Design and evaluation of CrOS: a touch screen interaction technique for manipulating on surfaces

Author 01 Author 02 Author 03

ABSTRACT

In this paper, we present and evaluate a new interaction technique, called Cursor On Surface (CrOS). It consists in moving a cursor on the surface of 3-D objects using touch screen input. CrOS is based on an algorithm which maps 2-D inputs into cursor displacements on 3-D surfaces. The technique relies on two principles. Firstly, it restricts the manipulation space to a 2-D space. Secondly, it reduces the complexity of the task by reducing the number of degrees of freedom manipulated by users. After presenting the technique, we provide the results of an experimental study in which we compare CrOS to two other manipulation techniques. The results show that CrOS increases the users' performance and satisfaction for selection task and curve steering task. The results suggest that these benefits come mainly from the reduction of the complexity of the task.

Categories and Subject Descriptors

H.5.1 [**INFORMATION INTERFACES AND PRESENTATION**]: Multimedia Information Systems—*Artificial, augmented, and virtual realities*; I.3.6 [**COMPUTER GRAPHICS**]: Methodology and Techniques Interaction techniques

Keywords

Interaction Technique, Touch Screen Interaction, Virtual Reality, Geometric Modelling

1. INTRODUCTION

In Virtual Reality (VR), many tasks can be assimilated to moving a point of interest (virtual pointer for editing tasks, camera viewpoint for navigation tasks) along a constrained or freely defined path. For example, painting on the surface of an object usually requires to be able to draw long strokes on the surface of an object, sometimes on parts of the object that are initially hidden from the view of the user [1]. Sketching tasks also require to edit paths that define the general shape of an object to create, such as in the Terrain Sketching application [2].

In many contexts, navigating can also be assimilated to following a path. When navigating in 3-D virtual worlds, one generally freely decides the path to follow, and displaces the camera view-point by moving it on a 2-D surface at the users' view height. In other situations of use, however, the task can be much more complex. For example, multi-scale navigation imply to navigate through 3-D scenes that extend at many levels: country, city, houses, or even objects laying on a table [4]. Medical applications also present complex sets of 3-D objects, containing overlapping objects with numerous holes and bends. In those cases, defining a path for navigation on the objects' surface seems to be an efficient solution [5].

We propose a new interaction technique called CrOS. With this technique, the user moves a Cursor of Interaction (CI) on a given surface using touch screen interaction. The displacements on the 2-D surface of the touch screen are mapped, through a mapping algorithm, on the surface of the object. The technique allows to displace a CI to either select an object or define a path on a surface. It also allows to explore the surface of the object by simply moving the CI to displace it towards a distant point of interest. In the remaining of this paper, we will focus on the context of path-based interaction for editing tasks, since the work we present here was conducted in the context of Geometric Modelling Applications (GMA). However, we think that the technique can be used in any situation where the task can be assimilated to moving a CI along an implicit or explicit surface.

In the next section, we introduce the previous works related to our purpose. Then, in Section 3, we describe the implementation of CrOS. In particular, we describe the two algorithms we propose: the first one which computes the geodesic displacements, the second one which computes the reorientation of the object. In Section 4 we present the experimental study we have conducted to evaluate this technique. Finally, in Section 5, we present the results we obtained. The discussion of these results and a few perspectives are given in Section 6 before concluding in Section 7.

2. PREVIOUS WORKS

In this section, we discuss the previous works related to our topic, focusing on navigation and manipulation, two fundamental interaction tasks in VR [3], and the use of 2-D devices for 3-D interaction tasks.

2.1 3-D interaction

In order to provide an intuitive way to interact in VR, many techniques are based on direct manipulation [8]. They allow distant manipulation through several metaphors. The *Virtual hand* [11] technique, extended by the *Go-Go* technique [12], allows to manipulate objects by directly grabbing them with an avatar of the hand. The *Ray-Casting* [13], introduces a beam that allows to manipulate objects at long distances. However, these techniques prevent users from manipulating hidden objects, or the occluded parts of an object. Therefore, users have to constantly orientate the object during the manipulation. They also lack precision for designating precise locations on an object.

Using 3-D devices for interaction is known to introduce specific issues for users that still need to be addressed to propose efficient interaction [9].

- Users are required to perform tasks involving six Degrees of Freedom (DoF) simultaneously: usually three DoF of position and three DoF of orientation. Such manipulation increases significantly the cognitive load. According to Bowman et al. [9], "*most real-world tasks aren't fully 3-D; they are 2-D or 2-1/2-D*". Indeed, in the real world, due to physical constraints, the tasks involve fewer dimensions. For instance, the gravity makes objects usually lay on a surface such as a desk. They propose that offering 2-D alternatives for 3-D user interfaces' tasks is an excellent way to increase usability;

- The whole manipulation is constantly in the mid-air. Such manipulation can introduce fatigue [10] that can impair users' comfort and limit their precision. Again, constraining the input to 2-D tasks might reduce the fatigue and provide higher accuracy [9].

2.2 Using 2-D devices for 3-D interaction

Several techniques propose to use 2-D devices (e.g. a mouse or a touch screen) to manipulate in a 3-D environment. The *Balloon Selection* technique, designed by Benko et al. [14], consists in positioning a selection cursor in the 3-D space using touch screen interaction. Smith et al. propose also to manipulate objects in a 3-D environment using 2-D devices [6]. To simplify the task, they introduce constraints on the possible positions of objects. They show that their technique is much more efficient than free interaction with 3-D devices.

Using 2-D devices for navigation tasks has also been proposed. The *Navidget* technique uses 2-D inputs for navigating in 3-D environments [7]. The user simply designates the part of an object he/she wants to explore, and the technique automatically computes a position for the camera that will ensure an adequate view of the object. However, it is a flying technique, that does not allow to control the path followed by the user view-point. Hanson et al. [21] propose a technique based on the idea of a constrained manipulation to navigate in a 3-D space using a mouse. To this purpose, they use a *surface for navigation* to move in a 3-D scene. The user navigates in the 3-D environment by navigating on this surface.

One major difficulty to implement this technique is to compute the displacements on the surface for navigation. Their approach is quite similar to the approaches used in the parametrisation of surfaces. They unfold the surface over a 2-D surface. By moving on the unfolded surface, the user moves on the original surface. This technique however has been designed for controlling camera viewpoints during navigation tasks and isn't adapted for other tasks (such as modelling tasks for instance). Moreover, this technique has not been tested using complex surfaces, presenting hidden parts, or numerous holes and bends.

3. CROS

3.1 Mapping design principles

Our objective is to propose an intuitive technique for precise and efficient interaction on the surface of an object. With this technique, we want to perform several complex modelling operations (e.g. sculpting, colouring or topological manipulation) in VR in a more efficient and more comfortable way than direct manipulation techniques. It must allow to precisely designate a point of interaction or define a path on the surface. It is also important that it is adapted to complex surfaces that present numerous holes and bends.

To that purpose, we propose to retain, from the previous works, the following ideas.

- Reduce the complexity of the task by reducing the number of degrees of freedom controlled. This will be ensured by the use of a 2-D input device, and the automatic orientation of the object;

- Introduce a physical support for the manipulation, in order to reduce the fatigue of users and increase their precision.

The CrOS interaction technique consists in performing interaction tasks on 3-D objects using touch screen interaction. To this purpose, we place a CI on the surface of the object. The cursor is then moved using 2-D displacements through touch screen interaction. In our VR environment, the tactile interface is the lower screen of the Workbench (which is at arms' reach). Using the pressure sensors placed at the extremity of the fingers of the data gloves, we are able to detect whether the user is touching the screen or not. The displacements of the hands are computed using the optical tracking system (see Section 4.4).

CrOS maps the users' hands movements into geodesic displacements on the surface of the 3-D object. In the remainder of this paper, we consider that this is performed by the **mapping function**. These geodesic displacements are then used to move the CI. From a geometrical point of view, mapping 2-D displacements on a 3-D surface is similar to the problem of finding a parametrisation of a surface. However, algorithms used to find a local or a global parametrisation of a surface [15] are generally time consuming [16] and may introduce latency. This is known to decrease the users' performance by reducing accuracy and increasing achievement times [17]. This is why, in the algorithm we propose, we focus on trying to introduce a low computational time. Moreover, the algorithm tries to best comply with the principles of the Directional Stimuli - Response Compatibility (DSRC) (which is not possible when using a parametrisation).

In order to reduce the computational cost, our solution is based on a method using only the local geometry of the object around the CI. The idea is that, for a given point on the surface of the object, each direction within the tangent plane to the surface at this point can be projected onto a geodesic path on the surface (see Figure 1). It is important to note that this statement is only true for 2-manifolds (i.e. surfaces that are locally similar to 2-D euclidean space at each position on the surface).

This geodesic path can be determined using the local geometry of the surface around the CI. More precisely, the geodesic path can be considered as a part of the intersection between a plane (called **geodesic plane**) and the surface. Avoiding the calculation of the entire intersection reduces significantly the computation time. The geodesic plane is orthogonal to the tangent plane, contains the 2-D movement and passes through the position of the CI (see Figure 2). These constraints ensure a good compatibility between the displacements performed by the user on the touch screen and the CI. The details to geometrically define the geodesic plane are given in Appendix A.

The method we use to compute the geodesic path is the following: compute the local intersection between the geodesic plane and

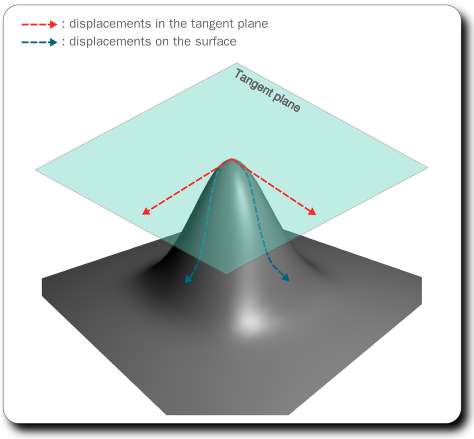


Figure 1: Two 2-D displacements transformed in geodesic displacements on the surface of the object (used to move the cursor located on the top of the bump).

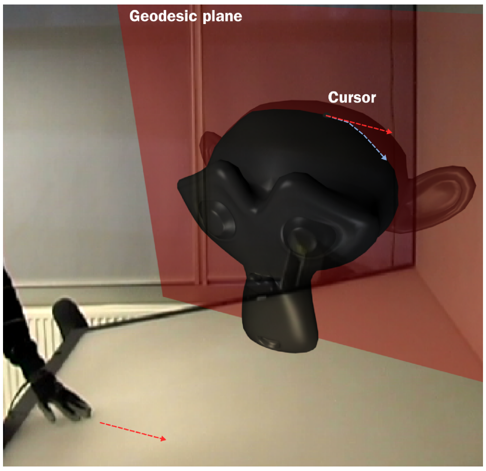


Figure 2: We can see that the geodesic displacement on the surface is defined as a part of the intersection between the 3-D object and the geodesic plane.

the object and move the CI along the intersection until the travelled distance is equal to the length of the movement performed by the user (ensuring a 1:1 manipulation).

3.2 Automatic orientation

Because of the possible complex geometry of the object, the user might have to repeatedly modify the orientation of the object, introducing multiple switches between the cursor manipulation and the object's orientation correction. To avoid this problem, we propose to add to the technique an orientation algorithm, that automatically rotates the object according to the displacements of the cursor on the surface and the position of the head of the user. This will reduce the number of DoF, and thus the complexity of the task. For users, this should mean a reduction of the cognitive load allowing them to achieve higher performances and increase usability, despite the fact that such a technique reduces predictability.

We implemented an algorithm to automatically orientate the object in order to provide the user with an appropriate point of view toward the cursor. This algorithm follows the two following rules.

1. When the angular distance between the normal of surface where the cursor lies and the viewing direction of the user toward the cursor is over a threshold (e.g. 85 degrees) we orientate the surface to align these two directions;

2. The orientation correction is performed when the criterion before-mentioned is not respected for at least one second.

This mechanism ensures an optimal DSRC during the whole manipulation and provides an appropriate point of view to the cursor being manipulated. To avoid sudden changes of the object, all corrections are progressively and smoothly applied.

4. EXPERIMENTAL STUDY

In this section we describe the experimental study we have conducted to evaluate the CrOS technique.

4.1 Interaction techniques

To validate the design principles we followed to develop CrOS, we wanted to evaluate separately the impact of two different factors: the reduction of the manipulation space to a 2-D surface and the impact of the automatic orientation on the participants' performance. To that purpose, we developed three techniques that we present in this section. To ensure a maximum homogeneity between the experimental conditions, none of these techniques introduced a collocated manipulation (the positions of the device and the cursor were not confounded).

4.1.1 *Visually Constrained Manipulation (VCM)*

In this condition, the user manipulates the spherical cursor using free manipulation gestures. The cursor's movements, however, are constrained to the surface of the object. By activating the thumb and index pressure sensors of the dominant hand simultaneously (through a pinch gesture), the user virtually grabs the cursor to move it. When the user performs a pinch gesture and moves the dominant hand, the displacement of the hand are mapped on the surface and applied to the cursor. Movements that are orthogonal to the surface on which the cursor lays are ignored. Using the non dominant hand, the user can orientate the object by simply closing and rotating the hand.

4.1.2 *Touch Screen Interaction (TSI)*

This condition implements the use of touch screen interaction using the method we describe in Section 3. In order to propose a condition that is homogeneous, the user rotates the object using touch screen interaction. By touching the screen with the non dominant hand, the user can rotate the object around the lateral axis by performing back and forth movements. The user can also rotate the object around the vertical axis by performing left and right movements. The user can combine these two axes of rotation by performing diagonal movements on the screen.

4.1.3 *Cursor On Surface (CrOS)*

This last technique is similar to the TSI technique since the manipulation of the cursor is done using the horizontal screen of the workbench. In this condition, however, the user does not have to orientate the object since the technique implements the orientation algorithm described in Section 3.

4.2 Tasks

Each participant performed two tasks. The first task is a *target selection task*. The participants had to achieve four blocks of five selections with each technique. The task consisted in bringing the manipulated cursor into a spherical volume representing the target (the size difference between the cursor and the target is 0.5cm). The position was automatically validated if the cursor remained in the target during 0.8s. Both the target and the initial position of the CI are on the surface of the object.

The second task is a *curve steering task*. This task is repeated nine times. Three different curves, entirely on the surface of the object, are presented three times in a random order. The cursor is initially at one extremity of the curve. The task is to bring the cursor to the other extremity of the curve. The instructions we gave to the participants were to find the best compromise between speed and accuracy during the task.

Both selection and curve steering tasks are performed on a spherical shape selected randomly in a set of 3-D objects of different sizes and shapes. Bumps and holes on the objects are generated randomly and ensures that the object is sufficiently complex to encounter situations of occlusion and necessitate reorientations during the task (see Figure 3).

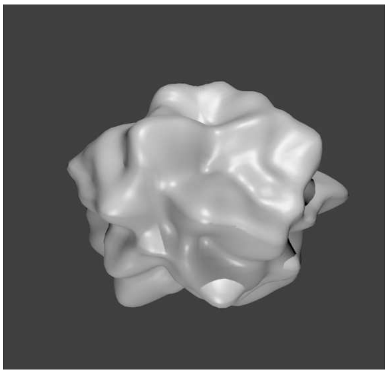


Figure 3: One of the shapes used for the experiment.

We intentionally chose two tasks that are fundamentally different. The selection task introduces a ballistic and a control phase during its achievement. On the other hand, the curve steering task necessitates a finer control during the whole fulfilment. Our aim is to know the ability to use this technique for various interaction situations.

4.3 Experimental procedure

Twelve voluntary participants (students and persons working at the University) took part in this study (mean age = 26.1, SD = 5.8). They all have already used VR with various levels of experience (ranging from occasional use to regular use). We used a within subject design: each participant carried out the selection and curve steering tasks with the three different techniques. To avoid any ordering effect, we used a Latin square distribution of the order of presentation of the techniques.

The experimental session takes place as follow. We first present the experimental environment to the participant. Each block starts with a training session during which the technique is presented orally. The participant could then use the technique freely as long as wanted. During this phase, one of the 3-D objects is selected randomly and presented to the user. We asked the participants to take their time during this training to understand how each interaction technique works. The objective is to familiarize the subject both with the environment and the various techniques.

The participant is then asked to perform the selection and curve steering tasks. After completing these trials, the subjects were asked to fill a subjective evaluation questionnaire. Finally, the subjects were asked to freely discuss each technique, their advantages and drawbacks during an informal debriefing.

4.4 Apparatus

Each experiment was conducted on a Barco Consul, a VR environment providing stereoscopic display. The two screens have a resolution of 1400×1050 with a refresh rate of 120 Hz. The CrystalEyes CE-2 glasses are synchronized with the screens to ensure the stereoscopy. The images were generated using two workstations (one for each screen) fit out with a 3 GHz Intel Quad Core and a NVIDIA Quadro FX 5600. The head and hands tracking were provided by an ART tracking system and optical trackers. The apparatus is completed by two XIST Data Gloves to interact the environment.

4.5 Measurements

During the experiments, we took the following measurements.

1. Achievement time (AT, in second): the time elapsed between the first and the last movement of the cursor for a given task;

2. Euclidean error (E, in mm): for the curve steering task, we computed the mean euclidean distance between the cursor and the curve.

We chose not to measure the precision during the selection task. A previous study interested in the influence of the use of touch screen interaction on positioning tasks found that the physical support had no influence on the precision during precise positioning tasks [18]. However, precise positioning task rely only on the position of the object at the end of the task. Thus, it requires precision only during the control phase of the task and not during the ballistic phase of the positioning. On the contrary, curve steering tasks necessitate to precisely follow the trajectory and require an acute control all along the task. In that context, the factors we want to evaluate may have an influence on the precision. That is why we decided to measure euclidean error only for the curve steering tasks. However, we probably would have measured precision in every condition if one of the technique had introduced the use of a finer muscular group than the forearm-wrist muscular group (through the use of a device introducing fingertip precision).

The questionnaire was designed to investigate the ease of use of the techniques, their impact on the physical and mental demand, the appropriateness to the tasks and the subjective evaluation of the performance in both tasks. For each question, the participant had to rate each technique on a 7 level Likert scale (1 = I strongly disagree ; 7 = I strongly agree).

4.6 Hypothesis

We had the following expectations concerning the impact of the interactions techniques (and therefore the factors) on the users' performance.

H1: the TSI technique leads to higher performance when compared to VCM. This is due to the fact that this condition reduces the dimension of the manipulation space (from a 3-D to a 2-D manipulation space). The technique also introduces a physical support that reduces fatigue.

H2: by removing the need, for the user, to reorient the object, the CrOS technique reduces the complexity of the manipulation task. This leads to significantly improved performances in terms of task completion times for the selection task. However, since the user does not control the orientation of the object, the technique may prove detrimental for the performance during curve steering tasks.

5. RESULTS

In this section, we present the results of the experimental study. An ANOVA with repeated measurements on the three different techniques was conducted. If the ANOVA showed a significant impact of the technique, a post-hoc analysis using the Tukey HSD procedure was performed to identify the significant differences between the techniques.

5.1 Selection task

The performance for the selection task are presented in Figure 4. The statistical analysis shows a significant impact of the technique ($F(2, 9)=10.189$, $p<0.05$). The post-hoc analysis reveals that the selection times are significantly reduced when using CrOS compared to VCM and TSI (for all the comparisons, $p<0.05$). However, there is no statistically significant difference between VCM and TSI ($p>0.05$).

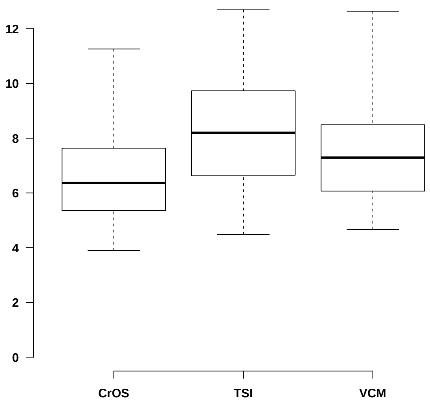


Figure 4: Quantitative measurements (box plots) for the selection task. The mean achievement times are given in seconds.

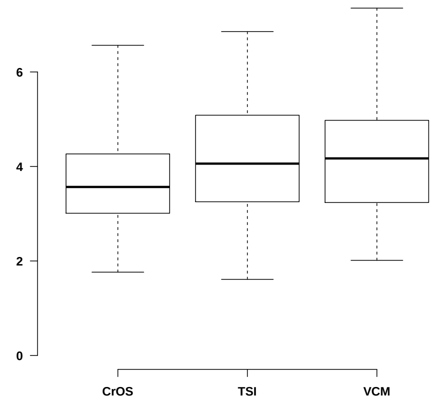
5.2 Curve steering task

Figure 5 presents the results for the curve steering task. The technique has a significant influence on the euclidean error ($F(2, 9)=7.5167$, $p<0.05$). The pairwise comparisons show that the mean euclidean error is significantly reduced (by 14%) when using CrOS compared to VCM and TSI ($p<0.05$). However, there is no statistically significant difference between VCM and TSI ($p>0.05$). Concerning achievement times we found no significant influence of the technique on the results ($F(2, 9)=0.5946$, $p>0.05$). These results indicate that the CrOS technique is more precise than the other techniques, but does not influence the achievement times.

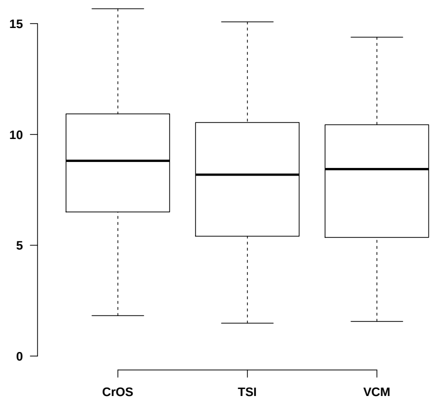
5.3 Subjective evaluations

Now, we present the results of the subjective evaluation. The quantitative results are given in Figure 6.

It seems that the interaction technique has a significant influence on Ease of Use (EU) with $F(2, 9)=5.91$ and $p<0.05$, Fatigue (F) with $F(2, 9)=7.08$ and $p<0.05$ and Mental Demand (MD), $F(2, 9)=10.87$ and $p<0.05$). However, we found no statistically significant influence of the technique either on Appropriateness for Curve steering (AC), $F(2, 9)=2.66$ and $p>0.05$, nor on Appropriateness for Selection (AS) with $F(2, 9)=1.2$ and $p>0.05$.



(a) Euclidean error



(b) Mean achievement time

Figure 5: Quantitative measurements (box plots) for the curve steering task (a) Mean euclidean error for each technique (b) Mean achievement times for each technique (in seconds).

The post-hoc analysis indicates that, for EU, the difference between VCM and TSI is significant ($p<0.05$), as well as the difference between VCM and CrOS ($p<0.05$). However, there is no difference between TSI and CrOS.

Regarding F, the post-hoc analysis shows that CrOS significantly reduces the fatigue compared to VCM ($p<0.05$). All the other pairwise comparisons were not significant. Finally, concerning MD, CrOS led to a significantly reduced mental demand when compared to TSI and VCM ($p<0.05$) but we found no difference between TSI and VCM ($p<0.05$).

6. DISCUSSIONS

With this study, we wanted to evaluate the performance od CrOS in an experimental situation of use, and assess the design principles that we used to conceive our technique.

6.1 Design principles

The results we obtained show no difference between the TSI and

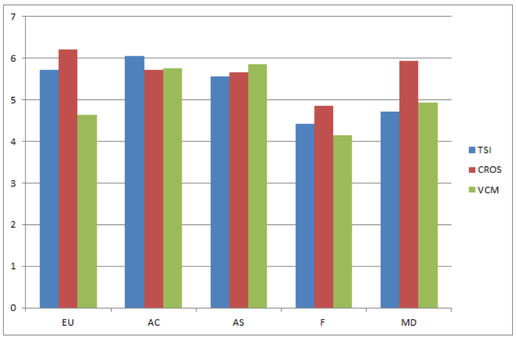


Figure 6: The results of the subjective evaluation for the different criterion. EU : ease of use ; AC : appropriateness for curve steering ; AS : appropriateness for selection ; F : fatigue ; MD : mental demand. The higher the evaluation, the better the results. Regarding F and MD, the user were asked to rate higher the techniques introducing the lesser fatigue.

VCM conditions in terms of task completion times during the selection and the curve steering tasks. Moreover the results show no differences between those techniques in terms of precision during the curve steering task. These results contradict our first hypothesis. Our idea was that the reduction of the dimension of the manipulation space and the presence of a physical support in the TSI condition would help the participants to achieve better performance.

We think that these results may be partly explained by the fact that the TSI technique does not modify the mental representation of the task. Indeed, that condition does not introduce a reduction of the complexity of the task. In the VCM condition the displacements of the cursor are already reduced to two DoF since they are constrained to the surface of the object. Likewise, the orientation of the object can be considered as a 2-D task, since any orientation of the surface can be performed through two DoF of rotation. Consequently, the TSI technique only reduces the dimension of the manipulation space to the dimension of the DoF of the pointer on the surface. It does not modify the mental representation of the task and does not affect the complexity of the task. This hypothesis is supported by the absence of difference between TSI and VCM regarding the mental demand.

The results also suggest that there is no impact of the presence of the physical support in the TSI condition, since they show no influence on fatigue. We think that this result can be explained by the experimental set-up we used during our evaluation. The results show no difference between TSI and VCM regarding fatigue, while previous results stress that the presence of a physical support should lead to a reduction of the physical demand [18]. We think that this is due to the short manipulation sessions. The participants only had to use each technique for approximately 15 minutes during the experiment. They also had the possibility to rest between two blocks of manipulation. We think that these experimental conditions did not allow the subject to feel the fatigue induced by the manipulation in the mid-air. It is likely that, with longer experimental sessions, an effect of the TSI technique on fatigue would have been identified.

Contrary to the TSI condition, the statistical analysis shows that the CrOS technique led to higher performance in terms of task completion times for the selection task. This result validates the first proposition of our second hypothesis. This indicates that reducing the dimensionality of the task can actually lead to higher performance in terms of task completion times. We were surprised to notice that, contrary to the second proposition of H2, CrOS also led to a lower euclidean error, and had no influence on task completion times, for the steering task. Indeed, we thought that unanticipated reorientations of the object while the participants were moving the pointer might be detrimental. However, it seems that the participants were able to handle these reorientations and adapt the trajectory of the pointer to minimize the euclidean error during curve steering, without significantly increasing cognitive load, as supported by the results concerning MD. This is also illustrated by the fact that the subjective evaluations for EU show no statistical differences between CrOS and TSI.

These results suggest that the improvements in performance mainly come from the reduction of the number of DoF controlled by the participants. As a consequence, the technique allows a reduction of the cognitive demand of the task. The cognitive demand have been identified in a previous study as a key-point for efficiency when using touch screen interaction to interact in virtual environments [20].

6.2 Evaluation of CROS

Our objective for this study was to validate the use of CrOS in VR. In particular, we wanted to know if the technique is usable in various interaction situations. The experimental results are very encouraging and suggest that the technique is efficient for the selection and the curve steering tasks. It leads to lower task completion times for the selection task, and to a higher precision for the curve steering. Moreover, contrary to what we thought before those experiments, the choice to add an orientation algorithm to the technique is not detrimental to the intuitivity of the manipulation.

It is important to notice that the evaluation sessions in this experimental study lasted only about 50 min. These are short sessions compared to usual manipulation sessions (such as 3-D modelling sessions). This can explain the limited differences between the techniques concerning performance and subjective evaluations. We think that the benefits we observed during this first experiment may increase proportionally to the duration of sessions.

For these reasons, we think that this technique can be efficient in Computer Aided Design (CAD) or GMA applications. Since many tools necessitate users to perform curve steering and selection tasks on the surface of models, it can offer an efficient way to apply editing operators. We are currently using it in a GMA we have developed called *VRModelis* (an example of an editing task in *VRModelis* is given in Figure 7). This application integrates some of the basic tools used to model 3-D objects, including topology manipulation, geometry manipulation (warping, sculpting, etc.) and appearance edition (colouring or texturing).

Within this environment, we are planning to further evaluate the technique, in a more complex situation of use and with longer evaluation sessions. We also want to improve the technique through several means. We want to introduce a non-linear control-display ratio, to allow users to cover long distances more easily, without harming precision when needed. CrOS can also be used to introduce pseudo-haptic information to have an enhanced perception of the relief on the surface of the model. Moreover, we think that the orientation algorithm can be improved in order to match the displacements of the CI and limit the divergence between the reorientation and the trajectory.

We would also like to stress that our apparatus does not allow fingertip precision. Since involving finer muscular groups in the interaction has been proven to increase users' performance for precise tasks, we think that using the last generations of touch screen devices could lead to even higher precision with significantly lower

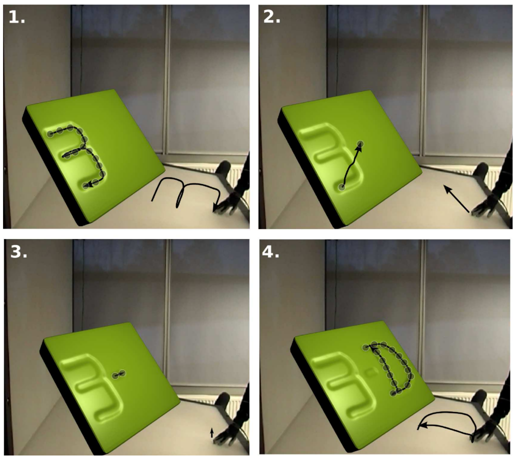


Figure 7: The user utilizes the CrOS technique to sculpt the model.

achievement times [19]. Moreover, using such a device, we could benefit from multi-touch input to design new modelling tools (e.g. physically-based operators which allow user to fold or stretch the surface), that could present richer interaction possibility. This last point seem very promising and we are planning to develop new tools exploiting this idea.

7. CONCLUSIONS

In this paper, we proposed and evaluated a new technique of interaction for editing 3-D objects, called CrOS. This technique uses touch screen interaction to control a cursor constrained to the surface of the object. It also implements an orientation algorithm that allows users to focus on the task at hand: the manipulation of the cursor. The results of the experimental study we have conducted indicate that the technique significantly increases the users' precision during curve steering tasks, and reduces selection times. These benefits may be particularly interesting in environments using complex editing tools, such as CAD and GMA.

8. REFERENCES

[1] CW. Fu, J. Xia, and Y. He. LayerPaint: a multi-layer interactive 3-D painting interface. In Proceedings of the 28th international conference on Human factors in computing systems (CHI '10). ACM, New York, NY, USA, 811-820, 2010.

[2] J. Gain, P. Marais, and W. StraSSer. Terrain sketching. In Proceedings of the 2009 symposium on Interactive 3-D graphics and games (I3D '09). ACM, New York, NY, USA, 31-38, 2009.

[3] D. A. Bowman, E. Kruijff, J.J. LaViola, and I Poupyrev, 3-D User Interfaces : Theory and practice. Redwood City, CA, USA : Addison Wesley longman Publishing Co. Inc., 2004.

[4] J. McCrae, I. Mordatch, M. Glueck, A. Khan, Multiscale 3-D navigation, Interactive 3-D Graphics and Games, 7-14 2009.

[5] B. Van Schooten, E. Van Dijk, A.Suinesiaputra, J. Reiber, Interactive navigation of segmented MR angiograms using simultaneous curved planar and volume visualizations. International Journal of Computer Assisted Radiology and Surgery,6,5:591-599, 2011.

[6] G. Smith, T. Salzman, W. Stuerzlinger. 3-D Scene Manipulation with 2-D Devices and Constraints, Graphics Interface 2001, pp. 135-142, June 2001.

[7] M. Hachet, F. Decle, S. Knĺodel, and P. Guitton. Navidget for Easy 3-D Camera Positioning from 2-D Inputs. In Proceedings of IEEE Symposium on 3-D User Interfaces (3DUI), pages 83-89, 2008.

[8] B. Shneiderman. Direct manipulation for comprehensible, predictable and controllable user interfaces. International Conference on Intelligent User Interfaces (IUI'97), 33-39, 1997.

[9] D. Bowman, S. Coquillart, B. Froehlich, M. Hirose, Y. Kitamura, K. Kiyokawa and W. Stuerzlinger. 3-D User Interfaces: New Directions and Perspectives. IEEE Computer Graphics Applications (CGA'08), 28:20-36, 2008.

[10] K. Hinckley, R. Pausch, J. Goble and N. Kassell. A Survey of Design Issues in Spatial Input. ACM Symposium on User Interface Software & Technology (UIST'94), 213-222, 1994.

[11] D. Sturman, D. Zeltzer, and S. Pieper. Hands-on interaction with virtual environments. ACM Symposium on User Interface Software & Technology (UIST'89), 19-24, 1989.

[12] I. Poupyrev, M. Billinghurst, S. Weghorst and T. Ichikawa. The go-go interaction technique: non-linear mapping for direct manipulation. ACM Symposium on User Interface Software & Technology (UIST'96), 79-80, 1996.

[13] I. Poupyrev and T. Ichikawa. Manipulating Objects in Virtual Worlds: Categorization and Empirical Evaluation of Interaction Techniques. Journal of Visual Languages & Computing (1999), vol. 10, pp. 10:19 - 35, 1999.

[14] H. Benko and S. Feiner. Balloon Selection: A Multi-Finger Technique for Accurate Low-Fatigue 3-D Selection. IEEE Symposium on 3-D User Interfaces (3DUI'07), 2007.

[15] K. Hormann, K. Polthier and A. Sheffer. Mesh parameterization: theory and practice. ACM SIGGRAPH ASIA courses, 1-87, 2008.

[16] A. Sheffer, E. Praun and K. Rose. Mesh parameterization methods and their applications, Foundations and Trends. Computer Graphics and Vision 2:105-171, 2006.

[17] A. Pavlovych and W. Stuerzlinger. The tradeoff between spatial jitter and latency in pointing tasks. ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS'09), 187-196, 2009.

[18] M. Veit, A. Capobianco and D. Bechmann. An Experimental Analysis of the Impact of Touch Screen Interaction Techniques for 3-D Positioning Tasks. IEEE Conference On Virtual Reality (VR'11), 2011.

[19] S. Zhai, P. Milgram and W. Buxton. The influence of muscle groups on performance of multiple degree-of-freedom input. SIGCHI conference on Human Factors in Computing Systems (CHI'96), 308-315, 1996.

[20] R.K. Hippler, D.S. Klopfer, L.M. Leventhal, G.M. Poor, B.A. Klein and S.D. Jaffee. More than speed? An empirical study of touchscreens and body awareness on an object manipulation task. Proceedings of the 14th international conference on Human-computer interaction: interaction techniques and environments,2:33-42, 2011.

[21] A. Hanson and E. Wernert. Constrained 3-D navigation with 2-D controllers. Conference on Visualization (VIZ'97), 175-183, 1997.

APPENDIX

A. DETAILS OF CrOS IMPLEMENTATION

To implement the mapping function, we do not make any assumptions on the data structure used to represent the 3-D objects. In order to be efficient, it is preferable to use a data structure providing information about the topology of the object. More specifically, it is recommended to have access for each face of the object, to all adjacent faces.

To formally describe the algorithm, we use the following geometrical entities:

- P_c (the control plane): the plane in which the user performs the 2-D displacements. In our case, the interaction surface is the surface defined by the screen of the Workbench which is at arms' reach. Therefore, P_c is defined as the plane containing the surface of the screen;
- $\vec{N_c}$: the normal associated to the plane P_c. In our case, $\vec{N_c} = (0,1,0)$;
- P_g (the geodesic plane): the plane used to compute the displacement on the surface of the object;
- $\vec{N_g}$: the normal associated to the plane P_g;
- $\vec{D}$: 2-D movement performed by the user into the control plane. In practice, the movement is a 3-D movement whose y coordinate is considered as null;
- I: the intersection between P_g and the 3-D object. The intersection is stored as a polygonal line (closed or open) containing the list of the points of intersection between the edges and the vertexes, and the plane.

We now present how to define the plane P_g.

A.1 Geodesic plane computation

The constraints used to define the plane P_g are the following:

- It passes through the cursor position;
- It contains the displacement specified by the user;
- It is orthogonal to the tangent plane at the current cursor position.

From these characteristics, it is possible to define geometrically the plane P_g. First, we define $\vec{N_g}$ as follows:

$$\vec{N_g} = A \times (\vec{D} \times \vec{N_c})$$

A is the matrix representing the minimum geometric transformation between $\vec{N_c}$ and the surface normal at the cursor position. In other words, if $\vec{N_s}$ is normal of the surface at the cursor position, $\vec{N_s} = A \times \vec{N_c}$.

By defining the normal of the plane, we define three of the four unknowns in its equation. The fourth unknown can be easily computed using the fact that CI is contained in the plane. Now, we are able to define P_g and to compute the intersection in order to define the geodesic path.

A.2 How to compute the intersection

The approach we propose to compute the intersection is incremental. A naive approach would be to compute the entire intersection between P_g and the 3-D object to finally select the part of the intersection corresponding to the geodesic path. This calculation, with an adapted data structure, would be in $O(n)$ where n is the number of edges of the mesh. We propose an approach where the intersection is constructed step by step. The incremental computation reduces the computation time to $O(m)$ where m is the number of vertexes or edges belonging to the geodesic path. This gain is significant especially for objects with a high level of details. The detailed algorithm is given at the end of this section.

A difficulty appears for the computation of the geodesic path. Consider that the CI is on a face F. Consider also that the geodesic path is defined as a sequence of vertexes forming a polygon, the first vertex of the geodesic path is the CI itself. The intersection between F and P_g is a line with at least two vertexes that are on the edges of F or be vertexes of F. The question that arises is: "Which vertex belongs to the geodesic path?". By choosing the wrong vertex, we introduce an inversion in the relationship between the direction of the CI displacement and the direction of the user's movement.

To choose the correct the vertex, we first consider $\vec{V_g}$, the vector defined by:

$$\vec{V_g} = \text{tested vertex} - \text{position of the CI}$$

We then choose the vertex so that:

$$\vec{V_g}.\vec{D} > 0 \qquad (1)$$

This test indicates whether the displacements on the surface will induce a movement whose projection on P_c is in the same direction as $\vec{D}$ or not. If the condition is satisfied, the vertex belongs to the geodesic path. This condition ensures the DSRC.

Now, we have found the first vertex to be added to the geodesic path. Once this first element added, the construction of the rest of the geodesic path can be done incrementally and without ambiguity. For this we use the Algorithm 1.

Algorithm 1 Compute the local intersection

Ensure: $geodesicPath$
  $geoPathLength = \| \vec{D} \|$
  $firstInter$ = first intersection
  $interEdge$ = edge on which is $firstInter$
  $curFace$ = adjacent face to F and incident to $interEdge$
  $geodesicPath = cursorPosition, firstInter$
  $curGeoPathLength = length(cursorPosition - firstInter)$
  while $curGeoPathLength < geoPathLength$ **do**
    $curInterEdge$ = edge intersected by P_g different of $interEdge$
    $curInter$ = point of intersection between P_g and $curInterEdge$
    $curGeoPathLength += \| curInter - geodesicPath.back() \|$
    add the intersection with $curInterEdge$ to $geodesicPath$
    $interEdge = curInterEdge$
    $curFace$ = face adjacent to $curFace$, incident to $curInterEdge$
  end while

Finally, the cursor position after displacement on the geodesic path is defined as the position on the geodesic path as the geodesic distance between the previous position and the current cursor position is equal to the length of the journey made by the user in the control plane.