MRTouch: Adding Touch Input to Head-Mounted Mixed Reality

Robert Xiao, Julia Schwarz, Nick Throm, Andrew D. Wilson and Hrvoje Benko

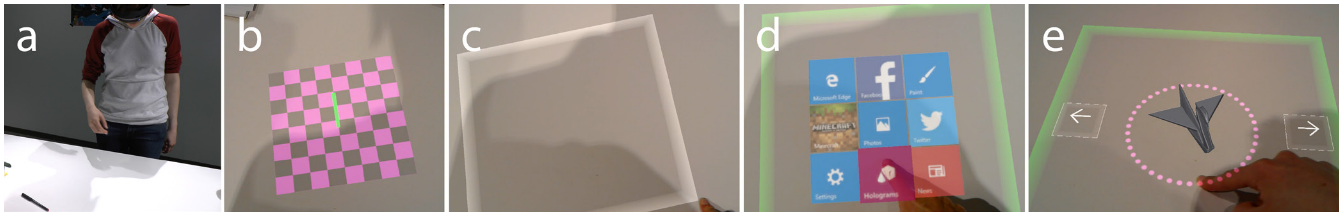

Fig. 1. MRTouch enables touch interaction in head-mounted mixed reality. (a) When a user approaches a surface, MRTouch detects the surface and (b) presents a virtual indicator. (c) The user touch-drags directly on the surface to (d) create a launcher and start an app. (e) In this app, the user uses touch to precisely rotate a 3D model.

Abstract— We present MRTouch, a novel multitouch input solution for head-mounted mixed reality systems. Our system enables users to reach out and directly manipulate virtual interfaces affixed to surfaces in their environment, as though they were touchscreens. Touch input offers precise, tactile and comfortable user input, and naturally complements existing popular modalities, such as voice and hand gesture. Our research prototype combines both depth and infrared camera streams together with real-time detection and tracking of surface planes to enable robust finger-tracking even when both the hand and head are in motion. Our technique is implemented on a commercial Microsoft HoloLens without requiring any additional hardware nor any user or environmental calibration. Through our performance evaluation, we demonstrate high input accuracy with an average positional error of 5.4 mm and 95% button size of 16 mm, across 17 participants, 2 surface orientations and 4 surface materials. Finally, we demonstrate the potential of our technique to enable on-world touch interactions through 5 example applications.

Index Terms— Augmented reality, touch interaction, depth sensing, sensor fusion, on-world interaction

1 Introduction

Head-mounted, mixed-reality devices can overlay realistic 3D content onto the physical environment, enabling a wealth of interactive possibilities. Current-generation commercial mixed reality devices, like Microsoft HoloLens or Meta 2, are primarily driven by in-air hand gesturing, gaze and voice. While convenient, these indirect input modalities are not particularly precise or rapid [10][14]. Furthermore, these input modalities are fatiguing to use for long periods of time – neither gaze nor voice input are suitable for continual input while gestural input suffers from "gorilla-arm" effects [24], precluding prolonged use. In response, many systems ship with accessory physical controllers, offering finer-grained input, though at the expense of occupying the hands with a special-purpose device that then impedes free-hand manipulations.

On the other hand, touch interaction is precise, tactile, familiar to users, and comfortable to use for extended periods. However, interestingly, it has been overlooked as an input modality in head-mounted mixed reality systems, despite the inherent ability for many systems to affix virtual interfaces to physical surfaces (which are then *e.g.*, gestured at instead of touched).

To explore the feasibility and utility of enabling touch interaction in mixed reality settings, we developed *MRTouch*: a multitouch input solution for mixed reality. By overlaying appropriate virtual content, our approach transforms ordinary surfaces into expansive virtual touchscreens, enabling interactions with coordinated virtual content. MRTouch is implemented on Microsoft HoloLens and requires no additional tracking infrastructure or hardware beyond the existing on-board cameras – users need only put on the headset to obtain touch tracking capabilities. We make the following contributions in this paper:

- A method for real-time detection of surface planes suitable for hosting touch-enabled applications.
- A robust finger-tracking pipeline able to segment touch inputs even while the user's head is in motion.
- A practical, self-contained implementation of our approach on an off-the-shelf Microsoft HoloLens.
- A user study assessing our system's tracking accuracy across common surface materials and orientations, the results of which show very high touch accuracy.
- A suite of example applications and interactions that are enabled by on-world touch input.

- *Robert Xiao is with Microsoft Research and Carnegie Mellon University. E-mail: brx@cs.cmu.edu*
- *Julia Schwarz and Nick Throm are with Microsoft. E-mails: jschwarz@microsoft.com, nithrom@microsoft.com*
- *Andrew D. Wilson and Hrvoje Benko are with Microsoft Research. E-mails: awilson@microsoft.com, benko@microsoft.com*

Manuscript received 11 Sept. 2017; accepted 8 Jan. 2018. Date of publication 19 Jan. 2018; date of current version 18 Mar. 2018. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TVCG.2018.2794222

2 Related Work

Our present work concerns mixed-reality input techniques and on-world touch tracking, two areas of related work which we now discuss.

2.1 Non-Touch Techniques in Mixed Reality

A vast range of input techniques have been proposed for use in mixed-reality systems [61], such as gaze input [4][51], hand gestures [16][34][36], voice input [8][44], physical controllers (using physical buttons [46], motion sensing [45], or external tracking [27]), tangible interfaces [7][29], or multimodal combinations of these approaches (*e.g.*, [8][14]). Of these, gaze, gesture and voice provide no haptic feedback, while controllers and tangible interfaces require additional external hardware to function.

2.2 Touch Input in Mixed Reality

The most common intersection of touch input with mixed reality is for providing haptic feedback, which adds a sensation of physicality to virtual objects. To this end, a number of haptic interaction techniques have been proposed for mixed reality, including special-purpose haptic controllers (*e.g.*, articulated probes [1][40], gloves [6][9], handheld devices [5] or even body suits [38]), in-air haptics [12][54], or robotically-presented textures [2].

For providing touch input, many systems use special-purpose touch input devices. Toucheo [20] augments a touchscreen table with a reflected stereoscopic display. In mobile augmented reality applications, touch input can be provided on the mobile device itself, providing a form of indirect touch input [28]. With head-mounted displays, touchpads can be mounted on the side of the device [53] or even across the entire front face of the device [19] to enable indirect touch interaction. Alternatively, touch input can be provided on handheld controllers [27] or on a dedicated mobile device [41].

More closely related to our conceptual approach, systems can also co-opt existing objects in the environment to provide input and haptic feedback. Vrui [33] uses video pass-through to allow users to use real keyboards and mice to provide input to a virtual reality system. Haptic Retargeting [3] uses perceptual warping in VR to allow users to grasp physical objects as proxies for virtual ones. Walsh *et al*. [55] propose projecting onto existing objects to co-opt them into a tangible user interface. Finally, Annexing Reality [23] opportunistically matches virtual objects with physical objects present in the user's environment, mirroring user's interactions with the physical objects onto the virtual objects.

2.3 Ad hoc Touch Tracking on Surfaces

Most related to our technical approach are systems that provide touch tracking on surfaces in the environment. One option is to instrument the surface itself, *e.g.*, with capacitive [35][56], optical [21] or acoustic [50][60] sensors. Alternatively, cameras can be used to avoid directly instrumenting the underlying surface. Several schemes for camera-based touch sensing have been proposed in the literature, including finger template matching [32], skin-color segmentation [37], contour segmentation [13], thermal imprint tracking [52], or LIDAR [49].

Many of these approaches require instrumenting the user or calibrating to the background environment, and in general methods based only on optical sensing have difficulty determining whether a finger has contacted the surface or not (*i.e.*, disambiguating hover from touch). The recent availability of inexpensive depth cameras has led to a wide range of touch sensing techniques built upon depth sensing, *e.g.*, Wilson [57], KinectFusion [30], WorldKit [58], OmniTouch [22], and DIRECT [59]. However, these systems rely on static, pre-acquired background models, with the sole exception being OmniTouch, which instead requires a more constrained finger pose (pers. comm.).

In contrast to these prior systems, we explore a different interaction modality (head-mounted mixed reality) built on a piece of commodity hardware (Microsoft HoloLens). Additionally, we demonstrate that our touch sensing approach achieves higher spatial accuracy than prior approaches on a variety of surfaces and orientations, including those that are typically challenging for infrared sensing.

3 Implementation

The MRTouch prototype is implemented on a Microsoft HoloLens development kit device, sold commercially for developer use since 2016 [43]. The HoloLens features a time-of-flight depth camera similar to the Microsoft Kinect for Xbox One [42], which can operate in both short-throw (maximum distance ~1 m) and long-throw modes, used for hand tracking and environment sensing, respectively. We use the short-throw mode exclusively, due to reduced noise characteristics.

HoloLens features a customized ASIC, dubbed the Holographic Processing Unit (HPU) used for depth data processing, hand tracking, localization and mapping, combined with a traditional Intel CPU running at 1 GHz with 2 GB of RAM. The HPU allows HoloLens to map the 3D environment and to provide a continual estimate of its pose and position within the space. However, unlike KinectFusion [30], the computed environment map is too rough to be directly useful for touch detection.

3.1 Software Architecture

MRTouch uses only the raw short-throw depth data and infrared imagery from the depth camera through a private API. The algorithm should therefore be portable to any device which provides a similar API, *e.g*. Google Project Tango devices. The tracking pipeline runs at 25 FPS and consists of three software components: 1) *Image Streamer*, 2) *Tracker Engine* and 3) *Client Library*.

Image Streamer extracts the raw depth data and infrared imagery from the on-board depth camera and exports it over a TCP socket, allowing us to stream data to either a Tracker Engine running locally on the HoloLens device, or to a GUI-enabled debug version of the Tracker Engine running on a connected desktop computer (shown in Figure 2).

Tracker Engine is implemented as a C++ program designed to run natively on HoloLens' Intel CPU. It receives depth and infrared imagery from Image Streamer (Figure 2a, 2b), and estimated head pose and position data from the spatial mapping API on the HoloLens device. Tracker Engine maintains a set of *known* touch surfaces in world coordinates. A touch surface is a 3D-positioned rectangle, represented uniquely by three 3D corner points (its bottom-left, bottom-right and top-left corners). On each frame, the tracker locates each known surface in the current depth frame, then acquires additional surfaces by fitting planes to the observed depth data. It then detects touch points over the visible planes, implements touch filtering and hover detection, and reports resultant surface plane and touch data to any connected client applications.

To receive touch data, a client application links against the *Client Library*, which is written in C#. This library opens a TCP connection to Tracker Engine, through which it receives plane position data and touch information. The library normalizes these positions into the application's coordinate system, and presents a simple event-based framework for consuming new plane and touch data. The client can also use the library to transmit new plane data to the Tracker Engine component, *e.g.*, if the user launches a new application on a surface (thereby locking the surface into the *known* set), or moves an existing in-air application onto a surface.

In the following sections, we detail the touch and surface tracking process performed by the Tracker Engine.

3.2 Surface Refinement and Detection

Many prior ad hoc touch systems first calibrate a background model of the environment, allowing them to perform a simple depth subtraction to extract hands and fingers. However, on a head-mounted display, the camera may be constantly in motion, precluding development of a simple background model. Furthermore, although HoloLens can determine its own pose and position in the physical world, the estimated position is not perfectly accurate, causing established planes to appear to shift in position relative to their physical substrates as the head moves around.

Consequently, the Tracker Engine component performs a depth-based refinement of each visible known plane, using the random sample consensus (RANSAC) algorithm [17]. This algorithm first projects the original plane into the depth image, producing an image mask. Each RANSAC iteration selects three random depth pixels from the mask and counts the number of pixels in the mask that approximately fit the resulting plane (the inliers). The final output is the plane with the largest number of inliers. This refinement procedure is robust to objects and hands above the touch plane, as these obstructions would be considered outliers and thus ignored.

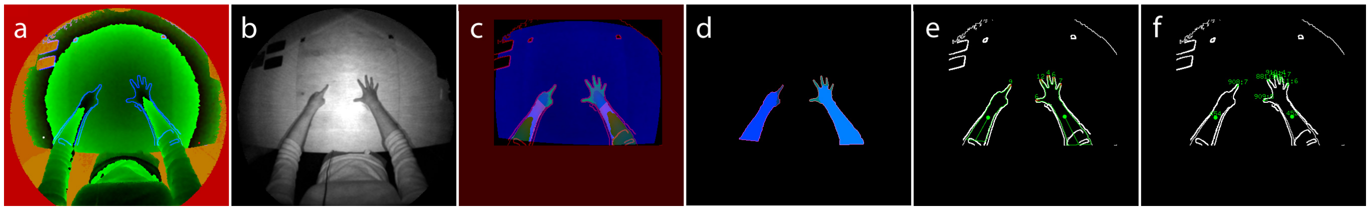

Fig. 2. Touch tracking pipeline. (a) depth data and (b) infrared reflectivity data from HoloLens depth camera. (c) detected ephemeral plane (blue), flood-filled hands (light blue) and fingers (teal). (d) extracted hand masks and contours. (e) hands, fingers and estimated finger distance from plane. (f) hands and fingers after touch filtering, with IDs corresponding to objects in past frames.

Our procedure also fits a single *ephemeral* plane to each frame (Figure 2c), defined as the plane centered at the user's gaze center (if present). This is also achieved by using RANSAC, by selecting three points at random near the gaze center and selecting the plane that fits the most depth points across the entire image. Unlike known planes, the ephemeral plane is not stored – it is discarded after the touch tracking step is complete and recomputed on the next frame. The ephemeral plane is used to allow users to walk up and touch surfaces without previously defining touch areas, providing a way for users to instantly begin using a touch surface.

3.3 Touch Finding

For each visible surface plane, including the ephemeral plane, Touch Engine initiates touch finding. The overall touch tracking approach merges depth data with infrared data, providing precise fingertip positions, an approach first demonstrated in DIRECT [59]. However, while DIRECT required a stabilized background and noise profile, our MRTouch implementation uses a RANSAC-refined touch plane, eliminating the prior profiling requirement and enabling true ad hoc touch tracking with no prior calibration of the surface.

Touch finding begins by re-projecting all depth pixels within the plane boundaries into *heights* relative to the surface plane (blue channel of Figure 2c). Then, the algorithm computes an *edge map* by merging a Canny edge map [11] of the infrared image with a threshold-based edge map of the heightmap (a pixel is labelled as a depth edge if it differs from a nearby pixel by more than 50 mm). Next, the algorithm flood fills pixels that are more than 40 mm off the surface, combining connected regions into tentative *high regions* (which typically includes arms, hands, and fingers that are far off the table). It then rejects high regions that do not contact the edge of the touch plane (*i.e.*, regions that lie entirely in the plane), as these are likely to simply be objects sitting on the surface.

From each remaining high region, the system fills downwards towards regions that lie closer to the surface, while respecting the edge map (Figure 2c, teal regions). The edge map ideally stops the flood fill at the user's fingertips, providing a clean segmentation even when the user's fingertips disappear into the background noise. If the flood fill doesn't stop (filling farther than a reasonable human finger length, 15 cm), this fill operation is rolled back, and the algorithm perform a more cautious fill that avoids filling into noisy background pixels (effectively returning to a depth-only tracking approach). On complex infrared surfaces, this ensures that the finger is still located, albeit with lower precision.

This step is followed by a smoothing procedure applied to the resulting hand+finger mask, and the conversion of this smoother mask into a contour map (Figure 2d). Finally, the algorithm walks across the contour, extracting convex points into fingertip points (Figure 2e). This contour-finding process ensures that the fingertip is extracted, even when the finger is very short (*e.g.*, if it is viewed from an oblique angle). Furthermore, as the segmentation is performed on the combined mask, this approach naturally segments hands and fingers in-air as well, enabling in-air hand gesture detection.

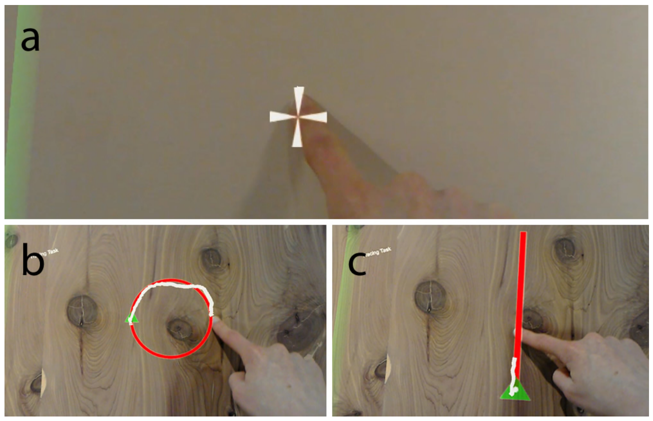

Fig. 3. Experiment tasks: (a) crosshair clicking task, (b) circle tracing task, and (c) line tracing task.

3.4 Touch Tracking and Filtering

Hand points (centroids) and fingertip positions, in world coordinates, are retained between frames to enable touch tracking and filtering. Hands from the current frame are matched to hands in the previous frame by Euclidean distance with a fixed upper limit on movement (*i.e.*, assuming hands do not move more than 10 cm in a single frame, or 2.50 m/s). Then, for each matched pair of hands, fingers are matched by applying the hand movement vector to the previous finger positions, then matching individual fingertips by Euclidean distance (again applying the same upper limit on movement distance). By matching hands first, it is possible to track rapidly-moving hands without mixing up fingers. Figure 2f shows the final hand and finger positions.

To detect if the fingertips are touching or not, our algorithm analyzes a 7x7 patch of pixels centered on the fingertip's contour position. Each patch is split into S, the set of pixels within the hand+finger mask, and T, the set of pixels outside the mask. The estimated height of the finger is then given by

$$\max(z_s \mid s \in S) - \min(z_t \mid t \in T)$$

where the use of *max* and *min* help to stabilize the estimated finger and background distances against noise, as well as choosing points that have maximum discriminative power.

The hand position, fingertip position, and fingertip height values are smoothed using an exponentially-weighted moving average filter, providing resilience to noise and tracking failures. To confirm contact with the surface, the algorithm applies a simple pair of hysteresis thresholds – a fingertip is declared as touching the surface if the smoothed fingertip height descends below 10 mm, and declared to have left the surface if its height later ascends past 15 mm. This hysteresis approach mirrors the hover detection approaches seen in other systems, such as Wilson [57], OmniTouch [22] and DIRECT [59].

4 Experiment

To quantify the spatial tracking accuracy and performance of MRTouch, we performed a user study with 17 participants – 5

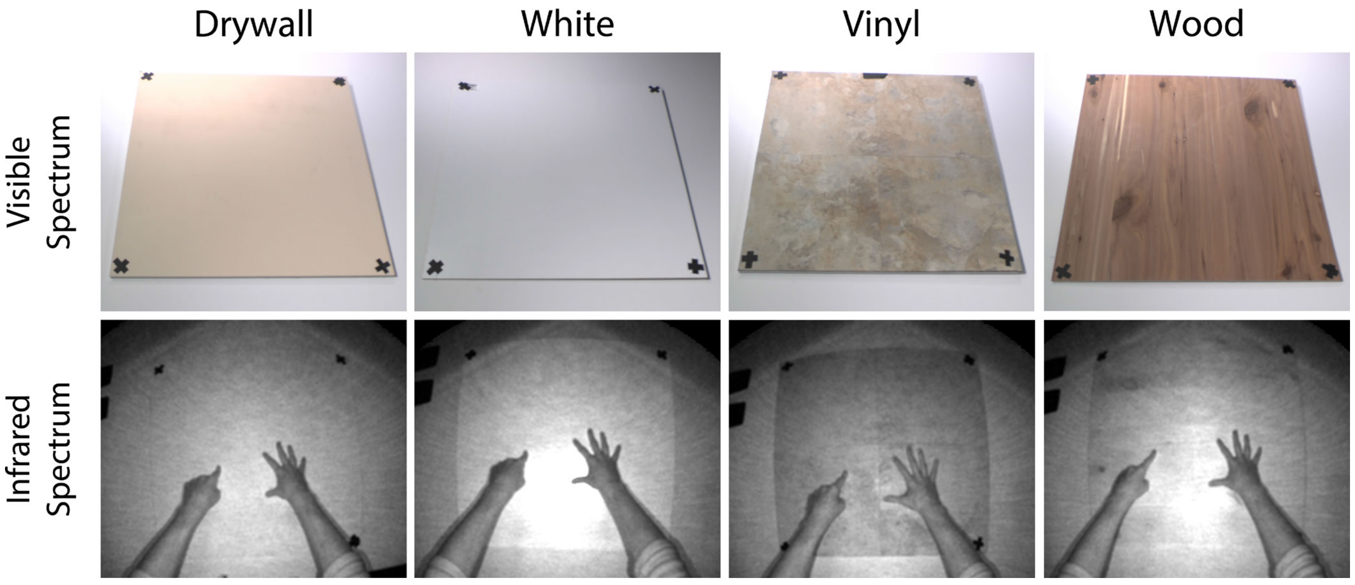

Fig. 4. Surfaces used in the experiment. User hands shown in infrared for contrast and comparison.

females, average height 68.8 in (174 cm), mean age 34, 2 left-handed, with Fitzpatrick skin types ranging from II to VI [18].

Participants were asked to perform a series of touching and tracing tasks on various surfaces in two separate orientations, to check the system's robustness to surface conditions and surface orientation. Users performed all tasks with their dominant hand, and held a wireless mouse in their other hand, which was used to advance trials.

4.1 Surfaces

We chose a set of four surface types representative of common, everyday surfaces in homes and offices, as well as providing a range of different surface profiles in infrared:

Drywall: a piece of drywall painted uniformly with acrylic paint, representing most typical wall surfaces. Under infrared illumination, this surface appears relatively bright and diffuse.

White: a piece of medium-density fibreboard (MDF) coated with melamine, producing a smooth and glossy surface typical of contemporary tables, desks and cabinets. Under infrared illumination, this surface appears bright and specular.

Vinyl: a set of vinyl tiles with slightly rough exterior stone-like textures, used to represent stone and textured surfaces typical of countertops and floors. Under infrared illumination, this material is dark and diffuse.

Wood: sanded plywood with a cedar veneer, used to represent wooden surfaces typical of tables and desks. Under infrared illumination, this material appears darker, specular and non-uniform.

Each surface was cut to a 60×60 cm square tile, allowing them to be swapped in and out during the experiment. Images of these surfaces in visible light and under infrared are shown in Figure 4.

Prior to the study, we hypothesized that the darker, more complex infrared background of the vinyl surface would degrade our infrared tracking, and thus decrease accuracy.

4.2 Orientations

We also tested the system in two common orientations: with the user seated and touching a vertically-oriented (wall-mounted) surface, and with the user standing and touching a horizontal (table-mounted) surface. In the Wall condition, the user sat on an adjustable chair, with the virtual surface centered at chest level (Figure 5, right), with the HoloLens at an average distance of 31 cm from the wall surface. In the table condition, users stood 6 inches from the table edge and looked down (Figure 5, left), averaging 51 cm from the HoloLens to the table's touch surface.

4.3 Tasks

For each combination of orientation and surface type (8 conditions in all), users performed two tasks: *crosshair targeting* and *tracing*.

The *crosshair targeting* task required the participant to place their fingertip on the surface corresponding to the displayed crosshair (Figure 3a), then click the mouse to confirm that they were touching the surface. The system recorded all detected touch contacts and their locations on click, allowing us to study the system's contact detection rate. There were 16 crosshair locations, placed in a 4×4 grid spaced evenly across a 20×20 cm square area. Crosshairs were displayed in random order.

In the *tracing* task, each participant was presented with eight shapes to trace: four circles (Figure 3b; each combination of clockwise/counterclockwise and 10 cm diameter/20 cm diameter) and four lines (Figure 3c; running left/right/up/down, all 20 cm long). A green arrow indicated where the participant should begin the trace, and the system automatically ended the trial when the participant completed the shape. Participants then lifted their fingers clear of the surface and clicked the mouse to start the next trial.

Prior to the main experiment, participants performed one trial each of the crosshair and tracing tasks to familiarize themselves with the experimental procedure; these training trials were not included in the final analysis.

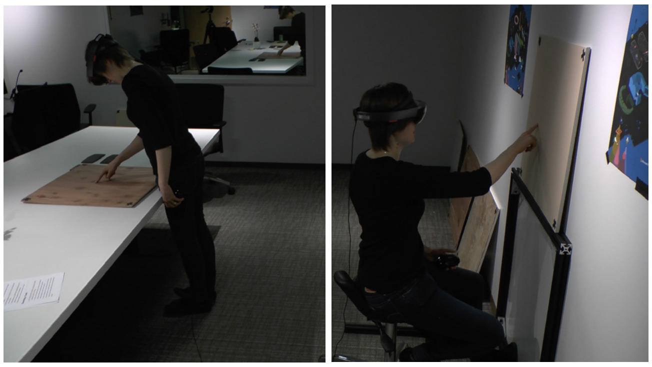

Fig. 5. Surface orientations used in the study. Left: Table orientation. Right: Wall orientation.

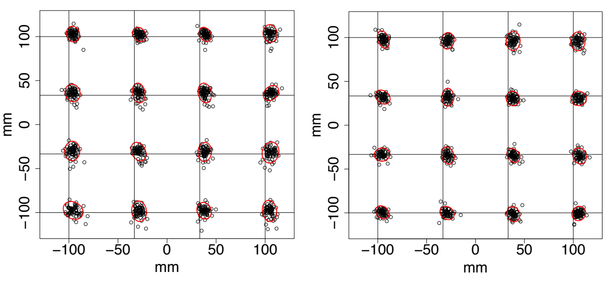

Fig. 6. Scatterplot of touch points in the *wall* condition (left) and *table* condition (right), plotted with 95% confidence ellipses. Grid intersections correspond to crosshair points.

5 Results

5.1 Crosshair Targeting

Across 17 participants, we obtained 2176 crosshair trials, recorded at the moment that the participant pressed the mouse button. Of these, 77 (3.54%) reported no touch contact, so the system located at least one touch point 96.5% of the time. In 416 trials (19.12%), the system detected two or more touch contacts; of these, 49 (2.25%) found three or more touch contacts. In the event of multiple detected contacts, touch accuracy was computed using the contact nearest to the target.

The distributions of touch contact counts were significantly different (p<0.0001, ANOVA) between the Wall and Table conditions. In the Wall condition (1088 trials), only 2 trials (0.2%) reported no touch contact and 185 trials (17%) reported two or more contacts. In contrast, on the Table, 75 trials (6.9%) reported no touch contact, and 240 trials (22%) reported two or more contacts. No significant differences (p>0.01, ANOVA with Tukey HSD) in touch contact count were found between materials.

5.1.1 Spatial Accuracy

Analysis of our crosshair data demonstrated a systematic offset between the target positions and the received touch position. Reported touch positions were an average of 5.0 mm to the right of the crosshair (Figure 6) with a negligible (<0.1 mm) vertical shift. This shift held constant across all orientations, materials, and user handedness, and thus, for further analysis, we subtracted the global average offset from all touch points. In practice, we would add this global offset correction as part of the touch detection pipeline, as it is user-, material- and orientation-independent. To allow for direct comparison with the OmniTouch system, we followed the same data analysis procedure as Harisson et al. [22] and removed 44 outlier points (2.0%) which lay more than three standard deviations from the target point. Across all remaining points, we achieved a global mean Euclidean error (μ) of 5.4 mm (SD=3.2 mm).

Spatial accuracy showed significant (p<0.0001, ANOVA) differences between the Wall (μ=5.7 mm, Figure 6, left) and Table (μ=5.0 mm, Figure 6, right) orientations. In terms of materials, pairwise Tukey HSD comparisons showed that the Vinyl material (μ=6.2 mm) was significantly less accurate than all other materials (Drywall μ=5.0 mm p<0.0001, White μ=5.3 mm p<0.0001, Wood μ=5.0 mm p<0.0001), but no other significant differences were found (p>0.01).

Plotting the 95% confidence ellipses for the Wall and Table conditions (Figure 6) shows that, on average, a 16 mm diameter button would capture 95% of touches, which compares favorably with the 15 mm button diameter for capacitive touchscreens found by Holz, *et al*. [26].

5.2 Tracing

For our shape tracing task, we continually computed the absolute Euclidean distance between each incoming touch point and the nearest point on the shape. Because graphical feedback was provided, we did not apply any post-hoc offset correction, reasoning that users would naturally adjust their motions to compensate for any observed inaccuracies. The mean Euclidean distance across all users, conditions, and orientations was 4.0 mm (SD=3.4 mm), with no significant differences across shapes, materials or orientations.

5.3 Latency

A static analysis of latency in our research prototype (as determined through timestamping various stages in the pipeline) suggests an expected latency between 105~175 ms, or a mean latency of 140 ms, from physical touch event to the first displayed frame incorporating the event. However, due to smoothing factors, the observed latency is somewhat higher, at around 180~200 ms in empirical testing.

5.4 Comparison with Existing Approaches

In terms of spatial accuracy, our results compare favorably to past ad-hoc touch tracking approaches. Wilson [57] does not provide a formal evaluation of spatial touch accuracy, but suggests a positional error of 7 mm at a sensor distance of 75 cm from a fixed, pre-calibrated surface. KinectFusion [30] demonstrates, but does not formally evaluate the touch tracking methodology. DIRECT [59] demonstrates 4.8 mm average Euclidean error, albeit with a significantly different setup over a rigidly pre-calibrated table surface.

Finally, OmniTouch [22] shows an average positional offset of 11.7 mm (dependent on surface orientation), compared with our 5.0 mm global offset. OmniTouch also plotted the 95% confidence ellipses to evaluate its accuracy; in the best condition (touching on a wall), the confidence ellipses measure an average of 28 mm in diameter (compared with MRTouch's 16 mm buttons).

6 Discussion

6.1 Click Detection

Our crosshair study demonstrated a surprisingly high rate of both missed touches (3.5%) and spurious extra touches (19%). From reviewing the study logs and recordings, we determined that poor hover sensing was primarily to blame for both issues. When we missed touches, the finger itself was often located but the finger's height did not cross our heuristic threshold. On the flip side, we observed that many participants held their thumbs out while performing the crosshair task, which were correctly detected as an additional finger. However, in a few trials, the thumbs were detected as touching the surface. This often happened when participants unconsciously brought the thumbs close to (but not touching) the surface during the experiment, and the lack of visual feedback meant that people were unaware that the system was detecting a contact event.

Contact detection was not the primary subject of our initial MRTouch implementation; instead, we sought to tackle the orthogonal problem of spatial accuracy. Because divining the finger's distance from the surface is difficult solely from an overhead view, we felt that solving the contact detection problem was out of scope for this paper, and thus chose a hover detection implementation similar to those used by other ad-hoc touch tracking systems we reviewed. However, we expect that more sophisticated finger height estimation algorithms could produce better results in the future, and we look forward to exploring this issue further.

6.2 Latency

Latency is a crucial consideration for touch input. While our system latency of ~180 ms is comparable with the higher end of touchscreen latencies (50~200 ms, per [47]), people can notice

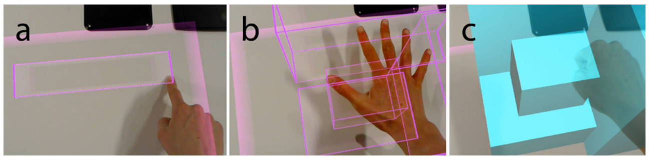

Fig. 7. Blueprint extrusion app. (a) The user draws a blueprint with their finger. (b) They use an in-air open-hand gesture to extrude it, using hand height to set the extrusion distance. (c) The user closes their hand to lock-in the extrusion.

latencies as low as 20 ms [31]. The latency problem is often exacerbated on large displays (*e.g.*, wall-sized displays) because sensing larger displays typically requires more processing time, and simultaneously users can move much faster across the display (increasing the observed gap between physical actions and virtual responses). Thus, for larger displays, people may be even more sensitive to latency issues.

We emphasize that our system latency is largely a prototyping limitation. Most of the HoloLens' existing tracking algorithms run on the dedicated HPU ASIC, which has very low latency access to depth data combined with high input priority (*i.e.* low-latency access to the system input event queue). Our prototype does not currently run on the HPU, but rather on the HoloLens' CPU. We estimate that the latency for an HPU implementation would average around 60 ms (depth camera average latency + touch pipeline processing time + rendering average delay + display latency), which is comparable to high-end commercial touchscreens.

Decreasing latency is thus a major topic of future work. Besides an HPU implementation, we could also explore techniques such as forward prediction to forecast the user's finger position and pre-touch [25] to predict when the user will contact the surface.

6.3 Effects of Orientation

We found significant orientation-dependent effects on detected touch count. In our study, participants sat much closer to the Wall (31 cm) than the Table (51 cm). Our analysis of the data suggests that the longer Table distance increased depth noise but not infrared noise, causing more touch contacts to exceed the predefined hover thresholds, resulting in reduced touch detection accuracy.

Although the Wall orientation was significantly less accurate than the Table orientation (5.7 mm vs 5.0 mm), this difference is relatively slight, suggesting that the use of the infrared map helps ensure a high level of spatial accuracy. Future work will need to account for varying depth sensor noise levels in order to provide accurate click detection.

6.4 Effects of Material

From our study, we found that touch tracking had significantly worse spatial accuracy on the vinyl material. This may be due to a combination of surface roughness (which increases depth noise) and dark infrared profile (which reduces contrast with the hands). However, even on that material, we still achieve a high degree of spatial accuracy relative to prior work.

In general, some materials will perform worse than others, and in the extreme case, some materials will not work at all. For instance, we initially considered a fifth material, MDF coated in black paint, which absorbed a significant amount of infrared radiation (*i.e.*, it is black in the infrared spectrum too). We were unable to use this material for the study, though, because the low reflectivity prevented the depth camera from even sensing the surface. Furthermore, transparent surfaces (*e.g.*, glass) will likely never be supported by depth-based touch tracking, necessitating some means of distinguishing suitable from unsuitable surfaces and conveying this in the interface so that the user's expectations are properly set.

The infrared reflectivity of a surface is likely to be a strong predictor of its ability to support MRTouch interactions – for instance, dark or highly patterned surfaces may not support interactions as well. Therefore, at a distance, we should be able to determine how well touch tracking will work before users attempt to interact with the surface, and we can present feedback to users to allow them to choose an appropriate surface. For example, we could choose to only highlight suitable surfaces with a surface indicator (Figure 1b), so that users would learn to associate the indicator with the ability to perform touch interaction.

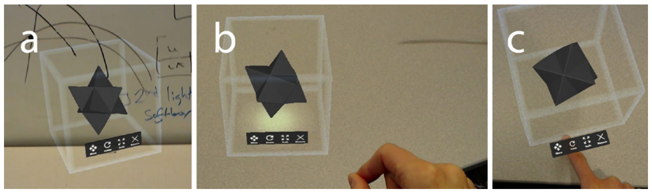

Fig. 8. Snapping interfaces to touch surfaces. (a) AR apps often float in mid-air with gaze/gesture-input buttons. (b) With MRTouch, the user can move the app near a surface, which highlights in response. (c) When the user lets go, the interface and toolbar snap to the surface, enabling touch interaction.

7 Mixed-Reality Touch Interfaces

To demonstrate the utility of MRTouch, we built several example applications, which we describe below. We also encourage the reader to inspect these applications in greater detail in the accompanying Video Figure. Note that the digitally-overlaid content was not edited to synchronize with the color video camera – the video figure accurately reproduces the visual feedback and perceived latency of the system as seen by the user. The perceived latency to head motion is low due to the HoloLens' use of late-stage reprojection (LSR) functionality (which uses interpolated rendering [39] similar to *e.g.* Oculus Asynchronous Time-Warp [48]), which visually warps content to compensate for post-render head motion.

7.1 Creating Touch Interfaces

When a person walks up to a surface that is not hosting an existing interface, a visual representation of the computed ephemeral surface plane (Figure 1b) is shown to indicate that the surface supports touch interaction. Inspired by OmniTouch [22], the user can simply touch the surface to begin defining the interactive region. As the user drags their finger along the surface, the system displays a proposed rectangular interactive area on the surface (Figure 1c) oriented according to the user's point of view. After the finger is released, the interactive region instantiates with an application menu (Figure 1d), from which the user can select an app to run on that region (Figure 1e). In a practical implementation, a delimiting input (*e.g.*, a voice command or unique hand gesture) could be added before the initial drag input to avoid inadvertent input on inactive surfaces (the Midas touch problem).

7.2 Standard Multitouch Interaction

Applications running on the surface support standard single-touch and multi-touch inputs, such as clicking, panning and zooming. Furthermore, the ability to support standard multitouch input opens the possibility of running traditional tablet PC applications within virtual touchscreens (*e.g.*, a document reader or web browser), greatly expanding the number of available applications.

7.3 Interactive Control of 3D Objects

Touch input can also be used to control 3D objects, such as controlling their rotation, size and scale, by manipulating appropriate on-surface control elements (as proposed in [6]). In our example, the person can select a 3D object in their view, dock them by moving them closer to the surface, and then use on-surface touch control elements (*e.g.*, a rotation dial, Figure 1e) to manipulate the object. Crucially, the precision of touch input allows users to make fine adjustments to objects.

7.4 Combining Touch Interaction and In-Air Gestures

MRTouch naturally supports finger input both on and off the surface, making it possible to combine touch interaction on a surface with gestures performed above it. For instance, after drawing the blueprint for a building using touch input (Figure 7a), a user can use an in-air gesture (Figure 7b) to adjust its *z*-height, thus naturally extruding a 2D cross-section into a 3D volume (Figure 7c). Many more interactions are possible; see *e.g.,* [6][15] for an exploration of mixed-modality gestures.

7.5 Converting In-Air Interfaces into Touch Interfaces

HoloLens applications often feature menus or toolbars, which users ordinarily interact with using head gaze and hand gestures (Figure 8a). With MRTouch, a person can move such an application onto a surface (Figure 8b), whereupon the in-air menu transforms into a touch surface (Figure 8c). By enabling a seamless transition between on-surface and in-air modes, a person can choose an interaction modality that suits their work. Furthermore, because of the higher precision of touch input, the on-surface mode could be used to surface additional functionality – for example, expanding a simple palette of image operations into a complete Photoshop-like toolbar.

8 Conclusion

We have presented MRTouch, a touch tracking solution which brings un-instrumented touch input, on a wide variety of common surfaces, to head-mounted mixed reality. In this way, mixed reality systems can benefit from a precise, tactile and familiar way to interact with virtual content, neatly complementing the existing input modalities. Through a user evaluation, we determined that our system can provide near-touchscreen levels of spatial accuracy, enabling a host of useful interactions. By demonstrating usable touch input using our commodity setup, we hope to demonstrate that touch input is a feasible and useful addition to the mixed-reality input toolbox, and hope that it can serve to inspire further work in this area.

References

[1] M. Adcock, M. Hutchins, C. Gunn. "Haptic Collaboration with Augmented Reality," *ACM SIGGRAPH 2004 Posters (SIGGRAPH '04)*, pp. 41, 2004.

[2] B. Araujo, R. Jota, V. Perumal, J. X. Yao, K. Singh and D. Wigdor. "Snake Charmer: Physically Enabling Virtual Objects," *Proc. 10th Int. Conf. Tangible, Embedded and Embodied Interaction (TEI '16)*, pp. 218-226, 2016.

[3] M. Azmandian, M. Hancock, H. Benko, E. Ofek and A. D. Wilson. "Haptic Retargeting: Dynamic Repurposing of Passive Haptics for Enhanced Virtual Reality Experiences," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '16)*, pp. 1968-1979, 2016.

[4] M. Bâce, T. Leppänen, D. G. de Gomez and A. R. Gomez. "ubiGaze: ubiquitous augmented reality messaging using gaze gestures," *Proc. SIGGRAPH ASIA 2016 Mobile Graphics and Interactive Applications (SA '16)*, Article 11, 5 pages, 2016.

[5] H. Benko, C. Holz, M. Sinclair and E. Ofek. "NormalTouch and TextureTouch: High-fidelity 3D Haptic Shape Rendering on Handheld Virtual Reality Controllers," *Proc. 29th Ann. Symp. User Interface Software and Technology (UIST '16)*, pp. 717-728, 2016.

[6] H. Benko, E. W. Ishak and S. Feiner. "Cross-Dimensional Gestural Interaction Techniques for Hybrid Immersive Environments," *Proc. IEEE Virtual Reality (VR 2005)*, pp. 209–116, 2005.

[7] M. Billinghurst, H. Kato and I. Poupyrev. "Collaboration with tangible augmented reality interfaces," *Proc. HCI Int.*, pp. 234-241, 2004.

[8] R. A. Bolt. "Put-that-there: Voice and gesture at the graphics interface," *Proc. 7th Ann. Conf. Computer graphics and interactive techniques (SIGGRAPH '80)*, pp. 262-270, 1980.

[9] G. Burdea, J. Zhuang, E. Roskos, D. Silver and N. Langrana. "A portable dextrous master with force feedback," *Presence: Teleoper. Virtual Environ.* 1, 1 (January 1992), 18-28.

[10] M. C. Cabral, C. H. Morimoto and M. K. Zuffo. "On the usability of gesture interfaces in virtual reality environments," *Proc. Lat. Am. Conf. Human-computer interaction (CLIHC '05)*, pp. 100-108. 2005.

[11] J. Canny. "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Anal. Mach. Intell.* 8, 6 (June 1986), 679-698.

[12] T. Carter, S. A. Seah, B. Long, B. Drinkwater and S. Subramanian. "UltraHaptics: multi-point mid-air haptic feedback for touch surfaces," *Proc. 26th Ann. Symp. User Interface Software and Technology (UIST '13)*, pp. 505-514, 2013.

[13] J. S. Chang, E. Y. Kim, KC Jung and H. J. Kim, "Real time hand tracking based on active contour model," *Proc. Int. Conf. Computational Science and Applications (ICCSA '05)*, pp. 999-1006, 2005.

[14] I. Chatterjee, R. Xiao and C. Harrison. "Gaze+Gesture: Expressive, Precise and Targeted Free-Space Interactions," *Proc. ACM Int. Conf. Multimodal Interaction (ICMI '15)*, pp. 131-138, 2015.

[15] X. Chen, J. Schwarz, C. Harrison, J. Mankoff and S. E. Hudson. "Air+touch: interweaving touch & in-air gestures," *Proc. 27th Ann. Symp. User Interface Software and Technology (UIST '14),* pp. 519-525, 2014

[16] K. Dorfmuller-Ulhaas and D. Schmalstieg. "Finger tracking for interaction in augmented environments," *Proc. IEEE and ACM Int. Symp. Augmented Reality*, pp. 55-64, 2001.

[17] M. A. Fischler and R. C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM* 24, 6 (June 1981), 381-395.

[18] T. B. Fitzpatrick. "Soleil et peau," *J Med Esthet* 2.7 (1975): 33-34.

[19] J. Gugenheimer, D. Dobbelstein, C. Winkler, G. Haas and E. Rukzio. "FaceTouch: Enabling Touch Interaction in Display Fixed UIs for Mobile Virtual Reality," *Proc. 29th Ann. Symp. User Interface Software and Technology (UIST '16)*, pp. 49-60, 2016.

[20] M. Hachet, B. Bossavit, A. Cohé and J-B de la Rivière. "Toucheo: multitouch and stereo combined in a seamless workspace," *Proc. 24th Ann. Symp. User Interface Software and Technology (UIST '11)*, pp. 587-592, 2011.

[21] J. Y. Han. "Low-cost multi-touch sensing through frustrated total internal reflection," *Proc. 18th Ann. Symp. User Interface Software and Technology (UIST '05)*, pp. 115-118, 2005.

[22] C. Harrison, H. Benko and A. D. Wilson. "OmniTouch: wearable multitouch interaction everywhere," *Proc. 24th Ann. Symp. User Interface Software and Technology (UIST '11)*, pp. 441-450, 2011.

[23] A. Hettiarachchi and D. Wigdor. "Annexing Reality: Enabling Opportunistic Use of Everyday Objects as Tangible Proxies in Augmented Reality," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '16)*, pp. 1957-1967, 2016.

[24] J. D. Hincapié-Ramos, X. Guo, P. Moghadasian and P. Irani. "Consumed endurance: a metric to quantify arm fatigue of mid-air interactions," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '14)*, pp. 1063-1072, 2014.

[25] K. Hinckley, S. Heo, M. Pahud, C. Holz, H. Benko, A. Sellen, R. Banks, K. O'Hara, G. Smyth and W. Buxton. "Pre-Touch Sensing for Mobile Interaction," *Proc. 2016 SIGCHI Conf. Human Factors in Computing Systems (CHI '16)*, pp. 2869-2881, 2016

[26] C. Holz and P. Baudisch. "The generalized perceived input point model and how to double touch accuracy by extracting fingerprints," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '10)*, pp. 581-590, 2010.

[27] HTC Corporation. HTC Vive Controller. https://www.vive.com/us/accessory/controller/

[28] W. Hürst and C. van Wezel. "Gesture-based Interaction via Finger Tracking for Mobile Augmented Reality," *Multimedia Tools and Applications 62*, 1 (January), pp. 233-258.

[29] H. Ishii and B. Ullmer. "Tangible bits: towards seamless interfaces between people, bits and atoms," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '97)*, pp. 234-241.

[30] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison and A. Fitzgibbon.

"KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera," *Proc. 24th Ann. Symp. User Interface Software and Technology (UIST '11)*, pp. 559-568, 2011.

[31] R. Jota, A. Ng, P. Dietz and D. Wigdor. "How fast is fast enough?: a study of the effects of latency in direct-touch pointing tasks," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '13)*, pp. 2291-2300, 2013.

[32] H. Koike, Y. Sato and Y. Kobayashi. "Integrating paper and digital information on EnhancedDesk: a method for realtime finger tracking on an augmented desk system," *ACM Trans. Comput.-Hum. Interact.* 8, 4 (December 2001), pp. 307-322.

[33] Oliver Kreylos. Vrui VR Toolkit. http://idav.ucdavis.edu/~okreylos/ResDev/Vrui.

[34] Leap Motion, Inc. Leap Motion Mobile VR Platform. https://www.leapmotion.com/product/vr

[35] SK Lee, W. Buxton and K. C. Smith. "A multi-touch three dimensional touch-sensitive tablet," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '85)*, pp. 21-25, 1985.

[36] T. Lee and T. Hollerer. "Handy AR: Markerless Inspection of Augmented Reality Objects Using Fingertip Tracking," *Proc. 11th IEEE Int. Symp. Wearable Computers (ISWC '07)*, pp. 1-8, 2007.

[37] J. Letessier and F. Bérard. "Visual tracking of bare fingers for interactive surfaces," *Proc. 17th Ann. Symp. User Interface Software and Technology (UIST '04)*, pp. 119-122, 2004.

[38] R. W. Lindeman, R. Page, Y. Yanagida and J. L. Sibert. "Towards full-body haptic feedback: the design and deployment of a spatialized vibrotactile feedback system," *Proc. ACM Symp. Virtual Reality Software and Technology (VRST '04)*, pp. 146-149. 2004.

[39] W. R. Mark, L. McMillan, G. Bishop. "Post-Rendering 3D Warping," *Proc. Symp. on Interactive 3D Graphics (I3D '97)*, pp. 7-16. 1997.

[40] T. H. Massie and J. K. Salisbury. "The PHANToM Haptic Interface: A Device for Probing Virtual Objects," *ASME Winter Annual Meeting*, DSC-Vol. 55-1, pp. 295–300, 1994.

[41] D. Medeiros, L. Teixeira, F. Carvalho, I. Santos and A. Raposo. "A tablet-based 3D interaction tool for virtual engineering environments," *Proc. 12th ACM SIGGRAPH Int. Conf. on Virtual-Reality Continuum and Its Applications in Industry (VRCAI '13)*, pp. 211-218. 2013.

[42] Microsoft Corporation. Kinect hardware. https://developer.microsoft.com/en-us/windows/kinect/hardware

[43] Microsoft Corporation. Microsoft HoloLens. https://www.microsoft.com/microsoft-hololens/

[44] Microsoft Corporation. Mixed Reality – Voice Input. https://developer.microsoft.com/windows/mixed-reality/voice_input

[45] Microsoft Corporation. Use the HoloLens clicker. https://support.microsoft.com/help/12646

[46] Microsoft Corporation. Use your Xbox Wireless Controller on Samsung Gear VR. https://support.xbox.com/en-US/xbox-one/accessories/use-samsung-gear-vr-with-xbox-controller

[47] A. Ng, J. Lepinski, D. Wigdor, S. Sanders and P. Diet. "Designing for low-latency direct-touch input," *Proc. 25th Ann. Symp. User Interface Software and Technology (UIST '12)*, pp. 453-464, 2012.

[48] Oculus VR. "Asynchronous TimeWarp (ATW)." https://developer.oculus.com/documentation/mobilesdk/latest/concepts/mobile-time-warp-overview/.

[49] J. A. Paradiso, K. Hsiao, J. Strickon, J. Lifton and A. Adler. "Sensor systems for interactive surfaces," *IBM Syst. J.* 39, 3-4 (July 2000), pp. 892-914.

[50] J. A. Paradiso, C. K. Leo, N. Checka, and K. Hsiao. "Passive acoustic sensing for tracking knocks atop large interactive displays," *Proc. IEEE Sensors '02*, pp. 521-527. 2002.

[51] H. M. Park, S. H. Lee and J. S. Choi. "Wearable augmented reality system using gaze interaction," *Proc. IEEE/ACM Int. Symp. Mixed and Augmented Reality (ISMAR '08)*, pp. 175-176, 2008.

[52] E. N. Saba, E. C. Larson and S. N. Patel. "Dante vision: In-air and touch gesture sensing for natural surface interaction with combined depth and thermal cameras," *Proc. IEEE Emerging Signal Processing Applications (ESPA '12)*, pp. 167-170, 2012.

[53] Samsung. Gear VR: About the Touchpad. http://www.samsung.com/au/support/skp/faq/1073201

[54] R. Sodhi, I. Poupyrev, M. Glisson and A. Israr. "AIREAL: interactive tactile experiences in free air," *ACM Trans. Graph.* 32, 4, Article 134 (July 2013), 10 pages.

[55] J. A. Walsh, S. von Itzstein and B. H. Thomas. "Ephemeral Interaction using Everyday Objects," *Proc. 15th Australasian User Interface Conference (AUIC '14)*, pp. 29-37, 2014.

[56] D. Wei, S. Z. Zhou and D. Xie. "MTMR: A conceptual interior design framework integrating Mixed Reality with the Multi-Touch tabletop interface," *Proc. IEEE Int. Symp. Mixed and Augmented Reality (ISMAR '10)*, pp. 279-280, 2010.

[57] A. D. Wilson. "Depth sensing video cameras for 3D tangible tabletop interaction," *Proc. 2nd Int. Wkshp. Horizontal Interactive Human-Computer Systems (Tabletop '07)*, pp. 201-204, 2007.

[58] R. Xiao, C. Harrison and S. E. Hudson. "WorldKit: rapid and easy creation of ad-hoc interactive applications on everyday surfaces," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '13)*, pp. 879-888, 2013.

[59] R. Xiao, S. E. Hudson and C. Harrison. "DIRECT: Making Touch Tracking on Ordinary Surfaces Practical with Hybrid Depth-Infrared Sensing," *Proc. 2016 ACM Int. Conf. Interactive Surfaces and Spaces (ISS '16)*, pp. 85-94, 2016.

[60] R. Xiao, G. Lew, J. Marsanico, D. Hariharan, S. E. Hudson and C. Harrison. "Toffee: enabling ad hoc, around-device interaction with acoustic time-of-arrival correlation," *Proc. 16th Int. Conf. Human-computer interaction with mobile devices & services (MobileHCI '14)*, pp. 67-76, 2014.

[61] F. Zhou, H. B-L. Duh and M. Billinghurst. "Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR," *Proc. 7th IEEE/ACM Int. Symp. Mixed and Augmented Reality (ISMAR '08)*, pp. 193-202, 2008.