Jason Hochreiter*
Institute for Simulation and Training
University of Central Florida
3100 Technology Parkway
Orlando, Florida 32826

Salam Daher
Arjun Nagendran
Laura Gonzalez
Greg Welch
University of Central Florida

Optical Touch Sensing on Nonparametric Rear-Projection Surfaces for Interactive Physical–Virtual Experiences

Abstract

We demonstrate a generalizable method for unified multitouch detection and response on various nonparametric and parametric surfaces to support interactive physical–virtual experiences. The method employs multiple infrared (IR) cameras, one or more projectors, IR light sources, and a rear-projection surface. IR light reflected off human fingers is captured by cameras with matched IR pass filters, allowing for the detection and localization of multiple simultaneous finger-touch events. The processing of these events is tightly coupled with the rendering system to produce auditory and visual responses displayed on the surface using the projector(s) to achieve a responsive, interactive, physical–virtual experience. We demonstrate the method on two nonparametric face-shaped surfaces and a planar surface. We also illustrate the approach's applicability in an interactive medical training scenario using one of the head surfaces to support hands-on, touch-sensitive medical training with dynamic physical–virtual patient behavior.

1 Introduction

Touch sensing for interactive computer graphics applications is typically implemented on surfaces that can be mathematically parameterized via analytical equations, such as planar surfaces or spheres. We present a method for unified multitouch detection and response on nonparametric surfaces with rear-projection animated content. Similar to work by Benko, Wilson, and Balakrishnan (2008), our method comprises infrared (IR) cameras, projectors, IR light sources, and a rear-projection surface. IR light from below the surface passes through it, reflects off the user's fingers back through the surface, and is captured by cameras below the surface with matched IR pass filters. This allows for the localization of multiple finger-touch events over the surface. Detection of the touch events is tightly coupled with the rendering system to produce a highly responsive interaction.

The presented method is generalizable both to nonparametric and to parametric rear-projection surfaces. We demonstrate the use of the proposed method in allowing for touch sensing on two nonparametric human head surfaces and one planar surface. Furthermore, we illustrate the applicability of a

Presence, *Vol. 25, No. 1, Winter 2016,* 33–46
doi:10.1162/PRES_a_00242
© 2016 by the Massachusetts Institute of Technology

*Correspondence to jhochrei@cs.ucf.edu.

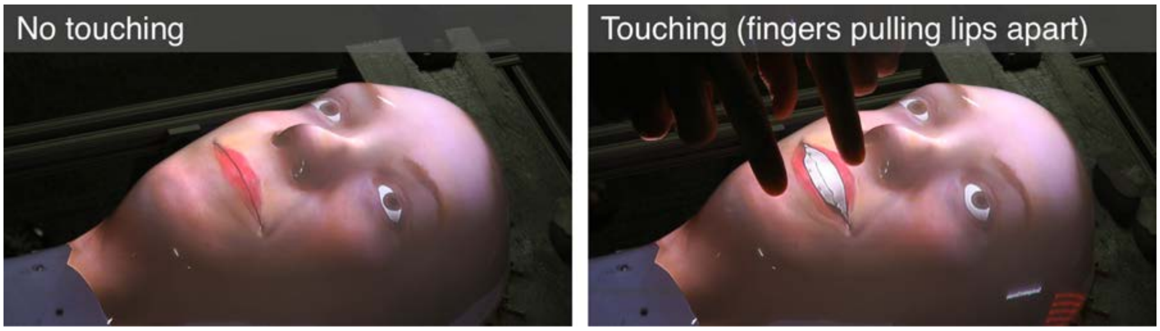

Figure 1. *Our touch-sensitive physical–virtual head for hands-on healthcare training. The prototype comprises a translucent head-shaped shell with a digital projector, IR light sources, and cameras underneath. Left: No touch. Right: A nurse diagnosing a potential stroke uses her fingers to pull the lips apart to inspect the teeth and gums.*

physical prototype in a medical training scenario using one of the head-shaped surfaces we refer to as the *physical–virtual patient head*. Touch events on this surface prompt graphical and audio responses—for instance, a participant can examine the patient's teeth and gums by spreading the lips using her fingers (see Figure 1), and certain actions result in the patient speaking. Our approach is complementary to existing training interfaces and will support a wide range of physical–virtual medical training scenarios involving touch, including diagnostic, therapeutic, and comfort touch. As with human patient actors and robotic mannequins, our method offers a direct hands-on paradigm without the need for a head-worn display (HWD) or a separate virtual content display. Like virtual patients, our physical–virtual patient head can present abnormal physical findings that can be diagnosed via visual appearance, either passively (just visually) or interactively (through touch).

Our proposed method relies on the construction of a lookup table relating points in various coordinate spaces. We have demonstrated an initial application of this method on a human head-shaped surface (Hochreiter, Daher, Nagendran, Gonzalez, & Welch, 2015). Here, we present an extension of this approach that provides a more efficient mechanism for scanning the 3D geometry of a surface and eliminates the need for external photogrammetry. Moreover, we have simplified the overall representation and relationships between the coordinate spaces of the cameras, the projectors, the surface, and the 3D model projected onto the surface. We include a projector calibration routine that is incorporated into the 3D graphics rendering system, a step which originally required manual approximation. Finally, we illustrate how the proposed method generalizes to three rear-projection surfaces, both parametric and nonparametric.

2 Related Work

A number of methods exist for touch sensing on various surfaces. Capacitive sensing approaches (Dietz & Leigh, 2001; Gu & Lee, 2011; Rekimoto, 2002) are popular, though they typically assume flat surfaces. Conductive materials can be molded to fit more complex surfaces (Rekimoto, 2002); however, the density of the capacitive elements must be sufficiently high compared to the complexity (spatial frequency) of the surface. In any case, capacitive methods would interfere with projected imagery.

Computer vision approaches use either visible or IR light to detect touch events. One class of methods uses *frustrated total internal reflection* (Han, 2005; Roudaut, Pohl, & Baudisch, 2011; Villar et al., 2009; Wang, Cao, Ren, & Irani, 2009), but such methods cannot handle surfaces with arbitrary curvature and discontinuities. Hands and fingers that interact with a surface reflect IR light through the material; this light can be used directly for touch sensing (Benko, 2009; Benko et al., 2008;

Matsushita & Rekimoto, 1997; Wilson, 2004). Touch sensing has been performed on planar and parametric nonplanar surfaces, such as spheres (Benko, 2009; Benko et al., 2008), in this manner. When extending this technique to arbitrary surfaces, it can be challenging to find an appropriate mapping between touch events detected by a camera and 3D coordinates on the surface. Moreover, it may be difficult to provide uniform or smoothly varying IR illumination across the surface.

Geometric registration techniques have been explored on parametric planar surfaces using linear methods, nonlinear methods, and piecewise linear methods (Majumder & Brown, 2007). Piecewise linear methods have also been applied to parametric nonplanar displays. For nonparametric display surfaces, one approach involves finding a mapping between projected imagery and the viewer's eye, represented using a camera located at the intended location of the viewer.

Our work here is specifically motivated by medical training. As part of a growing desire to leverage the best of virtual and physical aspects of medical training, we have previously developed both avatar and agent-based physical–virtual patients (Lincoln et al., 2010; Rivera-Gutierrez et al., 2012; Welch et al., 2011). Kotranza and Lok (2008) explored a Mixed Reality (MR) paradigm aimed at realistic breast exams. The trainee stands over a prone physical mannequin that includes a physical breast simulator—a rubberized breast that provides the feel of breast skin, tissue, and underlying breast masses, and contains twelve pressure sensors to detect the user's touch. Through an HWD, the trainee sees a virtual patient overlaid on the mannequin and breast simulator, with a dynamic face and gown continuously rendered via camera-based tracking of a physical gown on the mannequin. Chuah et al. (2013) have since surveyed existing research and formalized the idea of physicality for embodied conversational agents.

3 Correspondences

In conventional touch systems, the geometric relationship between the coordinate spaces of the input device (touch) and the output device (display) can usually be modeled by a parametric function, such as a 6D rigid transform or a homography. Our situation differs in two respects: the 3D touch and 3D graphics spaces corresponding to the physical surface are discretely sampled, and the relationship between the spaces cannot be described by a parametric function. As such, we construct and use a lookup table to directly link the coordinate spaces. This has the additional benefit of temporally decoupling the rendering and touch systems so that rendering events can be carried out independently and at a higher rate than the processing of touch events. In total, we have three types of coordinate spaces: $\mathcal{I}$ 2D spaces for the cameras, $\mathcal{P}$ 2D spaces for the projectors, and one 3D space for rendering on the physical surface. We describe these three spaces next.

3.1 Coordinate Spaces

3.1.1 CAM2. Using the touch sensing system (described in the Touch Sensing section), touch events are detected and localized as 2D (x, y) coordinates in the *2D camera space (CAM2)*. Each camera capable of imaging a touch event records the event's position within its coordinate space.

3.1.2 PRO2. A detected touch event must trigger an appropriate graphical response in the rendering system, requiring a mapping between 2D (x, y) camera coordinates and 2D (u, v) projector coordinates in the *2D projector space (PRO2)*, which represents the 2D image sent to the projector. To obtain these mappings, we employ a structured coded light approach using pattern images of white circles, discussed next.

3.1.3 GFX3. In a typical multiple view geometry scenario, some form of feature matching provides correspondences between the different views. As our rear-projection surfaces are both uniform in color and smooth, they exhibit few features; it is therefore difficult to find correspondences via a feature matching approach. Instead, we use the white projected circles from the structured coded light patterns as manually generated "features," as their detected positions within each camera's imagery are corresponding points. These

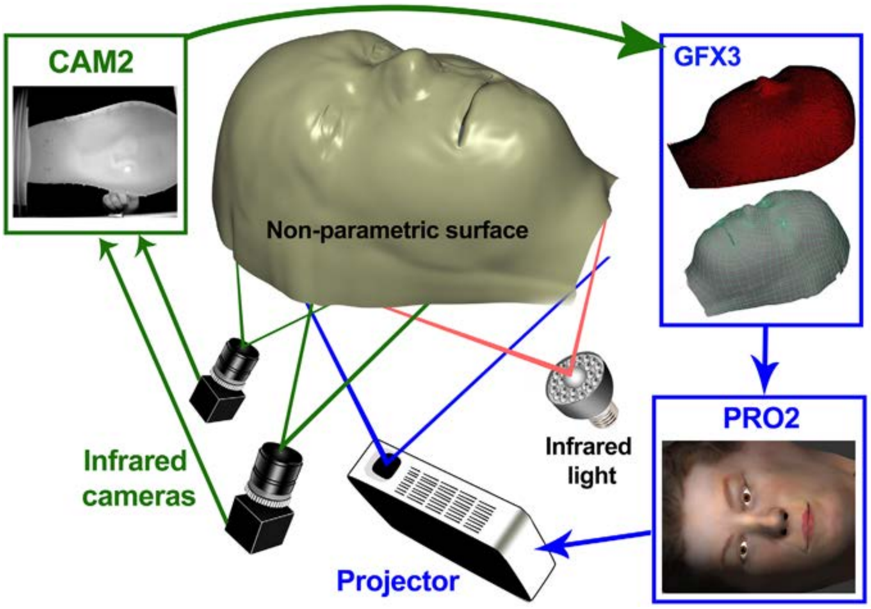

Figure 2. *Descriptions of and relationships among the various coordinate spaces of the proposed method. Touch events detected in the 2D camera space (CAM2) are converted to coordinates in the 2D projector space (PRO2) and coordinates in the 3D graphics space (GFX3), as needed. Correspondences in CAM2 are triangulated in 3D space to form GFX3, which is then textured. At run time, GFX3 is rendered in 2D and sent to the projector (PRO2). This method is general to any nonparametric surface; we show imagery of the physical–virtual patient head only for the purpose of illustration.*

correspondences are triangulated in three-dimensional space using data from camera calibration, resulting in 3D (X, Y, Z) vertices in the *3D graphics space (GFX3)*. Together with the 3D positions of the cameras and projectors obtained through calibration (described in the Setup section), this provides a 3D model of the entire setup (shown in Figure 6 for the physical–virtual patient head).

By interpolating the camera correspondences, we can make this 3D vertex point cloud arbitrarily dense. We backproject all 2D pixels within each camera image plane (CAM2) and find the corresponding 3D points on this dense mesh. When a touch event is detected within a particular camera's imagery, the associated 3D position on the surface is thus available through a constant-time lookup. The dense mesh is simplified and retopologized for animation purposes to create a 3D graphical model (GFX3), as described in the Setup section.

Figure 2 shows a summary of these coordinate spaces and their relationships.

3.2 Structured Coded Light

We employ the use of *structured coded light* (Salvi, Pagès, & Batlle, 2004) for two purposes: to obtain a three-dimensional model of the surface and to obtain correspondences among projector pixels, camera pixels, and the three-dimensional coordinates of the surface. First, we create a series of binary-coded images such that a given pixel is assigned a unique identifier based on the sequence of binary values for that pixel across all pattern images. We sequentially project these images onto the surface and capture imagery with each camera. We decode the patterns within each camera's imagery to obtain the identifiers for the projected pixels, which establishes a bidirectional correspondence between camera and projector pixels. Then, using the results of camera calibration, we triangulate these points in 3D space to obtain the 3D model and projector-to-3D correspondences.

Initially, we experimented with patterns using binary-coded rectangular stripes. However, we encountered illumination artifacts along the boundaries of these stripes—particularly across regions of high curvature on the rear-projection surfaces—making accurate localization of the stripes challenging. To address this, we created patterns of binary-coded white circles on a black background. First, we project the "full" pattern containing all possible white circles that appear throughout the pattern images. By detecting the circles present within each camera's capture of this image and storing their positions, we simplify the detection of circles when processing subsequent pattern captures, as we need consider only these locations and check for the presence of predominantly white or black pixels.

Creating patterns with a large number of small circles provides more correspondences than patterns with a smaller number of larger circles at the cost of decreased reliability. It is important to note that the sizes and shapes of the circles vary across a nonparametric surface, particularly in regions with high curvature; the size of the projected circles must be sufficiently observable across the entire surface. To generate additional correspondences without decreasing the size of the circles, we also created phase-shifted patterns—duplicate copies of pattern images shifted in one or both of the spatial dimensions by some amount (Salvi et al., 2004).

If the projection image extends beyond the boundaries of the surface, some circles may appear on other objects, resulting in erroneous points. We filter out decoded points that are significantly far away from the others. Next, we triangulate the decoded points and tessellate them into quads, filtering out any quads with drastically different surface normals than their immediate neighbors. Both of these filtering steps happen automatically. To form a dense, smooth mesh, we apply cubic interpolation to the projector-3D correspondences. We then use this dense mesh to create very dense correspondences among all camera pixels, all projector pixels, and the 3D points of the mesh, as discussed in the Lookup Table section.

For the physical–virtual patient head, we projected 40 pattern images and obtained 1588 reliable vertices out of a maximum of 2304. We then applied cubic interpolation to form a mesh with 70511 vertices (see Figure 3).

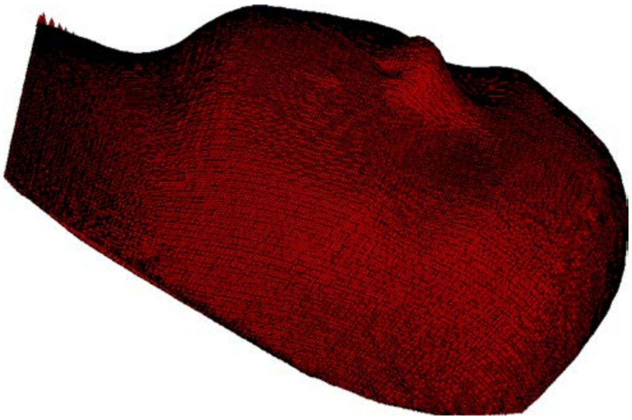

Figure 3. *The resulting structured coded light scan for the physical–virtual patient head after interpolating to* 70511 *vertices.*

3.3 Lookup Table

To generate camera-to-3D and projector-to-3D correspondences, we backproject all CAM2 and PRO2 pixels to rays and find their points of intersection on the dense mesh. To generate dense projector-to-camera correspondences, we take each projector pixel and find the corresponding 3D coordinate on the surface, which we then forward project onto each camera's image plane. We use this same process to generate dense camera-to-projector correspondences. As a result, we can create a lookup table storing correspondences between all camera pixels, projector pixels, and 3D graphics coordinates.

A lookup table relating the coordinates of $\mathcal{I}$ IR cameras, $\mathcal{P}$ projectors, and the 3D graphics space will have $2\mathcal{I} + 2\mathcal{P} + 3$ columns: an (x, y) coordinate for each IR camera, a (u, v) coordinate for each projector, and an (X, Y, Z) coordinate on the surface model. Each row of the table consists of points that correspond directly to one another in the aforementioned coordinate spaces. Every camera pixel and projector pixel appears in the table.

An illustration of the general format and dimensions of an example lookup table relating coordinates within $\mathcal{I}$ cameras, $\mathcal{P}$ projectors, and the 3D graphics space is shown in Table 1. For example, suppose the user touches a region on the surface that is imaged at position $(x, y)_1^{C_1}$ in camera C_1 and at position $(x, y)_1^{C_2}$ in

Table 1. *Example Lookup Table with Correspondences among $\mathcal{I}$ Cameras, $\mathcal{P}$ Projectors, and the 3D Graphics Space*

	Cameras (CAM2)			Projectors (PRO2)			GFX3
Row	C_1	$\cdots$	$C_{\mathcal{I}}$	P_1	$\cdots$	$P_{\mathcal{P}}$	3D graphics space
1	$(x,y)_1^{C_1}$	$\cdots$	$(x,y)_1^{C_{\mathcal{I}}}$	$(u,v)_1^{P_1}$	$\cdots$	$(u,v)_1^{P_{\mathcal{P}}}$	$(X,\Upsilon,Z)_1$
2	$(x,y)_2^{C_1}$	$\cdots$	$(x,y)_2^{C_{\mathcal{I}}}$	$(u,v)_2^{P_1}$	$\cdots$	$(u,v)_2^{P_{\mathcal{P}}}$	$(X,\Upsilon,Z)_2$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
n	$(x,y)_n^{C_1}$	$\cdots$	$(x,y)_n^{C_{\mathcal{I}}}$	$(u,v)_n^{P_1}$	$\cdots$	$(u,v)_n^{P_{\mathcal{P}}}$	$(X,\Upsilon,Z)_n$

Note. Each row contains corresponding points within each of these coordinate spaces. For instance, the 2D point $(x,y)_1^{C_1}$ in camera C_1 corresponds to the 2D point $(x,y)_1^{C_2}$ in camera C_2, the 2D point $(u,v)_1^{P_1}$ in projector P_1, and the 3D point $(X,\Upsilon,Z)_1$ on the surface.

camera C_2. The corresponding entry $(x,y)_1^{C_3}$ might be "empty" to indicate that camera C_3 is unable to image this particular location due to the camera's position and viewing angle. The entry $(u,v)_1^{P_1}$ in the same lookup table row indicates the corresponding coordinate in the space of projector P_1: a projection at this coordinate would appear at the location of the user's touch on the surface. Likewise, the corresponding coordinate $(X,\Upsilon,Z)_1$ stores the 3D position of this point within the graphical model. In the lookup table, these correspondences are stored within the same row, and a point in any of the coordinate spaces can be used as an index to find the corresponding point in any of the other spaces.

4 Prototype

We have developed a prototype system to demonstrate our method on nonparametric surfaces—in particular, the physical–virtual patient head. The prototype rig (see Figure 4) is composed of four Point Grey Blackfly monochrome cameras with IR filters (780 nm), two IR illuminators (850 nm), and one AAXA P300 pico projector. We use monochrome cameras with removable IR filters so that the cameras can capture visible-light imagery of the projections on the surface to build the CAM2 to PRO2 correspondences in the lookup table. Once this preprocessing stage is complete, the IR filters are reinserted so that the cameras can detect touch events in the IR spectrum. We also demonstrate the use of the prototype and method on another head surface and a planar surface in the Surfaces section.

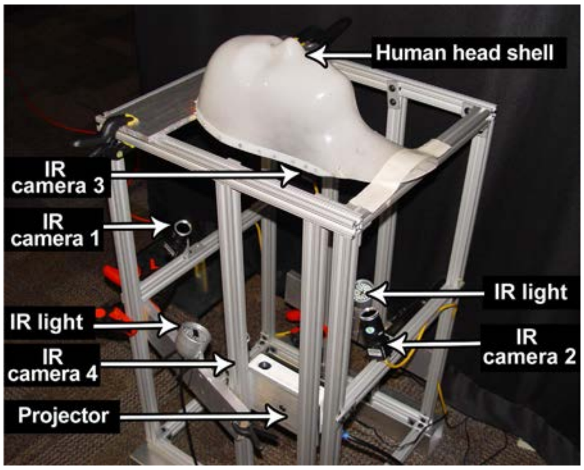

Figure 4. *In our prototype rig, we used four IR cameras, two IR light sources, and a projector.*

4.1 Setup

4.1.1 Camera, Projector, and IR Light Placement. We place our cameras and projectors in positions that allow for sufficient visual coverage of the head surface (see Figure 6). Likewise, we position the IR illuminators to provide ample lighting for touch detection.

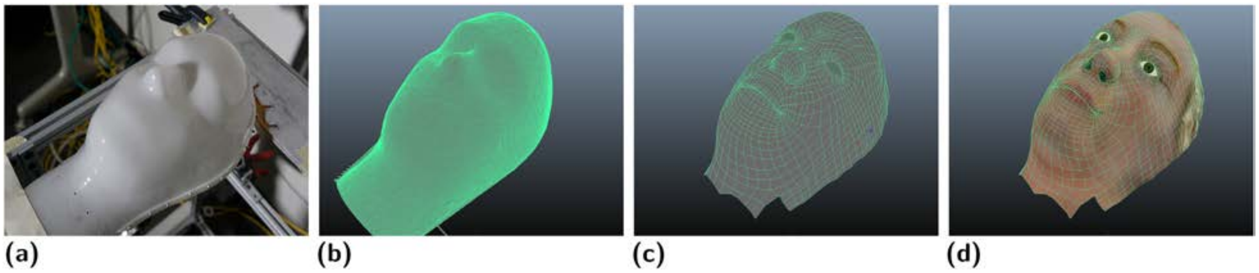

Figure 5. *Developing an animatable 3D mesh from the head surface. (a) Head surface with no projection. (b) Dense mesh from structured coded light scan. This mesh is not suitable for texturing or animation. (c) Retopologized mesh. (d) Retopologized mesh with head texture, suitable for animation.*

4.1.2 Camera and Projector Calibration. We calibrate the IR cameras using a standard checkerboard calibration approach in OpenCV (Bradski, 2015). Each camera is first individually calibrated; their computed intrinsics are used as initial estimates for subsequent stereo camera calibration among all pairs of cameras. We use the projector-to-3D correspondences obtained from the structured coded light scan to calibrate the projector: let $\mathbf{x}_i = (u, v)_i$ and $\mathbf{X}_i = (X, Y, Z)_i$ be a projector coordinate and its corresponding 3D model coordinate, respectively. We solve for the projection matrix P that best satisfies $\mathbf{x}_i = P\mathbf{X}_i$ for all i, from which we can obtain the intrinsics and extrinsics for the projector (Hartley & Zisserman, 2004). As a result, we obtain relative position information between all cameras, the projector, and the surface (see Figure 6). To simplify operations, we take one camera to be the origin of this 3D coordinate system.

4.1.3 3D Model. We use the 3D graphics space point cloud to construct a 3D model to be projected onto the surface (Figure 5[a]) using Unity (Unity—Game Engine, 2014). The point cloud is dense and not easily textured or animated (see Figure 5[b]). Based on this point cloud, we generate a lower density mesh that uses quads with an edge flow topology instead of the grid of quads produced by the structured coded light scan; this is more suitable for animation as the use of quads minimizes artifacts generated by subdivision during smoothing. Quads are suitable for edge-loop modeling, shown in Figures 5(c) and (d) (Murdock & Allen, 2006). Poles are vertices with three or five or more edges; they are positioned at specific locations to reduce artifacts during animation. For the physical–virtual patient head, we create eyeballs and inner mouth geometry and attach them to the mesh by merging vertices using Maya (3D Animation Software, Computer Animation Software | Maya | Autodesk, 2014). The mesh is textured (see Figure 5[d]) and then rigged using a combination of joints and blendshapes for various animations. A convex mesh is created for each semantically defined region of the face—for example, the eyes, nose, and mouth—and used as a collider in Unity so that touch events can trigger any region-specific responses.

The 3D model rigged with joints and blendshapes is exported to Unity and positioned in a 3D coordinate space based on the results of camera and projector calibration as in Figure 6—that is, the coordinates of the surface within Unity correspond to the triangulated coordinates from the structured coded light scan. The resolution of the Unity scene is set to match the maximum resolution of the projector (1920 × 1080 pixels) as described in the manufacturer's manual. A virtual projector—a camera in Unity—whose position and rotation match the physical rear projector's extrinsic parameters is created; additionally, we compute the field of view for this virtual projector using the projector calibration results. As in our prototype rig, the virtual projector is located underneath the surface. The resulting view rendered from this virtual projector is sent as imagery to the physical projector to be displayed on the actual surface.

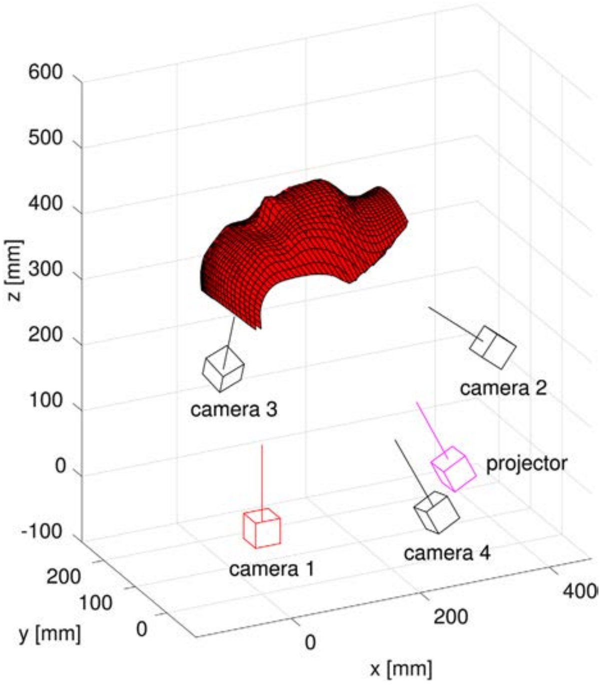

Figure 6. *The coordinate space containing the cameras, projector, and 3D surface for the physical–virtual patient head. The IR cameras and projector are shown as cubes with a line pointing along their principal axes. The 3D surface plotted here is simplified for visualization purposes.*

4.2 Run Time

To reduce latency, we developed a multithreaded application that performs image acquisition, touch detection, and rendering on separate threads. For each camera, one thread continuously captures new imagery, while a second thread segments potential touch events in this imagery into contours. A single thread processes the contours from all cameras and assigns confidence scores to them, with higher scores corresponding to likelier touch events. On average, the touch sensing thread processes potential touches across the four cameras of our prototype rig at 40–60 frames per second, and the rendering thread sends images to the projector at 60–80 frames per second.

4.2.1 Touch Sensing. Initially, each camera captures imagery of the surface when no touch events are occurring. New imagery captured by the cameras is compared to these background images, thresholded, and segmented into contours. Each *camera contour* is converted via the lookup table into a corresponding *projector contour* in the 2D projector space, where it is evaluated and scored. In order to assign a confidence score for a potential touch event, we consider the degree to which multiple cameras agree on the event as well as the reliability of each camera in imaging the particular region of the surface on which the potential event is occurring.

A region of high intensity that does not correspond to an actual touch may appear in a camera's imagery, for example, if the user's hand hovers close to the surface. These erroneous events must be discarded. We combine the projector contours converted from each camera into a single matrix with the same resolution as the projector. We declare cameras as "agreeing" on a touch event when their projector contours have sufficiently similar positions and sizes. We sum up the ratios of the area of c cameras agreeing on a projector contour to the area of the entire contour, weighted by empirically obtained values $w_a[c]$ for $c \in \{1, 2, 3, 4\}$, and normalized by the sum of the weights.

Each camera images different regions of the surface with varying amounts of distortion. A camera with a better "view" of a particular region should have its contributions weighted higher than one with a comparatively less accurate view. The CAM2 to GFX3 correspondences in the lookup table provide rays connecting each camera's position to a 3D position on the surface, passing through a given pixel in the camera's image plane. We compute the surface normal at this point of intersection. Regions that have surface normals that are nearly parallel to the camera ray are locally "flat" when imaged by that camera and should appear with minimal distortion; potential touch events that occur in such regions are more reliable. Regions with surface normals nearly orthogonal to the camera ray will be highly distorted when imaged by that camera. Hence, the dot products between surface normals and camera rays form a measure of "viewing" confidence. These dot products are precomputed for all pixels of all camera image planes. At run time, we aggregate these dot products for the segmented contours within camera imagery and weight the respective camera contributions accordingly.

The camera agreement and camera reliability measures are added together as an overall confidence score. Projector contours with a sufficiently high confidence are accepted as touch events and sent to the rendering system to trigger appropriate graphical or other responses.

4.2.2 Rendering. Accepted touch events are sent to Unity in two forms, using the lookup table to perform the necessary coordinate conversions. First, a monochrome image mask in 2D projector space is created in which touch events are represented as white pixels. This mask is overlaid onto the rendered output from Unity, allowing for the touch event to be displayed in a different color. The touch event is also converted to 3D graphics space coordinates and sent to Unity for the activation of blendshapes. The upper and lower lips, upper and lower eyelids for both eyes, and nose of the physical–virtual patient head have associated blendshapes. When a touch occurs, the rendering system uses Unity's physics engine to determine if it happens inside a collider region associated with one of these blendshapes; if so, the blendshape is activated. As the touch position varies over the surface, we compute a percentage between the current touch position and the maximum position the blendshape can reach to update the blendshape accordingly. In this way, a user can "spread" the lips of the patient with her fingers, revealing the teeth and gums (see Figure 1). If the touch event triggers an audio response, lip-synced animations are activated to move the mouth appropriately. Finally, a shader in Unity renders the buffers for the mesh in reverse order so that the eyeballs and mouth bag are rendered correctly.

4.3 Evaluation

During normal operation of our system, touch events detected in the 2D camera space are mapped via the lookup table to the 3D graphics space, and the appropriate graphical effect is rendered based on the location of the touch. To assess the detection accuracy of our prototype, we effectively reversed this process for a large number of samples (over 600). Specifically, we projected a grid of visual targets—a set of concentric circles with a "crosshair" at the center—onto the head shell, one at a time. A user was instructed to touch each visual target with a small wand as carefully and precisely as possible, and both the detected location of each touch event in PRO2 and the corresponding 3D position in GFX3 were recorded.

The positions of the projected targets and user touches are shown in Figure 7 for an earlier prototype of our system (described in Hochreiter et al., 2015). The mean distance between the projected circles and the detected touch events in projector space is 16.7 pixels with a standard deviation of 12.5 pixels, which corresponds to a mean 3D distance in the 3D graphics space of 4.3 mm and standard deviation of 4.3 mm (same as mean). In general, the largest errors correspond to highly curved regions of the shell—such as the nose—and to a portion of the shell that is largely parallel to the projector's principal axis. In such regions a small pixel displacement in projector space can result in a comparatively large displacement in physical 3D coordinates. If we ignore the points with the top 10% error, the mean distance in projector space is 13.2 pixels with a standard deviation of 6.5 pixels, which corresponds to a mean 3D distance in the 3D graphics space of 3.2 mm and standard deviation of 1.7 mm. Because the errors are static, we could potentially factor corrections into our lookup table if needed. However, we would first attempt to improve the camera and projector placement to minimize error sensitivities in the regions with higher accuracy requirements.

It is worth noting that while the user attempted to touch the surface carefully and precisely during this experiment, there is undoubtedly noise introduced by his inability to know whether or not his touching was indeed centered on the circle. In fact, for finger touches in general it is not clear what being "centered" means as fingers pressed onto a surface are not circular. In the end what matters is the user's perception of where he or she is touching and how the graphical content responds, all in the context of the application.

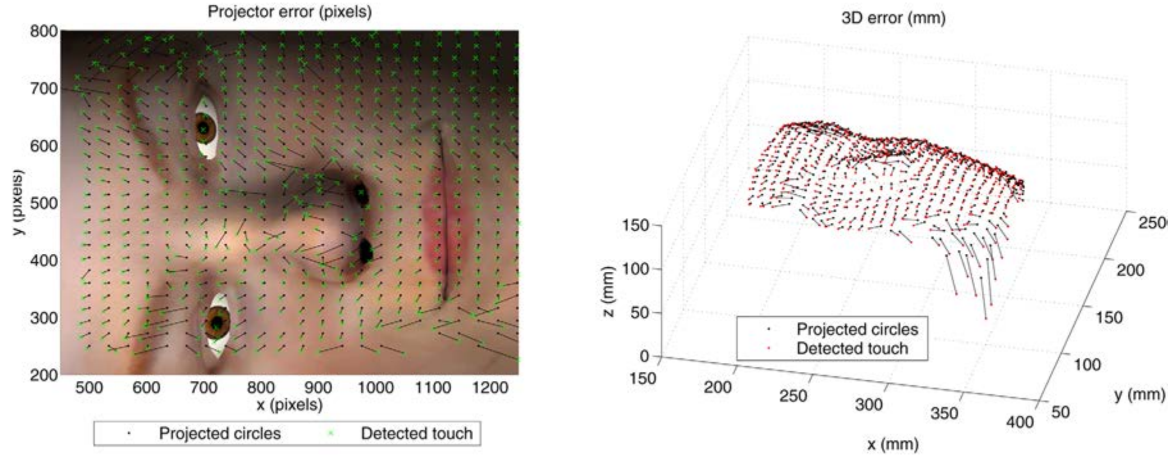

Figure 7. *Errors between a user's touch and the touch location detected by the system. Left: Errors in 2D projector space (average* 16.7 *pixels with a standard deviation of* 12.5 *pixels). Right: Errors in 3D graphics space (average and standard deviation of* 4.3 *mm).*

4.4 Surfaces

We demonstrate the general nature of our method using three rear-projection surfaces: two different human head-shaped surfaces and a planar surface (see Figure 8). The first head surface is modeled after a real human head—it serves as our physical–virtual patient head for training applications. The second head surface has a more robotic or cartoon-like head shape. Finally, the planar surface simply demonstrates the applicability of this approach on an ordinary parametric surface. The first row of Figure 8 shows these surfaces without any projected imagery.

The pipeline for turning each of these three surfaces into touch-sensitive ones is the same. First, we scan each surface using the structured coded light approach detailed earlier. After retopologizing the resulting 3D mesh, we rig it with joints and blendshapes as appropriate and texture it. We semantically define regions of the 3D model so that touch events can trigger region-based responses. Finally, we build lookup tables relating camera pixels, projector pixels, and 3D surface points. In the remaining rows of Figure 8 we show the detection of touch events, the 3D model of the virtual head, the resulting projection onto the surface, and touch-triggered blendshape changes on the three surfaces.

5 Application to Stroke Assessment

We have demonstrated our touch-rendering methods using a physical–virtual patient head in a medical training scenario. Specifically, we chose a stroke scenario to illustrate the capabilities of our prototype for assessment of patients. The American Stroke Association (ASA) suggests the FAST mnemonic for early stroke recognition and survival: Face drooping, Arm weakness, Speech difficulty, and Time to call emergency services (Warning Signs of Stroke, 2014). At a receiving center, a healthcare provider will perform a neurological and a psychomotor assessment that includes examining the patient for an asymmetric smile, lid lag, irregular pupils, garbled speech, the ability to show teeth, numbness, or difficulty sensing touch. At present, healthcare providers are trained in medical and nursing schools to recognize these findings. Training consists of task trainers and standardized patients. The use of simulation for stroke training is currently limited, possibly due to the limitations of current simulator technology, which has not been designed with stroke-related focal neurologic assessment in mind (Garside, Rudd, & Price, 2012).

Most human patient simulators do not have the capability to respond and exhibit typical stroke symptoms, such as smile asymmetry, visual gaze, and sensation

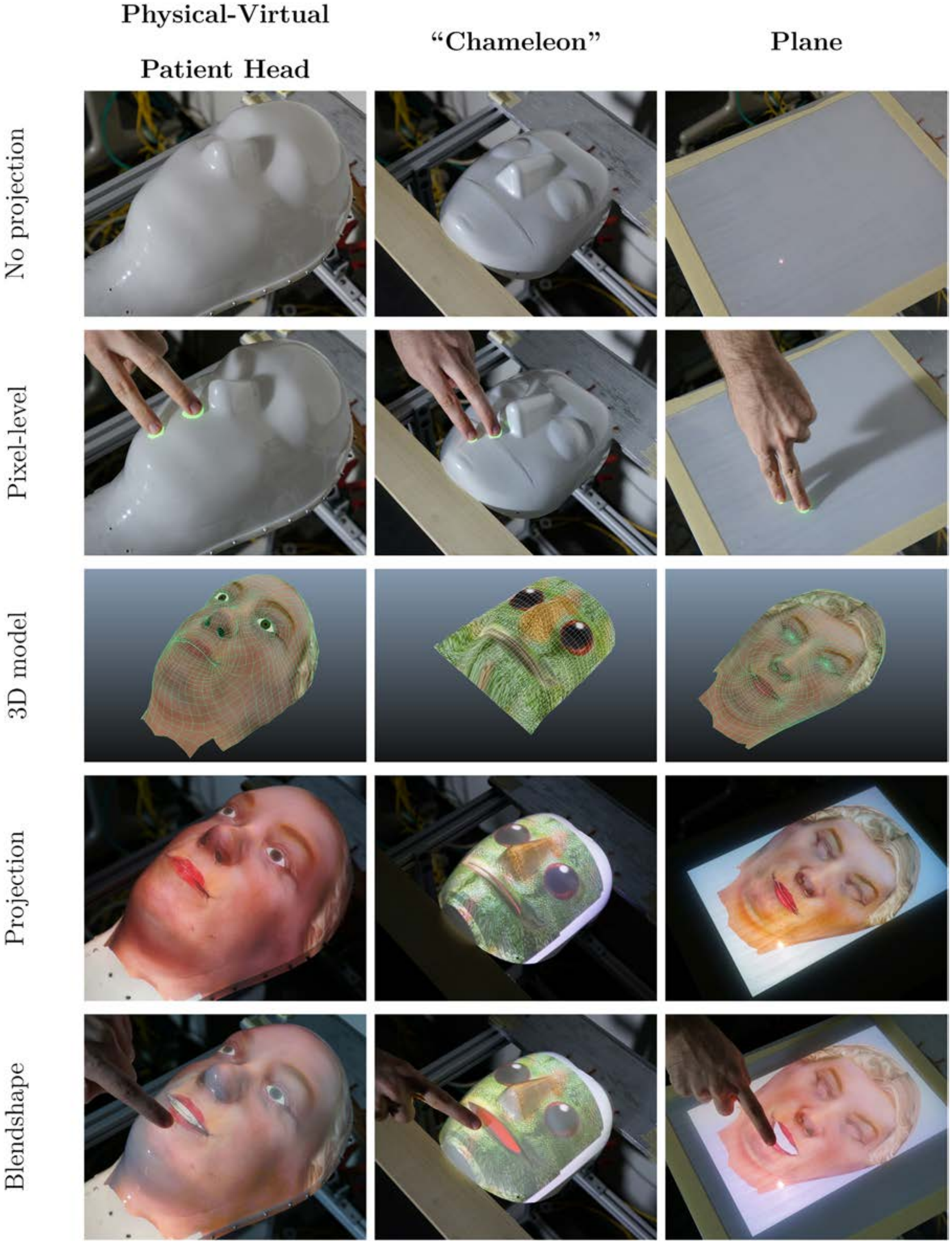

Figure 8. *Touch sensing on various nonparametric and parametric surfaces. The second and fifth rows show the effects of two kinds of touch events: pixel-level changes at the points of contact and geometric changes due to a touch-triggered lower-lip-tug blendshape, respectively.*

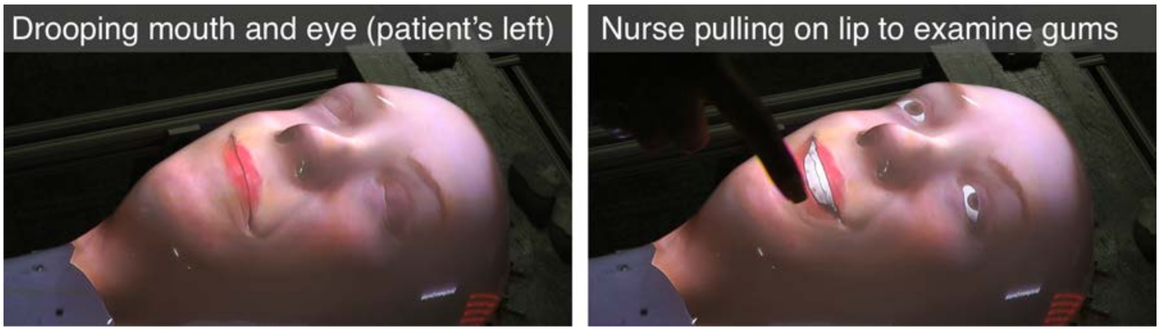
Drooping mouth and eye (patient's left)
Nurse pulling on lip to examine gums

Figure 9. *Example stroke scenario. The patient's mouth and eye are drooping on her left side, where she cannot sense touch, and she is unable to smile. Animated blendshapes still function properly on the drooping mouth. Figure 1 shows similar interactions on a healthy patient.*

awareness. Similarly, a trained human actor would be unable to simulate facial droop and lid lag in a realistic manner. However, our physical–virtual patient head is capable of exhibiting many of the visual and auditory symptoms of a stroke, as shown in Figure 9. The projected 3D model can be switched between "normal" mode and "neurological event" mode whenever desired.

Moreover, as the physical–virtual patient head can respond to touch events, a healthcare practitioner can follow the aforementioned neurological and psychomotor assessments. Compared to a two-dimensional avatar or a non-touch-sensitive robot, this could afford a more natural and engaging training experience. To this end, we apply our physical–virtual patient head as a simulated "patient" in a stroke assessment training scenario. Various regions of the graphical model are semantically defined: the forehead, chin, left and right eyes, left and right cheek, and nose. A detected touch event, once converted from 2D camera space to 3D graphics space, can be classified as belonging to one of these parts of the face, and a prerecorded sound file of the patient verbally identifying the location of the touch can be triggered. In this way, the healthcare provider can test for sensory function by touching different locations on the head and asking the patient to indicate where the touch is perceived. Blendshapes that control the opening and closing of the graphical model's eyes and lips are created and linked to the touch system so that the trainee can examine the simulated patient by physically interacting with it. In neurological event mode, when the patient's mouth and eye are drooping on the left side, the blendshapes function naturally.

As appropriate, additional prerecorded verbal and nonverbal responses can be triggered manually. Joint animations in the jaw are activated when the patient speaks, and the lips are synchronized to the audio source. Nonverbal responses include emotions, such as happiness, surprise, contempt, or anger, and nonemotional behaviors, such as closing eyes, pupil dilation, eyebrow movement, and tongue movement. Each verbal and nonverbal response behaves appropriately depending on the current mode of the patient; for instance, the patient exhibits slurred speech, asymmetric eyebrow movement, and impaired ability to recognize the locations of some touches in neurological event mode.

6 Future Work

We plan to extend the proposed technique to handle multiple projectors, allowing for touch sensing and rendering on larger surfaces, by expanding the lookup table to provide correspondences in the additional coordinate spaces of these projectors. The primary difficulty concerns the blending of projected imagery in regions of overlap, as these regions can have arbitrary shapes, sizes, and orientations with respect to the nonparametric surfaces. If multiple projectors can "cover" a particular section of the surface, one projector may be able to do so with less distortion or brightness falloff depending on its position and orientation. There are also tradeoffs to consider regarding the number of projectors used and their positioning: limiting the number of projectors and amount of overlap can help reduce the complexity of the overall system, but this may reduce the brightness and contrast in certain regions.

To prevent the need for time-consuming recalibration, we are considering automatic continuous self-calibration methods. Constant updates to the lookup table should improve correspondence accuracy and increase the overall reliability of the system.

We are also investigating the ability to sense proximity or contact with objects beyond fingers and hands, both passive (e.g., medical instruments) and active (e.g., a wand with an IR LED in the tip). The ability to detect non-contact events will allow for additional interactions, such as having the patient follow a moving flashlight with the eyes.

We are experimenting with creating our own surfaces through a combination of 3D printing and vacuforming. To be suitable, a surface must satisfy a few constraints. First, it must allow for the formation of a sharp projection image when viewed from above. If projecting an image onto the surface leads to visible light reflections below the surface, it will disrupt the scanning process. Similarly, the surface must permit IR light to pass through it without the presence of intense hot spots in the IR domain to allow for accurate touch sensing. We are also investigating different translucent skin materials to simulate a tactile feeling closer to that of human skin. The material we are considering appears to be simultaneously opaque enough to support the formation of a projector image and transparent enough to pass IR light in both directions, thus maintaining our ability to achieve optically-based touch sensing on a skin-like surface.

Finally, from an application standpoint, we are working to apply these methods to an entire life-size physical–virtual body.

7 Conclusion

We have presented a generalizable approach for touch detection on nonparametric rear-projection surfaces using IR cameras. We developed a physical prototype and applied our method to three rear-projection surfaces on which a projector displays a graphical model. Touch events on these surfaces prompt visual and auditory responses in the model. We demonstrated an example application scenario on a head-shaped surface where touch events and resulting graphical updates could be used to train a healthcare professional in stroke assessment.

A primary goal of our approach is a direct touch-graphics architecture to ensure the appropriate semantic response of the interactive graphics application while minimizing the perceived latency. Toward this end, our method includes precalibration steps to populate a lookup table that is used during run time to directly map points in the 2D camera space to the 3D graphics space, as well as decoupled touch and rendering coordinate spaces and processes to allow each to proceed at an appropriate rate.

We analyzed the accuracy of our prototype using a reverse-touch approach in which we chose a 2D projector point, attempted to touch that point, and assessed how well the detected touch event mapped back into the projector space. The results revealed greater static error (mismatch) in regions where camera/projector-surface normals are closer to being orthogonal (ill-conditioned). We should be able to reduce this error by various means, including better camera/projector placement and the use of dense correction factors.

We look forward to extending our methods to a full human body surface with multiple projectors and cameras and to evaluating more extensive medical training scenarios incorporating hands-on full-body diagnostic, therapeutic, and comfort touch.

References

Benko, H. (2009). Beyond flat surface computing: Challenges of depth-aware and curved interfaces. *Proceedings of the 17th ACM International Conference on Multimedia*, 935–944.

Benko, H., Wilson, A. D., & Balakrishnan, R. (2008). Sphere: Multi-touch interactions on a spherical display. *Proceed-*

ings of the 21st Annual ACM Symposium on User Interface Software and Technology, 77–86.

Bradski, G. (2015, October 31). *Dr. Dobb's Journal of Software Tools.* OpenCV. Retrieved from https://github.com/Itseez/opencv/wiki/citeOpenCV

Chuah, J. H., Robb, A., White, C., Wendling, A., Lampotang, S., Kopper, R., & Lok, B. (2013). Exploring agent physicality and social presence for medical team training. *Presence: Teleoperators and Virtual Environments*, *22*(2), 141–170.

Dietz, P., & Leigh, D. (2001). DiamondTouch: A multi-user touch technology. *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, 219–226.

Garside, M. J., Rudd, M. P., & Price, C. I. (2012). Stroke and TIA assessment training: A new simulation-based approach to teaching acute stroke assessment. *Simulation in Healthcare*, *7*(2), 117–122.

Gu, J., & Lee, G. (2011). TouchString: A flexible linear multi-touch sensor for prototyping a freeform multi-touch surface. *Proceedings of the 24th Annual ACM Symposium Adjunct on User Interface Software and Technology*, 75–76.

Han, J. Y. (2005). Low-cost multi-touch sensing through frustrated total internal reflection. *UIST '05: Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, 115–118.

Hartley, R. I., & Zisserman, A. (2004). *Multiple view geometry in computer vision* (2nd Ed.). Cambridge, UK: Cambridge University Press.

Hochreiter, J., Daher, S., Nagendran, A., Gonzalez, L., & Welch, G. (2015). Touch sensing on non-parametric rear-projection surfaces: A physical–virtual head for hands-on healthcare training. *2015 IEEE Virtual Reality*, 69–74.

Kotranza, A., & Lok, B. (2008). Virtual human + tangible interface = mixed reality human: An initial exploration with a virtual breast exam patient. *Virtual Reality Conference, IEEE*, 99–106.

Lincoln, P., Welch, G., Nashel, A., State, A., Ilie, A., & Fuchs, H. (2010). Animatronic shader lamps avatars. *Virtual Reality*, *15*(2), 1–14.

Majumder, A., & Brown, M. S. (2007). *Practical multi-projector display design*. Natick, MA: A. K. Peters, Ltd.

Matsushita, N., & Rekimoto, J. (1997). HoloWall: Designing a finger, hand, body, and object sensitive wall. *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, 209–210.

3D Animation Software, Computer Animation Software | Maya | Autodesk. (2014). Retrieved from http://www.autodesk.com/products/maya/overview

Murdock, K. L., & Allen, E. (2006). *Edgeloop character modeling for 3D professionals only.* New York: John Wiley & Sons, Inc.

Warning Signs of Stroke. (2014). Retrieved from http://www.stroke.org/site/DocServer/TIA.pdf?docID=405

Rekimoto, J. (2002). SmartSkin: An infrastructure for free-hand manipulation on interactive surfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 113–120.

Rivera-Gutierrez, D., Welch, G., Lincoln, P., Whitton, M., Cendan, J., Chesnutt, D. A., et al. (2012). Shader lamps virtual patients: The physical representation of virtual patients. *Studies in Health Technology and Informatics, Volume 173: Medicine Meets Virtual Reality, 19*, 372–378.

Roudaut, A., Pohl, H., & Baudisch, P. (2011). Touch input on curved surfaces. In D. S. Tan, S. Amershi, B. Begole, W. A. Kellogg, & M. Tungare (Eds.), *CHI* (pp. 1011–1020). Victoria, BC: ACM.

Salvi, J., Pagès, J., & Batlle, J. (2004). Pattern codification strategies in structured light systems. *Pattern Recognition*, 37, 827–849.

Unity—Game Engine. (2014). Retrieved from http://unity3d.com/

Villar, N., Izadi, S., Rosenfeld, D., Benko, H., Helmes, J., Westhues, J., et al. (2009). Mouse 2.0: Multi-touch meets the mouse. *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, 33–42.

Wang, F., Cao, X., Ren, X., & Irani, P. (2009). Detecting and leveraging finger orientation for interaction with direct-touch surfaces. *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, 23–32.

Welch, G., Rivera-Gutierrez, D., Lincoln, P., Whitton, M., Cendan, J., Chesnutt, D. A., et al. (2011). Physical manifestations of virtual patients. *Simulation in Healthcare*, *6*(6), 488.

Wilson, A. D. (2004). TouchLight: An imaging touch screen and display for gesture-based interaction. *Proceedings of the 6th International Conference on Multimodal Interfaces*, 69–76.