Depth-Sensing Video Cameras for 3D Tangible Tabletop Interaction

Andrew D. Wilson
Microsoft Research
Redmond, WA
awilson@microsoft.com

Abstract

Recently developed depth-sensing video camera technologies provide precise per-pixel range data in addition to color video. Such cameras will find application in robotics and vision-based human computer interaction scenarios such as games and gesture input systems. We present an interactive tabletop system which uses a depth-sensing camera to build a height map of the objects on the table surface. This height map is used in a driving simulation game that allows players to drive a virtual car over real objects placed on the table. Players can use folded bits of paper, for example, to lay out a course of ramps and other obstacles. A projector displays the position of the car on the surface, such that when the car is driven over a ramp, for example, it jumps appropriately. A second display shows a synthetic graphical view of the entire surface, or a traditional arcade view from behind the car. Micromotorcross *is a fun initial investigation into the applicability of depth-sensing cameras to tabletop interfaces. We present details on its implementation, and speculate on how this technology will enable new tabletop interactions.*

1. Introduction

Tabletops distinguish themselves from other surfaces in the everyday world by their ability to support objects placed on them. This property lends tabletops to a wide variety of complex, productive tasks involving physical objects. A garage workbench, for example, is often the site of intricate assembly tasks, while architects still build physical models of buildings and landscapes on horizontal surfaces. Accordingly, many interactive tabletop systems include some ability to sense and use physical objects placed on them. Often these capabilities are based on capacitive sensing, RFID, active infrared emitting devices, generic computer vision-based object recognition, or visual barcode recognition.

However, while these objects live in a 3D world, interactions with them on interactive surfaces are typically 2D in nature. For example, a game piece placed on the surface would typically indicate the 2D position of a player on a board game. Beyond tabletop interfaces, the field of augmented reality offers sophisticated techniques to reason about 3D objects, but these efforts are often based on manipulating some aspect of the physical object, or are primarily concerned with determining camera position.

A few interactive tabletop systems deal with 3D objects more generally. Illuminating Clay [2], for example, uses a laser scanner, while SandScape[1] uses infrared illumination, an infrared camera, and translucent beads to deduce a height map of the surface. The Perceptive Workbench [3] combines views of objects placed on its surface to perform 3D reconstruction. Meanwhile, 3D gesture interfaces over tabletops are typically based on augmented reality or VR techniques such as magnetic trackers or other motion capture techniques.

In this paper we propose the use of recently developed depth-sensing video cameras to support a wide range of 3D interactions on interactive tabletops. This new camera technology enables the real-time 3D capture of everyday objects placed on the surface for example. We present an early demonstration system which uses the calculated height map in a driving simulation that allows the player to drive a virtual car over real objects placed on the table.

2. Depth-sensing cameras

We refer to camera systems which recover depth information throughout the captured scene (i.e., depth per pixel) as *depth-sensing.* While we acknowledge the utility of other camera-based means of 3D capture for interactive surfaces, such as recognizing an object from

[1] http://tangible.media.mit.edu/projects/sandscape

two views [4], a fully detailed range image permits a great deal of flexibility.

Laser scanners have been used in robotics and other fields to calculate accurate depth images. Despite being available for many years, such technology is still expensive, and often is barely fast enough for interactive applications (Illuminating Clay reports 3Hz scan rate).

Correlation-based stereo is another old approach which suffers from a number of difficulties. For example, stereo matching typically fails on regions of the scene with little or no texture. Secondly, even today stereo matching requires a great deal of computational power to obtain interactive rates at reasonable resolution. Finally, stereo camera setups typically require fine calibration.

New camera technologies under development observe depth information in a more direct fashion, and so address many of the drawbacks of previous approaches.

In the present work we use the ZSense camera by 3DV Systems, Ltd [1]. The ZSense times the return of pulsed infrared light: reflected light from nearer objects will arrive sooner. A Gallium-Arsenide (GaAs) solid-state shutter makes this possible (see Figure 1). The result is an 8 bit depth image, over a variable dynamic range (70cm to 3m). Figure 1d illustrates an example ZSense depth image. The ZSense camera also includes a separate color camera. The output of the color camera is registered with the depth image to obtain a complete "RGBZ" image at 30Hz.

3. Micromotocross

3.1 Motivation

To explore the application of depth-sensing cameras to interactive tabletops, we built a projection-vision system which uses the camera to recover the height map of the table surface and the objects placed on it. Graphics are then projected on the same surface.

We are interested in exploring the range of interactions made possible by this configuration. To begin, we implemented a simple driving simulation using the XNA game development platform. Players drive a virtual dune buggy around and over any objects placed on the table surface. The players can arrange folded bits of construction paper, for example, to lay out a course of ramps and other obstacles. A projector is used to display the position of the car on the surface, such that when the car drives over a ramp, for example, the car jumps appropriately. Players control their cars with a wireless Xbox 360 controller.

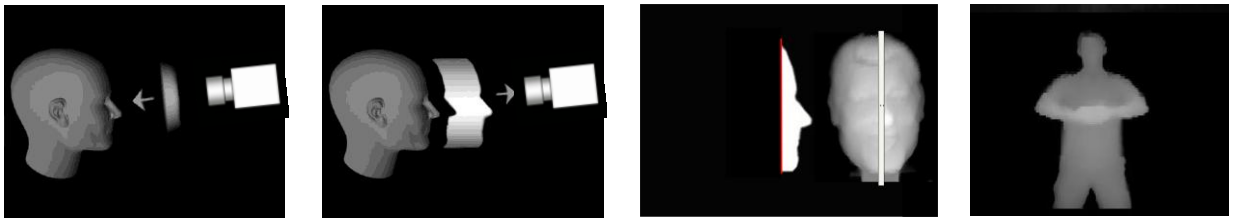
3.2 Terrain model

The 320x240 depth image is returned by the camera at a frame rate of 30Hz. The camera is configured to place its depth-sensing dynamic range at the height of the table and about 70cm above the table.

Because the focal length of the camera is known, it is possible to calculate the 3D position (in centimeters, for example) indicated at each pixel in the depth image. It is straightforward to then construct a vertex buffer for this height map, and it is similarly easy to texture map this mesh with the color image also returned by the camera.

In the present implementation, instead of converting the mesh to world coordinates as suggested above, the depth image is normalized between a minimum depth image collected offline when the surface is clear of objects, and a maximum depth image, collected when the surface is raised about 20cm.

The ZSense depth image is somewhat noisy. For some applications it will be necessary to smooth the image to mitigate the effects of shot noise. In Micromotocross, where every bump due to noise is potentially a huge pothole, we found it necessary to use a mixture of spatial and temporal smoothing. This has the effect of adding a significant delay from when an object on the surface is moved, to when this change is reflected in the modeled terrain.



a. b. c. d.

Figure 1. 3DV ZSense uses pulsed infrared light (a) and solid state shutter to compute depth image. Reflected light from closer surfaces arrives sooner (b). Fast shutter truncates light (c) while imaging sensor integrates more light from closer surfaces. A second captured image is used to normalize for differences in reflectance. Example depth image (d). Illustrations adapted from [1].

3.3 Projection-vision system

In order to project graphics that are correctly registered with the sensed image, it is necessary to model the coordinate transform that relates a position in the sensed image to a position in the projected image. In the common case of a 2D camera and projection on a 2D surface, this can be handled using a projective transform and a simple calibration procedure which maps the four corners of the table surface in the camera view to display coordinates [5].

When using a 3D camera to sense objects and various depths, projected graphics need further correction such that, for example, the virtual dune buggy appears the same size whether it is on the table surface or on some object of significant height placed on the table surface.

We use a simple extension of the 2D case which allows an easy two step calibration process. Offline, the calibration of the 2D projective transform mapping image coordinates to display coordinates is run twice: first when the table is clear, and second with the table raised about 20cm from its normal height. In each calibration, the user clicks on 4 projected points in the color video image. This yields two 2D projective transforms, one for each table height.

The display coordinates of any point in the normalized depth image can then be determined by interpolating between the results of both projective transforms, at an amount given by the depth value. It is convenient to incorporate these calculations in the graphics vertex shader, replacing the usual graphics camera projection, so that the application developer need not be aware of such details when constructing the graphics scene, and so that this transformation imparts no overhead to the system. Figure 2 illustrates the tabletop with projection and extracted terrain.

In addition to the table projection display, the Micromotocross system includes a secondary LCD display which shows a purely synthetic overview of the scene. In this view, the terrain is textured with the color image. Players can switch to a conventional arcade view, in which the graphics camera is placed just behind the car (Figure 3). A player can toggle this view by hitting a button on the controller. The second player can switch the display back to the overview by hitting the same button on their controller.

Figure 3. Traditional arcade view for blue player shows other player and ramp ahead.

3.4 Physics model

Two players can control their dune buggies using standard driving controls on wireless Xbox 360 controllers. The car's movement is calculated by the Newton physics library, which includes a detailed vehicle dynamics model of acceleration, steering, tire friction effects and chassis rigid body dynamics. Figure 2 shows a car airborne after a jump, for example.

Most physics libraries include facilities for collision detection that are specialized to efficiently handle many separate rigid bodies. Typically the geometry of each object is constrained to be static, so that various structures such as convex hulls and BSP trees can be pre-computed to speed collision detection. In Micromotocross, however, these techniques are inappropriate when detecting the collision of the cars

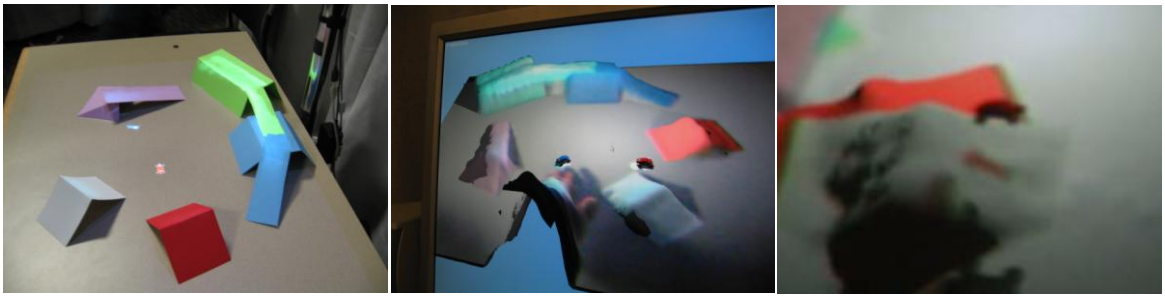


Figure 2. Micromotocross demonstration. Left: Tabletop with paper ramps and obstacles, and top-down projected cars. Middle: Synthetic (graphics) overview of tabletop, showing extracted terrain model, two cars, synthetic shadows, and user's hand in the scene. Right: In the synthetic view, a car is airborne after taking a jump.

with the dynamically changing height map. Fortunately, because the height field is a regular grid of points, determining which parts of the mesh are potentially in contact with the car's tires, for example, is a simple lookup operation.

Unfortunately, most available physics libraries do not support dynamically changing meshes in the calculation of collision response. Fast changes in the mesh can result in unpredictable dynamics. If the player moves an object quickly against the virtual car, the response of the car may not look convincing, or may even penetrate the terrain mesh.

Finally, it is important to note that a height map-based terrain model is indifferent to the motion of distinct objects as it relates to the simulation of friction. For example, if the car is sitting on a physical object that the user then moves across the table, the car will not stay on the object (even if its brakes are on). An appropriate analogy may be moving one's hand under a bed sheet, with objects sitting on the sheet: the objects are likely to stay in place.

3.5 User experiences

Micromotocross has been experienced by hundreds of people of a wide variety of ages and backgrounds. People have fun manipulating objects on the table, setting up ramps and bridges from one object to another. Many are impressed by the magical quality of the interaction.

Some users prefer to look at the synthetic view (overview or behind-car), while others prefer to look at the table projection. Some may prefer the synthetic view because it is easier to see the car as it goes through the air over a jump, or because it has higher resolution.

Many people are amused to find that the system will incorporate their hands into the scene when they are placed on the table. Some try to drive their buggy over their friend's hand, or even up their arm, while a few try to hold the car in the palm of their hand to move it (if the hand cupped and moved slowly, this is often successful). Children often will have fun knocking the car driven by their friend off the table, but are annoyed when this happens when it is their turn to drive.

4. Further interactions

It is interesting to watch people attempt to interact with the virtual cars directly with their hands. Once they realize that the system sees their hands, and that the virtual objects tend to react in somewhat appropriate ways to real physical objects on the table, many people try to pick up and move the car. These observations suggest that users would be able to use a gesture-based interface to manipulate virtual objects, just as they would manipulate real objects on the table.

In fact, it has been our goal all along to explore the use of gestures in the 3D space on and above the table surface. The depth image will make certain gestures potentially easier to recognize, and the 3D information should ease interactions with the physical objects.

The current prototype supports an initial implementation of a pinching gesture that may be suitable for picking up objects (see [6]). Presently, this gesture must be done well above the surface of the table, and instead of grasping a virtual object, the gesture creates an obstacle (a heavy block) which is dropped into the scene.

We envision the ability to pick up a virtual object with this grasping gesture, place it on a physical object sitting on the table, or even hold the virtual object in the palm of the hand. So that such interactions behave in the way that users expect, it may be desirable to drive such interactions from the physics engine (in contrast to recognizing the hand as a special object, for example). Such an approach will require significant upgrades to the physics simulation's support of dynamic meshes.

4. Acknowledgements

Thanks to 3DV Systems, Ltd., for providing the ZSense prototype which made this work possible.

5. References

[1] G. J. Iddan and G. Yahav, "3D Imaging in the Studio," *SPIE*, vol. 4298, pp. 48, 2001.

[2] B. Piper, C. Ratti, and H. Ishii, "Illuminating Clay: A 3-D Tangible Interface for Landscape Analysis," presented at CHI 2002 Conference on Human Factors in Computing Systems, 2002.

[3] T. Starner, B. Leibe, D. Minnen, T. Westeyn, A. Hurst, and J. Weeks, "The perceptive workbench: Computer-vision-based gesture tracking, object tracking, and 3D reconstruction for augmented desks," *Machine Vision and Applications*, vol. 14, pp. 51-71, 2003.

[4] A. Wilson, "TouchLight: An Imaging Touch Screen and Display for Gesture-Based Interaction," presented at International Conference on Multimodal Interfaces, 2004.

[5] A. Wilson, "PlayAnywhere: A Compact Tabletop Computer Vision System," presented at Symposium on User Interface Software and Technology (UIST), 2005.

[6] A. Wilson, "Robust computer vision-based detection of pinching for one and two-handed gesture input," presented at UIST, 2006.