Opportunistic Controls: Leveraging Natural Affordances as Tangible User Interfaces for Augmented Reality

Steven J. Henderson   Steven Feiner

Columbia University*

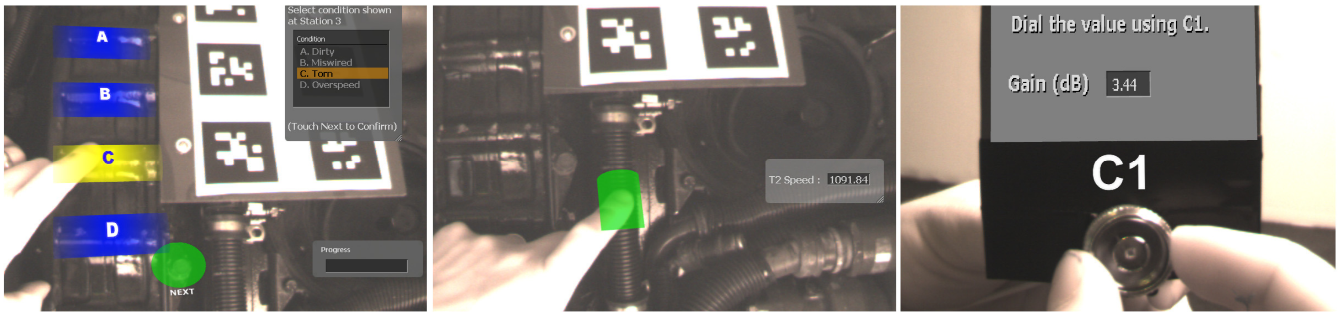

Figure 1: Opportunistic Controls in action. (left) A user manipulates a virtual button while receiving haptic feedback from the raised geometry of the underlying engine housing. (center) A user leverages the grooves in a wiring harness to help position a virtual slider. (right) A user turns the collar of an antenna connector to change a virtual text box value.

Abstract

We present *Opportunistic Controls*, a class of user interaction techniques for augmented reality (AR) applications that support gesturing on, and receiving feedback from, otherwise unused affordances already present in the domain environment. Opportunistic Controls leverage characteristics of these affordances to provide passive haptics that ease gesture input, simplify gesture recognition, and provide tangible feedback to the user. 3D widgets are tightly coupled with affordances to provide visual feedback and hints about the functionality of the control. For example, a set of buttons is mapped to existing tactile features on domain objects. We describe examples of Opportunistic Controls that we have designed and implemented using optical marker tracking, combined with appearance-based gesture recognition. We present the results of a user study in which participants performed a simulated maintenance inspection of an aircraft engine using a set of virtual buttons implemented both as Opportunistic Controls and using simpler passive haptics. Opportunistic Controls allowed participants to complete their tasks significantly faster and were preferred over the baseline technique.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation]: User Interfaces—*Input devices and strategies*, *Interaction Styles*; H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—*Artificial, augmented, and virtual realities*; I.3.6 [Computer Graphics]: Methodology and Techniques—*Interaction techniques*

*{henderso,feiner}@cs.columbia.edu

General Terms

Human Factors

Keywords

3D interaction, selection metaphor, tangible user interfaces, augmented reality.

1 Introduction

Many current and potential augmented reality (AR) application domains pose two sets of competing constraints. The first set of constraints limits extraneous head, eye, and hand movements beyond the immediate vicinity of a user's current task. For example, a mechanic servicing the internals of a turbine engine may find it impractical (or impossible) to reposition their hands to manipulate any device not currently within reach or sight. Likewise, head and eye movements that cause the mechanic to avert their gaze from the repair area can break context and increase task completion time. The second set of constraints relates to various policies, material properties, and physical space limitations that restrict modifications to the application's environment. For example, safety regulations and a confined repair space might preclude the mechanic from bringing in, or installing, certain interface devices (e.g., portable devices or keypads) that might otherwise compensate for limited head, eye, and hand movement.

To support these types of AR scenarios, we have developed a class of interaction techniques we call Opportunistic Controls, examples of which are shown in Figure 1. An *Opportunistic Control* (OC) is a tangible user interface [Ishii and Ullmer 1997] that leverages naturally occurring, tactilely interesting, and otherwise unused *affordances*—properties of an object that determine how it can be used [Gibson 1986; Norman 1988]. These affordances serve as tactile landmarks [Blaskó and Feiner 2004] that provide inherent passive haptic feedback [Lindeman et al. 1999] for hand gestures and are augmented with overlaid 3D widgets to provide visual feedback. Ideally, OCs are "harvested" from compatible surfaces in the physical task domain of the AR application. As we describe later, certain characteristics of the tactile landmarks are exploited to simplify gesture recognition.

An OC interface enables a user to interact with an AR application by touching naturally occurring surfaces within an application's task environment. For example, the aforementioned mechanic servicing the turbine engine might use various fasteners (e.g., screws and bolts) located on individually serviced components to display documentation specific to each component. A rotating washer or other surface on the same component can be used to page through the documentation or select entries from a list of observed conditions. A grooved surface in the vicinity of the component, such as a wiring harness or door hinge, might map to a virtual spinner used to enter diagnostic data or set various component parameters.

This approach creates a tangible user interface with three distinguishing properties: (1) leveraging otherwise unused, and unassociated objects that are already in the task domain as primary user interface components, (2) deliberately exploiting certain features of these objects for passive haptics and hand gesture recognition, and (3) minimizing the need for external user interface artifacts. As we describe below, this generalizes earlier work on passive haptics.

2 Related Work

There is much previous work on the use of haptic feedback in user interfaces in general and 3D user interfaces in particular. Some of this involves *active haptics* (e.g., [Brooks et al. 1990]), in which active devices, typically using motors, create forces and torques as part of the user interface. Here, we concentrate on previous work on *passive haptics*, in which passive elements in the environment respond to user interaction.

Buxton and colleagues [1985] added a cardboard overlay with cutout holes to a 2D touch tablet, creating a set of separate widgets, each of which could be discriminated through tactile feedback, encouraging eyes-free use. Weimer and Ganapathy [1989] positioned a set of 3D virtual buttons operated with a DataGlove to be coplanar with a physical desktop, providing what they called "a natural source of tactile feedback." Later, Hinckley and colleagues [1994] used a ball or doll's head and a small plastic panel, both outfitted with 6DOF trackers as "passive interface props" with which a physician could control an interactive visualization of a patient's head when planning neurosurgery.

Several groups have used tracked hand-held tablets with tracked fingers or styli to provide a supportive mobile surface on which to operate 2D widgets in AR (e.g., [Szalavari and Gervautz 1997]) or VR (e.g., [Lindeman et al. 1999]). Lindeman, Sibert, and Hahn [1999] referred to this as "passive haptics" or "passive-haptic feedback." Later work by Insko and colleagues [2001] demonstrated the advantages of passive haptics in virtual environments, positioning styrofoam blocks to coincide with the walls of an otherwise virtual environment. (In fact, one could argue that essentially any immersive virtual environment in which the virtual floor is coplanar with the real floor is using passive haptics.)

Research on *tangible user interfaces* [Ishii and Ullmer 1997] uses a variety of physical artifacts, often tracked or recognized wirelessly, as physical representations of otherwise virtual data and to physicalize otherwise virtual interaction techniques.

All of this previous work either uses simple naturally occurring surfaces (e.g., [Weimer and Ganapathy 1989]) or introduces new objects into the environment, whether simple (e.g., [Szalavari and Gervautz 1997]) or more complex (e.g., [Hinckley et al. 1994]). In contrast, we are interested in the opportunistic use of objects that already exist in a particular task domain, and whose possibly complex surface geometry provides affordances that lend themselves well to certain kinds of interactions. Thus, OCs apply Buxton and colleagues' notion of 2D haptically discriminable widgets to generalize and extend Weimer and Ganapathy's early example of a set of 3D widgets laid out on a single undifferentiated existing surface, without adding additional objects to the environment.

3 Alternative User Interfaces

Prior to designing OCs, we considered some of the many alternative interaction techniques involving devices such as keyboards, keypads, and touch screens. If these devices are not readily available within the task domain, they can be added or mobile versions can be used. We rejected these alternatives because of the two sets of competing constraints highlighted in Section 1. Some AR task domains (e.g., aviation maintenance) are not amenable to the introduction of objects that are not indigenous to the domain. Even if mobile devices are made available, they may require a user to momentarily shift their hands and eyes away from a specific task. Some require that the user hold them in one hand (e.g., a Handykey Twiddler) or momentarily engage both hands (e.g., a wrist-worn device operated with the other hand).

In contrast, OCs use existing features of the domain environment to provide a suitable tangible user interface. If the user's eyes and hands must remain in a certain area, then affordances within that area may be able to be exploited as part of the user interface. Finally, when the user finishes their task, nothing remains behind that must be maintained, hidden, or removed.

It is important to address potential situations when the task domain lacks sufficient suitable features for our technique. For example, a user might encounter areas that do not offer enough of the right kind of features to satisfy a task's required number and type of OCs. In these cases, our technique would offer a smooth fallback to conventional passive haptic feedback techniques by binding one or more OCs to undifferentiated available flat surface regions.

4 Definition

We define an OC as the six tuple $\Omega = (\tau, \psi, \alpha, \Gamma, \beta, \rho)$, where:

- τ represents a continuous physical region bounding the naturally occurring affordance(s) serving as one or more tactile landmarks for hand gestures. This region is specified by a 3D physical model capturing the physical geometry used by the OC.
- ψ is a 3D widget satisfying the definition and design specifications provided by Conner and colleagues [1992]. Each instance of ψ consists of a virtual model representing the widget's geometry and an augmented transition network (ATN) specifying the widget's behavior.
- α is a function mapping the encapsulated virtual geometry of the widget (ψ) to the physical geometry of the affordance region (τ). This function serves to dynamically register the 3D widget's model at the correct location in τ based on the current state of the widget's ATN.
- $\Gamma = \{\gamma_1, \gamma_2, .. \gamma_n\}$ is the set of visually recognized hand gestures associated with the OC. These gestures share a common 3D model space and grammar.

- β represents the functional mapping from the grammar of Γ to the ATN of ψ and defines how an individual widget responds to gesturing.
- ρ is the 3D transformation required to map locations in the model space of Γ to the model space of τ. This transformation is used to detect when and where a gesture intersects with an OC's physical geometry.

As defined above, each OC consists of a particular contiguous physical region paired with a particular 3D widget that together respond to one or more gestures. This definition distills the OC to an atomic building block that can be collectively combined to form more complex interfaces.

It is useful to place our definition of OCs within the broader context of tangible user interfaces. Using Fishkin's taxonomy of tangible user interfaces [Fishkin 2004], OCs present a *nearby embodiment* to the user. That is, the output of applications featuring OCs will take place near the primary input device (the OC's physical affordance region, τ). Continuing the classification, each OC presents a *fully realized metaphor* to the user. Given the definition above, the virtual component of the OC (the 3D widget, ψ) is paired to the physical system (the physical affordance region, τ). When the user gestures on an OC, the 3D widget and physical affordance region respond and feel as one control.

5 Prototype

We developed a hardware and software architecture for studying OCs in an indoor laboratory setting. This architecture allowed us to create a specific prototype implementation that we evaluated by means of the user study described in Section 6.

5.1 Affordance Design

We experimented with three kinds of affordances in our prototype. The first kind includes unused objects in the environment that tangibly resemble buttons. These objects have physical contours that are easily distinguished by a user's hand. Examples include various fasteners (e.g., screws, bolts, and nuts), raised geometry, small holes, dimples, or the intersection of hard edges, as shown in Figure 2. OCs based on these types of surfaces support binary gestures in which the OC is activated when the user's hand intersects any part of the button. Here, passive haptic feedback associated with button-based OCs need only provide information about the button's *location* to prove useful (a result demonstrated in the user study). However, certain types of elastic surfaces might provide additional feedback about the state of the button. For example, objects made of rubber or malleable plastic could provide elastic feedback as the user presses the button.

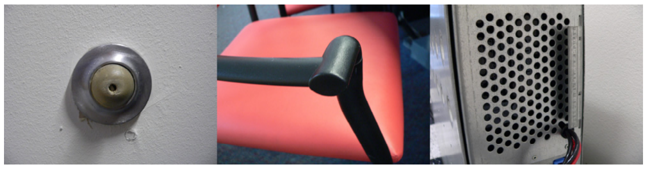
Figure 2: Objects that could support button OCs.

The second kind of affordance we explored includes linear or curved *static* surfaces in the environment that could support valuator-based OCs. These include smooth edges, pipes, cords, or natural surfaces, as shown in Figure 3. Gestures interacting with these types of surfaces require more precise tracking of the user's hand and subsequent positioning of 3D widgets. More interesting versions of these affordances are characterized by grooves, notches, and other textures that provide discretized feedback to the user as they gesture along the control (e.g., in the spirit of ridged surfaces that are specifically designed to provide haptic feedback [Murray-Smith et al. 2008]).

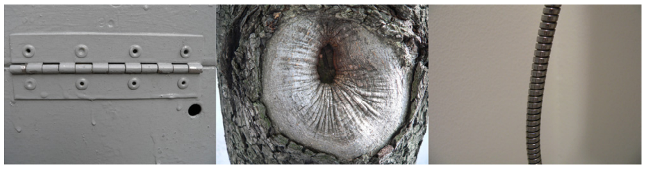
Figure 3: Objects that could support valuator OCs.

The third kind of affordance we studied involves surfaces associated with *moveable* objects in the environment. Examples include objects that rotate (e.g., free spinning washers, quick-release fasteners, and disconnected wiring connectors), objects that slide (e.g., large poster clips on top of dry erase boards, and control rods), and objects that bend (e.g., rubberized tubes and hoses), as shown in Figure 4. These objects allow for richer controls whose underlying physical geometry (τ) moves with the 3D widget (ψ) in response to the user's gestures.

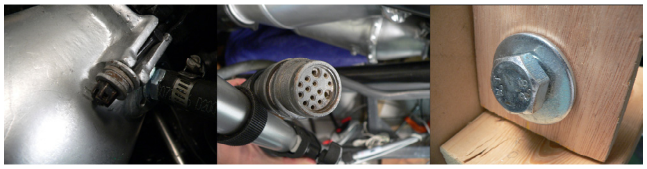
Figure 4: Objects that could support moveable OCs.

Throughout this exploration of the space of possible affordances, we adopted the following initial set of guidelines governing the selection of OCs:

- OCs should avoid desensitizing the user to a function of an overloaded object (e.g., using switches and knobs on a control panel for functions outside their design specification).
- OCs should not endanger the user or desensitize them to surfaces that could prove dangerous outside the context of the OC (e.g., using the tip of a spark plug as a button).
- When applicable, affordances should not overload objects that might become damaged through gesturing (either while the user is manipulating the OC or when the user tries to execute the gesture when the object is assuming its designed purpose).

5.2 Gesture Recognition Design

Gesture recognition is performed optically with a single camera mounted overhead with clear line of sight to all OCs in our environment. The camera is tracked by using the ARTag optical marker tracking library [Fiala 2005] to detect a fiducial array within the camera's current frame. We use a separate dedicated camera (as opposed to the cameras supporting the user's display) to free the user from having to look at the OCs. This allows the user to look in another location while gesturing and supports eyes-free interaction.

A separate execution thread analyzes each camera frame for the user's gesture and is implemented in three phases—data reduction, gesture matching, and gesture parsing. In the *data reduction* phase, we build on the appearance-based approach developed by Kjeldsen and Kender [1996] to segment each frame to locate the user's hands. The segmentation process first defines the collective gesture model space as one sharing the camera's 2D coordinate system. In doing so, the segmentation algorithm ignores any depth information in the scene. Despite several notable disadvantages discussed in Section 5.4, this relaxation speeds gesture recognition and provides a sufficient grammar for our OCs. We next define the physical model for each OC (τ) as a convex polyhedron that generally matches the physical contours of a particular OC. Each polyhedron is defined by 3D points positioned in a common physical interface coordinate system. The algorithm then defines the transformation ρ that enables conversion of coordinates in gesture space (camera coordinates) to and from physical interface coordinates.

This is an important step in the data reduction chain, and a particular advantage afforded by OCs, because it focuses the amount of follow-on image processing required for segmentation. Because the interaction technique is only concerned with gestures that might intersect with specific physical areas, segmentation algorithms can restrict processing to the 2D pixel regions that overlap with each OC's physical model. Moreover, because we track the position and orientation of the camera, ρ is computable in real-time by solving for the inverse model-view matrix received from the ARTag library. The algorithm calculates a segmentation window for each OC by using the value of ρ to construct a 2D bounding box encapsulating each OC's physical geometry (Figure 5, left). Each segmentation window is filtered for significant values of the primary color red in the source image's 24-bit RGB color format. When complemented by a controlled lighting environment, this filtering can effectively isolate a user's gesture from other objects in an image and supports a wide range of skin pigmentation. The result is a binary image that represents possible locations of the user's skin touching (or overlapping) each OC's geometry (Figure 5, right).

The algorithm then executes a connected component analysis for each OC bounding box and assumes the largest component in each is the user's hand, finger, or set of fingers. A high-pass filter is applied to the size of each maximum component to prevent noise from triggering buttons when skin is not present. During this step, the reduced pixel area provided by each OC's segmentation window again helps reduce data processing by limiting the breadth and depth of recursive connected component analysis.

During the *gesture matching* phase of the algorithm, the largest connected component C in each OC is evaluated for the location of point p_h, where p_h approximates the location of the user's finger tip in the connected component. This point is determined by selecting the leftmost point on the highest scan line of C. This approach assumes p_h is the highest leftmost point of the user's gesture in the camera's coordinate system. The algorithm then uses ρ^{-1} to translate the point p_h to the corresponding point t_h in the physical coordinates of the OC (τ). The location of t_h is used to match each OC's gesture (γ).

Gesture parsing is accomplished with a finite state machine (FSM) for each OC that resembles the ATN of the accompanying 3D widget (ψ). Each state in the FSM represents a command (e.g., "BUTTON_1_DOWN" or "SLIDER_2_UP") in the shared OC grammar Γ and the ATN's transitions are mapped to the OC gestures γ . The gesture algorithm then uses the functional mapping of β to translate the current command to the appropriate state in the corresponding 3D widget ψ . This final step ensures the widget's ATN is synchronized with the user's gestures.

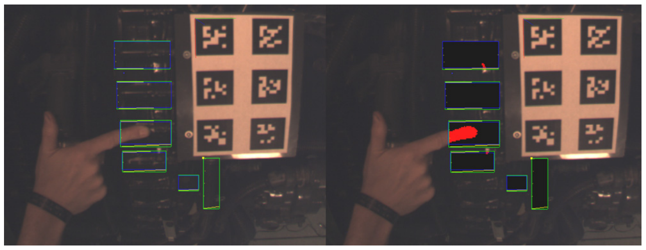
Figure 5: Unsegmented (left) and segmented (right) bounding boxes for a set of OCs. Graphics are added in debugging interface. (The user does not see this camera's view.)

5.3 Widget Design

We experimented with several designs for 3D widgets (ψ) as part of our prototype development. In each case, we sought to create an appropriate 3D model that matched the particular geometry of the OC. In all cases, we found that increasing the transparency of the widget models was helpful to allow users to partially view the OCs underlying geometry. The transparency also allows the user to partially view any gestures that might be occluded by the 3D Widget. The ATNs for each widget are modeled as specified by Conner and colleagues [1992]. This was a trivial process for button-type OCs, and involved slightly more complicated transitions for valuator and moveable OCs

5.4 Design Limitations

Our design suffers from several limitations. First, it relies on an optical marker-based tracking scheme to compute the value of ρ . Therefore, markers must be added to the domain environment, contradicting our vision of OCs as not requiring modifications of or additions to the task domain. We view the use of such markers as a temporary and minor violation of this premise, and believe that it will be possible to build on recent advances in markerless tracking (e.g., [Klein and Murray 2007; Bleser and Stricker 2008]) to replace our current use of markers. Moreover, these markers are cheaper and arguably easier to add than alternative interface devices, and minimally disturb the task environment. Second, our segmentation algorithm's relaxation of depth information limits the type of interactions one can perform—specifically clutching and hovering. Finally, because each OC's bounding box is segmented separately, the gesture algorithm can produce multiple gestures from multiple OCs. This was a deliberate design decision to support the user gesturing on more than one OC simultaneously (i.e., for multi-touch interactions). However, this feature requires more sophisticated program logic to reconcile potentially conflicting gestures. When coupled with our algorithm's lack of depth information, this feature can create situations in which hovering and clutching movements overlap neighboring controls and are erroneously interpreted as active gestures. We discuss this further in the description of our user study.

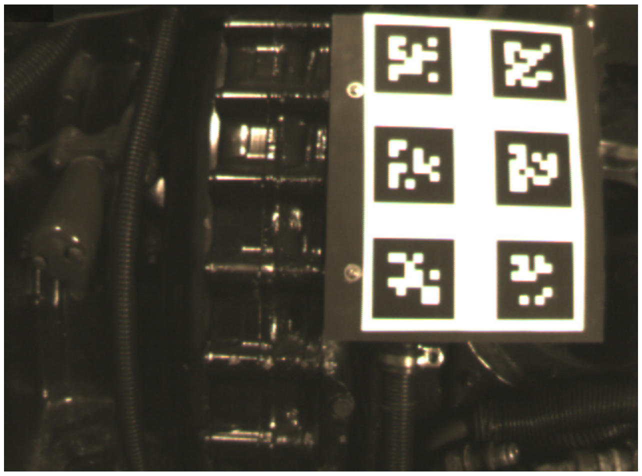
Figure 6: Dart 510 Engine (without OCs).

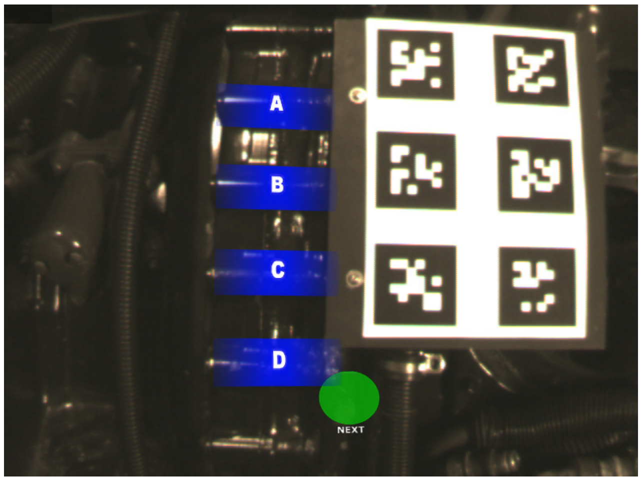

Figure 7: Dart 510 Engine (with five button-type OCs).

5.5 Prototype Implementation

We implemented our current prototype using two locally networked computers, one for managing gesture recognition for the OCs, and one for rendering OC widgets as part of a broader AR application testing the OCs in various scenarios. The decision to use two machines resulted in part from concerns about the resource load required to drive a binocular stereo video see-though display, while also supporting hand-gesture recognition. Additionally, we are interested in the ability of our software architecture to support scenarios where a single, relatively fixed server and attached cameras could provide gesture recognition to multiple users.

5.5.1 Implemented OCs

Our current implementation features five button-type OCs on a Rolls-Royce Dart 510 turboprop aircraft engine in our lab, as shown in Figures 6–8. Four of these OCs (Buttons A–D) map to large smooth protrusions on the outside of the engine's compression section and are used to select items in a virtual menu. The fifth button OC maps to a nearby bolt, and is used as a "next" button to navigate between menus. The menu button 3D widgets were modeled to resemble the underlying protrusions, while the "next" button 3D widget is a semi-transparent circle. All button models use multiple textures to provide visual feedback when the buttons are pressed.

We also implemented two other types of OCs. One is a valuator-based OC that maps a slider to a grooved wiring harness on the Dart engine (Figure 1, center). This slider is used to control a numeric value recorded in a text box. The other is a rotating OC that maps an antenna connector to a virtual text box (Figure 1, right). As the user rotates the connector's collar, the text box changes value.

5.5.2 Gesture Recognition

The gesture-tracking algorithm runs on a dedicated Dell M1710 XPS laptop connected to a fixed Point Grey Firefly MV 640×480 resolution color camera tracked by a single ARTag fiducial array mounted near the five button-type OCs (Figure 8). The gesture recognition application segments the five button-type OCs and parses gestures at a 30 fps rate.

5.5.3 Opportunistic Control Application

We implemented a central OC application that integrates all aspects of gesture recognition, rendering, user tracking, and AR task management. This application executes on a PC running Windows XP Professional, with a single NVIDIA Quadro 4500 graphics card. We then attached a custom-built stereo video see-through head worn display (HWD). This HWD was constructed from a Headplay 800×600 resolution color stereo gaming HWD with two Point Grey Firefly MV 640×480 resolution color cameras mounted to the front and connected to separate IEEE 1394a buses on the PC (Figure 8). The cameras are equipped with 5mm micro lenses and capture at a 30 fps rate.

Tracking is provided by two systems. For the user's head, we used a ceiling-mounted InterSense IS900 6DOF tracker to track a single station mounted on the HWD. Head tracking data is used to position all virtual content, with the exception of the 3D widgets that are part of the OCs. These widgets are positioned using the same optically tracked ARTag fiducial array used by the gesture recognition camera, sensed with the HWD's left camera. Note that the HWD cameras operate independently of the fixed gesture recognition camera in order to facilitate eyes-free gesture recognition.

The primary AR application software was developed as a game engine "mod" using the Valve Source Engine Software Development Kit. The engine's "player" serves as a virtual proxy for the

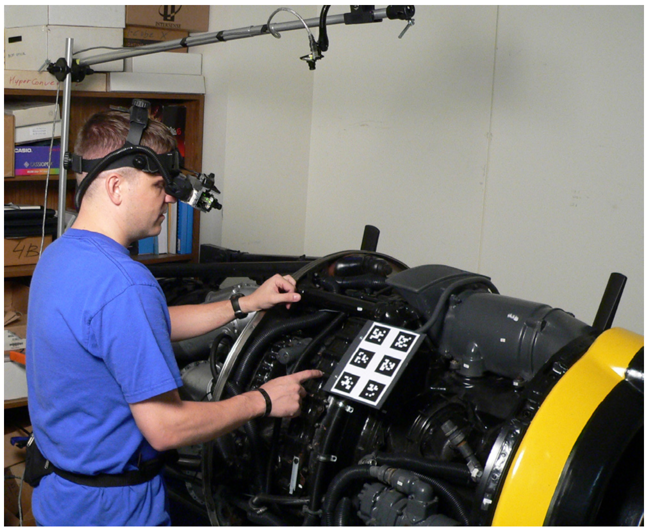
Figure 8: A user (wearing a stereo video see-through HWD) manipulates OCs with our prototype. The fixed gesture recognition camera appears at the top of the photo.

user and is positioned by tracking information from the IS900. All virtual content in the AR scene is provided by custom game engine models, GUI elements, and other components. Full resolution stereo video from the two Firefly MV cameras is stretched to the scene's back buffer via an external DLL that hooks the game engine's instance of the DirectX graphics interface via the Windows Detours library. The entire scene is rendered in stereo at 800×600 resolution with an average frame rate of 60 fps, synchronized to the display refresh rate. (Note: the effective video frame rate is approximately 25 fps due to the software upscaling from 2×640×480 to 2×800×600).

6 User Study

We designed a user study to compare the performance and general acceptance of our OC prototype to that of a more standard tangible user interface technique. Fifteen participants (11 male and 4 female), ages 20–34, were recruited by mass email to the Computer Science students at our university and by flyers distributed throughout the campus, and were paid $10 each. All participants were frequent computer users, but only two had experience with VR or AR techniques or technology. All participants but one identified themselves as right handed. Eight participants identified themselves as requiring corrective contact lenses or glasses. All of these users determined that the separate left and right eye focus adjustments on the Headplay display provided adequate correction.

6.1 Baseline Comparison Technique

We selected virtual buttons projected on a single undifferentiated surface as the baseline comparison technique for the study (herein referred to as BL). This technique is similar to the one used by Weimer and Ganapathy [1989]. More recent versions optically track the user's fingers, and have proven robust enough for commercialization as "virtual keyboards" [Roeber et al. 2003; Tomasi et al. 2003]. In order to adapt this technique to our prototype, we installed a 60 cm (width) x 78 cm (height) x 0.3 cm (thickness) panel of PVC plastic over the top of the part of the Dart engine that we used to implement the OCs described in Section 5.5.1. The panel, shown in Figure 9, was positioned and curved such that the virtual buttons would appear in the same locations and could use the same tracking and segmentation algorithms, as their OC counterparts, but on an undifferentiated surface. The panel was attached with quick release hardware to facilitate rapid transitioning between the two techniques during our within-subject study.

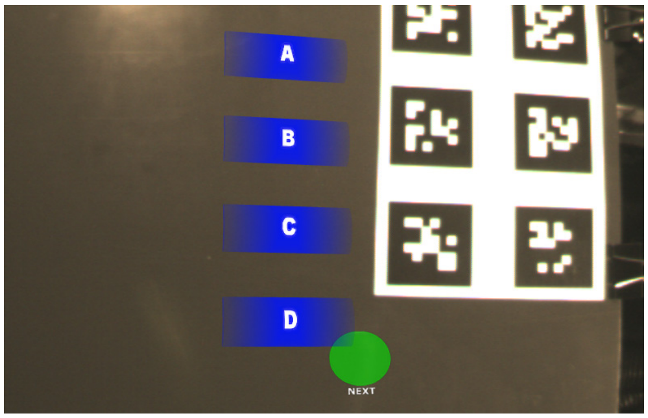

Figure 9: Baseline comparison technique (BL).

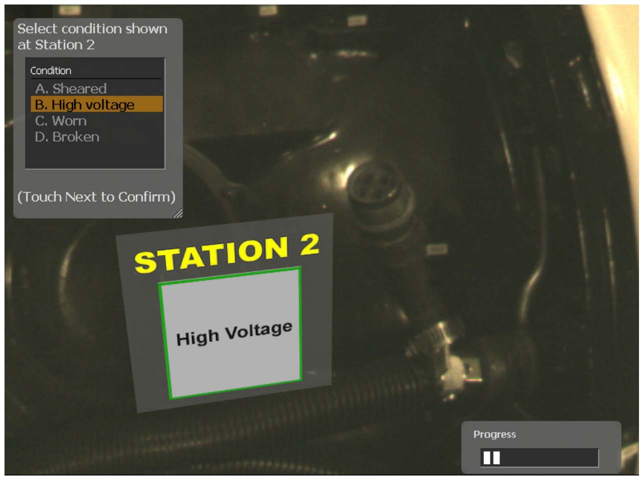

Figure 10: Sample selection task featured in the user study.

6.2 Task

Participants were asked to perform a selection task simulating the mechanical inspection of an actual Rolls Royce Dart 510 turboprop engine in our lab. This selection task consisted of matching target text displayed on 3D virtual placards positioned at locations on the engine with a corresponding text entry in a virtual 2D list displayed at a fixed location on the HWD. The 3D placards are registered to subcomponents of the engine to simulate specific items checked during the inspection. Each target text entry corresponds to a technical maintenance failure condition that might be recognized, observed, and recorded by a trained mechanic (e.g., "Broken" or "Cracked"). This target failure condition was randomly chosen from a list of thirty-two actual failures codes sampled from [DA PAM 738-751].

To successfully complete an individual trial, the user must use virtual buttons to highlight and confirm the target condition in the 2D list. The list contains four positions and randomly populates these positions with the target and three other alternate conditions. Participants use four virtual buttons mapped to each position in the list for the highlight step, and confirm the highlighting with a fifth virtual button. Figure 10 shows an example 3D placard with target text and the accompanying 2D menu, as seen in the HWD. In order to simplify the study design, we did not evaluate the valuator-based or rotation OCs.

6.3 Procedure

A within-subject, repeated measures design was used consisting of two techniques (OC and BL) and five inspected locations on the engine. The experiment lasted approximately 60 minutes and was divided into two blocks with a short break between blocks. Each block consisted of all trials for one of the two techniques, and the order was counterbalanced across participants. At the start of the experiment, each participant was shown an instructional video demonstrating the techniques. Before each block, each participant was afforded an opportunity to rehearse the technique using practice trials until they felt comfortable.

The timed portion of the block consisted of 50 trials divided uniformly over five locations on the engine. Each trial began by first presenting a single virtual placard at one of the five randomly chosen locations. Cueing information was presented to the user prompting them to locate and read the target condition displayed on the placard. This portion of the trial was not timed. When then user positioned and oriented their head so that the placard was under a crosshair in the middle of their field of view, the 2D list appeared and the trial timer started. Once the user highlighted and confirmed any condition (right or wrong) in the 2D list, the trial ended. The experiment logic then logged the overall completion time, the displayed target condition, and the user's selection from the list. The block then proceeded to the next trial in repeated fashion until the participant had experienced ten random target conditions at each of the five locations.

6.4 Hypotheses

Prior to the experiment, we proposed the following hypotheses:

(1) OC would be faster than BL, as the differentiable tactile landmarks would reduce homing time and facilitate eyes-free manipulation of the virtual buttons.

(2) OC would be more accurate than BL, as the tactile landmarks would focus gestures and prevent stray entries.

7 Results

We first filtered our collected data for outliers, which we defined as selection tasks lasting longer than 10 seconds. These outliers accounted for 3.5% of all trials, with a total of 23 occurring during the OC block and 29 occurring during the BL block. The majority of outliers resulted when the user paused in the middle of a selection task to adjust the HWD or ask a question. We then analyzed the remaining data set for completion time, error rate, and subjective ratings, with $\alpha = 0.05$.

7.1.1 Completion Time Analysis

We applied a 2 (Technique) × 5 (Location) repeated measure ANOVA on mean selection time from a subset of the outlier free data with our participants as the random variable. This subset included only those trials where the user correctly selected the target condition from the menu (96% of our outlier-filtered trials).

Technique had a significant main effect on selection completion times ($F_{(1,28)}$=8.11, $p < 0.001$). On average, the OC technique was 16% faster (Figure 11) than the BL baseline technique, which was statistically significant ($t_{(14)}$= 4.983, $p < 0.001$). This result confirms our first hypothesis. Finally, the interaction of Technique and Location did not have a significant main effect on completion time for the selection task.

7.1.2 Error Rate Analysis

We applied a 2 (Technique) × 5 (Location) repeated measure ANOVA on mean error rate data, with our participants as random variables. However, we failed to identify any significant effects on error rates. Specifically, we found no evidence of Technique affecting error rates and thus failed to confirm our second hypothesis. We attribute this result to two design shortcomings. First, based on our observations of the experiment and user input,

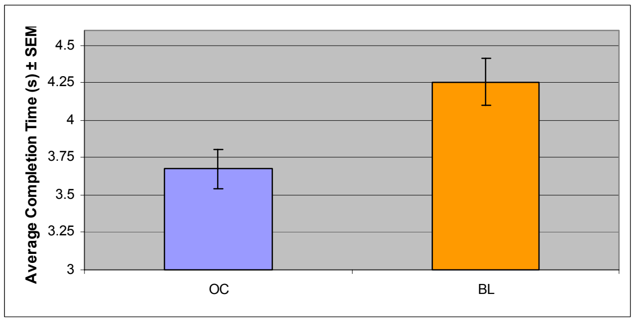

Figure 11: Average completion times (seconds) for OC (left) and BL (right). OC was 16% faster than (BL), which was a significant speedup.

Technique/*Factor*	MEAN	MEDIAN	MODE	MIN	MAX
OC					
Simple to use	4.00	4	4	1	5
Level of satisfaction	3.87	4	4	1	5
Intuitiveness	4.67	5	5	4	5
BL					
Simple to Use	3.43	3	3	1	5
Level of satisfaction	3.40	3	3	2	5
Intuitiveness	3.97	4	5	2	5

Figure 12. Survey results.

the "next" virtual button was placed too close to the physical protrusion on the engine that was mapped to the virtual button used to select the bottom item in the menu. As a result, the user's hand gesture could accidentally stray into the segmentation window of this bottom button just prior to activation of the next button. This would erroneously update the user's selection without allowing time to detect the stray gesture before confirmation. Second, our gesture recognition algorithm does not provide a depth filter. As a result, if the user's hand hovers over the top of any buttons while transitioning, the algorithm will detect this hovering as button activation. Including depth information in our gesture recognition algorithm and more careful selection of OC affordances could decrease the number of these errors.

7.1.3 Subjective Analysis

We asked each participant to complete a post-experiment questionnaire. This questionnaire featured five-point Likert scale questions (where 1 is most negative, 5 is most positive) to evaluate ease of use, satisfaction level, and intuitiveness for each interaction technique. These summary results from these ratings, shown in Figure 12, are difficult to generalize given our small population size and individual rating systems. However, we offer them as interesting indicators of how our technique might be perceived among a larger population. Collectively, the subjects rated the OC technique as better than the baseline in terms of ease of use (4.00), satisfaction (3.87), and intuitiveness (4.67). When asked to rank the technique they would rather use to perform the task, 11 of 15 participants selected the OC technique. General participant comments reflected a preference for tactile landmarks to help with homing and feedback. The majority of participants expressed frustration with the top-to-bottom button layout because of the inability of the gesture algorithm to distinguish hovering from selection.

We also noticed several interesting behaviors in participants. First, many participants were uncomfortable touching physical parts of the aircraft engine. As one participant recounted, touching the plastic surface of BL felt more familiar than touching louvers and bolts on an engine. Second, several participants used additional passive haptics from the task environment that were not linked to our button OCs to assist in the selection task. These techniques involved incorporating surfaces adjacent to the buttons as homing points between gestures. Third, even though we deliberately did not mention two-handed techniques to the participants, several participants quickly incorporated them into their technique. The fastest recorded completion time originated from one such participant.

Additionally, although our user study did not explicitly feature tasks mandating eyes-free interaction, several users did attempt this technique during both OC and BL trials. Multiple users commented on how they felt more comfortable attempting eyes-free interaction using OCs as opposed to BL.

8 Conclusions And Future Work

We were pleased that our initial prototype implementation of OC was able to support faster completion times than those of the baseline. Moreover, we were encouraged by the level of enthusiasm expressed by the user study participants for our technique. We also believe that minor modifications to our design (e.g., selecting a better arrangement of buttons) could result in a significant improvement over the baseline in error rate performance.

Our immediate research focus is on improving the segmentation algorithm to replace marker-based tracking with a feature-based approach. We believe many of the same rich features embodied in tactilely interesting OCs could also be leveraged for tracking. Other planned improvements in the segmentation process include adding depth information; for example by using a stereo pair of cameras or a depth camera [3DV Systems 2008].

We are also interested in developing tools that would allow a user to quickly designate promising looking elements in the environment as OCs. This would require having the user locate a physical object, select a widget type, and specify how the physical object is mapped to the widget. It might even be possible for the system to recognize certain types of features to automatically suggest possible OCs to support the task at hand.

In closing, we have presented a class of user interaction techniques for AR applications that support gesturing on, and receiving feedback from, otherwise unused affordances already present in the domain environment. A collection of Opportunistic Controls was demonstrated to be faster than a similarly laid out set of controls on an undifferentiated surface. While not suitable for all user interface scenarios, this technique may be a good choice for tasks requiring eye and hand focus and restricting other interaction techniques.

Acknowledgments

This research was funded in part by DAFAFRL Grant FA8650-05-2-6647 and ONR Grant N00014-04-1-0005, and a generous gift from NVIDIA. We also thank Bengt-Olaf Schneider of NVIDIA for providing the StereoBLT SDK used to support the display of stereo camera imagery.

References

DA PAM 738-751, 1992. *Functional Users Manual for The Army Maintenance Management System - Aviation (TAMMS-A)*. Washington D.C: U.S. Army.

BLASKÓ, G. AND FEINER, S. 2004. An interaction system for watch computers using tactile guidance and bidirectional segmented strokes. *Proc. ISWC 2004*. 120-123.

BLESER, G. AND STRICKER, D. 2008. Advanced tracking through efficient image processing and visual-inertial sensor fusion. *Proc. IEEE Virtual Reality Conf. (VR '08)*, 137-144.

BROOKS, F. P., OUH-YOUNG, M., BATTER, J. J. AND KILPATRICK, P. J. 1990. Project GROPE-Haptic displays for scientific visualization. *Proc. 17th Annual Conf. on Comp. Graphics and Interactive Techniques*, 177-185.

BUXTON, W., HILL, R. AND ROWLEY, P. 1985. Issues and techniques in touch-sensitive tablet input, *Proc. 12th Annual Conf. on Comp. Graphics and Interactive Techniques*, 215-224.

CONNER, B. D., SNIBBE, S. S., HERNDON, K. P., ROBBINS, D. C., ZELEZNIK, R. C. AND DAM, A. V. 1992. Three-Dimensional Widgets. *Proc. 1992 Symp. on Interactive 3D Graphics*, 183-188.

FIALA, M. L. 2005. ARTag, a fiducial marker system using digital techniques. *Proc. 2005 IEEE Computer Society Conf. on Comp. Vision and Pattern Recognition (CVPR'05) - Volume 2*, 590-596.

FISHKIN, K. P. 2004. A taxonomy for and analysis of tangible interfaces, *Personal Ubiquitous Computing* 8, 5, 347-358.

GIBSON, J. 1986. *The Ecological Approach to Visual Perception*. Hillsdale, N.J.: Lawrence Erlbaum Associates.

HINCKLEY, K., PAUSCH, R., GOBLE, J. C. AND KASSELL, N. F. 1994. Passive real-world interface props for neurosurgical visualization. *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, 452 - 458.

INSKO, B. E., MEEHAN, M. J., WHITTON, M. C. AND FREDERICK P. BROOKS, J. 2001. *Passive haptics significantly enhances virtual environments*. Technical Report 0-493-17286-6. The University of North Carolina at Chapel Hill.

ISHII, H. AND ULLMER, B. 1997. Tangible Bits: Towards seamless interfaces between people, bits and atoms. *Proc. SIGCHI Conf. on Human Factors in Comp. Sys.* 234-241.

KJELDSEN, R. AND KENDER, J. 1996. Finding skin in color images. *Proc. 2nd International Conf. on Automatic Face and Gesture Recognition (FG '96),* 312.

KLEIN, G. AND MURRAY, D. 2007. Parallel tracking and mapping for small AR workspaces. *Proc. International Symp. on Mixed and Augmented Reality (ISMAR'07).*

LINDEMAN, R. W., SIBERT, J. L. AND HAHN, J. K. 1999. Hand-held windows: towards effective 2D interaction in immersive virtual environments. *Proc. IEEE Virtual Reality Conference*, 205-212.

MURRAY-SMITH, R., WILLIAMSON, J., HUGHES, S. AND QUAADE, T. 2008. Stane: Synthesized surfaces for tactile input. *Proc. of the Twenty-sixth annual SIGCHI Conf. on Human Factors in Comp. Systems*, 1299-1302.

NORMAN, D. 1988. *The Psychology of Everyday Things*. New York: Basic Books.

ROEBER, H., BACUS, J. AND TOMASI, C. 2003. Typing in thin air: The Canesta projection keyboard - a new method of interaction with electronic devices. *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, 712-713.

3DV SYSTEMS, 2008. http://www.3dvsystems.com

SZALAVARI, Z. AND GERVAUTZ, M. 1997. The personal interaction panel - a two-handed interface for augmented reality *Computer Graphics Forum* 16, 3, 335-346.

TOMASI, C., RAFII, A. AND TORUNOGLU, I. 2003. Full-size projection keyboard for handheld devices, *Communications of the ACM* 46, 7, 70-75.

WEIMER, D. AND GANAPATHY, S. K. 1989. A synthetic visual environment with hand gesturing and voice input. *Proc. SIGCHI Conf. on Human Factors in Comp. Systems*, 235-240.