Direct Drawing on 3D Shapes with Automated Camera Control

Michaël Ortega[1], **Thomas Vincent**[2]
[2]UJF-Grenoble 1 / UPMF-Grenoble 2 / Grenoble-INP / [1]CNRS / INRIA
[1,2]Laboratoire d'Informatique de Grenoble, UMR 5217 F-38041, Grenoble, France
firstname.lastname@imag.fr

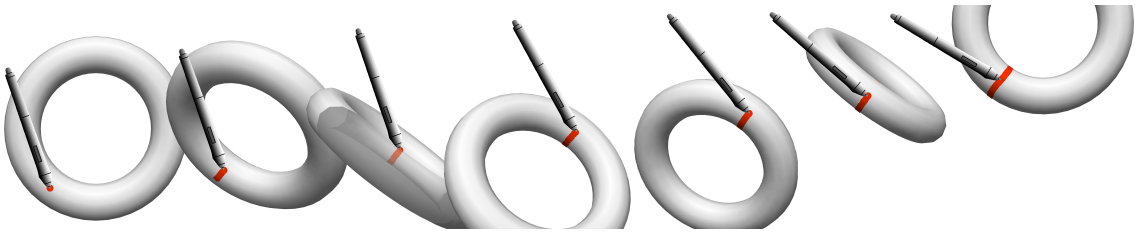

Figure 1. With Automated Camera Control for Drawing (*ACCD*), the small radius of a torus can be drawn in a single continuous curve.

ABSTRACT
We present ACCD, an interaction technique that allows direct drawing of long curves on 3D shapes with a tablet display over both multiple depth layers and multiple viewpoints. ACCD reduces the number of explicit viewpoint manipulations by combining self-occlusion management and automated camera control. As such it enables drawing on occluded faces but also around a 3D shape while keeping a constant drawing precision. Our experimental results indicates the efficacy of ACCD over conventional techniques.

Author Keywords
3D interaction technique; 3D painting; camera controls.

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation: User Interfaces - Interaction styles, Input devices and strategies.

INTRODUCTION
Drawing on 3D shapes is a generic elementary action for performing various tasks: 3D models painting [5], 3D sculpture [14], sketch-based 3D objects modelling [10], mesh segmentation [3], specification and control of navigation in 3D environments [6]. In WYSIWYG interfaces, drawing consists of mapping 2D input strokes into curves on the 3D shape.

The status-quo configuration uses a tablet display for direct drawing, and a standard projection (pinhole camera metaphor) for a "realistic" representation of the shape [1]. Drawing is performed with stylus strokes on the tablet display, while viewpoint manipulation is done with separated tools (rotation, pan and zoom). However, this configuration requires the users to frequently switch between drawing and viewpoint manipulation.

Indeed, some parts of the 3D shape might not be reachable from a single viewpoint: (1) faces occluded by the object itself (i.e., ***self-occlusion***); and (2) ***back faces***. Viewpoint manipulations are required to access such faces. Moreover, the ***drawing precision*** depends on (1) the distance from the viewpoint to the shape, and on (2) the angle between the line of sight (viewpoint direction) and the normal of the surface [12]. Controlling this precision during drawing thus requires even more viewpoint manipulations. This number of viewpoint manipulation can be very high for complex 3D shapes, leading to tedious and time-consuming drawing task.

Furthermore, switching between viewpoint manipulation and drawing results in discontinuities at the junctions of the different sub-curves. For example, in 3D sculpture tools, like Pixologic ZBrush [14], the dig out tool is controlled by user strokes on the 3D shape. The junctions of those strokes result in visual gaps that the user must correct afterwards. Minimizing the number of strokes required to draw a single long curve is therefore key to avoiding visual gaps and hence reducing the correction step.

This paper presents *ACCD* (Automated Camera Control for Drawing), a new approach that minimizes the number of explicit viewpoint manipulations, and so the number of strokes while drawing long curves over both multiple layers and multiple viewpoints on 3D shapes. To do so, *ACCD* integrates self-occlusion management and automated camera control for accessing back faces and controlling the drawing precision.

CHI 2014, April 26–May 1, 2014, Toronto, Ontario, Canada.

http://dx.doi.org/10.1145/2556288.2557242

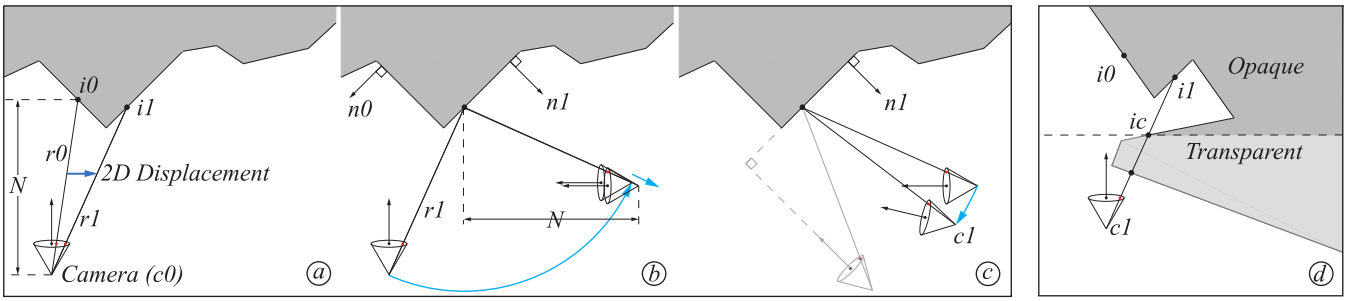

Figure 2. Computing the new position and orientation of the camera: *(a)* A new intersection *i1* is computed from the new position of the stylus. *(b)* *n0* and *n1*, the normals from the faces, give the angle of camera rotation around *i1*. The camera is rotated and translated to keep the distance to the shape (*N*) constant. *(c)* The camera rotates about 10% of the angle needed for having the line of sight orthogonal to the shape. *(d)* When self-occlusion occurs, a clipping plane is computed at *ic* (first intersection from *i1* to *c1*) and the occluding part of the shape is renderered transparently.

RELATED WORK

To address self-occlusion, Fu et al. proposed *LayerPaint*, a multi-layer approach [5]. They proposed a layer-aware drawing technique that allows to access self-occluded faces. Also, users can display a layer above another, and draw on it. However, back faces remain unreachable. Moreover, changing the layers order could be disturbing as it breaks the users' depth perception. In this paper we propose a transparency-based approach rather than a layer-based approach to self-occlusion.

To address the back-faces issue, some commercial softwares use the *UVMap* technique [7]. This technique displays all the parts of a shape by unfolding it on a single 2D plan. However, with complex shapes:
- context is lost, the shape is displayed outside its 3D scene;
- understanding the new representation requires a high cognitive load, which is even higher for untextured shapes;
- most of the 3D shapes cannot be unfolded in a single 2D one: It has to be cut in pieces, preventing drawing continuity.

According to Schmidt *et al.* [12], the drawing precision depends on both the distance between the viewpoint and the shape (*D*) and the angle between the line of sight and the shape normal (*A*). The lower are *D* and *A*, the higher is the precision. Controlling *D* and *A* with the status-quo configuration requires frequent viewpoint manipulations, which is tedious and time-consuming. Balakrishnan *et al.* [2] proposed to delegate the viewpoint manipulation to the non-dominant hand. Both drawing and viewpoint control can then be done in parallel. In our evaluation, we compare *ACCD* with this bimanual approach for direct drawing and viewpoint manipulation.

Automated camera control has been explored to assist different tasks in 3D environments [1, 4, 9, 11, 13, 15]. The most complete study has been presented by Kahn *et al.* [8] with the *Hovercam* technique. *Hovercam* proposes to ease navigation around 3D objects at constant and close proximity by integrating camera controls into a single 2D gesture. Thus it allows to access back faces while keeping *D* constant and trying to minimize *A*. However, this technique is designed for mouse-based interaction and it does not support the direct mapping of stylus inputs on the tablet display.

To sum up, existing techniques does not address self-occlusion, back-faces and drawing precision issues at once. Also, some techniques are not adapted to tablet display.

ACCD

To reduce the number of viewpoint manipulations and to keep a rather constant drawing precision, the *ACCD* approach provides self-occlusion management combined with automated camera control inspired by the *HoverCam* technique [8].

Self-Occlusion

In ACCD, the self-occlusion management uses a GPU approach that interactively changes the transparency while the viewpoint changes. No pre-computation is needed, and the principle is simple (illustrated in figure 2d). For a camera position (*c1*), ACCD looks for intersections with the 3D object along the segment [*i1*-*c1*]. The closest intersection to *i1* (*ic*) is used for separating the 3D scene into two parts. Camera frustum is changed, and the part of the scene from *ic* to *c1* is rendered in a texture. Then, the other part of the scene is rendered (adapting camera frustum too), and finally the texture is transparently superimposed.

Camera movements

Figure 2 shows an overview of the 4 main steps of the *ACCD* computation. The whole computation is done for each 2D stylus displacement on the screen. The first step consist of computing the new *intersection point* (*i1*) on the object from the 2D displacement (figure 2a). *ACCD* takes the intersection between the object and *r1* (the ray defined by the camera position and the final position of the 2D displacement). The camera is rotated around *i1* of an angle defined by *n0* and *n1*, the normals of the previous and the current intersection faces (figure 2b). Next, the camera is translated to keep the distance to the object constant, and to keep *i1* under the current stylus position. Finally, the camera is rotated once more to have its viewpoint direction orthogonal to the current face. Yet, the rotation is not done at once (figure 2c). It only takes a percentage of the complete movement (10% in our implementation), and the orthogonality is only obtained after a few 2D displacements. This allows to smooth the transitions between the 3D object faces.

Clutching on screen sides

Because of the constraint of direct manipulation, the 3D object does not stay centered on the screen during the drawing. Manual translations of the camera has to be integrated in the process of drawing, without breaking the continuity of the curve. This is done with a translation mode, controlled by a switch button. It allows the user to translate the camera position along the projection plan, while keeping the drawing cursor under the stylus. Then, it is possible to switch between drawing and translating without lifting the stylus.

EVALUATION

In this experiment we evaluated three approaches to combine viewpoint's manipulation with long curves drawing over both multiple layers and multiple viewpoints on a 3D shape. The goal of this experimentation is to evaluate whether or not the user takes advantage of ACCD to improve curve continuity. To quantify this, we mainly measured both the number of strokes (NS) and the number of viewpoint manipulations (NVM) required by each technique. With all these techniques, drawing is performed with stylus strokes on the screen of a tablet display. Yet, the viewpoint manipulation control differs:

- **One-handed** (*One-H*): Viewpoint rotations (arcball) are available as a quasi-mode triggered by a tablet display shortcut key (on the edge of the tablet display) and controlled with the stylus.
- **Bimanual** (*Biman*): Viewpoint rotations (arcball) are controlled with finger strokes on the screen, with the non-dominant hand. It is therefore possible to draw and rotate the viewpoint in parallel.
- **ACCD**: As described above. Viewpoint is automatically controlled during the drawing. For the clutching issue, viewpoint's translations are available as a quasi-mode triggered by a screen shortcut key and controlled with the stylus.

As we wished to focus on viewpoint control, all three techniques benefited from the same self-occlusion management.

Procedure, Apparatus and Participants

We asked participants to draw long strokes along a guiding path on the surface of different 3D shapes with the three techniques. Participants were standing-up and holding the stylus in their dominant hand. We decided to do so as the sitting posture can constraint participants' gestures.

For each of the three techniques, participants first freely practiced on an **infinity loop** object (figure 3a). Then they performed guided drawing tasks on three different 3D models (figure 3-[b-d]) presented in the following order:

- A **Sphere** as a simple convex shape;
- A **cut Cylinder** as a more complex shape with concavity and thus possible self-occlusion; and
- A **Trefoil Knot** with a very long guiding path.

For the Sphere and the Cylinder, participants first performed 5 training tasks with a large brush (1cm on screen) and then 5 trials with a smaller brush (0.5cm on screen). As the Trefoil Knot was much longer, participants performed only 2 trainings with the large brush, and 2 trials with the smaller one.

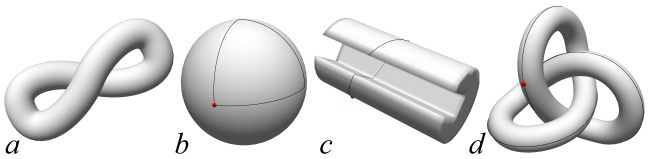

Figure 3. Models used for the evaluation: (*a*) The training "infinity loop"; *b*, *c*, and *d* were successively presented for each technique. The guiding path is the black line. Starting and ending points are reds spheres.

Participants were instructed to keep the guiding path within the brush while drawing. After each trial, participants were informed of the percentage of the guiding path that lies outside the path they drew. We asked them to try to keep this percentage around 5%. This aimed at having a consistent accuracy across participants.

Presentation ordering of the three techniques was counterbalanced across participants using a Latin-square. For each participants, we recorded 36 drawing tasks. The experiment was about one hour long.

Twelve right-handed novices (from 22 to 42 years old) participated in our experiment. We used a Wacom Cintiq 24HD Touch multi-touch and stylus sensitive screen (resolution: 1920x1200, pixel pitch: 0.27mm). We developed an OpenGL-based ad hoc application for this evaluation.

Results

The analysis of variance below were performed separately for each 3D shape on the mean aggregated by participants with participant as a random factor and *Technique* as a factor. Means were compared using Tukey's HSD tests. We discarded 5 unfinished trials (accidental application closure by participants) from the 432 recorded drawing tasks.

Performances

Average *Completion Time* (CT) were 29.8s for the *Sphere*, 20.3s for the *Cylinder* and 82.5s for the *Knot*. *Technique* had a significant effect on CT only for the *Sphere* (see table 1) where *ACCD* was found significantly faster than both *One-H* ($p<.0001$, mean difference=9.49s, SE=2.57) and *Biman* ($p<.05$, mean difference: 6.50s, SE=2.57).

3D Curves Continuity and Regularity

The *Technique* factor had a significant effect on both the *Number of strokes* and the *Number of viewpoint manipulations* (see table 1 and figure 4).

The *Number of strokes* (NS) was found significantly smaller with *ACCD* than with the two other techniques for both the *Cylinder* and the *Trefoil Knot* (all with $p<.0001$). For the *Sphere*, *ACCD*'s NS was only significantly smaller than *Biman*'s NS (with $p<.01$).

The *Number of viewpoint manipulations* (NVM) was found significantly lower with *ACCD* than with *Biman* on all 3D shapes (all with $p<.0001$). *ACCD*'s NVM is also significantly lower than *One-H*'s NVM on the *Cylinder* ($p<.0001$) and the *Knot* ($p<.05$). *One-H*'s NVM was significanlty lower than *Biman*'s NVM for the *Sphere* ($p<.05$).

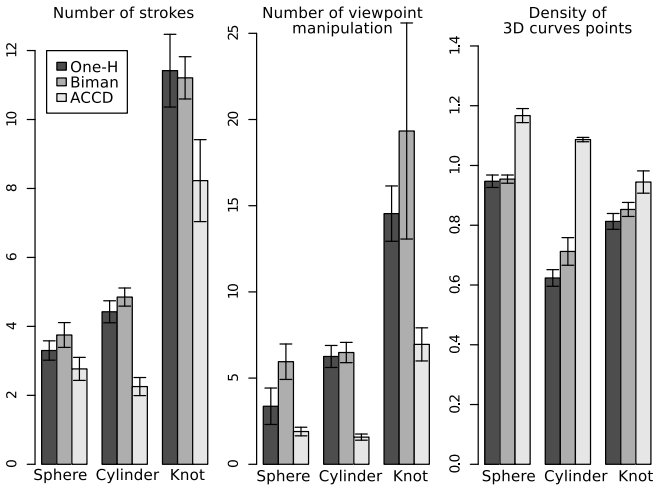

Figure 4. Means and 95% confidence intervals: (Left) Number of strokes (NS); (Middle) Number of viewpoint manipulations (NVM); and (Right) Density of input points of the 3D curves (ρ).

We also explored the effect of *Technique* on two measures of drawn curves regularity and found significant effects. First, the *Density of input points* on the 3D curve (ρ) was found significantly higher with *ACCD* than with the two other techniques for all 3D shapes (with at least p<.001). The only significant ρ difference between *One-H* and *Biman* was found for the *Cylinder* (though with p<.05). Second, the *Standard deviation of the distances between input points* along the 3D curve (σ_d) was found significantly smaller with *ACCD* than with the two other techniques for the *Sphere* and the *Cylinder* (with at least p<.01 but for the *ACCD-Biman* comparison on the *Sphere* where p<.05). For the *Knot*, *ACCD*'s σ_d was only significantly smaller than *One-H*'s σ_d (p<.05).

User Preference

We asked participants for their preferred technique, 9/12 participants said it was *Biman* and 7/12 said *ACCD*, but none answered *One-H* (multiple answers allowed). Those measurable results are consistent with participants' comments. All participants were comfortable using *ACCD*, and no participant complained about the camera rotations. Only one participant found clutching of *ACCD* annoying. Another participant said that *Biman* needs more training than *ACCD* and two further participants clearly preferred *ACCD*, as the technique avoids jumps between different parts of the same shape. Some participants expressed the need for occasional explicit control of the viewpoint while drawing with *ACCD*. This could be useful for occasionally reorienting the 3D shape in a more convenient way, to draw in more comfortable directions and to avoid hand occlusion.

Table 1. F and p values of the analysis of variance of the *Technique* effect on CT, NS, NVM, ρ and σ_d for each 3D shape.

	Sphere		Cylinder		Knot	
	$F_{2,22}$	p<	$F_{2,22}$	p<	$F_{2,22}$	p<
CT	7.14	.01	1.44	.26	1.14	.34
NS	5.76	.01	34.59	.0001	15.17	.0001
NVM	7.28	.01	44.36	.0001	7.91	.01
ρ	60.63	.0001	115.96	.0001	19.13	.0001
σ_d	7.46	.01	15.30	.0001	3.57	.05

CONCLUSION AND FUTURE WORK

The evaluation provides encouraging results on the benefits and potential of *ACCD*. First, *ACCD* reduces the number of strokes performed to draw a curve on a 3D shape. This indicates that it allows to draw longer curves in a single stroke. Thus it requires less switching between drawing and viewpoint control. Secondly, *ACCD* provides a higher density and more homogeneity of input points along the drawn curve. This indicates that it provides a more constant drawing precision. Third, it tends to be faster than the other techniques even if this was only significant for the *Sphere* 3D shape. These three points show that *ACCD* improves drawing continuity.

An outcome of our experiment was that occasional explicit viewpoint control would be helpful to reorient the viewpoint while drawing with *ACCD*. *ACCD* could be extended with explicit finger strokes based viewpoint rotation. The user would then rotate the viewpoint with its finger with the current drawing brush position as the pivot point.

ACKNOWLEDGMENTS

This work was supported by the FUI project 3DCI (AAP14); the ANR/JST AMIE project; and Région Rhônes-Alpes. LIG is partner of PERSYVAL-Lab (ANR–11-LABX-0025).

REFERENCES

1. Bae, S.-H., Balakrishnan, R., and Singh, K. Ilovesketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proc. of UIST*, ACM (2008), 151–160.
2. Balakrishnan, R., and Kurtenbach, G. Exploring bimanual camera control and object manipulation in 3d graphics interfaces. In *Proc. of CHI*, ACM (1999), 56–62.
3. Chen, X., Golovinskiy, A., and Funkhouser, T. A benchmark for 3d mesh segmentation. In *Proc. of SIGGRAPH*, ACM (2009), 73:1–73:12.
4. Foskey, M., Otaduy, M. A., and Lin, M. C. Artnova: Touch-enabled 3d model design. In *Proc. of VR*, IEEE (2002), 119–.
5. Fu, C.-W., Xia, J., and He, Y. Layerpaint: a multi-layer interactive 3d painting interface. In *Proc. of CHI*, ACM (2010), 811–820.
6. Hagedorn, B., and Döllner, J. Sketch-based navigation in 3d virtual environments. In *Proc. of SG*, Springer-Verlag (2008), 239–246.
7. Hormann, K., Levy, B., and Sheffer, A. Mesh parameterization: Theory and practice. In *SIGGRAPH Course Notes* (2007).
8. Khan, A., Komalo, B., Stam, J., Fitzmaurice, G., and Kurtenbach, G. Hovercam: interactive 3d navigation for proximal object inspection. In *Proc. of Interactive 3D graphics and games*, ACM (2005), 73–80.
9. Ortega, M. 3D object position using automatic viewpoint transitions. In *Proc. of CHI*, ACM (2013), 193–196.
10. Owada, S., Nielsen, F., Nakazawa, K., and Igarashi, T. A sketching interface for modeling the internal structures of 3d shapes. In *Proc. of Smart graphics*, Springer-Verlag (2003), 49–57.
11. Phillips, C. B., Badler, N. I., and Granieri, J. Automatic viewing control for 3d direct manipulation. In *Proc. of I3D*, ACM (1992), 71–74.
12. Schmidt, R., Khan, A., Kurtenbach, G., and Singh, K. On expert performance in 3d curve-drawing tasks. In *Proc. of SBIM*, ACM (2009), 133–140.
13. Veit, M., Capobianco, A., and Bechmann, D. Cros: A touch screen interaction for cursor manipulation on 2-manifolds. In *Proc. of VRST*, ACM (2012), 97–100.
14. ZBrush. http://pixologic.com/zbrush.
15. Zeleznik, R., and Forsberg, A. Unicam - 2d gestural camera controls for 3d environments. In *Proc. of I3D*, ACM (1999), 169–173.