Touch Sensing on Non-Parametric Rear-Projection Surfaces: A Physical-Virtual Head for Hands-On Healthcare Training

Jason Hochreiter* Salam Daher † Arjun Nagendran ‡ Laura Gonzalez § Greg Welch ¶

University of Central Florida

ABSTRACT

We demonstrate a generalizable method for unified multitouch detection and response on a human head-shaped surface with a rear-projection animated 3D face. The method helps achieve hands-on touch-sensitive training with dynamic physical-virtual patient behavior. The method, which is generalizable to other non-parametric rear-projection surfaces, requires one or more infrared (IR) cameras, one or more projectors, IR light sources, and a rear-projection surface. IR light reflected off of human fingers is captured by cameras with matched IR pass filters, allowing for the localization of multiple finger touch events. These events are tightly coupled with the rendering system to produce auditory and visual responses on the animated face displayed using the projector(s), resulting in a responsive, interactive experience. We illustrate the applicability of our physical prototype in a medical training scenario.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Animations, Artificial, Augmented, and Virtual Realities; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality; I.3.8 [Computer Graphics]: Applications

1 INTRODUCTION

Touch sensing for interactive computer graphics is typically implemented on surfaces that can be mathematically parameterized via analytical equations, such as flat/planar surfaces or spheres. We present a method for unified multitouch detection and response on a non-parametric surface with rear-projected animated content. Similar to [3] our method comprises infrared (IR) cameras, projectors, IR light sources, and a rear-projection surface. IR light from below the surface passes through the surface, reflects off the human 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. The touch events are tightly coupled with the rendering system to produce a highly responsive interaction.

While the method is generalizable to other non-parametric surfaces, we illustrate the applicability of our physical prototype in a medical training scenario using a human head-shaped surface. Touch events on the surface prompt graphical and audio responses—for instance, a participant can open the patient's mouth by spreading the lips using her fingers (Fig. 1), and certain actions result in the patient speaking. Additionally, we devised a medical training scenario incorporating these features. Our approach is complementary to existing approaches: As with human patient actors and robotic mannequins, our approach offers a direct hands-on paradigm without the need for a head-worn display (HWD) or a separate virtual content display. Like virtual patients, we are able to present abnormal physical findings that can be diagnosed via visual appearance, either passively (just look) or interactively (touch).

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 gums.

In Section 2, we present a brief introduction to existing touch sensing and projection techniques on non-parametric surfaces. Our proposed method relies on the construction of a lookup table relating points in various coordinate spaces, which are detailed in Section 3. We provide a description of our prototype in Section 4 and its applicability to a stroke assessment scenario in Section 5. Finally, we discuss potential extensions to this approach in Section 6 and provide a brief summary in Section 7.

2 RELATED WORK

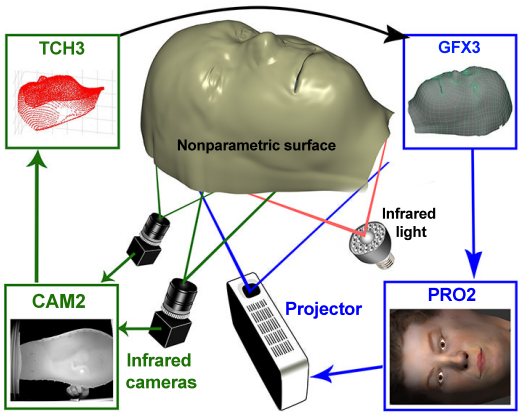
A number of methods exist for touch sensing on various surfaces. Capacitive sensing approaches [6, 8, 17] are popular though they typically assume flat surfaces. Conductive materials can be molded to fit more complex surfaces [17], however the density of the capacitive elements must be sufficiently high compared to the complexity (spatial frequency) of the surface. In any case, capacitive methods would interfere with the projected imagery.

Computer vision approaches use either visible or IR light to detect touch events. One class of methods uses *frustrated total internal reflection* [9, 19, 21, 22], 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 [2, 3, 13, 24]. Touch sensing has been performed on planar and parametric non-planar surfaces, such as spheres [2, 3], in this manner. When extending this technique to arbitrary surfaces, it can be challenging to find an appropriate mapping between detected touch events 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 [12]. Piecewise linear methods have also been applied to parametric nonplanar displays. For non-parametric display surfaces, one approach involves finding a mapping between projected imagery and the viewer's eye, represented

*email: jhochrei@cs.ucf.edu
†email: salam@knights.ucf.edu
‡email: arjun@cs.ucf.edu
§email: Laura.Gonzalez@ucf.edu
¶email: welch@ucf.edu

IEEE Virtual Reality Conference 2015
23 - 27 March, Arles, France
978-1-4799-1727-3/15/$31.00 ©2015 IEEE



CAM2—2D camera space: touch events are detected and localized within the imagery obtained by each camera.

PRO2—2D projector space: this space is used to compute a mapping between touch coordinates and projector coordinates. This allows for rendering via pixel shaders and aids in computation of the sparse 3D touch space.

TCH3—sparse 3D touch space: computed via points projected on the surface, the sparse 3D touch space serves as an intermediary between touch events detected by the infrared cameras and actual points on the 3D model being projected onto the surface.

GFX3—dense 3D graphics space: during rendering, 3D vertices on the model can be adjusted in response to touch or other events.

Figure 2: Descriptions of the various coordinate spaces of the proposed method. Touch events detected in the 2D camera space (CAM2) are converted to projector coordinates (PRO2) and 3D graphics coordinates (GFX3), as needed. This method is general to any non-parametric surface; we show imagery of the physical-virtual head only for the purpose of illustration.

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, researchers have developed physical-virtual patients (avatars and agents) [11, 23, 18]. Kotranza and Lok [10] 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. [5] have since surveyed existing research and formalized the idea of physicality for embodied conversational agents.

Here, we present a computer vision based method employing IR light to achieve tightly coupled touch sensing and rendering capabilities on a rear-projection surface. This approach will support a wide range of physical-virtual medical training scenarios involving touch including diagnostic, therapeutic, and comfort touch.

3 Coordinate Spaces

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 use a lookup table to directly link the coordinate spaces. This has the additional benefit of decoupling the rendering and touch temporally so that rendering events can be carried out independently (at a higher rate than) the touch events. In total, we have four types of coordinate spaces: 2D spaces corresponding to the cameras, 2D spaces corresponding to the projector, and 3D spaces corresponding to both the touch sensing and rendering on the 3D physical surface. We describe these four spaces next.

CAM2: Using the touch sensing system (Section 4.2.1) touch events are detected and localized as 2D (x,y) coordinates in the **2D camera space (CAM2)**. Each camera capable of imaging the touch event records its position within its coordinate space.

PRO2: A detected touch event must trigger an appropriate graphical response in the rendering system, requiring a mapping between camera coordinates and 2D (u,v) projector coordinates in the **2D projector space (PRO2)**. To obtain these mappings, we project white circles with a radius of 13 pixels across a grid of (u,v) projector coordinates, with 11 pixel spacing in each dimension, on a black background. Each IR camera segments the circle and records its centroid. In total we captured 11706 such circles. This provides a direct mapping between a projector pixel and pixels within all cameras. The resolution of the mapping is impacted by the number of circles projected; we perform cubic interpolation on the CAM2 to PRO2 correspondences to provide sufficient resolution.

TCH3: In a typical multiple view geometry scenario, some form of feature matching provides correspondences between the different views. As our rear-projection surface is both uniform in color and smooth, it exhibits few features; it is therefore difficult to find correspondences via feature matching approach. Instead, we can use each of the aforementioned projected circles as manual "features," treating their detected positions in each camera's imagery as corresponding points. These correspondences are triangulated in three-dimensional space, resulting in 3D (X,Y,Z) vertices in the **sparse 3D touch space (TCH3)**. Together with the 3D positions of the cameras as obtained through calibration (described in Section 4.1), this provides a three-dimensional model of the entire setup, as shown in Fig. 4.

GFX3: While TCH3 is a 3D triangulated point cloud of the surface, providing correspondences between points imaged by cameras and 3D locations on the surface, its resolution is limited both by the number of circles projected and by the imaging resolution of the cameras. Because the rendered and projected effects need to be relatively dense we create a separate renderable 3D model of the surface as a collection of vertices (X',Y',Z') in a **dense 3D graphics space (GFX3)** as described in Section 4.1. We use TCH3 as an intermediary between camera pixels in CAM2 and the 3D model in GFX3. Specifically we use an iterative closest point algorithm (implementation by Per Bergström [4]), estimating a rotation and translation matrix that relates the high resolution 3D model of the surface in GFX3 to the space of the triangulated point cloud in TCH3. From here, we backproject all 2D pixels within each camera image plane (CAM2) and find the corresponding 3D points on the model (GFX3). When a touch is detected within a particular camera, the associated 3D position on the graphical model is thus available as a constant-time lookup.

3.1 Lookup Table

A lookup table relating the coordinates of $\mathcal{I}$ IR cameras, $\mathcal{P}$ projectors, the sparse 3D touch space, and the dense 3D graphics space will have $2\mathcal{I}+2\mathcal{P}+3+3$ columns: an (x,y) coordinate for each IR camera, a (u,v) coordinate for each projector, an (X,Y,Z) coordinate within the sparse 3D touch space, and an (X',Y',Z') coordinate

Row	Cameras (CAM2) C_1	$\cdots$	$C_{\mathcal{I}}$	Projectors (PRO2) P_1	$\cdots$	$P_{\mathcal{P}}$	TCH3 Sparse 3D touch space	GFX3 Dense 3D graphics space
1	$(x,y)_1^{C_1}$	$\cdots$	$(x,y)_1^{C_{\mathcal{I}}}$	$(u,v)_1^{P_1}$	$\cdots$	$(u,v)_1^{P_{\mathcal{P}}}$	$(X,Y,Z)_1^{\text{TCH3}}$	$(X',Y',Z')_1^{\text{GFX3}}$
2	$(x,y)_2^{C_1}$	$\cdots$	$(x,y)_2^{C_{\mathcal{I}}}$	$(u,v)_2^{P_1}$	$\cdots$	$(u,v)_2^{P_{\mathcal{P}}}$	$(X,Y,Z)_2^{\text{TCH3}}$	$(X',Y',Z')_2^{\text{GFX3}}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
n	$(x,y)_n^{C_1}$	$\cdots$	$(x,y)_n^{C_{\mathcal{I}}}$	$(u,v)_n^{P_1}$	$\cdots$	$(u,v)_n^{P_{\mathcal{P}}}$	$(X,Y,Z)_n^{\text{TCH3}}$	$(X',Y',Z')_n^{\text{GFX3}}$

Table 1: Example correspondence lookup table; each row contains correspondences among all coordinates it contains.

within the dense 3D graphics space. 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, the sparse 3D touch space, and the dense 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 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 $(X,Y,Z)_1^{\text{TCH3}}$ stores the 3D position of this point in the sparse triangulated point cloud obtained from the camera imagery, and $(X',Y',Z')_1^{\text{GFX3}}$ 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.

4 Prototype

We adapt the methodology for touch detection and graphical response on non-parametric surfaces on a 3D physical-virtual human head. Our prototype rig (Fig. 3) contains four Point Grey Blackfly monochrome cameras with IR filters (780nm), two IR illuminators (850nm), and one AAXA P300 pico projector. We use monochrome cameras with removable IR filters so that the cameras can capture imagery of the projections on the surface, which appear in the visible light spectrum, 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.

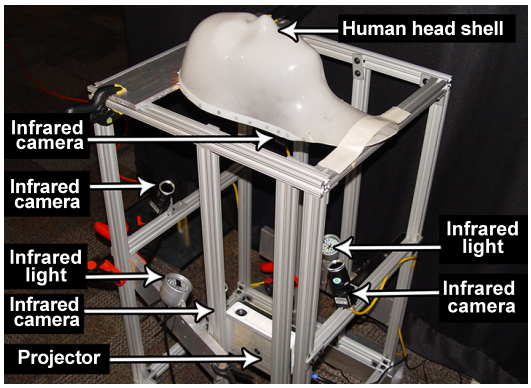


Figure 3: In our prototype rig, we used four IR cameras, two IR light sources, and a projector.

4.1 Setup

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 Fig. 4). Likewise, we position the IR illuminators to provide ample lighting for touch detection.

Camera calibration: we calibrate the IR cameras using a standard checkerboard calibration approach in OpenCV. Each camera is calibrated individually; their computed intrinsics are used as initial estimates for subsequent stereo camera calibration among all pairs of cameras. As a result, we obtain relative position information between all cameras. To simplify operations, we take one camera to be the origin of the sparse 3D touch space.

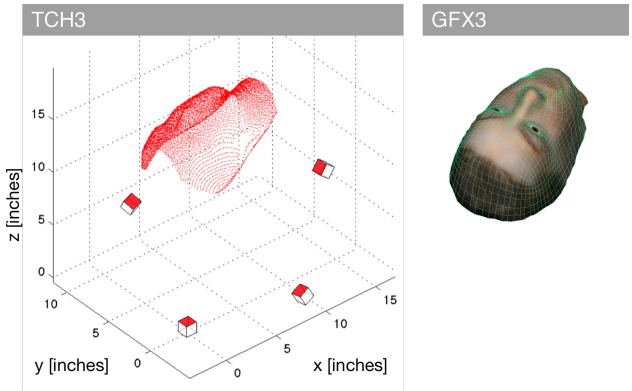


Figure 4: The sparse 3D touch space (TCH3) of our physical setup, which links touch events to the dense 3D graphics space (GFX3), shown on the right for comparison. The IR cameras are shown as cubes; the colored face of each cube represents the orientation of that particular camera.

Dense 3D information: photogrammetric techniques are used to generate a dense scan of the surface using overlapping photographs from all angles around it. To provide suitable features, a colored checkerboard is projected onto the surface (Fig. 5b). This greatly improved the accuracy of the scan, as the shell is of a uniform color (Fig. 5a). The photographs are sent to Autodesk 123D Catch, a free software package that converts a collection of photographs of an object into a dense 3D mesh [1]. The resulting mesh is cleaned, scaled and aligned using physical measurements of the rig.

The model obtained via 123D Catch is very dense and not easily textured or animated (Fig. 5c). Based on the dense mesh, we generate a lower density one that uses quads with an edge flow topology instead of triangles, which is more suitable for animation as the use of quads minimizes artifacts generated by subdivision during smoothing. The nature of quads makes them the basis of edge-loop modeling used here (Fig. 5d) [15].

Poles are vertices with three or five or more edges; they are positioned at specific locations to reduce artifacts during animation. For our head shell, eyeballs and inner mouth geometry are created and attached to the mesh by merging vertices using Maya [14]. The

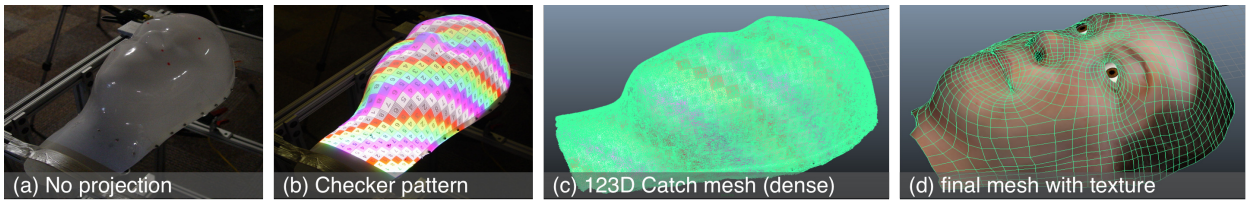


Figure 5: Developing an animatable 3D mesh from the head surface. (a) Head surface with no projection. (b) Head surface with colored checkerboard projected to aid photogrammetry. (c) Resulting dense mesh from 123D Catch; this mesh is not suitable for texturing or animation. (d) Final retopologized mesh with face texture (GFX3). This mesh is more appropriate for animation.

mesh is textured using one unwrapped image and then rigged using a combination of bones and blendshapes for various animations.

The animated 3D model is exported to Unity3D [20] and positioned in a scene with scene size matching the maximum resolution of the projector (1920×1080 pixels) as described in the manufacturer's manual. A virtual camera whose field of view, position, and rotation approximately match the rear projector's intrinsic and extrinsic parameters is created, using information from the manufacturer's specification sheet and measurements from the rig. As in the physical rig, it is located underneath the surface. The resulting rendered view from the camera is sent as imagery to the projector.

4.2 Run Time

Image acquisition, touch detection, and rendering are all executed 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 assigns confidence scores to potential touches across the four cameras of our prototype rig at 40 to 60 frames per second, while the rendering thread sends images to the projector at 60 to 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, e.g., if the user's hand hovers close to the surface. These erroneous events must be discarded. We combine the projector contours of 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 various 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 TCH3 correspondences in the lookup table provide rays connecting each camera's position and 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.

These two measures are added together as an overall confidence score. Projector contours with a sufficiently high confidence are accepted as touch events.

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 mask image in 2D projector space is created where touch events are represented as white pixels. This is sent to a pixel shader, which renders the touch event in a different color. The touch event is also converted to dense 3D graphics space and sent to Unity for the activation of blendshapes. If touch occurs near a region with an associated blendshape—including the upper and lower lips, the upper and lower eyelids for both eyes, and the nose—the blendshape is activated. As the touch position varies over the surface, each blendshape updates accordingly. In this way, a human participant can use her fingers to "spread" the lips of the model, revealing the gums (Fig. 1). If the touch event triggers an audio response, a bone animation is triggered that opens and closes the mouth. 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 camera space are mapped (via the lookup table) to the graphics space, and the graphical effect is rendered based on the results. 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 the detected location of each touch event in PRO2 and the corresponding 3D position in GFX3 (found via the lookup table) were both recorded. The collected data allows for visual and quantitative comparisons between points in the projector space and corresponding transformed points from the touch space.

The positions of the projected circles and user touches are shown in Fig. 6. 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 dense 3D graphics space of 4.3mm and standard deviation of 4.3mm (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 which 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 phys-

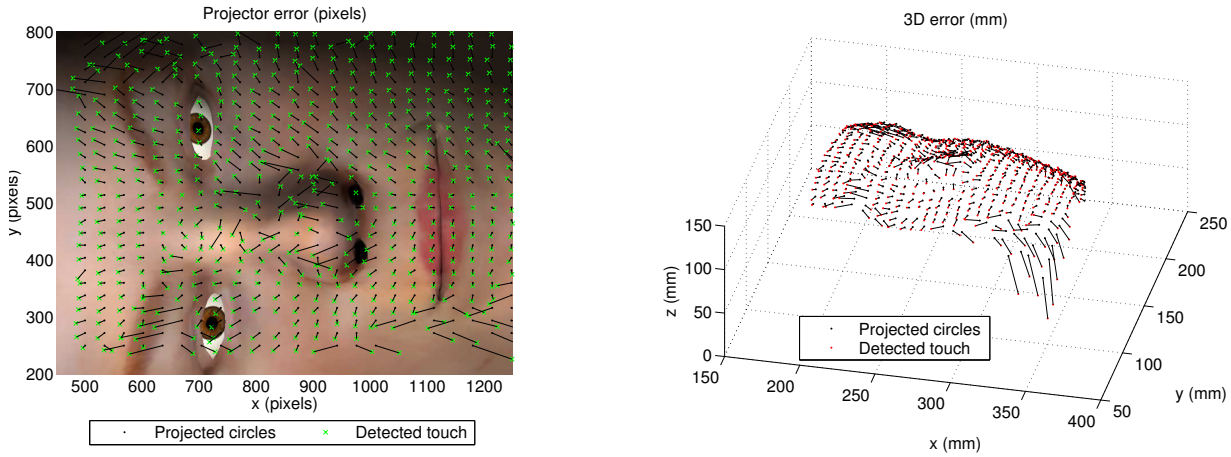


Figure 6: Errors between a user's touch and the touch location detected by the system. Left: errors in 2D projector space (average 16.7 pixels with a standard deviation of 12.5 pixels). Right: errors in 3D model space (average and standard deviation of 4.3mm).

ical 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 dense 3D graphics space of 3.2mm and standard deviation of 1.7mm. 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/projector placement to minimize error sensitivities in the regions with higher accuracy requirements.

It is worth noting that while we attempted to touch the surface carefully/precisely during this experiment, there is undoubtedly noise introduced by our inability to know whether or not our 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 they are touching and how the graphical content responds, all in the context of the application.

5 Application to Stroke Assessment

We have demonstrated our touch-rendering methods using a physical-virtual 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 [16]. 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 [7]. Most human patient simulators do not have the capability to respond and exhibit typical stroke symptoms, such as smile asymmetry, visual gaze, and sensation awareness.

While a trained human actor would be unable to simulate facial droop and lid lag in a realistic manner, our prototype rig, shown in Fig. 7, is capable of exhibiting many of the visual and auditory symptoms of a stroke. 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 head as a simulated "patient" in a stroke assessment training scenario. 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. 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 into the dense 3D graphics space, can be classified as belonging to one of these parts of the face, and a prerecorded sound file 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 touch is perceived. Bone animations in the jaw are activated when the patient speaks. As appropriate, other graphical responses—such as closing eyes and smiling—can be triggered manually.

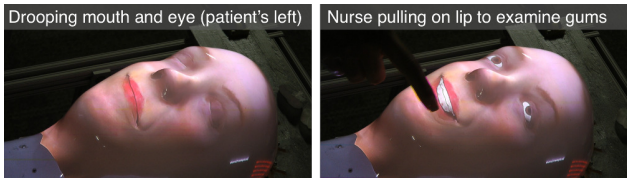


Figure 7: Example stroke scenario. The patient's mouth and eye are "drooping" on her left side, and she is unable to smile. Animated blendshapes still function properly on the drooping mouth. Fig. 1 shows similar interactions on a healthy patient.

6 Future Work

The proposed technique can be extended to handle multiple projectors, allowing for touch sensing and rendering on larger surfaces. The lookup table can be expanded 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 non-parametric surfaces. If multiple projectors can "cover" a particular section of the surface, it may be the case that one projector can 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. Con-

stant 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/contact with objects beyond fingers/hands, both passive (e.g., medical instruments) and active (e.g., a wand with an IR LED in the tip).

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

7 Conclusion

We have presented a generalizable approach for touch detection on non-parametric rear-projection surfaces using IR cameras. We developed a prototype using a human head-shaped surface on which a projector displays a graphical model of a head. Touch events on the surface prompt visual and auditory responses in the model. We demonstrated an example application scenario 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 very direct touch-graphics architecture/approach to ensure the appropriate semantic response of the interactive graphics application while minimizing the perceived latency. Toward this end our method includes pre-calibration 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 spaces/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 parallel (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 evaluating more extensive medical training scenarios incorporating hands-on full-body diagnostic, therapeutic, and comfort touch.

Acknowledgements

This work was supported in part by the USA Office of Naval Research (Dr. Peter Squire, Award N000141410248), the UCF Interactive Systems and User Experience Research Cluster of Excellence (Prof. Joseph LaViola Jr.), and Florida Hospital (USA).

References

[1] Autodesk 123D Catch — 3d model from photos. `http://www.123dapp.com/catch`. Accessed: 2014-09-02.

[2] H. Benko. Beyond flat surface computing: challenges of depth-aware and curved interfaces. In *Proceedings of the 17th ACM international conference on Multimedia*, MM '09, pages 935–944, New York, NY, USA, 2009. ACM.

[3] H. Benko, A. D. Wilson, and R. Balakrishnan. Sphere: multi-touch interactions on a spherical display. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 77–86, New York, NY, USA, 2008. ACM.

[4] P. Bergström. Iterative closest point method. `http://www.mathworks.com/matlabcentral/fileexchange/12627-iterative-closest-point-method`. MATLAB Central File Exchange. Accessed 2014-09-18.

[5] J. H. Chuah, A. Robb, C. White, A. Wendling, S. Lampotang, R. Kopper, and B. Lok. Exploring agent physicality and social presence for medical team training. *Presence: Teleoperators and Virtual Environments*, 22(2):141–170, 2013/09/29 2013.

[6] P. Dietz and D. Leigh. Diamondtouch: a multi-user touch technology. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, pages 219–226, New York, NY, USA, 2001. ACM.

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

[8] J. Gu and G. Lee. Touchstring: a flexible linear multi-touch sensor for prototyping a freeform multi-touch surface. In *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology*, UIST '11 Adjunct, pages 75–76, New York, NY, USA, 2011. ACM.

[9] J. Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, New York, NY, USA, 2005. ACM.

[10] A. Kotranza and B. Lok. Virtual human + tangible interface = mixed reality human an initial exploration with a virtual breast exam patient. In *Virtual Reality Conference, 2008. VR '08. IEEE*, pages 99–106, March 2008.

[11] P. Lincoln, G. Welch, A. Nashel, A. State, A. Ilie, and H. Fuchs. Animatronic shader lamps avatars. *Virtual Reality*, pages 1–14, 2010. 10.1007/s10055-010-0175-5.

[12] A. Majumder and M. S. Brown. *Practical Multi-projector Display Design*. A. K. Peters, Ltd., Natick, MA, USA, 2007.

[13] N. Matsushita and J. Rekimoto. Holowall: designing a finger, hand, body, and object sensitive wall. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, UIST '97, pages 209–210, New York, NY, USA, 1997. ACM.

[14] 3D animation software, computer animation software — Maya — Autodesk. `http://www.autodesk.com/products/maya/overview`. Accessed: 2014-09-02.

[15] K. L. Murdock and E. Allen. *Edgeloop Character Modeling for 3D Professionals Only*. John Wiley & Sons, Inc., 2006.

[16] Warning signs of stroke. `http://www.stroke.org/site/DocServer/TIA.pdf?docID=405`. Accessed: 2014-09-18.

[17] J. Rekimoto. Smartskin: an infrastructure for freehand manipulation on interactive surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, pages 113–120, New York, NY, USA, 2002. ACM.

[18] D. Rivera-Gutierrez, G. Welch, P. Lincoln, M. Whitton, J. Cendan, D. A. Chesnutt, H. Fuchs, and B. Lok. Shader Lamps Virtual Patients: the Physical Representation of Virtual Patients. In *Studies in Health Technology and Informatics, Volume 173: Medicine Meets Virtual Reality 19*, Studies in Health Technology and Informatics, pages 372–378. IOS Press, 2012.

[19] A. Roudaut, H. Pohl, and P. Baudisch. Touch input on curved surfaces. In D. S. Tan, S. Amershi, B. Begole, W. A. Kellogg, and M. Tungare, editors, *CHI*, pages 1011–1020. ACM, 2011.

[20] Unity – game engine. `http://unity3d.com/`. Accessed: 2014-09-02.

[21] N. Villar, S. Izadi, D. Rosenfeld, H. Benko, J. Helmes, J. Westhues, S. Hodges, E. Ofek, A. Butler, X. Cao, and B. Chen. Mouse 2.0: multi-touch meets the mouse. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pages 33–42, New York, NY, USA, 2009. ACM.

[22] F. Wang, X. Cao, X. Ren, and P. Irani. Detecting and leveraging finger orientation for interaction with direct-touch surfaces. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pages 23–32, New York, NY, USA, 2009. ACM.

[23] G. Welch, D. Rivera-Gutierrez, P. Lincoln, M. Whitton, J. Cendan, D. A. Chesnutt, H. Fuchs, B. Lok, and R. Skarbez. Physical manifestations of virtual patients. *Simulation in Healthcare*, 6(6):488, December 2011.

[24] A. D. Wilson. Touchlight: An imaging touch screen and display for gesture-based interaction. pages 69–76. ACM Press, 2004.