Steerable Augmented Reality with the Beamatron

Andrew D. Wilson, Hrvoje Benko
Microsoft Research
One Microsoft Way
Redmond, WA 98052
awilson, benko@microsoft.com

Shahram Izadi, Otmar Hilliges
Microsoft Research Cambridge
7 JJ Thomson Avenue
Cambridge, CB3 0FB
shahrami, otmarh@microsoft.com

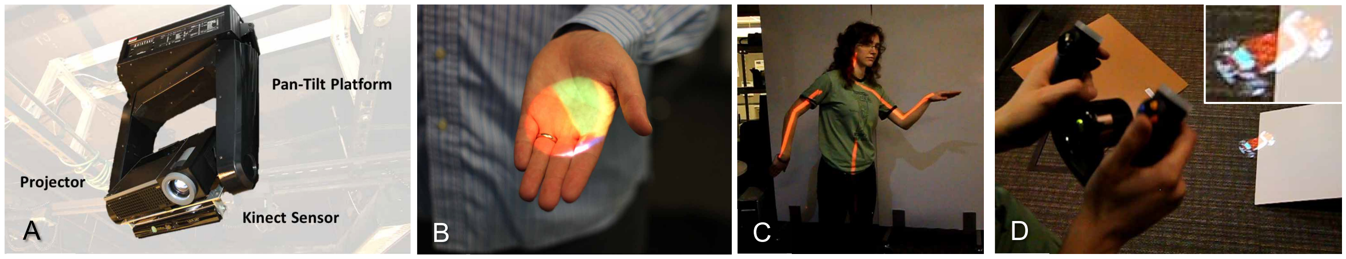

Figure 1. (A) The Beamatron mounts a projector and Kinect sensor to a pan and tilt motorized platform. (B) The Beamatron can project graphics in the user's hand, and follow the user as they move about the room. (C) Skeletal tracking results superimposed on a user. (D) Driving a projected car on real objects in the room with a wireless wheel controller.

ABSTRACT

Steerable displays use a motorized platform to orient a projector to display graphics at any point in the room. Often a camera is included to recognize markers and other objects, as well as user gestures in the display volume. Such systems can be used to superimpose graphics onto the real world, and so are useful in a number of augmented reality and ubiquitous computing scenarios. We contribute the Beamatron, which advances steerable displays by drawing on recent progress in depth camera-based interactions. The Beamatron consists of a computer-controlled pan and tilt platform on which is mounted a projector and Microsoft Kinect sensor. While much previous work with steerable displays deals primarily with projecting corrected graphics onto a discrete set of static planes, we describe computational techniques that enable reasoning in 3D using live depth data. We show two example applications that are enabled by the unique capabilities of the Beamatron: an augmented reality game in which a player can drive a virtual toy car around a room, and a ubiquitous computing demo that uses speech and gesture to move projected graphics throughout the room.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Human Factors

Keywords: Steerable displays, augmented reality, ubiquitous computing, depth cameras

INTRODUCTION

Augmented reality generally refers to viewing the real, physical world with superimposed computer-generated graphics. Today this usually involves rendering graphics onto a live video of the physical world, most commonly on a mobile device such as a smartphone. Graphics are updated continuously to follow the orientation of the device video camera, such that the graphics appear to be inserted into the real world. This approach has the drawback that rather than look at the real world, the user must instead look at a small video feed of it. Near-to-eye displays address this shortcoming, allowing the user to view the real world with superimposed graphics directly, but require wearing exotic gear [4,16].

An alternative approach to achieve direct viewing is to use projectors to render graphics onto the real world directly. *Spatial augmented reality* [4] entails correcting the projected graphics to account for the shape of the projection surface, possibly assuming the user's viewpoint (see [30] for a particularly compelling example). Whereas the effective field of view of a smartphone or near-to-eye display AR system is potentially unlimited as the user moves their head or their device, a projected spatial augmented reality system is limited by the field of view of the projector. This can be overcome somewhat by using multiple projectors and cameras, at the expense of the overall complexity of the system [35]. A handful of research prototypes have explored using a motorized platform to reorient a single projector and camera to view arbitrary locations throughout a room [22, 8, 6, 11]. Such *steerable display* systems trade the shortcomings of the multiple projector and camera approach for the problem of selecting the most appropriate orientation of the projector and camera at each moment.

This paper builds on previous examples of steerable displays by replacing the usual video camera with a depth camera, and by developing algorithms and techniques which allow precise rendering of graphics that appear to be correctly inserted into the real world from the user's point of view. Whereas much of the previous work on steerable displays exploits explicit calibration procedures against a set of discrete static planar surfaces of the real world [22], the real-time update of precise 3D shape information from the depth camera allows interactions with possibly moving objects (including the user's body). This capability allows certain interactions new to steerable displays, such as holding a virtual (projected) object in the hand while walking around the room.

We contribute the Beamatron, a hardware device design, calibration strategy and associated control algorithms to enable the use of a steerable projector and depth camera for spatial augmented reality throughout a room (see Figure 1). We believe the Beamatron is the first steerable display to

- incorporate a depth camera to allow live changes in the 3D environment, such as moving a projection surface, or following a user's hand
- use 3D calculations to allow the application to reason in room coordinates and correctly render 3D objects without regard to the precise orientation of the platform
- use a hardware feedback mechanism to more accurately stabilize projected graphics when the unit is moving
- use projective texturing to appropriately render 3D graphics onto arbitrarily shaped surfaces for a given viewpoint
- incorporate 3D (stereo) projection
- use 3D sound source localization as a means to track user position when the user is not in view, to call the unit to view the user for subsequent gesture-based interaction or adjust rendering viewpoint

In the rest of the paper, we discuss related work, and detail the Beamatron hardware and algorithms. We demonstrate the system in an augmented reality scenario of driving a virtual toy car around the real physical world, and in a ubiquitous computing scenario involving the gesture-driven movement of projected 2D graphics. Finally we discuss a number of possible extensions to the Beamatron.

RELATED WORK

Spatial Augmented Reality

A number of previous works have explored using projectors to superimpose graphics onto the real world [4]. A key technical challenge for these systems is taking into account the shape of the projection surface when rendering graphics, so that the projected graphics either appear "painted" on the object [25], or so that the projection gives the impression of a real 3D object of the correct size and shape [2]. These approaches require the precise calibration of the projector (i.e., focal length and optical center).

Of particular relevance to the present work is the overall rendering pipeline of MirageTable [2], which uses Microsoft Kinect depth image data to distort projected graphics on a per-vertex basis. MirageTable also supports the correct rendering of 3D objects for a given viewpoint; this viewpoint is determined by tracking the user's head. Beamatron adopts a similar approach to rendering graphics.

Immersive Projection

In many application scenarios involving projecting onto the real world, it is desirable to project over wide areas to create an immersive experience [23, 30], and researchers have demonstrated the combination of multiple projectors to create bright, high resolution immersive displays [24].

Underkoffler et al.'s "I/O bulb" [32] is an intriguing vision of a single fixture in the ceiling which projects and senses throughout a room. While it is possible to use wide-angle optics to display and sense over a wide area (for example, see [1]), the limited brightness of projectors and limited sensing of cameras imposes a variety of tradeoffs to make a simple implementation of the I/O Bulb difficult. LightSpace [35] addresses brightness and resolution concerns by mounting multiple cameras and projectors to create a large combined fustrum of display and sensing to approximate an I/O Bulb. As with the present work, LightSpace uses depth cameras to correct projected graphics, and enable a variety of 3D interactions.

Steerable Projector-Camera Systems

Steerable displays use a motorized platform or rotating mirrors to direct projected graphics programmatically. This avoids the complexity of multi-projector, multi-camera systems, but retains the ability to place graphics anywhere in the room. The IBM Everywhere Displays prototype is the first steerable display to demonstrate the correction of projected graphics to account for the orientation of the (planar) projection surface [22]. It consists of a projector and a small mirror with 2 axes of rotation. In contrast with the present work, Everywhere Displays is limited to projecting on a discrete set of planar surfaces which are calibrated beforehand, each modeled by a 2D homography.

The Everywhere Displays project explored many applications of steerable displays, particularly in the retail domain. With the addition of a pan and tilt video camera, a number of gesture-based interactions are demonstrated, including touch-based manipulation of projected graphical widgets [12], as well as person tracking to enable graphics to follow the user.

A small number of later works contribute steerable displays designs which mount the projector and camera directly onto motorized platform with pan and tilt axes of motion [8, 6, 11]. Butz et al. [7] contribute a number of interactions in the tradition of ubiquitous computing, such as the ability to scan a room for special AR Toolkit markers [3] so that they later may be indicated to a viewer by a projected "searchlight" graphic, and to project graphical autonomous agents where they are needed. Molyneaux et al. [19] explore more generic object recognition techniques to find and annotate physical objects.

Ehnes et al. [11] contribute a steerable display design that comes closest to the spirit of the present work. In particular, they address the challenges of stabilizing projected graphics while the projector is in motion. Such functionality can be interesting for a wide variety of applications, such as tracking and projecting onto a moving object (such as a person, or a mobile display). They describe their approach to model the precise pose of the head without use of a hardware feedback circuit, and note the importance of accounting for the latency of the camera when matching image processing results with platform pose.

Handheld and Wearable Projectors

Recent advances in the miniaturization of projectors have encouraged the study of handheld and wearable interactive projectors [36, 33, 9, 13, 20]. These share many traits of the steerable displays described above, including problems related to the distortion of the projected image and device pose [26]. Handheld projectors differ from steerable projectors in at least one important way, however: the onus is primarily on the user to select the appropriate area to project. As we will show with the Beamatron, steerable projectors can split this responsibility between the user and the host computer, as appropriate to the application.

Interacting with Light

A number of early works demonstrate the interactive manipulation of graphics projected onto the environment. For example, Rekimoto at al.'s Augmented Surface [27] allows the user to drag a document off the laptop display and onto the projected desktop, and later bring the object from the desktop to the wall using a laser pointer.

The present work demonstrates the ability for the user to "hold" a projected object in the hand, as if it were a real physical object. This has been demonstrated in previous interactive projection-based interfaces. Micromotocross [34] uses a projector and depth camera to allow a player to drive a virtual projected car over real objects and possibly into the hand, with the projected graphics corrected for the shape of the hand. LightSpace [35] gives the user the capability to move a projected off a tabletop and into the other hand. The projected object stays with the hand until dropped back onto the table top or onto the wall. Furthermore, the motion of the ball is simulated such that it rolls downhill along any real physical surface such as the user's forearm or a piece of paper. MirageTable [2] adds the ability to correctly render 3D virtual objects for a given user viewpoint, as well as stereo rendering for active shutter glasses. Finally, LightGuide [29] demonstrates the use of projected graphics in the hand to guide the user to perform a particular motion in three dimensions.

THE BEAMATRON

Hardware configuration

The Beamatron hardware device consists of a video projector and Microsoft Xbox Kinect sensor rigidly mounted to a City Theatrical Autoyoke [10] moving light platform. The Autoyoke's configuration is similar to that of moving lights found at nightclubs and rock concerts, except that the included spot light is easily removed and replaced with other gear. The Autoyoke tilts 270° and pans 360° with a hard stop (i.e., it cannot rotate past 360°). Product documentation indicates pan and tilt movement speed of 90°/s and positional accuracy of 0.1°.

The Autoyoke uses the DMX lighting protocol and interface. Commands to move to a new pan and tilt configuration are issued programmatically using a USB to DMX adapter. As is common for DMX devices, there is no provision for providing feedback on the actual pan and tilt configuration as the device moves to new target pan and tilt angles. Real-time knowledge of the platform pose is critical to projecting stabilized graphics while the platform is in motion. In lieu of such feedback it may be possible to model and predict the position of the platform, as demonstrated in [11]. Instead, we developed a small circuit board which connects to the pan and tilt optical shaft encoders built into the Autoyoke. This circuit uses the quadrature decoders of an AVR XMega256 microcontroller to relay pan and tilt configuration to the PC by a separate USB interface at 100Hz. As far as we know, the Beamatron is the first steerable display to exploit accurate hardware-based feedback for the purposes of stabilizing graphics.

For projecting graphics, the Beamatron uses the Dell 1610HD 3D-ready projector, rated at 3500 lumens. This projector uses the Texas Instruments' DLP Link technology to optionally render stereo 3D graphics with matching stereo shutter glasses. Whereas other techniques employing active shutter glasses require the careful placement of an infrared beacon to synchronize the shutter glasses with the display, DLP Link instead hides a nonvisible synchronization signal in the projected video. This approach is particularly appropriate for steerable displays because it avoids the problem of placing a synchronization beacon so that it is visible regardless of platform orientation, or the complexity of rigging multiple beacons throughout the room. With DLP Link the glasses will synchronize with the video projection if the user can see the projection.

As with other steerable display prototypes that move the projector and camera directly (e.g., [6, 8, 11]), the Beamatron obtains a wide range of pan and tilt motion at the expense of large and possibly noisy stepper motors. When hung in the ceiling near the center of the room, it can reach most surfaces of the room, but also may operate sitting on a desk or the floor. In contrast, the Everywhere Displays prototypes use a small and quiet rotating mirror in front of a stationary projector and so have a more limited range of motion (pan 230°, tilt 50°). This more limited range of motion may place restrictions on where the unit is best placed for a given application.

Calibration

As in previous steerable displays work, precise calibration of the projector, camera and moving platform is critical to precisely superimpose projected graphics onto real objects. Much of the previous work in steerable displays (e.g., Everywhere Displays) is concerned with projecting graphics on to one or more discrete locations throughout the room while the platform is stationary. In this case calibration involves establishing the 2D transformation which warps the graphics for projection onto a given surface. Usually, this transformation is calculated to ensure that a rectangular source image appears rectangular when projected on a planar surface of arbitrary orientation. For this limited case, a camera is unnecessary.

Ehnes et al. [11] calibrate their projector, platform and camera for a spherical coordinate system. This allows for rendering graphics in precise registration with visual markers (AR tags) found by image processing of the video feed.

The Beamatron goes beyond existing work in steerable displays by performing all calculations in 3D. This is enabled primarily by the use of the depth camera of the Kinect sensor. For example, the platform may be moved such that a given 3D point in the room projects to a given 2D point in the projected graphics. Thus it is possible to place projected graphics precisely on an object in the real world, and, as we will describe later, to account for the shape of the projection surface. We describe the Beamatron calibration procedure in more detail in a later section.

Integrating Depth Images

Depending on the application, it may be necessary to aggressively smooth the depth image data. Later, we present an application which runs a physics simulation benefitting from a smooth surface estimate. Averaging multiple successive frames is challenging if the head is in motion. Minimally, before averaging, the previous smoothed frame must be rotated (in 3D) to match the platform motion. Another challenge is to reuse smoothed depth data for a particular view when the unit leaves and later returns to the same view. It is desirable for the system to build a coherent, smoothed model of the room as it pans and tilts so that smoothed data may be reused over static regions. Certain applications may also make use of such a room model in planning platform motion.

The Beamatron system addresses the problems of smoothing the depth data under motion and incrementally building a model of the room by using KinectFusion [14, 20]. KinectFusion incorporates successive depth images into a voxel representation of the surfaces in the room (see Figure 2). Calculations may be run against this volume directly, or it may be processed by successive ray casting to generate a smoothed depth image which is then processed. In its original formulation, KinectFusion uses 3D image processing techniques to solve for the pose of the camera. At first it may seem that this would obviate the need to calibrate the platform geometry, but in early experiments we experienced tracking loss and drift when relying exclusively on KinectFusion's pose estimate over longer time scales. In the Beamatron, this pose information is instead drawn directly from the known pose of the platform.

The present Beamatron implementation uses Kinect Fusion as a means to obtain smoothed depth images under platform motion, but we expect future iterations to exploit its capability to build a model of the room. This could be useful in determining where to next orient the unit if the desired position is out of view. For example, an application may wish to plan a movement to display a graphic on any available flat surface in the room that is also visible to the user. In our present work, KinectFusion is configured to model a 7.5m cubic volume centered under the unit, covering much of our 6m × 10m space.

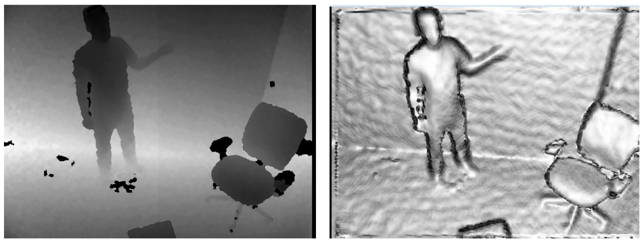

Figure 2. KinectFusion integrates multiple depth camera images (left) as the Beamatron pans and tilts to reveal different parts of the room. Kinect fusion provides stable, smoothed estimates of room geometry suitable for subsequent processing (right).

Rendering

Rendering graphics correctly onto the real world requires using a graphics projection and view matrix that incorporates the pose of the platform, and the projector focal length and principal point found in calibration. For graphics to appear correct when they are projected onto arbitrarily shaped objects, the rendering process must also take into account the shape of the projection surface [25]. Beamatron can use the depth image information to distort the projected graphics so that the projected graphics appear to be "painted" onto the real world object, but it also can correctly render 3D virtual objects for a particular viewpoint while taking into account the varied shape of the projection surface [2] (see Figure 3). This projective texturing-based technique uses multiple rendering passes: the first pass renders the real objects (the room) along with virtual objects from the point of view of the user. The second pass treats the output of the first pass as a texture map for a rendering of the real geometry.

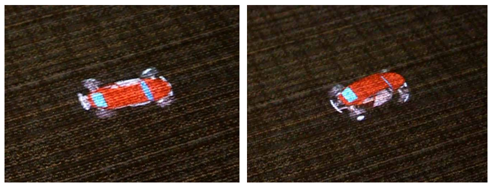

Figure 3. Beamatron rendering pipeline corrects projected graphics to account for projection surface distance and shape. This can be used to correctly project graphics for a particular viewpoint. Left: projective texturing disabled—the car is rendered from the projector's viewpoint. Right: projective texturing enabled—the car is rendered for the user's view.

By this technique, Beamatron matches many aspects of the presentation possible with near-to-eye augmented reality glasses or head mounted displays (HMDs). In some aspects the Beamtron's may be superior. Firstly, the focal distance of the projected graphics may be closer to the correct value, particularly when the virtual object is placed near a real world surface, whereas the focal distance of a rendering on glasses or an HMD is typically fixed. Secondly, graphics rendered on glasses or an HMD must account for the motion of the user's head, particularly if the graphics are meant to be stabilized against the real world. While this can be very difficult given sensor and rendering latencies, projection on the real world would be approximately correct for objects near the real world surface, for any head motion. Such a projection might not show the precisely correct side of a 3D object during motion, but it would not slide across the user's field of view as it might with glasses or an HMD, and therefore would appear more stable.

Beamatron's rendering pipeline supports 3D (stereo) rendering using DLP Link compatible stereo active shutter glasses. Stereo rendering can be used to render virtual objects that appear to have a 3D shape rising from the surface of projection, and even potentially render objects that are located in room coordinate space above any real surface. Consider, for example, a helicopter hovering in space: the projection of the left eye view may fall far from that of the right eye view, depending on the configuration of the room geometry. This can pose a challenge for platform control: instead of controlling the platform to point at the room coordinates of the helicopter, it must instead determine some view that includes the projections of both eyes as they fall on real surfaces. Our current implementation of Beamatron does not support this calculation. The effect of stereo 3D is noticeable, but weaker than that of MirageTable [2], primarily because the distance to the rendered object may be beyond the range in which binocular disparity is a powerful depth cue.

Reasoning in Room Coordinates

The computational strategies described above provide the ability to relate all sensing data to a room coordinate frame, find the real time 3D pose of the moving platform and render viewpoint-corrected 3D graphics regardless of projector pose. Together they provide a fully 3D model of sensing and display.

This model is useful for steerable displays because it allows application developers to perform reasoning in room coordinates, without regard to the pose of the steerable display, provided that any real time changes in the scene are within the field of view of the depth camera.

In fact, for many applications it may be useful to conceptually decouple the reasoning involving the room model (e.g., placing virtual objects in 3D) from the process that controls platform motion, and instead think of the Beamatron as a "camera man" that is controlled by a process that selects the most active or relevant area of the room to monitor. In the case of an augmented reality application where the user controls an object moving around the room, it may be natural to simply have the platform follow the object.

External Sensing with Sound Source Localization

One shortcoming of placing a steerable display in the center of the room is that it is difficult to use the unit's camera to monitor events outside the view of the unit's single camera. KinectFusion could be used to plan the placement of graphics onto known locations throughout the room as long as they do not move while the unit is looking away. However, in applications that would make use of person-tracking or gesture recognition, a model of the room's static surface features may not be useful. In these cases, a means of drawing the unit's attention is needed.

A number of solutions to monitor the wider area come to mind, such as mounting one or more additional Kinect cameras on the unit, using multiple cameras throughout the space, or even moving the unit's single camera to a second moving platform, decoupling the placement of display and sensing. Many of the possible solutions tend towards the more complex immersive setups (such as [34]) and therefore violate the spirit of the steerable display.

Solutions that serve a wide area, but with coarse sensing resolution, seem more appropriate. For example, a single wide angle conventional camera mounted on the unit might be used to detect the presence of the user's gesturing to gain the Beamatron's attention. The unit may then rotate to gain a more detailed view.

Rather than use more cameras and image processing, we exploit the beam-forming array microphone in the Kinect sensor to localize the user as they speak. The Kinect array microphone can be used to find the angle of the dominant sound source in the horizontal plane of the camera to an accuracy of 4° standard deviation at 4m. This is reported over a range of about 100°, approximately double that of Kinect's video camera field of view and well suited for mounting in the corner of a room. We use three Kinect sensors mounted in the corners of the room, one mounted in a vertical orientation, to solve for the 3D position of the dominant sound source. The pose of each sensor in room coordinates is determined by finding the same calibration pattern used to calibrate the Beamatron (described later). The sound source position is found by noting that the angle reported by each sensor constrains the sound to a plane orthogonal to the horizontal plane of the sensor. Thus the 3D position is calculated by computing the intersection of three planes indicated by the reported source angles (see Figure 4). This position is reported along with beam forming confidence values at 10Hz.

A Beamatron application may designate one of the cameras as an audio source for speech recognition. Speech recognition can be an easy way to gate the continuous updates of the sound source localization process: e.g., when a particular user says "Beamatron!" the unit may be commanded to orient itself to bring that user into view, much in the same way that a person reflexively turns their head to look at someone who just called their name from across the room. In the worst case, the platform must pan nearly 360° (about 4s) to put the user in view.

We have found the accuracy of the audio localization to be good enough for the two applications reported in the rest of this paper. To characterize audio localization accuracy, we compared localization results to ground truth at 13 locations throughout the room. Standing at each location, the subject uttered the word “Beamatron”. 3D triangulated sound source position was recorded at the precise moment the word “Beamatron” was automatically recognized. This process was repeated four times for each location, with the subject turning to face each of the four walls of the lab. Comparing the audio localization results to ground truth, we notice that accuracy varies depending on the location in the room (RMS error = 0.84m), and that there is a bias that may be caused by imprecise calibration. Translating each sample by the average error can correct for this bias somewhat (RMS error = 0.50m).

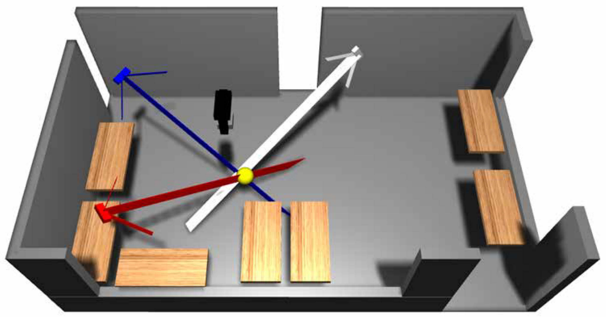

Figure 4. Audio sound source localization combines array microphone sound source output angles from three Kinect sensors (red and blue mounted horizontally, white mounted vertically) to find the 3D position of the sound source within the room (yellow). Each array microphone provides a planar constraint on the 3D position of the sound source (illustrated here by long thin rectangles, with minimum and maximum angles indicated by short rectangles). Plan view of the room is shown (6m × 10m).

UBICOMP EXAMPLE: MASTER AND COMMANDER

Our first Beamatron example application is an homage to Bolt’s Put-that-there system [5], the grandfather of all natural user interfaces. We also build on LightSpace [34], which demonstrated moving graphics from one surface to another as well as to holding a projected object directly in the hand.

With Master and Commander, the user moves projected graphics around the room with speech and gesture. They can request Beamatron’s attention by calling “Beamatron”, place projected objects around the room or take them and hold them in their hand. For example, while the Beamatron displays a graphic on the wall, a user in the room may say “take that”. This utterance is recognized using the Kinect sensor designated for speech recognition, and the 3D position of the speech audio source is determined as described above. The Beamatron is then oriented to the 3D position of the speaker sound source. With the user now in full view of the onboard Kinect sensor, Master and Commander enables Kinect skeletal tracking (see Figure 5).

Once the user’s skeleton is found, a graphical representation of the object previously projected on the wall now appears in the user’s hand, thus completing the “take that” command. This projected object may be passed from one hand to the other, and to another tracked user’s hand. This is accomplished simply by computing the distance of the hand currently holding the virtual object to all other available hands, and transferring the object to any sufficiently nearby hand, with some hysteresis to prevent instantly handing the object back.

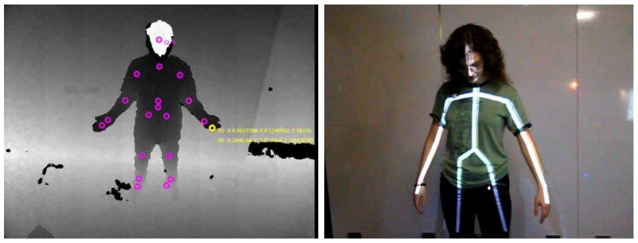

Figure 5. left: Master and Commander skeletal tracking. right: projecting skeleton onto the user.

The user may place the object in the room by pointing with their free hand to a destination position in the room while saying “put that there”. Recognition of this phrase triggers the movement of the virtual object from their hand to the 3D point indicated by the ray from the user’s head to the user’s hand, and its first intersection with a surface in the room (see Figure 6).

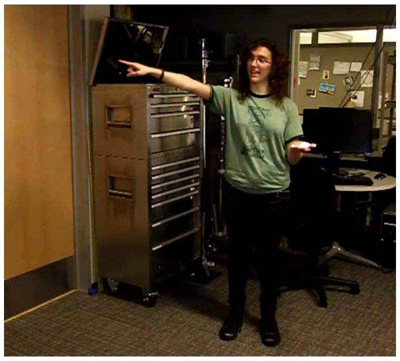

Figure 6. Master and Commander allows the user to take projected graphics from the wall into their hand and then place the objects on another part of the room. Here the user is holding a graphic in their left hand while pointing at a destination. Saying “put that there” will cause Beamatron to animate the object to the wall, where it will find a surface to “hang” the graphic.

Computing this destination position can be done in multiple ways. The voxel space representation of KinectFusion holds a representation of all surfaces viewed thus far as the unit has been reoriented to view various points throughout the room. It is thus possible to use this representation to find the intersection of the user’s pointing ray with the surface of the room. In the current implementation, we use an approach which does not rely on the model of the entire room. Instead, the Beamatron is continuously moved to look at points along the pointing ray, beginning with the 3D position of the hand. As the Beamatron is moved along the ray, the depth image data in the current view is tested for intersection with the ray after transformation to room coordinates. While this search along the ray is conducted, a graphical representation of the virtual object is rendered along the floor to illustrate the connection between the user's gesture and the movement of the virtual object.

Once the intersection point is found, the virtual object is "hung" on the wall or any other surface by finding the normal at the point of intersection from the depth image. We can also account for gravity by rotating the object to align with the room up vector. This is particularly useful when trying to place graphics or slides on the wall for a presentation.

In Master and Commander, gesture recognition depends on the Beamatron's ability to locate the user well enough that the Kinect skeletal tracking successfully tracks the user. Not all viewpoints work equally well. In particular, locations away from the Beamatron tend to work better, while in the area directly under the Beamatron tracking tends to fail, since the skeleton tracker was trained for frontal views.

In addition, sound source localization can provide noisy position information. When calling its attention, this can cause the Beamatron to move in the right direction, but not center on the user accurately enough to capture their entire body. We address this by randomly moving the Beamatron in a small search pattern around the sound source position until a skeleton is found. In practice, this process can take a few seconds, but due to the limitations of the skeletal tracker is not guaranteed to succeed. When the user's skeleton is found, the Beamatron orientation is trained on the hips of the recovered skeleton, and will then follow the user as they move about room.

AUGMENTED REALITY EXAMPLE: BEAMABUGGY

Our second Beamatron example application takes inspiration from Wilson's Micromotocross tabletop augmented reality depth camera demo [34].

Beamabuggy allows a player to drive a small virtual car throughout the entire room using a wireless steering wheel gaming controller and standard driving controls (see Figure 7 and Figure 1d). Calculations on the KinectFusion surface data enable the car to interact appropriately with real objects placed in the room. Driving the car over a cardboard ramp causes the car to jump appropriately, possibly flipping over. The movement of the car is determined by a gaming physics engine configured to model the dynamics of the car and perform collision detection against the ground mesh derived from the depth image. The Beamatron is programmed to keep the car in the center of the projected image.

Using a smooth surface representation rather than raw depth image data is critical for a reasonable driving experience. Consider that at the scale of a toy car, a bit of noise in the depth data may seem like a giant pothole! KinectFusion provides a smooth depth map for the current view, regardless of the Beamatron motion, as described previously. Transformed to room coordinates, this depth information is used to create the appropriate collisions with the car's wheels and chassis. We use the Newton gaming physics engine [21] for its flexible support of custom collision detection routines.

One of Beamatron's drawbacks in this application is that correct interaction with real world objects is limited to the combined view fustrums of the projector and depth camera. For example, our current prototype does not allow the player to drive under a table. In this case, the depth information at the transition from table to floor leads to a "curtain" in the collision geometry that the car cannot drive through. One approach to addressing this problem is to remove this curtain by rejecting mesh triangles that are at a very oblique angle to the camera, and by providing a dummy ground plane under the table to allow the car to drive under the table. Even with this approach, however, the player will not see the projection of the car under the table.

Beamabuggy uses the multi-pass projective texturing technique described earlier to render the car so that it appears to have the correct 3D shape given a particular user viewpoint, regardless of what surface the car is driving over. In our present implementation, the user viewpoint is deduced by the sound source localization procedure described above. When the user says "Beamatron", the 3D viewpoint is updated with the most recent 3D sound source localization values.

CALIBRATING THE BEAMATRON

To fully calibrate the Beamatron, we first find the focal length and principal point (optical center) of Kinect's infrared (depth) camera and color camera. In the case of the color camera we also find radial lens distortion parameters. These *intrinsic* parameters are computed using the technique described in [37], which uses multiple images of a known 2D printed calibration pattern (i.e., a checkerboard of known dimensions) placed at various orientations.

We must also find the intrinsic parameters of the projector. While it is useful to think of the projector as a camera, with its own focal length and principal point, projector calibration must be handled differently. First, we use the (calibrated) color camera to acquire an image of the calibration pat-

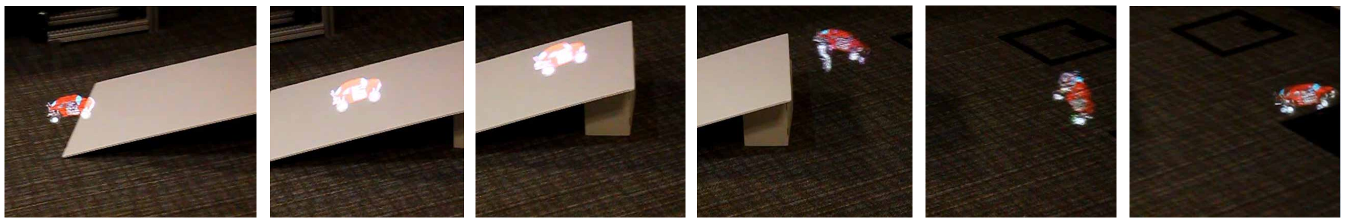
Figure 7. Beamabuggy takes a ramp. A gaming physics engine simulates vehicle dynamics and contact with room geometry modeled by KinectFusion.

tern lying on a large poster board. Because the calibration pattern is known, we can find the plane equation of the calibration pattern (and thus the poster board itself). Then, the printed pattern is taken away, and a similar calibration pattern is projected onto the poster board. This pattern is then imaged by the color camera. In this configuration, the real world dimensions of the projected pattern at the poster board are unknown (the poster board is at an unknown distance and orientation), but because the projected pattern lies in the plane of the poster board, we can compute the 3D position of all points of the calibration pattern using the plane equation. We may then use the usual calibration calculations designed for cameras (see [28] for more detail).

Once the intrinsic parameters of a camera are known, it is straightforward to recover the pose (position and orientation) of the calibration pattern. Thus it is possible to find the *extrinsic* parameters: the relative pose of the Kinect color camera and infrared camera, as well as that of the color camera and projector.

Remaining is the characterization of the moving platform geometry. This includes the conversion of DMX pan and tilt device parameters to radians, the pose of the camera relative to the platform axes of rotation and its pose relative to the room coordinate system. For these calculations we use a procedure described in [15] and [31] which again relies on several images of the same calibration pattern used above, under varying platform pan and tilt angles.

In summary, Beamatron calibration establishes the following quantities, listed here in the approximate order that they should be collected:

- focal length f, principal point (c_x, c_y) and lens distortion parameters of the Kinect color camera
- focal length and principal point of the infrared (depth) camera
- focal length and principal point of the projector
- $\mathbf{T}_{color\to infrared}$, the coordinate transform (rotation and position) relating the pose of the color and depth cameras
- $\mathbf{T}_{color\to proj}$, the coordinate transform relating the pose of the color camera and projector
- scalar values converting DMX pan and tilt commands to pan and tilt angles radians (α, β), and device feedback units (stepper motor counts) to (α, β)
- $\mathbf{T}_{base\to color}$, the coordinate transform relating the pose of the color camera with respect to the pan and tilt axes of rotation
- $\mathbf{T}_{room\to base}$, the coordinate transform placing the pan and tilt axes of rotation in room coordinates

We note that during runtime the color camera is not used, our current calibration formulation is formed around the color camera primarily because of its use in the calibration procedure.

Platform Movement

Two important tasks in working with the Beamatron are controlling its orientation to bring either a real or virtual object within view, and determining the pose of the unit as it moves to the target orientation.

For both we make use of the platform geometry calibration described above to find the pose of the color camera given pan and tilt angle values (α, β). A point in the room coordinate frame can be brought into the coordinate frame of the projector by

$$\mathbf{x}_{proj} = \mathbf{T}_{color\to proj}\mathbf{T}_{base\to color}\mathbf{R}(\alpha,\beta)\mathbf{T}_{room\to base}\mathbf{x}_{room}$$

where $\mathbf{R}(\alpha, \beta)$ is the matrix which rotates about the two axes of the platform. A similar formula may be used to convert a depth camera coordinate point to room coordinates.

To determine the real-time pose of the unit, the encoder feedback values are converted from device units to radians, and the formula for the pose of the color camera is used. The pose of the unit can be used to transform a point in the depth camera to a 3D room coordinate point. In this case it is important to note that if the head is in motion, the latency of the camera will cause the transformed point to be incorrect because the unit will have moved since the image data was acquired. This discrepancy can be eliminated by buffering the encoder feedback values, and selecting older values when dealing with camera-derived information. Our current implementation was manually tuned for a delay of eight encoder frames, which at the 100Hz reporting rate of encoder frames is about 80ms, which corresponds well with the latency of the Kinect depth image. This approach is suggested in [11].

Determining the pan and tilt values to bring an object into view is basic to a steerable display. Given a 3D point in the room, we calculate the pan and tilt angles such that the point projects to a given 2D point in the projected graphics (this 2D point is often chosen to be the center of the projected image). A 3D point in a camera or projector's coordinate is projected to image coordinates by the following standard camera model (neglecting lens distortion):

$$P(\mathbf{x}) = s\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}f & 0 & c_x\\0 & f & c_y\\0 & 0 & 1\end{bmatrix}\mathbf{x}$$

The relationship between pan and tilt angles and the unit pose is nonlinear due to the rotations involved. Thus it is appropriate to use a nonlinear optimization technique. We minimize an objective function that calculates the distance between the desired projector image position (u, v) and current guess for (α, β):

$$f(\alpha,\beta) = \left\|(u,v) - P_{proj}(\mathbf{x}_{proj}(\alpha,\beta))\right\|^2$$

The choice of optimization technique is complicated by the fact that for every pan and tilt solution there is a dual solution that pans 180°, and tilts to reflect about the vertical axis. Without constraining the optimization, the platform control algorithm runs the risk of making a drastic and disruptive motion to this second solution, perhaps only to return to the other side in the next moment. Accordingly, we use the Luus-Jaakola nonlinear optimization technique

[17], which is easy to implement to include the constraint that solutions must have a tilt value greater than 0°.

FURTHER WORK

The two very different example applications suggest the broad range of applications enabled by the Beamatron system. The many capabilities of the system, such as the ability to stabilize graphics while the platform is in motion, follow a given point in 3D, render 3D graphics for a given viewpoint and interpret the user's speech and gesture, invite a variety of further investigations into new interactions and applications enabled by the Beamatron. As one example, it is straightforward to contemplate rendering of a 3D object in the user's hand corrected for viewpoint and surface shape as the user wanders about the room, and furthermore use stereo rendering to make the graphic appear to rise off the user's hand.

Here we consider a few areas of further work directed at improving the basic capability of the device.

Automated Calibration

Our current method for calibrating the Beamatron involves several steps and is still very manual in nature. It may be possible to automate much of the calibration procedure. Ultimately, we care most to precisely project graphics on features in the depth image, yet our current calibration procedures are ill-suited to directly minimize any discrepancy between the projected graphics and the target feature in the depth image. Rather, this alignment comes about from multiple independent calibrations, which when composed can accrue noticeable alignment errors. We note that due to the movement of the platform and the need to accurately calibrate its internal geometry, the calibration of the Beamatron is more difficult than the calibration of a static projector and camera pair.

Deeper Integration with KinectFusion

Our present implementation uses KinectFusion primarily as a means to obtain smooth surfaces from noisy depth images even while the platform is moving. There are a number of other ways we could exploit KinectFusion, however.

The original development of KinectFusion includes calculating camera pose by matching features of the image. This pose information is used to correctly integrate new depth image data, and is a potentially useful by-product. Beamatron overrides this pose information with the pose calculated more directly by the known pan and tilt configuration of the platform and the calibration of its internal geometry. Whereas Beamatron's pose estimate will never drift or fail due to tracking failure, it may possess systematic errors due to inaccurate calibration. The complementary nature of these two sources of platform pose suggests that the two could be fused to adjust the calibration of the platform while the unit is used initially, for example.

Additionally, many Beamatron applications could make use of the voxel space model of the room that KinectFusion constructs as the camera is reoriented to view the entire room. This whole-room model could be useful in planning the Beamatron's next movement, such as where best to display a graphic. This model may also be used to integrate data from another camera, possibly a handheld unit or a second Beamatron in the same room. A handheld Kinect sensor, for example, could be used to obtain close-in, high resolution scans of parts of the room, and also complete geometry that is not visible to the camera on the platform, such as under furniture or just around corners. This more complete model could be useful in some applications; e.g., driving the car under the table.

Alternate Hardware Configurations

The Beamatron prototype is large in part because of the use of a regular DLP video projector. The device could be made much smaller with an LED or laser projector. While LED and laser projectors are not nearly as bright as a full size projector, this could be mitigated by the use of longer projection lenses. The smaller projection area of a longer lens is a reasonable tradeoff for a steerable display.

Automated control of focus and zoom might be useful for some applications. In our experience the depth of field of the projector is not a great problem, but it could be for close-in work. Of course, a laser projector requires no focus adjustment.

While the limited viewing and display areas are characteristic of steerable displays, it is tempting to consider more elaborate hardware designs that effectively enlarge the viewing and display areas without resorting to many cameras and many projectors. For example, multiple Beamatron units could be used to cover a large area at relatively little expense. Multiple units could be coordinated in useful ways, such as performing a handoff of projected graphics around a user as they move from one unit's service area to another (see Figure 8).

As suggested earlier, the use of multiple units could address the problem of split attention between the user and some distant focal point of the presentation. In such cases it may be desirable to task a unit with monitoring a user and another unit with monitoring or displaying on another part of the room, and in some cases it may suffice to include units that have only a camera and others that have only a projector.

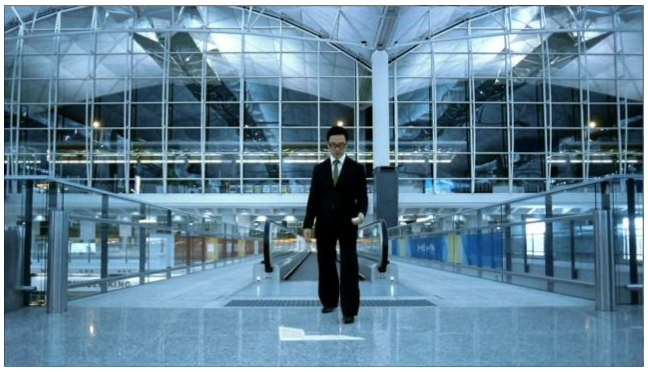

Figure 8. An arrow directs a man through an airport in Microsoft Office Labs' Productivity Future Vision video [18]. This could be realized by multiple coordinated steerable displays.

CONCLUSION

The Beamatron advances steerable displays by drawing on recent progress in depth camera-based interactions. Through its calibration, control and rendering algorithms, the Beamatron allows applications to reason in the 3D coordinate system of the room, enabling a wide range of augmented reality and ubiquitous computing scenarios with a minimal amount of hardware and computation.

ACKNOWLEDGMENTS

We thank Bruce Cleary for help building the Beamatron hardware prototype.

REFERENCES

1. Benko, H., and Wilson, A.D. Multi-point interactions with immersive omnidirectional visualizations in a dome. In *Proc. of Interactive Tabletops and Surfaces (ITS),* 2010, 19-28.
2. Benko, H., Jota, R., and Wilson, A.D. MirageTable: freehand interaction on a projected augmented reality desktop. In *Proc. of ACM SIGCHI,* 2012, 199-208.
3. Billinghurst, M., Weghorst, S., and Furness, T. Shared space: An augmented reality approach for computer supported collaborative work. *Virtual Reality,* 3(1), 1998. 25-36.
4. Bimber, O., and Raskar, R. Spatial Augmented Reality: Merging Real and Virtual Worlds. A.K. Peters, 2005.
5. Bolt, R.A. "Put-that-there": voice and gesture at the graphics interface. In *ACM Siggraph Computer Graphics.* 14, 3. 1980. 262-270.
6. Borkowski, S., Letessier, J., and Crowley, J.L. Spatial control of interactive surfaces in an augmented environment. In *Lecture Notes in Computer Science*. 2004. 228-244.
7. Butz, A., Schneider, M., and Spassova, M. SearchLight – a lightweight search function for pervasive environments. In *Proc. of Pervasive Computing*, 2004. 351-356.
8. Butz, A., and Kruger, A. Applying the peephole metaphor in a mixed-reality room. In *IEEE Computer Graphics and Applications* 26, 1 (2006). 56-63.
9. Cauchard,, J.R., Fraser, M., Han, T., and Subramanian, S. Steerable projection: exploring alignment in interactive mobile displays. In *Personal Ubiquitous Computing*. 2012. 27-37.
10. City Theatrical, Autoyoke: The First Automated Lighting Solution Designed for Theatrical Use. http://citytheatrical.com/rcyoke.htm, 2012.
11. Ehnes, J., Hirota, K., and Hirose, M. Projected augmentation – augmented reality using rotatable video projectors. In *Proc. of ISMAR*. 2004. 26-35.
12. Kjeldsen, R., Pinhanez, C., Pingali, G., Hartman, J., Levas, T., Podlaseck, M. Interacting with steerable projected displays. In *Proc. 5th Int. Conf. on Automatic Face and Gesture Recognition.* 2002.
13. Harrison, C., Benko, H., Wilson, A.D., OmniTouch: wearable multitouch interaction everywhere. In *Proc. of ACM UIST.* 2011. 441-450.
14. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., Fitzgibbon, A. KinectFusion: real-time 3d reconstruction and interaction using a moving depth camera, In *Proc. of ACM UIST.* 2011. 559-568.
15. Ngan, P.M., and Valkenburg, R.J. Calibrating a pan-tilt camera head. In *Image and Vision Computing Workshop*. 1995.
16. Levy, S. Google gets transparent with Glass, its augmented reality project, http://wired.com/epicenter/2012/04/epicenter-google-glass-ar/. 2012.
17. Luus, R, and Jaakola, T.H.I. Optimization by direct search and systematic reduction of the size of search region. In *American Institute of Chemical Eng. Journal,* 19(4), 1973. 760-766.
18. Microsoft Office Labs. Productivity Future Vision Video. http://www.microsoft.com/office/labs/index.html, 2009.
19. Molyneaux, D., Gellersen, H., Kortueum, G., and Schiele, B. Cooperative augmentation of smart objects with projector-camera systems. In *Proc. of Ubicomp.* 2007. 501-518.
20. Molyneaux, D., Izadi, S., Kim, D., Hilliges, O., Hodges, S., Cao, X., Butler, A., Gellersen, H. Interactive environment-aware handheld projectors for pervasive computing spaces, In *Proc. of Pervasive Computing*. 2012.
21. Newton Game Dynamics. http://newtondynamics.com.
22. Pinhanez, C. The Everywhere Displays Projector: a device to create ubiquitous graphical interfaces. In *Proc. of Ubiquitous Computing.* 2001. 315-331.
23. Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L. and Fuchs, H. The office of the future: a unified approach to image-based modeling and spatially immersive displays. In *Proc. of ACM SIGGRAPH.* 1998. 179-188.
24. Raskar, R., Brown, M.S., Yang, R. Chen, W.-C., Welch, G., Towles, H., Seales, B., and Fuchs. H. 1999. Multi-projector displays using camera-based registration. In *Proc. of Visualization.* 1999. 161-168.
25. Raskar, R., Welch, G., Low, K.-L., and Bandyopadhyay, D. Shader Lamps: Animating Real Objects With Image-Based Illumination. In *Proc. of Eurographics Workshop on Rendering Techniques.* 2001. 89-102.
26. Raskar, R., van Baar, J., Beardsley, P., Willwacher, T., Rao, S., Forlines, C. iLamps: geometrically aware and self-configuring projectors. In *ACM Trans. Graph., 22, 3.* 2003. 809-818.
27. Rekimoto, J. and Saitoh, M. Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments. In *Proc. of ACM SIGCHI.* 1999. 378–385.
28. Sodhi, R., and Jones, B. Kinect-Projector Calibration, http://augmentedengineering.com/ProCamCalibration/brjones2rsodhi2Final.pdf. 2010.
29. Sodhi, R., Benko, H., and Wilson, A.D. LightGuide: projected visualizations for hand movement guidance. In *Proc. of ACM SIGCHI.* 2012. 179-188.
30. Sony Entertainment Network. Great films fill rooms, http://greatfilmsfillrooms.com. 2011.
31. Tsai, R.Y, and Lenz, R.K. A new technique for fully autonomous and efficient hand/eye calibration. In *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 5(3), 1989. 345-358.
32. Underkoffler, J., Ullmer, B., and Ishii, H. Emancipated pixels: Real-world graphics in the luminous room. In *Proc. of ACM SIGGRAPH.* 1999. 385–392.
33. Willis, K., Poupyrev, I., Hudson, S.E., Mahler, M. SideBySide: ad-hoc multi-user interaction with handheld projectors. In *Proc. of ACM UIST.* 2011. 431-440.
34. Wilson, A. Depth sensing video cameras for 3D tangible tabletop interaction. In *Proc. of IEEE Tabletop*. 2007. 201-204.
35. Wilson, A., and Benko, H. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proc. of ACM UIST.* 2010. 273-282.
36. Xiang, C., Forlines, C., and Balakrishnan, R. Multi-user interaction using handheld projectors. In *Proc. of ACM UIST.* 2007. 43-52.
37. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(11), 2000. 1330-1334.