an arrow at least once (i.e. clicked on it); several participants used them on every scan. Overall they were ranked higher on average than the grey surface alone, showing that they can be a useful feature to have.

The scan scores obtained by automated scanning was as high if not higher than scan scores obtained by users, indicating that automated scanning can perform as well as humans.



(a) (b)

Fig. 8: (a) Average ranking of each visualization method. Participants were asked to rank each method on how hard (0) or easy (10) it was to use. Standard deviations are shown with black lines on each bar. (b) Average time per scan for all the visualizations, when scanning normal objects and ROSOs. Standard deviations are shown with black lines on each bar.

3.2 Average User Times Per Scan

The average times per scan are plotted for each display in Figure 8(b). The plot shows that the average times per scan are on average the smallest for the colored surface displays. This is consistent with the rankings of the participants, in which this display had higher rankings, since less time was used in order to find a good scan view. The surface with holes display seems to lead to higher average times per scan in part B, and hence may be somewhat confusing to participants. In both parts, the suggested view arrows seem to lead to higher scan times on average. An explanation for this effect is that participants spent extra time on exploring what the arrows are pointing to before deciding where to scan.

4 CONCLUSIONS AND FUTURE WORK

We have presented a user interface for 3D scanning of objects. By simulating the scanning process, we were able to evaluate how users perform in the task, with the goal being to either completely scan an object, or to improve the quality of an existing scan. Four different visualization methods were tested in both scenarios. The performance was not significantly different for the methods, signifying that they were all approximately equally effective. However users overall preferred the displays where the completion (or density) of an existing model is indicated by colouring.

We showed that an automated scanning approach is potentially as good in obtaining a complete scan or improving a scan as human users. On the other hand, we also saw that even users not overly familiar with 3D graphics can quickly learn to use a simple 3D interface to obtain or improve complete 3D models. This means that the task of 3D scanning is not that difficult from the user-perspective when a simple and intuitive user interface such as the one presented here is used. These results should be useful in further development of effective systems and user interfaces for 3D scanning using a real scanner and robotic platform for moving the object or the scanner. We will also investigate in the future how we can assist users in the case of unreliable data due to reflections or other scan artifacts. Furthermore, we will look at how we can directly visualize the effect of scan coverage and quality on later processing steps (such as simulations) during the acquisition process.

REFERENCES

[1] Alexa, M.; Behr, J.; Cohen-Or, D.; Fleishman, S.; Levin, D.; Silva C.T.: Point set surfaces, IEEE Visualization, 2001, 21–28.

[2] Amenta, N.; Bern, M.; Kamvysselis, M.: A new Voronoi-based surface reconstruction algorithm, ACM SIGGRAPH, 1998, 415–421.

[3] Besl, P.; McKay, H.: A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2), 1992, 239–256. DOI: 10.1109/34.121791

[4] Callieri, M.; Fasano, A.; Impoco, G.; Cignoni, P.; Scopigno, R.; Parrini, G.; Biagini, G.: RoboScan: an automatic system for accurate and unattended 3D scanning, 3D Data Processing, Visualization and Transmission, 2004, 805–812.

[5] Chen, X.; Golovinskiy, A.; Funkhouser, T.: A benchmark for 3D mesh segmentation, ACM SIGGRAPH, 2009, article 73.

[6] Fleishman, S.; Cohen-Or, D.; Silva, C.T.: Robust moving least-squares fitting with sharp features, ACM SIGGRAPH, 2005, 544-552.

[7] Hinkelmann, K.; Kempthorne, O.: Design and Analysis of Experiments: Introduction to experimental design, Wiley-Interscience, 2007.

[8] Lee, D.; Lin, A.: Computational complexity of art gallery problems, IEEE Transactions on Information Theory, 32(2), 1986, 276–282. DOI: 10.1109/TIT.1986.1057165

[9] Levoy, M.; Ginsberg, J.; Shade, J.; Fulk, D.; Pulli, K.; Curless, B.; Rusinkiewicz, S.; Koller, D.; Pereira, L.; Ginzton, M.; Anderson, S.; Davis, J.: The Digital Michelangelo project, ACM SIGGRAPH, 2000, 131–144.

[10] Low, K.-L.: An adaptive hierarchical Next-Best-View algorithm for 3D reconstruction of indoor scenes, Pacific Conference On Computer Graphics and Applications, 2006, 1-10.

[11] Massios, N. A.; Fisher, R. B.: A best next view selection algorithm incorporating a quality criterion, British Machine Vision Conference, 1998, 780-789.

[12] Pauly, M.; Mitra, N. J.; Guibas, L.: Uncertainty and variability in point cloud surface data, Symposium on Point-Based Graphics, 2004, 77–84.

[13] Ren, L.; Pfister, H.; Zwicker, M.: Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering, Computer Graphics Forum, 21, 2002, 461–470. DOI: 10.1111/1467-8659.00606
[14] Scott, W. R.; Roth, G.; Rivest, J.: View planning for automated three-dimensional object reconstruction and inspection, ACM Computing Surveys, 35(1), 2003, 64–96. DOI: 10.1145/641865.641868
[15] Shoemake, K.: Arcball: a user interface for specifying three-dimensional orientation using a mouse, Graphics Interface, 1992, 151–156.
[16] Stuerzlinger, W.: Imaging all Visible Surfaces, Graphics Interface 1999, 115-122.