Deep Window – 3D virtual camera control with a tablet and depth camera

Drew Fisher
University of California, Berkeley
University Ave.
Berkeley, CA, USA
dfisher@eecs.berkeley.edu

Björn Hartmann
University of California, Berkeley
University Ave.
Berkeley, CA, USA
bjoern@eecs.berkeley.edu

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.
ACM 978-1-4503-0268-5/11/05.

Abstract
This paper describes Deep Window, an implementation of a mobile window into a virtual world using a depth camera. Deep Window explores the use of 2.5D depth cameras to implement 3D interaction techniques. Users can move and orient a tablet to change their perspective of a 3D scene, showing that it is possible for a 3D interaction technique to be replicated with a depth camera. A user study comparing this interaction technique to standard mouse techniques for positioning a camera in a 3D scene turns out inconclusive.

Keywords
Virtual camera, depth camera, interfaces

ACM Classification Keywords
I.3.6 Methodology and Techniques – Interaction techniques. H.5.2 User Interfaces – theory and methods. D.2.2 Design Tools and Techniques – User interfaces.

General Terms
Depth camera – a camera which outputs an image in which every pixel has an associated distance value, representing distance from the camera.

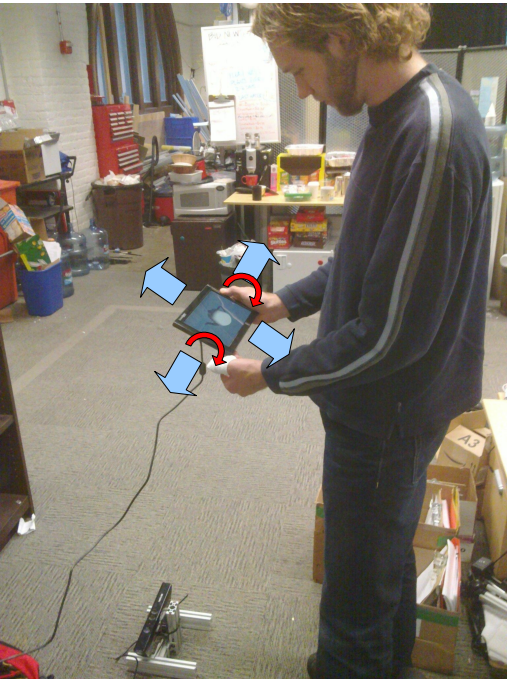

Figure 1: Interacting with Deep Window. The user holds a tablet, and the depth camera on the floor tracks said tablet. By moving the tablet left and right, the view shifts; by moving about the depth camera's axis, the view is rotated; by moving the tablet closer to the camera, the view is zoomed.

Introduction

Navigating 3D scenes is increasingly important as more of our movies, products, and designs are prototyped or produced using computer-aided design or 3D modeling and animation software.

A variety of interfaces for interaction with these software packages have been proposed, but most require elaborate hardware setups like VR helmets, mechanical linkages, or motion capture hardware. These setups restrict which environments and social situations the interfaces can be used in; their price further restricts the kind of tasks they are used for and their general applicability.

Depth cameras have recently become widely available. They are mobile, unobtrusive, and economical. While some work has been done on possible interaction techniques using depth cameras [1, 8], it remains to be seen what other rich 3D interaction techniques can be supported with this technology.

This paper investigates a particular 3D interaction technique – use of a spatially-aware display to explore a 3D scene as though the display were a window into a virtual world. This technique was demonstrated by the Boom Chameleon [7], but required the tablet be mounted to a boom. Movie producers have used tablets in conjunction with expensive motion-capture setups to produce camera paths in CGI scenes. We investigate if such tasks can be accomplished with less information about the user environment (thus widening their applicability) and less instrumentation (thus expanding their availability).

Related Work

Andy Wilson demonstrated closing the loop between sensing with depth cameras and interactions in his MicroMotorCross [8]. Benko and Wilson's DepthTouch [1] demonstrated the use of depth cameras to implement 3D gestures with static screens. While these laid important foundational work with respect to interacting with depth cameras, both of these projects made the assumption of a static screen.

Chris Harrison's Minput [4] and George Fitzmaurice's works on situated information spaces [3] demonstrate the use of small, motion-aware displays for creating rich interactions. The Boom Chameleon [7] is a prime example of using a spatially-aware display to provide a high level of interactivity, but requires a somewhat elaborate hardware setup.

Deep Window explores the intersection of depth cameras and spatially-aware displays, demonstrating that real 3D interaction techniques can be leveraged with 2.5D information from an inconspicuous, simple setup.

Interaction Technique

A depth camera sits on the floor, pointing upward. The user approaches the camera holding a tablet screen which shows a view of a 3D scene. As the user enters and moves about the field of view, the camera tracks the location and orientation of the tablet. By moving the tablet left and right in physical space, the user controls the virtual camera accordingly. Similarly, changing the orientation of the tablet in-place changes the direction of the virtual camera's viewing direction. Moving the tablet closer to the depth camera allows the user to zoom in on the target point of the 3D scene.

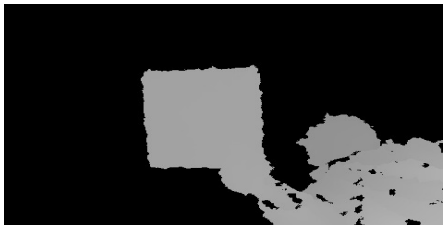

Figure 2: Thresholded depth image, drawn in greyscale. This is a person, viewed from below, holding a tablet in his right hand.

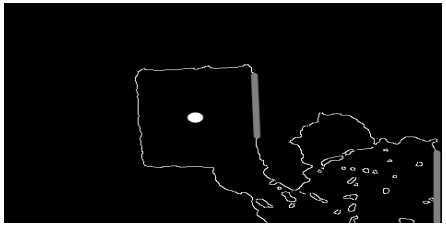

Figure 3: Edge image. The white circle indicates the tracking point. Straight lines from the Hough transform are shown as a thick grey line – note that this particular frame does not have enough Hough lines to acquire the tracking point, but is nonetheless adequate for tracking.

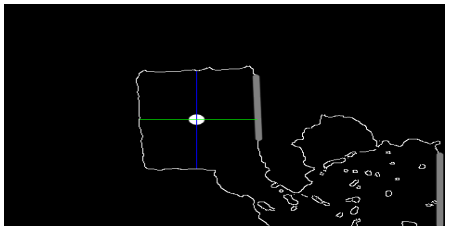

Figure 4: From the tracked (x,y) position, find edges above, below, and to the sides. The new (x,y) tracked point is the centroid of these four points. For objects with smooth borders, this operation converges to a point over time.

By pressing a button, or touching the touchscreen, the user can “clutch” the controls, disabling virtual camera position updates until the button or touch is released. Upon release of the button, the current position of the tablet is mapped to the current position of the virtual camera, and further interactions will be relative to this matching point. This is similar in functionality to lifting a mouse off the desk to match the current point on screen to a different point on the desk. The clutch functionality allows the user to explore the scene from all directions, even though the system cannot directly access all possible camera positions.

System Implementation

Deep Window has two important pieces of hardware: the depth camera, and the tablet monitor. The depth camera is used to track the tablet monitor, and the tablet is used to show the user the scene from a particular angle.

Due to the lack of current availability of a tablet platform with both a USB host and enough processing power to achieve interactive framerates running both computer vision and 3D graphics, we approximated this anticipated future setup with a 9” USB monitor from DoubleSight Displays connected to a laptop running Fedora Linux. The USB cable was small enough so as to be effectively invisible to the depth camera.

Image Acquisition

We used a Microsoft Kinect with the libfreenect drivers [6] to stream 11-bit 640x480 depth images at 30FPS.

Image Processing

The ultimate goal of the image processing is to extract from the image an effective (x,y,z) position of the tablet along with the tablet's normal vector.

From the raw depth image, we remove all pixels with distance values greater than 8 feet, assuming that nobody would be able to hold the tablet higher than that. The Kinect also has an intrinsic near-plane cutoff at about 3 feet from the device. The result is the image in Figure 2.

This thresholded image is then passed into the OpenCV toolkit [5], where we run the Canny edge detector [2] on the image. From there, we try to determine where the tablet is located in the image. We take an acquire-track approach – acquisition of the target looks for between three and five straight line segments at approximately right angles to each other using a Hough transform, then takes the centroid of the lines' endpoints as a point inside the tablet (Figure 3). For tracking (Figure 4), we find (from the previously tracked point) the first edge pixel to the left and right of the point in the same row, as well as above and below the point in the same column. If the distance between left and right edge is too great, or the distance between top and bottom edge is too great, or if there is no depth data at the previously-tracked point, the system assumes it has lost track of the tablet and reenters acquisition mode. Otherwise, it sets the new tracked point as the centroid of the four edge points it found, and proceeds from there.

Given the (x,y) position of the approximate center of the tablet, the z position can be taken straight from the value in the depth buffer for that particular pixel. By

calculating the gradient of the depth buffer at that (x,y) position, we can determine the normal vector's angle in the plane of the camera's vision. Finally, we can compute the z value of the normal vector with the magnitude of the slope in the direction of the gradient vector. This gives us our camera position and orientation – by letting the Kinect be a reference point in the scene, we can produce an “up” vector for the camera, which fully specifies the camera position in the scene.

Limitations

This technique has some noteworthy limitations: since the camera has difficulty seeing the tablet when held “feathered” to the depth camera, it cannot track the tablet's position or angle effectively in these orientations. Further, since the camera cannot tell the difference between the front of the tablet and the back, it is restricted to half of all possible viewing angles – the top hemisphere of a sphere. Additionally, the tablet has no knowledge of whether the tablet is rotated about its normal vector, so the camera cannot roll. Fortunately, for exploring 3D models (which usually have an implied orientation), this is not a terribly limiting constraint.

Perhaps the most limiting factor is that the tablet must be within the Kinect camera's field of view, which only spans about a 5 feet by 4 feet rectangle at a distance of 5 feet. As a result, this technique appears to be satisfactory for examining objects where the virtual camera is placed outside the object, but less well suited for exploring spaces where the camera is placed within the space.

Based on preliminary user feedback, our current implementation of object tracking is not sufficiently robust to provide for an adequate user interaction experience. As a result, we have postponed our user study until Deep Window hits a minimum threshold of polish. Nonetheless, we will discuss our proposed study in the following section.

Proposed Evaluation

One key interaction when using 3D modeling software is repositioning the camera to look at the model from a different orientation. To this end, we have designed a study that will investigate the effectiveness of Deep Window at exploring 3D models. We will test the following hypotheses:

Hypothesis 1: test subjects will be faster using Deep Window than with a comparable mouse and keyboard interface for 3D camera positioning tasks.

Hypothesis 2: test subjects will more accurately reproduce 3D camera views using Deep Window than with a comparable mouse and keyboard interface.

Method

We will take this system and load 3D models of several easily-recognizable objects for users to view. We will also prepare screenshots of a desired target image. Test subjects in the within-participants study will be tasked with manipulating the camera to reproduce the view shown in a screenshot. In one test condition, users will manipulate the camera by dragging the mouse; in another, users would use Deep Window to move the camera in the scene. By instrumenting the system, we can collect information on both the amount of time each user takes to produce the screenshot, as well as the camera parameters of the image eventually

produced. Participants will also be survey as to which interaction technique they preferred, and why.

Conclusion

This paper presents Deep Window, a spatial interaction technique that uses a mobile tablet in combination with a depth camera to implement more natural interactions with 3D virtual scenes.

Future work will explore the feasibility of other 3D and embodied interaction techniques. We also would like to compare Deep Window to the other natural user interfaces, such as similar efforts implemented with motion-capture devices or augmented reality VR goggles, to determine if there are situations in which the use of one implementation technique consistently outperforms others.

Acknowledgements

I thank Björn Hartmann for providing guidance and oversight, as well as for funding all the hardware involved.

References

[1] Benko, H and A. Wilson. "DepthTouch: Using Depth-Sensing Camera to Enable Freehand Interactions On and Above the Interactive Surface." Microsoft Research Technical Report MSR-TR-2009-23. March, 2009.

[2] Canny, J. "A computational approach to edge detection." Readings in computer vision: issues, problems, principles, and paradigms, 1987. pp 184-203.

[3] Fitzmaurice, G. "Situated information spaces and spatially aware palmtop computers." Communications of the ACM, Special issue on Augmented Reality and UbiComp, July 1993, 36(7), p.38-49.

[4] Harrison, C. and Hudson, S. E. 2010. "Minput: Enabling Interaction on Small Mobile Devices with High-Precision, Low-Cost, Multipoint Optical Tracking." In Proceedings of CHI 2010. ACM, New York, NY.

[5] OpenCV Wiki. http://opencv.willowgarage.com/wiki/

[6] OpenKinect Wiki. http://openkinect.org/

[7] Tsang, M, Fitzmaurice, G., Kurtenbach, G., Khan, A., and Buxton, B. "Boom chameleon: simultaneous capture of 3D viewpoint, voice and gesture annotations on a spatially-aware display." In Proceedings of SIGGRAPH 2003.

[8] Wilson, A. D. "Depth-Sensing Video Cameras for 3D Tangible Tabletop Interaction." TABLETOP 2007. pp 201-204.