Evaluating the Usability of Recent Consumer-grade 3D Input Devices

C. Siegl, J. Süßmuth, F. Bauer and M. Stamminger
Computer Graphics Group, FAU Erlangen-Nuremberg, Erlangen, Germany

Keywords: HCI, User Interface, Modeling, Interaction, 3D.

Abstract: Recently, 3D input devices such as the Microsoft Kinect sensor or the Leap Motion controller became increasingly popular - the later specialized in recognizing hand-gestures, advertising a very precise localization of tools and fingers. Such devices promise to enable a touchless interaction with the computer in three-dimensional space enabling the design of entirely new user interfaces for natural 3D-modeling. However, while implementing a modeling application we found that there are still fundamental problems that can not easily be solved: The lack of a precise and atomic gesture for enabling and disabling interaction (clicking gesture) and a poor human depth perception and localization within an invisible coordinate frame. In this paper, we show why the precision of the interaction is not limited by hardware but software constraints.

1 INTRODUCTION

Commercial 3D modeling software is traditionally operated using a 2D input device (e.g., a mouse or a graphics tablet) and a keyboard. The lack of affordable 3D input devices in the past has forced developers of such software to implement modeling paradigms that reduce the dimensionality of the user interaction. In most tools, manipulations are limited to a 2D plane, which may, for example, be perpendicular to the current viewing direction or conform with the tangent plane of the manipulated object. Some modeling tools (e.g., Blender) go even further and separate 3D movement into three distinct 1D movements along the x-, y- and z-axis. In practice, these concepts are usually different from software to software and often counter-intuitive, which makes traditional 3D modeling tools hard to learn and difficult to use for non-expert users.

With the recent advance of 3D motion tracking devices like the Microsoft Kinect^{TM}or the Leap Motion controller, input devices for 3D interaction became affordable for end users. This poses an entire new challenge for software developers to devise new paradigms for 3D modeling and scene manipulation. Future modeling tools will have to provide an intuitive and easy to use 3D gesture based user interface that supports artists in their everyday work.

While reviewing existing applications we found that most of them claim to make use of a 3D free-hand gesture based input device. However, in practice they do not really use all three dimensions. Often the interface resembles a multitouch interface without touching anything. The third dimension is solely used as an activation or clicking gesture.

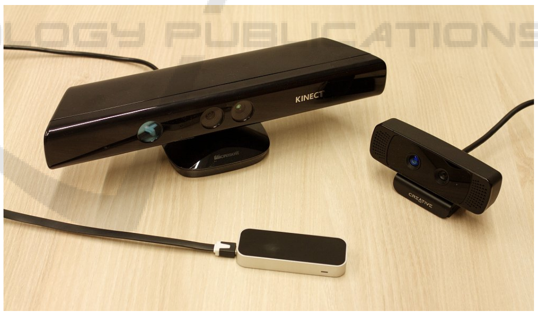

Figure 1: Three devices for 3D interaction. We concentrate on the Leap Motion controller (middle).

After realizing that there is no modeling software using true three-dimensional input, we wanted to leverage the new Leap Motion controller with its precise spatial hand detection algorithms to implement a simple to use software for common modeling tasks. While at first glance, this seems to be a trivial task, we found that there are fundamental obstacles preventing an intuitive and precise interaction. Since this is a problem concerning almost every software that relies on precise spatial three-dimensional interaction we decided to investigate this issue.

1.1 Consumer Hardware

There are multiple consumer grade input devices with variable technical specifications on the market. Considering the requirements of a 3D modeling and scene interaction application, we need a device that offers as much spatial and temporal accuracy as possible.

The first device that comes to mind when thinking about tracking human poses is the Microsoft Kinect sensor (Microsoft Corporation, 2010) (see Figure 1). Being one of the first affordable consumer devices in this area it is widespread. The Kinect is suitable for tracking the entire human body. However, it does not offer sufficient resolution for tracking individual fingers. Additionally, the maximum frame rate at the highest resolution is only 30 frames per second.

A similar device is the SoftKinetic (Intel Corporation, 2013) (see Figure 1). The SoftKinetic works in a range from 0.1m to 1.1m. It is geared towards tracking faces and hands, with frame rates around 30 frames per second.

The most recent device that is completely geared towards tracking hands is the Leap Motion controller (Leap Motion, Inc., 2012) (see Figure 1). While the Leap launched on July 22, we got a pre-production unit to work with. The Leap sensor is able to determine the location and orientation of the users hand with sub-millimeter precision. Compared to the other devices it provides the highest accuracy in the tracking of hands and fingers, and operates at up to 295 frames per second. Those specifications look promising enough to envision the use of this device in intuitive and interactive 3D modeling.

1.2 Previous Work

3D user interfaces have been an active topic in the computer vision and graphics communities and many interesting papers have been published over the last decades. For a broad overview over 3D spatial interaction, we refer our readers to the Siggraph 2011 course notes on “3D spatial interaction: application for art, design, and science” (LaViola and Keefe, 2011) and references therein.

While over the last years, a lot of research has been targeted at large scale 3D body gesture recognition using the Kinect Sensor (see (Ren et al., 2011; Gallo et al., 2011) and references therein), only little effort has been put into assessing the abilities of 3D interfaces for 3D object modeling and manipulation.

Using custom hardware, one very advanced approach towards modeling using 3D interaction was conducted by Araujo et al. (Araujo et al., 2013) by showing their tool “Mockup Builder”. They use a multitouch screen with 3D projection, a Kinect and mechanical tracking of positions in space. This complicated setup remedies a lot of problems we encountered but the sheer amount of hardware does not seem feasible to us.

Hilliges et al. (Hilliges et al., 2009) have implemented a system (hard- and software) that combines a multitouch device with depth information. In their system they can pinch an object and move it in 3D. They encountered similar problems with depth perception we will describe later on (see Section 4.2). In their application the scene is presented to the user from a birds eye view, therefore shadows are sufficient to correctly gauge relative depth.

Ren and O’Neill (Ren and O’Neill, 2013) address the problem of 3D selection using freehand gestures. They state that a single action for this task is not feasible because of accuracy issues. In their work they propose that a series of low level movements improves the precision of the selection. To get a better understanding for this we recommend the survey paper of Argelaguet and Andujar (Argelaguet and Andujar, 2013). This is a special case of handling clicking gestures. We will describe this in a more general manner later on.

Another approach that was taken in the past towards an accurate retrieval of hand gestures are HMI (human machine interface) gloves. See for example the work of Saggio et al. (Saggio et al., 2010). These gloves are equipped with sensors at every joint recovering every movement. They allows for a very accurate retrieval of relative finger positions. However the absolute position of the hand in 3D still can not be determined. Also the usability suffers and the price of such systems is quite high.

In this context, one last bit of related work we want to mention is the work with haptic feedback devices like for example the PHANTOM (Massie and Salisbury, 1994). Using such a device the user holds a pen that is connected to a “robot arm”. This enables a very accurate tracking and allows the user to perform a clicking gesture through a button on the pen. It also provides haptic or tactile feedback. Given such a device many of the problems we will encounter later on are remedied. However we want to concentrate on working with affordable, off the shelf, consumer-grade hardware.

2 DEVICE LIMITATIONS

Given that we want to work with affordable consumer-grade devices, challenges due to device limitations have to be addressed. As our primary device we use the pre-production Leap Motion controller. With this setup we can identify two major problems, a limited field of view and occlusion.

The field of view (in case of the Leap, a cone with an opening angle of approximately 90°) is especially problematic as the user can neither see it nor gets feedback upon leaving it. Given the restricted interaction space it is not feasible to use a second hand inside the view frustum (as was often suggested by users) as we simply would run out of space. In particular moving one hand will often occlude the other.

Devices that need a line of sight to the tracked hand, do not work in case of occlusion. That means, while working with two hands, one cannot be above the other. This also applies to individual fingers as can be seen in Figure 2. On the left side, the tracking will work, on the right side, with two fingers only slightly moved up and down, the tracking will fail for these two fingers. Also the whole finger has to be seen from below. As soon as the fingertips point too far downwards, tracking will fail. Touching of fingertips in a pinch gesture will also lead to failure of tracking for both fingers.

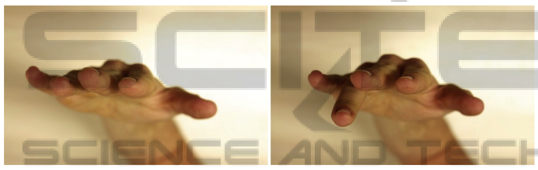
Figure 2: Using the Leap device robust hand tracking only works while every finger can be seen separately from the device (left side). The configuration on the right will lead to jittering of the middle and ring finger positions or a complete loss of the tracking.

3 SCENE INTERACTION

When first experimenting with our Leap device, we started to evaluate very simplistic three-dimensional gestures. We implemented a prototype application for camera navigation that provides the opportunity to interact with a simple 3D scene (see Figure 3).

For the camera control we wanted to achieve a flight-like interaction. Imagine your hand as an airplane. By moving this plane you control the camera. This seemingly simple task already revealed one fundamental problem. It is not obvious to determine whether the user wants to interact with the camera or merely wants to reposition the hand. To circumvent this we have to define an activation or clicking gesture. In our application this is implemented by a grab motion.

Using the Leap API, we can implement this gesture by monitoring the radius and center of a sphere that is fitted into the palm of the hand. The range of hand positions for which this sphere fitting works is quite narrow (see Figure 4). However, after a bit of training this gesture can be performed quite well.

The result of this free-flight interaction is an easy to control camera that can perform very nice and smooth pans that would not be possible by conventional input using mouse and keyboard (see video). We found that this is also very intuitive for first-time users.

Figure 3: A screenshot from our application for scene interaction.

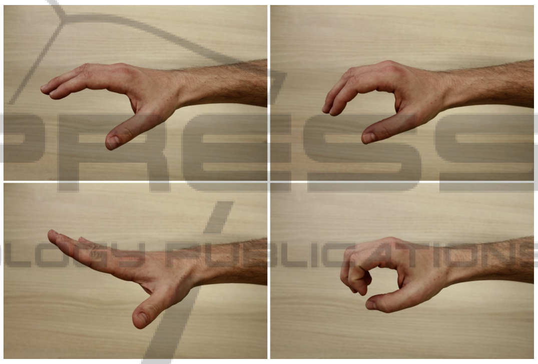
Figure 4: The grabbing feature only works in a narrow band. In the upper two images the gesture will work. For the lower two images the gesture will fail. While the left failure case could probably be resolved in software, the right failure case is more severe. Since the sensor only records the scene from a single direction, the palm of the hand is occluded by fingertips.

The second functionality implemented in this application is scene manipulation. For picking objects we tried to utilize the well known pinch motion. This gesture however is not robust using the Leap device. As the fingers come close to each other the tracking fails. Therefore, we use the "cross fingers" gesture shown in Figure 5. Having the fingers crossed (left image) corresponds to *not* having the mouse button pressed, while having separated the fingers (right im-

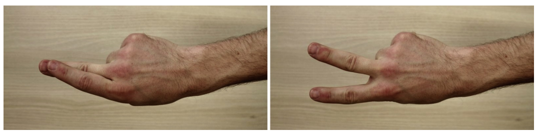
Figure 5: As a pinch gesture is not possible we substituted it with this crossing of fingers. In the left configuration the Leap will only recognize one finger. By separating the two fingers it is possible to select a model inside the scene.

age) corresponds to holding the mouse button down. While the user has his fingers crossed, we highlight the object that is closest to the position described by the two fingers. The user can then pick the highlighted object by separating the fingers and move it around. To drop the object, the user has to cross his fingers again. This is similar to drag and drop in mouse interaction.

The problem with the activation and clicking gestures is their accuracy. It is generally not possible to perform a hand movement for triggering a gesture without moving the hand. This however leads to an unwanted alteration of the camera or the scene. No matter how accurate the tracking device, accuracy of the interaction is impaired. In our opinion this is a fundamental problem of 3D input devices.

Given this rather simple application the Leap was able to perform quite well for navigating the scene. While some technical limitations became apparent it was possible to create an application that was sufficiently easy to use. However when trying to navigate to an exact position (aligning a cross in the scene with a cross on the monitor) we had problems to perform the fine grained alterations necessary to achieve a perfect alignment. This underlines the general problem of 3D interaction schemes having insufficient precision when using freehand gestures. When discussing our modeling application in the following chapter this will become more apparent.

4 MODELING

We first started with the idea of implementing a modeling application using true three dimensional input. Our goal was to model simple chess pieces, as these are widely known and easy to create. After observing the deficiencies regarding precise camera positioning, we decided to focus our analysis on accuracy.

In the resulting application, the user can build rotationally symmetric 3D models. Initially only a cylinder exists that is defined by two rings at the top and bottom. Using hand gestures only, additional rings can be added, existing rings can be moved up and down and their radius can be modified. To get a visually more pleasing result we use Catmull-Clark Subdivision on the resulting base mesh (see Figure 6).

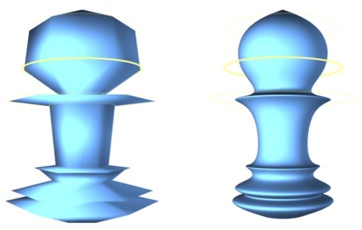
Figure 6: Modeling of rotationally symmetric models. Control mesh on the left, Catmull-Clark subdivided mesh on the right.

4.1 2D Interface

The 3D cursor can be positioned by moving the hand up and down. When the hand is moved away from the screen and penetrates a predefined plane, a new editable ring is added. To select an existing ring, the hand has to be moved closer to the screen and again penetrate a predefined plane. While a ring is selected it can be resized by moving the hand left and right and repositioned by moving the hand up and down. In this first stage we explicitely do not use real 3D interaction and therefore used the “glorified touchscreen” interface many other applications implement instead of true 3D interaction. However one problem that became apparent is the fact that users can not see or feel those planes. Therefore those users do not know the relative position of their hand and the planes, making it hard to select and deselect rings in a predictable way.

We evaluated this problem by giving the task to model a simple chess piece to a small group of users. The chess piece of our choice was the pawn shown in Figure 7(a), as this piece needs no further 3D editing and can be modeled with the application we have described so far. Each user had to model the same pawn twice, once using solely hand gestures, and once using a modified version of our modeler, where the hand gestures for creating a new, selecting an existing and deselecting the current ring were replaced by single key presses.

As we have already mentioned, participants had trouble applying the selection and deselection gestures without side effects. Especially deselecting a ring often led to unwanted alterations of the model. As a result the users became hesitant and lost confidence in achieving a good result. As soon as their model looked close enough to the template they stopped editing in anticipation of destroying the result with further editing steps.

Given the keyboard to substitute the clicking gestures, the average time per model did not change noticeably. However all users felt up to the task to add additional detail to the model (without encouragement from us). They also corrupted their model less often and no longer needed to restart the modeling process (this is the reason for the longer modeling time of User 4). The results are closer to the template model than those obtained using only hand gestures

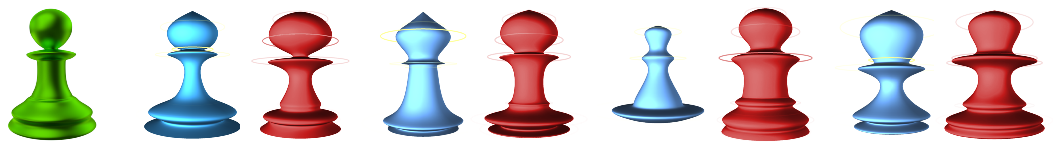

Figure 7: Some results of modeling a pawn. Users (with minor background in 3D modeling and without prior training) were given the task to model a pawn. The blue models were created using gestures for picking, the red models using keystrokes. The time-lapse session for user 2 can be found in the accompanying video. The second timings for this user show the time it took for modeling the base (comparable to the blue pawn) version and the refined version with additional detail.

as can be seen from Figure 7.

To get a better understanding why the users had problems triggering the interaction we investigated the problem: Selection and deselection is determined by penetrating a predefined, invisible plane. In order to select a ring, the user is moving his hand towards the screen. When intersecting our given plane, the software is entering the edit mode, thus giving feedback of a successful interaction. While editing the rings, the hand has to be moved while it stays behind the interaction plane. Intuitively the user will move his hand along an imaginary plane (blue plane in Figure 9), however this plane usually does not correspond with the one defined by the Leap coordinate frame (yellow plane in Figure 9). This results in the user unexpectedly changing the depth of the penetration having two observable side effects. If the user expects the tip of his finger to be well beyond the plane of interaction he might accidentally quit the editing mode. Even worse if he expects his finger to be close to the plane of interaction and wants to exit the editing mode, he will unexpectedly alter the model while having to move the hand further than expected. The problem could be avoided if the plane that the user is supposed to interact with would be sense-able.

4.2 3D Interface

Since most chess figures are not rotationally symmetric, we extended our application for true three dimensional vertex manipulation. Users can pick vertices using the “cross fingers” gesture shown in Figure 5. While a vertex is selected, the user can set its true 3D position by moving the hand. We use this extension to model the crown of the queen chess piece. The users have to extrude the prongs of the crown by picking a vertex of the base mesh and moving it perpendicular to the surrounding surface. We found that there are some problems with this seemingly intuitive modeling approach. The first one is the inaccuracy with picking we discussed in Section 3. The second one is the poor depth perception that makes an accurate movement along a three dimensional line very hard.

5 RELATIVE LOCALIZATION IN 3D

Having been surprised by the fundamental problems we ran into we wanted to investigate these issues further. For this purpose we implemented a simple voxel carving application with two different stencils. A sphere representing the tip of a pen and a box as a very crude representation of a hand.

Our first aim was to evaluate the lack of depth perception we described in Section 4.2. We asked some users to carve a music note with a penetration of constant depth into one side of the voxel cube using the spherical carving tool. In order to also evaluate the problem with the tilted imaginary plane we described in Section 4.1, we rotated the voxel cube such that its vertical sides are not parallel to the screen aligned coordinate planes. This means correct movement in all three dimensions is necessary to keep a constant penetration depth.

With just the sphere as visual cue for the position of the tool it was virtually impossible for the users to correctly gauge the relative depth of the sphere to the voxel volume (see 8). This underlines the negative impact of a lacking depth perception. To gauge how much help is needed to get a convenient user interface we implemented different levels of assistance that were given to the users.

The first and most obvious additional clue is visualizing the depth level of the sphere by rendering a plane through its center parallel to the screen. This way the user can see at which depth the sphere will hit the voxel cube before it touches (please refer to the accompanying video for more details). While this helps, it is still hard to perceive an accurate depth or know exactly when the sphere starts intersecting the volume and even more to gauge the depth of the actual penetration.

In the real world we have tactile feedback and resistance from a surface to help us carving at a constant depth. While tactile feedback can be provided by haptic devices like the Phantom (Massie and Salisbury, 1994), we wanted to concentrate one less ob-

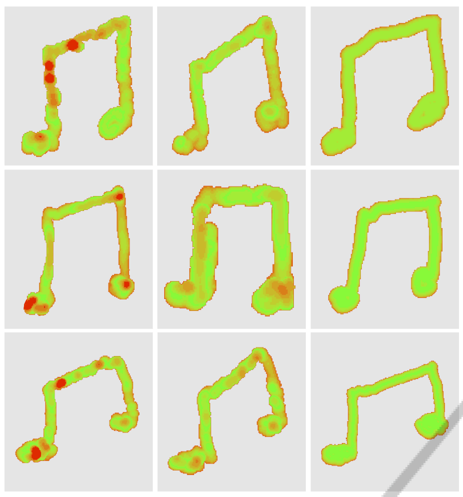

Figure 8: Results of our carving experiment. One user per row. From left to right: only visual feedback, audio feedback and snapping. Results are color coded by carving depth.

trusive, low-cost consumer devices mentioned in the introduction. As a work-around, we use acoustic instead of tactile feedback. We output sound similar to park distance control in modern cars when the tool is outside the voxel volume. The interval between beeping sounds becomes shorter as the tool approaches the volume until a continuous sound is played upon contact. A second form of acoustic feedback is provided for the penetration depth of the carving tool inside the volume. This is encoded by changing the pitch of the continuous sound.

When comparing the results using visual and acoustic feedback, it can easily be seen that all users had severe problems carving at a constant depth with visual hints only (see Figure 8, red color marks areas where the carving depth does not conform to the target depth). When provided with acoustic feedback, the users had less problems, however, the strokes are still shaky. Given the many degrees of freedom, it is difficult to simultaneously concentrate on penetration depth and stroke direction.

Since the results were still not satisfying, we implemented a snapping aid constraining the users movement to one side of the voxel cube. This concept is very similar to the modeling aids provided in current modeling tools that use mouse and keyboard as interaction devices. Our snapping aid restricts movement of the carving tool onto a plane aligned with the axis of the voxel volume. Therefore, we only consider the user interaction in x- and y- direction and discard the hand movement in z-direction. Please note that this is also a reduction of the dimensionality of the problem as only two out of three available degrees of freedom are used as input and should (in theory) correspond to well known mouse interaction semantics.

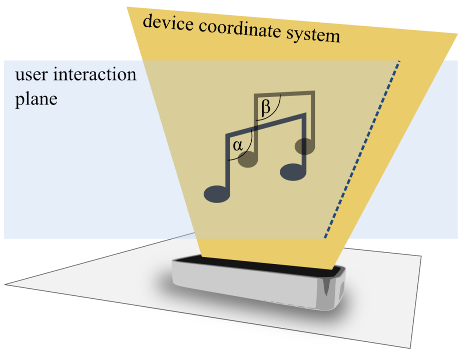

Figure 9: Restricting the user interaction to a 2D plane causes new problems: Since the plane in which the user interacts will hardly ever coincide with the coordinate system of the input device, projecting the user input onto this plane will distort it.

When the carving tool was snapped to one face of the cube, the users became much more fluent in their interaction, which led to notable better results. The video accompanying this paper shows how the interaction gets faster and less shaky. However the resulting notes still show some shakiness with the reduced degree of freedom, leading to the suggestion that those shaky paths are not caused by the additional degrees of freedom the Leap introduces.

When recording user gestures with a 3D input device, we expect the user to move on a predefined plane (yellow plane in Figure 9). The user intuitively will penetrate the plane with a certain depth and move along an imaginary plane (see Section 4.1). As the predefined plane can neither be seen nor felt, it is very likely that the plane imagined by the user (blue plane in Figure 9) does not correspond to the expected one. Given this configuration the user input will be distorted by a projection of the shape drawn on the imagined plane on to the expected one. For example, as depicted in Figure 9, the slope of a line drawn by the user will not be as expected (compare angles α and β). This effect is the reason for shaky results (see third column in Figure 8, the first bumps on the left side of the horizontal line), as the user is adjusting the slope while drawing. Vertical lines are not affected by this projection as the cube is only rotated along the vertical axis. This means the slope of the expected vertical line is identical to the carved one.

The problem could be avoided if the plane that the user interacts with would be known to the system, as the projection causing the observed discrepancy in slopes is neutralized. Unfortunately, this plane cannot be predicted for two reasons: First, the user most likely does not even interact on a plane but rather on a curved surface. Second and more importantly, to predict the plane from the user interaction, we would have to gather several frames to analyze the motion. However, this contradicts the real-time feedback that is required for interactive 3D modeling.

In order to increase depth perception in general, we used a 3D-TV. We found that the additional depth cue provided a level of assistance comparable to the acoustic feedback. However, a problem of this setup is that the coordinate system where the virtual scene is seen and the coordinate system where the hands move in still do not coincide. This causes another indirection which counteracts an intuitive interface and is confusing for novice users. The introduction of future virtual reality glasses with large view frustums like for example the Occulus Rift could potentially help solving this problem. This headset would allow us to merge the coordinate systems of the hand and the scene (assuming a stable tracking).

6 CONCLUSIONS

In this paper, we have examined the viability of real 3D interaction using recent consumer hardware, especially the Leap Motion controller. In our tests it became apparent that there are three fundamental problems. The lack of a clicking gesture, bad depth perception with standard displays and the discrepancy between the plane of interaction assumed by the software and the user.

As it turns out the precision of true 3D interaction is not limited by the resolution of the input device. In fact the Leap provides enough temporal and spatial resolution to support all our modeling needs. The imprecision in the modeling procedure is inflicted by the misalignment of the imagined and the actual coordinate system, as we showed in Section 5.

When interacting with a mouse the hand is stabelized by the table top. Using 3D interaction, it has to float above the table forcing the muscles to retain the position. This is another source of imprecision as the hand will move unintentionally.

In conclusion we think that the viability of true three dimensional interaction is not limited by hardware at the current point of time. The limiting factor rather is the software processing and interpreting the input.

ACKNOWLEDGEMENTS

We would like to thank the BlendSwap community for the models we used in our scene interaction application. The Space Marines were created by *hjmediastudios*, the House by *julsengt*.

REFERENCES

Araujo, B. R. D., Casiez, G., Jorge, J. A., and Hachet, M. (2013). Mockup builder: 3d modeling on and above the surface. *Computers and Graphics*.

Argelaguet, F. and Andujar, C. (2013). A survey of 3d object selection techniques for virtual environments. *Computers and Graphics*.

Gallo, L., Placitelli, A. P., and Ciampi, M. (2011). Controller-free exploration of medical image data: Experiencing the kinect. In *Computer-Based Medical Systems (CBMS), 2011 24th International Symposium on*.

Hilliges, O., Izadi, S., Wilson, A. D., Hodges, S., Garcia-Mendoza, A., and Butz, A. (2009). Interactions in the air: adding further depth to interactive tabletops. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09.

Intel Corporation (2013). Available from: `http://software.intel.com/en-us/vcsource/tools/perceptual-computing-sdk`.

LaViola, J. J. and Keefe, D. F. (2011). 3d spatial interaction: applications for art, design, and science. In *ACM SIGGRAPH 2011 Courses*, SIGGRAPH '11.

Leap Motion, Inc. (2012). Available from: `http://leapmotion.com`.

Massie, T. H. and Salisbury, J. K. (1994). The phantom haptic interface: A device for probing virtual objects. In *Proceedings of the ASME winter annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems*.

Microsoft Corporation (2010). Available from: `http://www.microsoft.com/en-us/kinectforwindows/`.

Ren, G. and O'Neill, E. (2013). 3d selection with freehand gesture. *Computers and Graphics*.

Ren, Z., Yuan, J., and Zhang, Z. (2011). Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. In *Proceedings of the 19th ACM international conference on Multimedia*.

Saggio, G., Bocchetti, S., Pinto, C., and Orengo, G. (2010). Wireless data glove system developed for hmi. In *Applied Sciences in Biomedical and Communication Technologies (ISABEL), 2010 3rd International Symposium on*.