Augmented Reality Interfaces for Procedural Tasks

Steven J. Henderson

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY
2011
ABSTRACT

Augmented Reality Interfaces for Procedural Tasks

Steven J. Henderson

Procedural tasks involve people performing established sequences of activities while interacting with objects in the physical environment to accomplish particular goals. These tasks span almost all aspects of human life and vary greatly in their complexity. For some simple tasks, little cognitive assistance is required beyond an initial learning session in which a person follows one-time compact directions, or even intuition, to master a sequence of activities. In the case of complex tasks, procedural assistance may be continually required, even for the most experienced users. Approaches for rendering this assistance employ a wide range of written, audible, and computer-based technologies.

This dissertation explores an approach in which procedural task assistance is rendered using augmented reality. Augmented reality integrates virtual content with a user’s natural view of the environment, combining real and virtual objects interactively, and aligning them with each other. Our thesis is that an augmented reality interface can allow individuals to perform procedural tasks more quickly while exerting less effort and making fewer errors than other forms of assistance. This thesis is supported by several significant contributions yielded during the exploration of the following research themes:
**What aspects of AR are applicable and beneficial to the procedural task problem?** In answering this question, we developed two prototype AR interfaces that improve procedural task accomplishment. The first prototype was designed to assist mechanics carrying out maintenance procedures under field conditions. An evaluation involving professional mechanics showed our prototype reduced the time required to locate procedural tasks and resulted in fewer head movements while transitioning between tasks. Following up on this work, we constructed another prototype that focuses on providing assistance in the underexplored psychomotor phases of procedural tasks. This prototype presents dynamic and prescriptive forms of instruction and was evaluated using a demanding and realistic alignment task. This evaluation revealed that the AR prototype allowed participants to complete the alignment more quickly and accurately than when using an enhanced version of currently employed documentation systems.

**How does the user interact with an AR application assisting with procedural tasks?** The application of AR to the procedural task problem poses unique user interaction challenges. To meet these challenges, we present and evaluate a novel class of user interfaces that leverage naturally occurring and otherwise unused affordances in the native environment to provide a tangible user interface for augmented reality applications. This class of techniques, which we call *Opportunistic Controls*, combines hand gestures, overlaid virtual widgets, and passive haptics to form an interface that was proven effective and intuitive during quantitative evaluation. Our evaluation of these techniques includes a qualitative exploration of various preferences and heuristics for Opportunistic Control-based designs.
Table of Contents

List of Figures .................................................................................................................................. iii
List of Tables ..................................................................................................................................... xi
Acknowledgements .......................................................................................................................... xii

1 Introduction .................................................................................................................................... 1
  1.1 Research questions and dissertation goals ................................................................. 5
  1.2 Contributions .................................................................................................................. 7
  1.3 Structure of Dissertation .......................................................................................... 25

2 Related Work .................................................................................................................................. 27
  2.1 The Nature of Procedural Tasks ................................................................................ 27
  2.2 Teaching, Learning and Assistance for Procedural Tasks ........................................ 29
  2.3 Designing Assistance for Procedural Tasks ............................................................... 30
  2.4 Non-automated Forms of Assistance ...................................................................... 32
  2.5 Computerized Forms of Assistance ........................................................................ 37
  2.6 AR Interfaces ............................................................................................................. 42

3 Leveraging Augmented Reality in Informational Phases of Procedural Tasks .... 45
  3.1 Related Work ............................................................................................................. 47
  3.2 Prototype .................................................................................................................... 50
  3.3 User Study Design .................................................................................................. 64
  3.4 Quantitative Results ............................................................................................... 74
  3.5 Qualitative Results ................................................................................................. 93
  3.6 Discussion ............................................................................................................... 98

4 Leveraging Augmented Reality in Psychomotor Phases of Procedural Tasks .... 101
  4.1 Related Work ........................................................................................................... 105
  4.2 Selecting a Psychomotor Task Environment .......................................................... 107
  4.3 Psychomotor Phase Assistance .............................................................................. 115
  4.4 User Study ............................................................................................................. 119
  4.5 Quantitative Results ............................................................................................. 134
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.6</td>
<td>Qualitative Results</td>
<td>140</td>
</tr>
<tr>
<td>4.7</td>
<td>Comparison to Physical Labels</td>
<td>142</td>
</tr>
<tr>
<td>4.8</td>
<td>Discussion</td>
<td>147</td>
</tr>
<tr>
<td>5</td>
<td>Opportunistic Controls</td>
<td>148</td>
</tr>
<tr>
<td>5.1</td>
<td>Related Work</td>
<td>151</td>
</tr>
<tr>
<td>5.2</td>
<td>Alternative User Interfaces</td>
<td>152</td>
</tr>
<tr>
<td>5.3</td>
<td>Opportunistic Controls Definition</td>
<td>154</td>
</tr>
<tr>
<td>5.4</td>
<td>Designing Opportunistic Control Interfaces</td>
<td>155</td>
</tr>
<tr>
<td>5.5</td>
<td>OC User Observation Study</td>
<td>163</td>
</tr>
<tr>
<td>5.6</td>
<td>Prototype Implementation</td>
<td>180</td>
</tr>
<tr>
<td>5.7</td>
<td>OC Interface Technique User Study</td>
<td>185</td>
</tr>
<tr>
<td>5.8</td>
<td>Discussion</td>
<td>194</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions and Future Work</td>
<td>197</td>
</tr>
<tr>
<td>6.1</td>
<td>Summary of Contributions</td>
<td>197</td>
</tr>
<tr>
<td>6.2</td>
<td>Lessons Learned</td>
<td>199</td>
</tr>
<tr>
<td>6.3</td>
<td>Future Work</td>
<td>201</td>
</tr>
<tr>
<td>6.4</td>
<td>Final Thoughts</td>
<td>209</td>
</tr>
<tr>
<td>Appendix A</td>
<td>The ARMAR Architecture</td>
<td>210</td>
</tr>
<tr>
<td>A.1</td>
<td>Architecture Requirements</td>
<td>211</td>
</tr>
<tr>
<td>A.2</td>
<td>Related Work</td>
<td>212</td>
</tr>
<tr>
<td>A.3</td>
<td>ARMAR Software Architecture</td>
<td>214</td>
</tr>
<tr>
<td>A.4</td>
<td>ARMAR Hardware Architecture</td>
<td>224</td>
</tr>
<tr>
<td>A.5</td>
<td>Evolution of the ARMAR Architecture</td>
<td>225</td>
</tr>
<tr>
<td>A.6</td>
<td>Value Source-based Version of ARMAR</td>
<td>225</td>
</tr>
<tr>
<td>A.7</td>
<td>Goblin XNA-based Version of ARMAR</td>
<td>237</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>247</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1.1: Examples of task guidance systems. On the left, an investigator follows a deliberate procedure for documenting a crime scene [Cross, Baber, and Smith 2007]. On the right, an electrician uses a hand-held device to navigate a troubleshooting procedure for a telecommunication network (http://www.mobiledataforce.com). .................................................................3

Figure 1.2: Examples of IETM task guidance systems. (Left) A U.S. Navy electrician services an MH-60 helicopter with the aid of a notebook-based IETM. (Right) An example screenshot from an IETM interface (http://www.oneil.com). .........................................................................................4

Figure 1.3: A user wearing a head-worn display (left) prepares for the next step in a procedural task using the ARMAR system described in this dissertation. The user’s view through the display (right) shows the natural review of the repair area, as well as virtual content pertaining to the next step in the procedure. (Image on right depicts the view through the head-worn display.) ..........5

Figure 1.4: (Left) A mechanic wearing a tracked, head-worn display performs a maintenance task inside an LAV-25A1 armored personnel carrier using ARMAR. (Right) A view through the head-worn display, captured in a similar domain, uses AR to provide information to assist the mechanic. The view through the head-worn display for the LAV-25A1 domain was not cleared for publication due to security restrictions, necessitating the substitution of an image from an alternative domain. (Image on right shows the view through the video see-through display.) ..................................................................................11

Figure 1.5: Examples of AR assistance offered using an ARMAR prototype during psychomotor phases of procedural tasks. (Left) During assembly of a combustion chamber, a user wearing a head-worn display is presented with information about the alignment of assembled subcomponents. (Right) Following the assembly, additional assistance is presented to the user to help with rotating and aligning the combustion chamber prior to installation. (Images captured by a video camera mounted inside and looking through an optical see-through display. A post-render filter was applied to remove camera distortion and vignetting.) .................................................................15
Figure 1.6: LCD-based 3D documentation system being used in our study of psychomotor phase assistance. .................................................................17

Figure 1.7: The idealized documentation used in our follow-up experiment. The inset view depicts the virtual version displayed on the LCD screen. ..........................19

Figure 1.8: Opportunistic Controls in action. A user (top) wearing a head-worn display uses (bottom) virtual buttons to record the results of an inspection task while receiving haptic feedback from the raised geometry of the underlying engine housing. (Image at bottom is an exact screen capture of the view presented to the user wearing a video see-through head-worn display.) .................21

Figure 1.9: (Left) A user holding a hand-held video see-through display selects affordances for a notional OC used to control a hypothetical menu selection task in a home entertainment user interface domain. (Right) A close-up screenshot of the view presented to the user via the hand-held video see-through display. “Wizard of Oz” feedback is used to change the menu’s selected item in response to the user’s gesture. ...........................................................22

Figure 1.10: (Left) Button-based OC interface evaluated in our performance and acceptance user study. (Right) The undifferentiated version of the interface used as the study’s baseline. (Images are exact screen captures of the imagery presented to the user wearing a video see-through head-worn display.) .........................23

Figure 1.11: A selection task support by a prototype OC interface. (Left) After viewing a target condition rendered on a virtual 3D placard, (right) a participant uses OC buttons to select the condition in a 2D menu. (Images are exact screen captures of the imagery presented to the user wearing a video see-through head-worn display.) ..................................................................................24

Figure 2.1: A completed workcard [National Transportation Safety Board 1984]. ........33

Figure 2.2: An example of non-automated, single-sheet, consumer assembly instructions (http://www.alvinit.com). .................................................................35

Figure 2.3: An example job chart used to assist gas-station attendants (http://justacarguy.blogspot.com). ..................................................................................36

Figure 2.4: Screenshots from COMET [Feiner and McKeown 1991]. .........................38

Figure 2.5: A mechanic references an IETM while performing troubleshooting inside a UH60 helicopter (U.S. Army). ...............................................................40

Figure 2.6: A prototype wearable IETM demonstrated by Siegel and colleagues [2001]. An F-18 aircraft mechanic (top) wears an untracked HWD over his right eye, which presents pages from an IETM (bottom). (Photos courtesy Jane Seigel and Carnegie Mellon University.) .................................................42

Figure 3.1: A mechanic in our user study, wearing an AR display and wrist-worn control panel, performs a maintenance task inside an LAV-25A1 armored personnel carrier.........................................................45
Figure 3.2: An example of assistance presented by our prototype during a typical maintenance activity. (Image shows the view through the video see-through display.)

Figure 3.3: Typical localization sequence (1 of 4). When a target task is located behind the user, a screen-fixed arrow indicates the shortest rotation direction to the target. (Image shows the view through the video see-through display.)

Figure 3.4: Typical localization sequence (2 of 4). As the target task enters the user’s field of view, a tapered red semi-transparent 3D arrow appears, directing the user toward the target. In order to promote legibility, the tail of the arrow is smoothly adjusted and placed along the far edge of the display at each frame. (Image shows the view through the video see-through display.)

Figure 3.5: Typical localization sequence (3 of 4). As the user focuses on the target, the red 3D arrow increases in transparency and eventually disappears. The blue screwdriver is animated to show the correct rotation to remove the oil pressure sensor screw. (Image shows the view through the video see-through display.)

Figure 3.6: Typical localization sequence (4 of 4). As the red 3D arrow disengages, the interface spawns a highlighting effect for five seconds at the location of the target. Following this highlight effect, the interface removes the blue animated model of the screwdriver, to avoid interfering with the psychomotor portion of the task. (Image shows the view through the video see-through display.)

Figure 3.7: An example animated model. (a) As the user localizes on the cable connector, (b) the 3D attention-directing arrow fades and (c) the 3D model of the channel locks begins to animate to show the proper (d-f) motion of the tool. The tool also fades after 5 seconds of animation. (Images show the view through the video see-through display.)

Figure 3.8: A mechanic uses a wrist-worn controller to cue the next task in a repair sequence. The inset view shows additional features that appear during applicable tasks for controlling animations.

Figure 3.9: A representative screenshot that shows the result of applying many of the previously published design heuristics we followed when designing our prototype. (Image shows the view through the video see-through display.)

Figure 3.10: Example screen shot from the currently used IETM interface.

Figure 3.11: LCD Condition. (Image shows the view presented on the LCD.)

Figure 3.12: HUD Condition. (Image shows the view through the video see-through display.)

Figure 3.13: Approximate task azimuths and distances as viewed from above the turret and looking down. Neighboring task identifiers are separated by commas.
Figure 3.14: Average task completion times (seconds) across all task types for AR, HUD, and LCD.

Figure 3.15: Average task completion times (seconds) by common task type for AR, HUD, and LCD. The edges of each box represent the 25th and 75th percentiles, and whiskers extend to the most extreme data points not considered outliers. Points are drawn as outliers if they are larger than $q_3 + 1.5(q_3 - q_1)$ or smaller than $q_1 - 1.5(q_3 - q_1)$, where $q_1$ and $q_3$ are the 25th and 75th percentiles, respectively.

Figure 3.16: Average task localization times (seconds) for AR, HUD, and LCD.

Figure 3.17: Ranges of head rotation (degrees yaw) for all participants across each task. Tasks are stacked in layers. Each task layer shows ranges for AR (bottom), HUD (middle), and LCD (top).

Figure 3.18: 2D Histograms shown as heatmaps of normalized head positions (viewed from above the turret looking down) across all participants and tasks for AR (upper left), HUD (upper right), and LCD (lower right). The heatmap scale represents the relative number of hits in each bin of a $150 \times 150$ grid covering the task area. Bin sizes are 0.004m ($X$) and 0.003m ($Y$). The schematic diagram in the lower-left corner depicts the general orientation of the mechanics while data was collected for the heatmaps (also viewed from above the turret looking down).

Figure 3.19: Distribution of normalized yaw angles for AR and LCD for Task T4. In each plot, the value $x=0$ indicates the population's mean yaw orientation.

Figure 3.20: Head orientation (yaw) trajectories for each participant (S1–S6) under AR and LCD conditions. The $x$ axis shows normalized elapsed time for each task, and the $y$ axis shows rotational head displacement about the forward facing direction.

Figure 3.21: Survey response histograms by condition for ease of use (top), satisfaction (middle), and intuitiveness (bottom). Median values for each condition are shown as triangles.

Figure 3.22: Survey response histograms by condition for rankings of most intuitive system (top) and most preferred (bottom).

Figure 4.1: AR assistance in the psychomotor phase of a procedural task. (left) A user aligns two combustion chamber components while (right) viewing continuous, dynamic feedback about the alignment presented by an optical see-through HWD. (The image on the right was captured by a video camera mounted in a dummy head wearing the HWD. A post-render filter was applied to remove camera distortion and vignetting.)

Figure 4.2: Procedural task environment. (a) The Dart 510 engine. (b) Combustion chamber. (c) Upper combustion chamber “cone.” (d) Lower combustion chamber “can.” (Small retroreflective spheres bolted to components are used for optical tracking.)
Figure 4.3: An initial prototype for removing and installing combustion chambers (1 of 2). An animated 3D arrow indicates the direction of rotation to achieve a target alignment about the long axis of the chamber. (The image was captured by a video camera mounted inside and looking through the optical–see-through HWD. A post-render filter was applied to remove camera distortion and vignetting.)

Figure 4.4: An initial prototype for removing and installing combustion chambers (2 of 2). Localization information (a) orients the user to the combustion chamber. As the user picks up the chamber, (b) an animated 3D arrow indicates the direction of rotation to achieve longitudinal alignment. As the user turns toward the engine, (c–d) a 3D label is used to suggest the correct lateral orientation. As the user nears the engine with the chamber, (e) a semi-transparent virtual version of the chamber indicates the correct location and orientation for (f) the user to emulate. (Images show view through a Vuzix VR920 video see-through HWD with attached CamAR camera which is used here to promote clarity in the images. An optical–see-through HWD is used in the user study.)

Figure 4.5: A large, red, dynamic arrow (a) indicates the direction and magnitude of the motion required to bring the can and cone into alignment. As the can and cone approach alignment (b–c), the arrow reduces size and changes color through yellow to green. If the motion overshoots the alignment, (d) the arrow changes direction to indicate the required correction and (e) slowly fades out when (f) alignment is achieved. (Images show view through a Vuzix VR920 video see-through HWD with attached CamAR camera which is used here to promote clarity in the images. An optical–see-through HWD is used in the user study.)

Figure 4.6: Example of dynamic highlights. Prior to alignment (left), distinct highlights are rendered on the top and bottom holes (indicated by “Q” and “19” leader lines). When the holes are aligned (right), the bottom highlight slowly fades out presenting the appearance of a single hole. (Images show enlarged view through a video see-through display)

Figure 4.7: Study task environment. LCD screen (upper right) is used as primary display in the LCD control condition and turned off in the AR condition.

Figure 4.8: Experiment trial (1 of 5). (Top) When the participant initiates the trial by pressing the confirm button, the system loads the first step and (bottom) presents assistance in locating the can. (Images show views captured by a camera mounted inside the optical–see-through HWD. A post-render filter was applied to remove camera distortion and vignetting.)

Figure 4.9: Experiment trial (2 of 5). (Top) When the user picks up the can, the system senses this motion and (bottom) loads the next step instructing the user to place the can on the turntable. (Images show views captured by a camera mounted inside the optical–see-through HWD. A post-render filter was applied to remove camera distortion and vignetting.)
Figure 4.10: Experiment trial (3 of 5). (Top) After the can is placed on the turn table, the system instructs the participant to pick up a cone. (Bottom) The system then prompts for the cone to be placed on the can. (Images show views captured by a camera mounted inside the optical–see-through HWD. A post-render filter was applied to remove camera distortion and vignetting.)

Figure 4.11: Experiment trial (4 of 5). (Top) Dynamic alignment instructions are presented after the system detects the cone is atop the can. (Bottom) The participant then completes the alignment and inserts both pins. (Images show views captured by a camera mounted inside the optical–see-through HWD. A post-render filter was applied to remove camera distortion and vignetting.)

Figure 4.12: Experiment trial (5 of 5). (Top) When the participant finishes the alignment and pinning steps, they inform the system by pressing the red confirm button. (Bottom) The system then presents instructions for clearing the turntable. (Images show views captured by a camera mounted inside the optical–see-through HWD. A post-render filter was applied to remove camera distortion and vignetting.)

Figure 4.13: A user pins the cone to the can in the LCD condition. (Used with permission of the participant.)

Figure 4.14: A screen capture of the documentation provided in our LCD condition.

Figure 4.15: Activity completion times (secs) for AR and LCD. An asterisk marks the mean task completion for each condition. Red lines inside each box represent median values. The edges of each box represent the 25th and 75th percentiles, and whiskers extend to the most extreme data points not considered outliers. Points are drawn as outliers if they are larger than $q_3 + 1.5(q_3 - q_1)$ or smaller than $q_1 - 1.5(q_3 - q_1)$, where $q_1$ and $q_3$ are the 25th and 75th percentiles, respectively.

Figure 4.16: Mean alignment error (radians) for the alignment activity in AR and LCDs. An asterisk marks the mean alignment error for each condition. This is a single Tukey plot. The edges of each box represent the 25th and 75th percentiles, and whiskers extend to the most extreme data points not considered outliers. Points are drawn as outliers if they are larger than $q_3 + 1.5(q_3 - q_1)$ or smaller than $q_1 - 1.5(q_3 - q_1)$, where $q_1$ and $q_3$ are the 25th and 75th percentiles, respectively.

Figure 4.17: Survey response histograms by condition for ease of use (top), satisfaction (middle), and intuitiveness (bottom). Median values for each condition are displayed as triangles.

Figure 4.18: (Top) Components labeled for the PRINTED condition. (Bottom) Graphics displayed on the LCD screen.

Figure 4.19: Survey response histograms by condition (PRINTED vs. AR) for ease of use (top), satisfaction (middle), and intuitiveness (bottom). Median values for each condition are displayed as triangles.
Figure 5.1: Opportunistic Controls in action. A user manipulates a virtual button to record the results of an inspection task while receiving haptic feedback from the raised geometry of the underlying housing of the engine being inspected. (Screen capture of the imagery presented to the user wearing a video see-through head-worn display)........................................................................148

Figure 5.2: Objects that could support button OCs. A rubber doorstop (left); an extruded handle on a chair (center); perforated metal supporting server ventilation. .................................................................................................................156

Figure 5.3: Objects that could support valuator OCs. A mailbox hinge (left); a knot on a tree (center); a metal clad cable (right). .................................................................................................................157

Figure 5.4: Objects that could support movable OCs. Chalkboard hooks (left); a disconnected rubber hose (center); a coax antenna connector (right). .......................158

Figure 5.5: An example ATN depicting the 3D widget design for a two-button OC interface. Note, the button regions are scanned sequentially implying only one button may be active at any given time. ....................................................................159

Figure 5.6: Unsegmented (left) and segmented (right) bounding boxes for a set of OCs as seen from an overhead camera. Graphics are added in debugging interface. (The user does not see the camera’s view) ................................................161

Figure 5.7: OC aircraft engine maintenance domain (MA).............................................167

Figure 5.8: OC home entertainment domain (HE)...........................................................168

Figure 5.9: User observation tasks..................................................................................169

Figure 5.10: A subject demonstrates a potential OC used during a menu selection task in the HE domain...................................................................................171

Figure 5.11: Selected user concept sketches of proposed OCs.......................................172

Figure 5.12: An extract from an example report provided to each subject to aid in their completion of the post-experiment questionnaire. The report includes (top) an image of the original stimulus and (bottom) screenshots, one of which is shown here, captured using the hand-held AR display of the user gesturing during the experiment. ...............................................................................173

Figure 5.13: A user (wearing a stereo video see-through HWD) manipulates OCs with our prototype. The gesture recognition camera appears at the top of the photo. .............................................................................................................181

Figure 5.14: Our prototype OCs interface shown (top) without and (bottom) with overlaid graphics. (Images depict view through the video see-through display.).................................................................182

Figure 5.15: Valuator-based OC. (Image depicts view through the video see-through display.).................................................................................................................183

Figure 5.16: Rotating OC. (Image depicts view through the video see-through display.).................................................................................................................184
Figure 5.17: Baseline comparison technique (BL). (Image depicts view through the video see-through display.) .................................................................186

Figure 5.18: OC interface technique user study task sequence. (Images depict view through the video see-through display.) .................................................................188

Figure 5.19: Average completion times (seconds) for OC (left) and BL (right). The average completion time when using the OC technique was 86% that of BL, which was a significant speedup. .................................................................191

Figure 5.20: Average errors per technique for OC (left) and BL (right). The difference in means was not significant. .................................................................192

Figure 5.21: Survey responses for all participants. Responses are plotted on Likert scales (where 1 is most negative, 5 is most positive), with mean and standard error below the horizontal axes .................................................................194

Figure 6.1: (a) A user is given a procedural task involving the receptacle for a computer peripheral (located at the far right of each frame). (b) The user responds to orientation-centric attention guidance directing their attention to the target task. This results in an obstructed view of the target task that requires repositioning of the user’s head to view the task .................................................................208

Figure 6.2: (a) A user responds to position-centric wayfinding guidance presented in AR as a hypothetical path (W1–W4) leading to the receptacle for a peripheral device on a computer (base of CPU at far right of scene). (b) The user progressively positions their head at each node of the path. (c) At node W3, the user requires an abrupt and isolated orientation step to find the (d) final node in the path facing the receptacle. .................................................................209
List of Tables

Table 3.1: Selected tasks (with descriptions expurgated for publication) and corresponding pitch and azimuth measured from 0.7 meters above the center of the left turret seat ................................................................. 70

Table 3.2: Groupings of common task types ................................................................. 75

Table 3.3: Ranges (in degrees) in head rotation across all tasks. .................................. 82

Table 3.4: Rotational Exertions and Velocities. ............................................................ 86

Table 4.1: Guo and Tucker's [1996] taxonomy of generic tasks with scored values depicting our own estimation of the level of psychomotor activity required in each task. An asterisk (*) indicates text we reworded to generalize the original taxonomy to a broader domain of tasks ................................................................. 109

Table 4.2: Steps of combustion chamber assembly task ............................................. 131

Table 4.3: Pairwise comparisons of mean completion time by activity type. .............. 136

Table 5.1: Frequency of OC affordance types by task (MA Domain). Note, because many participants invoked multiple OC types to accomplish their assigned goals, the column values are not mutually exclusive ........................................... 175

Table 5.2: Frequency of OC affordance types by task (HE domain). Note, because many participants invoked multiple OC types to accomplish their assigned goals, the column values are not mutually exclusive ........................................... 175

Table 5.3: Preferred physical features (%) in the MA and HE domains. A filter was applied to each image in this figure (but not used in the experiment) to accentuate the affordance of interest ................................................................. 178
Acknowledgements

First and foremost, I thank my advisor, mentor, and friend Steve Feiner for his endless support and guidance. I am eternally grateful to him for having invited me to study in the Computer Graphics and User Interfaces (CGUI) Lab at Columbia University. This opportunity allowed me to fulfill professional goals in my career while the location supported the stability and well-being of my family. I am indebted to Steve for introducing me to his incomparable research standards and then mentoring and inspiring me to produce results at his level. Finally, Steve has provided me a shining example of leadership with his endless devotion and loyalty to his students, his tireless work-ethic, and his impeccable integrity. As I enter a new chapter of life that will place me in my own role as teacher and mentor, I will lean heavily on his example.

Next, I would like to thank my committee members. John Kender introduced me to the world of visual interfaces to computers and inspired many of my ideas in this dissertation. Peter Allen assisted with my LAV-25A1 work and was a constant source of support as I roamed the halls outside the CGUI lab. Barbara Tversky introduced me to cognitive psychology and her work and that of her students has been a strong source of inspiration in this dissertation. Mark Livingston provided invaluable feedback and mentorship and brought an important real-world perspective and motivation to my work.
I thank the members of the CGUI lab for their endless support and friendship: Hrvoje Benko, Eddie Ishak, Sean White, Ohan Oda, Lauren Wilcox, Christian Holz, Nick Dedual, and Mengu Sukan. I will greatly miss our many talks, extreme coding crunch-time demos, and midnight dinner runs. I am especially grateful for Ohan’s work on GoblinXNA, which I leveraged to implement two of my prototypes.

I’d also like to thank the leadership and faculty of the United States Military Academy’s Department of Systems Engineering at West Point, New York. First, I thank Mike McGinnis for selecting me to return to graduate school and supporting my journey off the beaten path to obtain a doctorate in Computer Science. I’m also thankful to my supervisors Tim Trainor and Rob Kewley for their endless support and patience as I balanced my military duties with completion of my thesis. I’d also like to thank the many other faculty members who were constant sources of support, provided technical advice, and helped carry my load: Tom Kastner, Fred Meyer, Mike Kwinn. Pat Driscoll, Greg Parnell, Dale Henderson, Kenny McDonald, Dan McCarthy, Paul Kucik, Russ Schott, John Farr, and Sam Huddleston. I am also very grateful to the U.S. Army’s Advanced Civil Schooling program, which funded my tuition and salary during the past five years.

I thank Bengt-Olaf Schneider, who provided the StereoBLT SDK used to support the display of stereo camera imagery, and Kyle Johnsen, who advised on use of the OptiTrack system. I am also grateful for the assistance of cadre and students at Aberdeen Proving Ground, as well as the support of engineers at the Marine Corps Logistics Base, including Mike Shellem, Curtis Williams, Andrew Mitchell, and Alan Butterworth. I’m especially grateful to these individuals for loaning me their LAV-25A1 turret mock-up. I also thank David Madigan and Magnus Axholt for insights shared during design and analysis of my experiments.
I am forever grateful to my wife Maria and our children Eva and Anna. They have unselfishly stood by during the past five years and provided endless love, care, and support to me as I selfishly pursued this endeavor. I could not have finished this journey without them. Most of all, I thank Jesus for providing the grace, mercy and forgiveness to make it through each day despite my many shortcomings.

This research was funded in part by ONR Grant N00014-04-1-0005, DAFAFRL Contract FA8650-05-2-6647, NSF Grant IIS-09-05569, and generous gifts from Vuzix, NVIDIA and Google.

Permission to use the LAV-25A1 photos shown in this dissertation was granted by the USMC LAV Program Manager.
To Maria, Eva, and Anna
1 Introduction

This thesis explores new ways of using computers to assist in documenting physical tasks. In this exploration, we design, develop, and evaluate new computer systems and user interface techniques that leverage augmented reality technology to present in-situ documentation that is combined with a person’s natural view of a task. As detailed in subsequent chapters, these systems and techniques can improve a person’s performance and accuracy when completing physical tasks, while also reducing physical workloads. Moreover, in some noteworthy cases, users prefer using these systems and techniques and find them more intuitive compared to state-of-the-art computer-based documentation.

Physical tasks involve people performing activities to accomplish particular goals while interacting with objects in the physical environment. Physical tasks vary in complexity, and could involve a single person performing activities sequentially or multiple individuals working in parallel as part of a larger integrated effort. We encounter these tasks in almost all aspects of life and expend considerable time and effort in their completion. In some cases (e.g., preparing a simple sandwich), we are able to complete tasks with little to no cognitive assistance. The sequence of activities might be simple enough as to demand only human intuition for successful completion. Or, the sequence of activities might be arbitrary in nature (e.g., assembling personal items in a sparsely filled suitcase), where any reasonable combination of steps will lead to an ac-
ceptable result. However, in other instances (e.g., assembling an airplane), we require explicit steps or instructions to successfully accomplish a physical task. In these procedural tasks [Gagné 1977], we must execute a particular ordered set of prescribed activities, or a procedure, to reach our goal.

As Ellis, Whitehill, and Irick [1996] explain, procedures vary in the level of required planning, number of steps, number of decision points, required cuing, flexibility of activity ordering, and type of goal. Procedures also vary in the particular techniques and technologies used in their instruction and use. Some approaches aim for mastery of the procedure beforehand, relying heavily on deliberate classroom instruction and controlled practice. Other approaches incorporate the use of certain aids, such as printed checklists or instructions, during actual execution of a procedural task. These aids serve to instruct and remind, and are particularly useful for procedures involving numerous, unfamiliar, or contingent steps.

Several technological developments stand to significantly reshape our design and use of procedures. The first is the advent of increasingly small and inexpensive computer technology that allows for creation of what Ockerman and Pritchett [2000] call task guidance systems. These systems, examples of which are shown in Figures 1.1 and 1.2, are defined as portable, mobile, or wearable computers that provide guidance to workers. These devices supplant the printed aids mentioned above with small, portable, devices that workers interact with while executing procedural tasks. Substantial memory capacities allow these systems to provide greater levels of detail and richer content. Graphical user interfaces featuring menus, hyperlinked media, and search technology allow workers to navigate larger information corpora, react to changing conditions, and record data. Portability and wearability allow workers to retain greater freedom of movement and hand use while performing procedural tasks. A particular class of task guidance systems
known as *Interactive Electronic Technical Manuals (IETMs)* [Connell 1978; U.S. Department of Defense 2007] allows workers to access maintenance and repair documentation in the field using portable notebook or tablet computers. These systems, examples of which are shown in Figure 1.2, combine the equivalent of dozens of voluminous and bulky paper manuals into a single, integrated tool that serves as both a guide and a reference. Although there is great room for improvement in the hardware and software aspects of IETMs, these systems are already widely used in commercial and military settings.

![Figure 1.1: Examples of task guidance systems. On the left, an investigator follows a deliberate procedure for documenting a crime scene [Cross, Baber, and Smith 2007]. On the right, an electrician uses a hand-held device to navigate a troubleshooting procedure for a telecommunication network (http://www.mobiledataforce.com).](image-url)
A second emerging technology, augmented reality, complements the capabilities and functionality of task guidance systems with an enhanced form of visualization. *Augmented reality* (AR) integrates virtual content with a user’s natural view of the environment, combining real and virtual objects interactively, at real-time frame rates, and geometrically aligning them with each other [Azuma et al. 2001]. AR can support procedural tasks by complementing essential physical world characteristics with virtual constructs that assist in understanding and following a procedure, and that are visually integrated with the user’s view of the task itself. AR thus preserves the natural context, realism, and multi-sensory interaction of a task, while adding virtual enablers such as overlaid instructions, feedback, and cuing, as well as representations of additional physical objects that might be hidden or missing. These capabilities are depicted in Figure 1.3, which shows a user (and matched first-person view) performing a procedural task while using a prototype AR system. This system was built using the Augmented Reality for Maintenance and Repair (ARMAR) architecture developed as part of this dissertation research.

Figure 1.2: Examples of IETM task guidance systems. (Left) A U.S. Navy electrician services an MH-60 helicopter with the aid of a notebook-based IETM. (Right) An example screenshot from an IETM interface (http://www.oneil.com).
1.1 Research questions and dissertation goals

Our thesis is that AR techniques can make it possible for individuals to perform procedural tasks more quickly while exerting less effort and making fewer errors. Of particular interest are interfaces supporting procedural tasks within the domain of equipment maintenance, which we categorize here using the U.S. Army definition of essential maintenance functions: inspection, testing, servicing, alignment, installation, removal, assembly, repair, overhaul, or rebuilding of human-made systems [U.S. Army 2007, Table 1-5]. These activities represent an interesting and opportunity-filled problem domain for the application of AR. The majority of activities in this domain are conducted by trained personnel applying established procedures to documented designs in relatively static and predictable environments. These procedures are typically organized into established sequences of quantifiable tasks targeting a particular item in a specific location. These characteristics and others form a well-defined design space for AR.

Figure 1.3: A user wearing a head-worn display (left) prepares for the next step in a procedural task using the ARMAR system described in this dissertation. The user’s view through the display (right) shows the natural review of the repair area, as well as virtual content pertaining to the next step in the procedure. (Image on right depicts the view through the head-worn display.)
Our designs and evaluations cover complete prototype systems supporting procedural
tasks, with a focus on interaction techniques and methods for managing a worker’s cognitive and
psychomotor performance during these tasks. These focus areas are summarized in the following
research questions:

1. What are the benefits of using an AR interface to support procedural tasks? We hy-
pothesize that AR interfaces supporting procedural tasks can reduce time, improve accuracy, and
reduce physical movement expended by a worker when locating, transitioning between, and exe-
cuting task steps. These benefits could result from the integration and alignment of virtual con-
tent with the worker’s natural view of the procedure’s environment. This would reduce context
switching and head/eye movement by allowing the worker to synthesize information and make
decisions within a single, spatially accurate mental model. We hypothesize that these benefits
will occur in both phases of the procedural task model proposed by Neumann and Majoros
[1998]. This model, which is corroborated by Richardson and colleagues [2004] in the case of
assembly tasks, decomposes a procedural task into separate informational and workpiece phases
of user activity.

Activities in the informational phase of a procedural task are primarily cognitive and
concerned with directing user attention (localization), comprehending instructions, and transpos-
ing information from the instructions to the actual task environment. In this phase, AR interfaces
could present screen-fixed, context aware text instructions to help a worker understand the goals
and actions associated with a task. Virtual cues, such as labels and arrows registered with the
procedural task environment, could reduce localization times by guiding the worker’s view to
physical locations associated with particular procedural steps. Once the worker is focused on a
particular task, registered labels and 3D models could provide additional informational assistance such as exploded component views, cut-away views, and depictions of desired task end states.

In the workpiece, or psychomotor, phase of a procedural task, the user performs kinaesthetic activities including comparing, aligning, and adjusting configurations of components. In this phase, AR interfaces could provide tracked virtual guides and 3D models to help the user visualize target alignments and positions. Virtual labels, arrows, and 3D models could provide updated instructions reflecting a user’s ongoing activity compared to the overall task goal. These dynamic instructions could suggest optimized movements, correct suboptimal activities, and provide feedback as alignments and adjustments are finalized. Which types of dynamic instructions are most useful and what are their design parameters?

2. How can we develop effective user interaction techniques in AR interfaces, while also minimizing interference with the task environment and the worker? Many procedural tasks pose two sets of competing constraints. The first set of constraints limits extraneous head, eye, and hand movements beyond the immediate vicinity of a task, and occurs when the worker cannot easily, or safely, reposition their head or hands. The second set of constraints relates to various factors that restrict modifications to the application’s environment. This second set of constraints precludes the worker from carrying, wearing, or installing, certain interface devices (e.g., portable devices or keypads) that might otherwise compensate for the limited head, eye, and hand movement posed by the first set of constraints. How can we develop interaction techniques for AR that satisfy both sets of constraints?

1.2 Contributions

This dissertation makes three contributions:
1. Design, implementation, and evaluation of an AR interface supporting informational phases of procedural tasks. We designed a prototype AR interface that uses a tracked head-worn display to augment a mechanic’s natural view with text, labels, arrows, and animated sequences designed to facilitate task comprehension, localization, and execution. A within-subject controlled user study examined professional mechanics using our system to complete a set of common tasks under field conditions. An AR condition was tested against two baseline conditions: the same head-worn display used in the AR condition providing untracked text and graphics and a fixed flat panel display representing an improved version of the laptop-based documentation currently employed in practice. The AR condition allowed mechanics to locate tasks significantly faster than when using either baseline. These improvements in mean localization time were significant. Additionally, in some instances, the AR prototype resulted in significant reductions in overall head movement compared to the flat panel display. A qualitative survey showed that mechanics found the AR condition intuitive and satisfying for the tested sequence of tasks.

2. Design, implementation, and evaluation of an AR interface designed to support psychomotor phases of procedural tasks. Building on our experience addressing task localization in AR, we developed and evaluated a prototype AR user interface designed to provide improved assistance to users during the relatively under-explored psychomotor phase of procedural tasks. Our prototype tracks both the user and the components in a typical maintenance assembly task, and provides dynamic, overlaid instructions on a see-through head-worn display in response to the user's ongoing activity. We conducted a within-subject user study comparing our prototype to a task guidance system presenting 3D-graphics–based assistance on a stationary liquid crystal display (LCD). The study compared the average completion times during psychomotor aspects.

* Significance in this dissertation is defined as differences in population means detected at an α=0.05 confidence level
of the assembly task under both the AR and LCD conditions. This comparison revealed the average completion time under the AR condition was 46% that of the average completion time under the LCD condition, which was a significant reduction. We also compared the accuracy of the assembly task, which revealed alignment error under the AR condition was 22% that of the LCD condition, which was a significant improvement. A smaller, pilot experiment found no significant differences in performance between our AR condition and an idealized, but often impractical, condition in which labels are physically embedded in the task domain. Qualitative results from the study indicated that participants overwhelmingly preferred the AR condition, and ranked it as more intuitive than the LCD condition.

3. Design, implementation, and evaluation of Opportunistic Controls, a novel class of interaction techniques. We developed a new class of AR user interaction techniques, known as Opportunistic Controls, that support gesturing on, and receiving feedback from, otherwise unused affordances already present in the domain environment. This eliminates the introduction of foreign interaction devices into the task domain or the need to use wearable or hand-held devices that might interfere with a person’s gaze or use of hands. We describe examples of Opportunistic Controls that we have designed and implemented, and present the results of two user studies. In the first study, participants proposed and demonstrated user interfaces incorporating Opportunistic Controls for two domains, allowing us to gain additional insights into how user interfaces featuring Opportunistic Controls might be designed. In the second study, participants performed a simulated maintenance inspection of an aircraft engine using a set of virtual buttons implemented both as Opportunistic Controls and an interface using simpler passive haptics. The completion time when using the Opportunistic Controls interface was 84% that when using the simpler form of haptics, which was a significant improvement.
We developed a hardware and software architecture known as Augmented Reality for Maintenance and Repair (ARMAR), which we adopted to build the various AR prototypes used in each of these three contributions. The architecture integrates multiple hardware and software systems into a highly configurable, scalable framework for constructing AR interfaces supporting procedural tasks. This includes provisions for integrating various display, tracking, and user input technologies into the AR interface. The architecture also provides design patterns and application program interfaces facilitatng rapid authoring AR-supported documentation. Our development of and experience with ARMAR yielded several practical rules of thumb for designing and implementing AR interfaces for procedural tasks.

Each of these contributions is further summarized in the following sections.

1.2.1 Design, implementation, and evaluation of an AR interface supporting informational phases of procedural tasks.

In Chapter 3, we present a prototype AR interface to support the informational phases of procedural tasks encountered in the maintenance and repair domain. This interface, which is depicted in Figures 1.3 and 1.4, features attention-directing graphics, 2D text instructions, and registered 3D models that are overlaid on a mechanic’s natural field of view to provide in situ task instructions. These instructions provide guidance for common maintenance and repair activities including the alignment, installation, removal, and inspection of various components. The prototype, which is implemented using the ARMAR architecture described in Appendix A, is highly configurable and adaptable to various maintenance and repair scenarios.
We evaluated our prototype in a field setting involving professional military mechanics performing maintenance and repair procedures in the turret of a United States Marine Corps LAV-25A1 armored personnel carrier [Henderson and Feiner 2009; 2011]. A within-subject controlled user study examined six mechanics using our system to complete 18 common tasks under field conditions. These tasks included installing and removing fasteners and indicator lights, and connecting cables, all within the cramped interior of an armored personnel carrier turret. Our prototype’s AR condition was tested against two baseline conditions: an untracked head-worn display with text and graphics and a fixed liquid crystal display (LCD) representing an improved version of the laptop-based documentation currently employed in practice.

A qualitative survey administered with the evaluation found strong user support for the AR condition. When we asked participants to rank the techniques as to how intuitive they were,
4 of the 6 participants ranked the AR condition first. A majority of participants also classified the AR condition as providing a high level of satisfaction.

The evaluation also supported several hypotheses, which represent notable findings of this dissertation:

H2: Mean localization time for AR would be less than that for HUD or LCD.
H4: Mean head rotation for AR would be less than that for HUD or LCD.
H5: Mean head translation for AR would be less than that for HUD or LCD.

A repeated measure analysis revealed display condition produced a significant main effect on the time required to locate each task \( (F_{(2,34)}=42.444, p < 0.001) \). A pairwise comparison of mean task localization time revealed that AR was 53% that of LCD, which was statistically significant \( (p = 0.007) \). This result supported hypothesis H2 and revealed that AR interfaces can provide value by saving the mechanic time in locating procedural tasks. This is a significant advantage afforded by the use of AR when one considers the total time a mechanic might spend localizing during a protracted sequence of procedural tasks.

A similar repeated measure analysis of head movement also revealed a significant main effect attributable to display condition. A pairwise comparison of mean rotational distance found mechanics accumulated less head and neck exertion in the pitch, yaw, and roll axes when experiencing the AR condition than when experiencing the LCD condition, all of which were statistically significant\(^\dagger\) \( (p < 0.05) \). These results supported hypothesis H4 and reveal a significant advantage afforded by the use of AR interfaces in reducing head and neck rotational movement during procedural tasks. A similar analysis of translation head movement revealed that translation

\(^\dagger\) In this dissertation, a Bonferroni correction is applied when claiming significance involving the testing of multiple hypotheses.
tional head exertion when using the AR condition was 37% that of the LCD condition. This result supports hypothesis H5. The ability of the AR interface to reduce both rotational and translation head movement is notable as decreasing such movement might reduce user fatigue and job-related injuries.

The evaluation also revealed several unsupported hypotheses:

H1: Mean completion time for AR would be less than that for HUD or LCD.

H3: Mean error rate for AR would be less than that for HUD or LCD.

H6: Mean head velocity for AR would be less than that for HUD or LCD.

A repeated measure analysis did reveal that display condition produced a significant main effect on the overall time required to complete each task ($F_{(2,34)}=5.252, p = 0.028$). However, a pairwise comparison found no evidence to suggest the mean completion time under the AR condition differed from that of the LCD condition ($p = 0.51$). This result failed to support hypothesis H1. A review of our approach revealed opportunities to improve the assistance offered by our prototype in the psychomotor phase of each task.

Participants in our study made very few errors under any of the display conditions ($F_{(2,34)}=1.00, p=0.410$). Consequently, an analysis of errors failed to detect any effects attributable to display condition. This result failed to support our hypothesis H3.

A repeated measure analysis revealed that display condition produced a significant main effects on mean translational head velocity ($F_{(2,34)}=5.252, p = 0.028$), and mean rotational velocity for the pitch ($F_{(2,34)}=12.205, p = 0.002$), yaw ($F_{(2,34)}=44.191, p < 0.001$) and roll ($F_{(2,34)}=48.875, p < 0.001$) axes. However, a pair-wise comparison of mean translational and rotational velocity failed to reveal any instance when AR was faster than LCD. These results failed to support hypothesis H6.
We also examined task focus, which was measured by estimating the average Distance from Center Point (DFCP) [Axholt, Peterson, and Ellis 2008]. In the case of our evaluation, this metric represents the accumulated distance a mechanic’s view direction deviates from an optimal view direction to the current task. A repeated measures analysis found display condition exhibited a significant main effect on mean DFCP ($F_{(2,34)}=1043.6$, $p < 0.001$). Post-hoc comparisons found the mean DFCP under the AR condition was 37% that of the LCD condition, which was significant ($p < 0.001$). This result reveals an advantage afforded by the use of AR interfaces in allowing mechanics to remain more focused on the task at hand. This increase in task focus can reduce cognitive load and the need to memorize instructions.

1.2.2 Design, implementation, and evaluation of an AR interface designed to support psychomotor phases of procedural tasks.

Encouraged by the results of the LAV-25A1 study, we set out to explore how the benefits afforded by AR in the informational phase of a procedural task could be extended into the more complex psychomotor phase. We designed and evaluated a prototype AR interface designed to provide specific assistance in this underexplored phase, in which the user performs physical manipulations, and thus alters aspects of the underlying task environment. Assistance offered during the psychomotor phase must continuously evaluate the evolving spatial arrangement of the user and objects in the task domain to dynamically update instructions. We applied our prototype to several realistic procedural tasks, depicted in Figure 1.5, that include significant psychomotor activities. In these tasks, our prototype tracks both the user and the task objects and provides dynamic, overlaid instructions presented on a see-through head-worn display in response to the me-
chanic’s ongoing activity. These instructions help with performance of psychomotor activities of aligning, rotating, comparing, and adjusting.

Figure 1.5: Examples of AR assistance offered using an ARMAR prototype during psychomotor phases of procedural tasks. (Left) During assembly of a combustion chamber, a user wearing a head-worn display is presented with information about the alignment of assembled subcomponents. (Right) Following the assembly, additional assistance is presented to the user to help with rotating and aligning the combustion chamber prior to installation. (Images captured by a video camera mounted inside and looking through an optical see-through display. A post-render filter was applied to remove camera distortion and vignetting.)

We selected one of these tasks, the assembly of combustion chambers from a Rolls Royce Dart 510 turboprop aircraft engine (depicted in Figure 1.5, left), for formal evaluation of our prototype’s effectiveness at providing assistance during psychomotor activities. Assembling an individual combustion chamber requires precise alignment of two components—a truncated conical upper section (which we will refer to as a “cone”) and a mostly cylindrical lower section (referred to as a “can”). During this alignment step, which we identify as the main psychomotor activity within the larger assembly task, our prototype presents several forms of assistance. A dynamic 3D arrow indicates the optimal direction of rotation to bring the cone and can into the
desired alignment. The size and color of the arrow are varied in response to user activity to reflect the magnitude of the motion required to achieve the desired alignment. Virtual labels are rendered at locations registered with the connection points (i.e., holes) on the can and cone where the worker will insert fasteners to secure the assembly after finalizing the alignment. Our prototype also presents dynamic highlights that help identify these connection points and provide feedback when they are aligned.

A within-subject controlled user study examined 22 users, all of whom were recruited from our university’s student population, assembling 14 combinations of combustion chamber cans and cones. Each participant experienced both our AR prototype and an LCD-based 3D electronic documentation system, depicted in Figure 1.6, which we developed as a control condition. This control condition is similar to the one used in our earlier LAV-25A1 armored personnel carrier field study (described in Section 1.2.1). We formulated the following hypotheses regarding our prototype’s effectiveness in assisting with the combustion chamber assembly task:

H1: AR would be the fastest technique during psychomotor activities.
H2: AR would be the most accurate technique during psychomotor activities.
H3: AR would be the most preferred technique.
H4: Participants would rank the AR technique as most intuitive.
A repeated measure analysis revealed display condition provided a significant main effect ($F_{(1,21)} = 37.09, p < 0.001$) on the mean time to complete the psychomotor phase (i.e., the alignment of cone and can) of the combustion chamber assembly task. A pairwise comparison of means showed that participants assisted by our prototype were 46% faster than when assisted by the LCD-based 3D documentation, which was a significant difference ($p < 0.001$). This result supported hypothesis H1. We conducted a similar analysis of mean alignment error, which we defined as the angular difference between the optimal orientation of can and cone and that achieved by the user at the completion of the task. This analysis revealed a significant main effect of display condition on mean alignment error ($F_{(1,21)} = 48.75, p < 0.001$). The mean difference between the optimal orientation and that achieved by the user was 0.08 radians for AR (0.25 inter-hole widths) and 0.36 radians (1.15 inter-hole widths) for the LCD condition. A comparison of means revealed that the AR condition was 0.265 radians more accurate than the LCD condi-
tion, which was significant ($p < 0.001$). We also tested the success rate in achieving task alignment, where success is recorded as a binary variable indicating the user successfully achieved the target alignment for each pair of cans and cones at the end of the task. The AR condition allowed participants to achieve a mean success rate of 95.3% compared to 61.7% when experiencing the LCD condition. A McNemar’s test revealed this was a significant difference ($\chi^2_{(22,1)} = 266.76$, $p < 0.001$). These results supported hypothesis H2.

Qualitative results showed participants overwhelmingly preferred our prototype, with 20 of the 22 users ranking AR as the most preferred technique, which a Friedman’s test revealed as a significant ranking ($\chi^2_{(22,1)} = 11.64$, $p < 0.001$). When we asked the participants to rank the techniques according to their intuitiveness, 19 of 22 users ranked our AR prototype as the most intuitive, which was also a significant ranking ($\chi^2_{(22,1)} = 14.73$, $p < 0.001$). These results support hypotheses H3 and H4.

Following up on this experiment, we were interested in how AR would compare to documentation that was physically embedded in the task; for example, by physically labeling or otherwise modifying all components to clearly disambiguate them from each other and clearly distinguished the different ways that the components might be configured. We had considered implementing this idealized condition as our original study baseline, but felt it was not ecologically valid for the engine combustion chambers, which are not actually labeled this way, and many similar domains. However, we decided to perform a pilot study comparing AR to such a baseline in order to situate our results relative to task domains in which physical labeling would be possible. We created a modified version of the LCD condition in which we printed and glued small physical labels to all possible connection points on each can and cone, as depicted in Figure 1.7.
We also added virtual versions of these printed labels to the virtual models displayed on the LCD.

Six additional participants experienced the same experiment design as our larger study population with the printed label condition substituted for the LCD condition. A repeated measure analysis failed to detect significant main effects on task completion time or accuracy as a result of display condition ($F(1,5) = 0.67, p < 0.451$). Therefore, we found no statistical support to suggest that the performance of the AR prototype differs from this idealized baseline.

Figure 1.7: The idealized documentation used in our follow-up experiment. The inset view depicts the virtual version displayed on the LCD screen.

1.2.3 Design, implementation, and evaluation of Opportunistic Controls, a novel class of interaction techniques.

In our exploration of AR interfaces supporting procedural tasks, we identified the need for efficient and intuitive interaction techniques. These techniques must not detract from a mechanic’s focus on the physical task, which may require near exclusive use of their hands and
eyes. The techniques must also satisfy constraints in the procedural task domain that preclude the use of traditional input devices. In an effort to provide an interaction technique that satisfied these requirements, we designed and tested Opportunistic Controls [Henderson and Feiner 2010; 2008], a novel class of interaction techniques supporting AR interfaces for procedural tasks. Opportunistic Controls (OCs) combine the passive haptics from unused affordances in the task environment with overlaid 3D virtual widgets to create a natural tangible user interface [Ishii and Ullmer 1997]. *Tangible user interfaces* are ones in which users employ objects from the physical environment to manipulate digital information. These objects are typically pre-selected and deliberately integrated into the user interface design.

In the case of OCs, these physical objects are harvested from objects that are already part of the environment, that are not normally used as interface constructs, and which afford certain tactile and visual characteristics that support an interface task. For example, a mechanic inspecting an airplane wheel might navigate a checklist using an AR interface that repurposes the wheel’s hardware as primary interface widgets. More specifically, when the mechanic glances at a hex nut, 3D graphics presented in AR and aligned with the hex nut might depict a physical button for cuing the next step in the inspection. When the mechanic touches this virtual button, the protruding physical geometry of the nut provides the feel of a real button.
We designed a user observation study examining how users might perceive naturally occurring affordances as components of OCs [Henderson and Feiner 2010]. We recruited fifteen participants from our university’s student population and presented each with hypothetical user interface scenarios. Each scenario required the user to design their own versions of notional OCs to complete common 2D and 3D interface tasks (e.g., menu selection and manipulation of virtual

Figure 1.8: Opportunistic Controls in action. A user (top) wearing a head-worn display uses (bottom) virtual buttons to record the results of an inspection task while receiving haptic feedback from the raised geometry of the underlying engine housing. (Image at bottom is an exact screen capture of the view presented to the user wearing a video see-through head-worn display.)
3D objects). The scenarios were presented to the user on an untracked, hand-held, magic lens display [Billinghurst, Kato, and Poupyrev 2001], as shown in Figure 1.9. As participants responded to each scenario, we asked them to devise an OC interface to match the scenario and to demonstrate this OC’s associated gestures and matching affordances in the physical world. As the user gestured, we provided “Wizard of Oz” feedback to simulate tracked virtual content and system responses. Figure 1.9 depicts a user demonstrating one such hypothetical OC interface for a menu selection task.

![Figure 1.9: (Left) A user holding a hand-held video see-through display selects affordances for a notional OC used to control a hypothetical menu selection task in a home entertainment user interface domain. (Right) A close-up screenshot of the view presented to the user via the hand-held video see-through display. “Wizard of Oz” feedback is used to change the menu’s selected item in response to the user’s gesture.](image)

The results of the study yielded several qualitative findings. First, participants selected a plurality of valuator-based affordances—static linear or curved surfaces that support sliding hand or finger gestures similar to those employed on a track pad. Participant preferences for these affordances extended to interface tasks normally associated with buttons (e.g., discrete menu selections). A second finding was that users often selected multiple affordances from the physical en-
vironment while constructing notional OCs. A third finding indicated a user’s perception of individual affordances within a domain (e.g., an inability to envision screws on the back of a television as buttons) might be influenced by their perceptions of surrounding context (e.g., a person’s reluctance to touch the wiring configuration of a television). Finally, a fourth finding was that participants preferred surfaces located at eye-level and within arm’s reach, suggesting affordances selected for OC designs should minimize a user’s physical exertion.

We created a set of OCs that use an appearance-based vision tracking technique [Kjeldsen and Kender 1996] to recognize manual interaction with virtual buttons, sliders, and knobs. We evaluated a subset of these OCs with a within-subject controlled user study [Henderson and Feiner 2008]. This study evaluated fifteen participants using our prototype OC interface to conduct an inspection procedure involving a Rolls Royce Dart 510 turboprop aircraft engine. This OC prototype interface, depicted in Figure 1.10 (left) featured five virtual 3D buttons registered with raised geometry located on the engine’s compression section.

Figure 1.10: (Left) Button-based OC interface evaluated in our performance and acceptance user study. (Right) The undifferentiated version of the interface used as the study’s baseline. (Images are exact screen captures of the imagery presented to the user wearing a video see-through head-worn display.)
During the study, participants wearing a head-worn, video see-through display first located a 3D placard labeled with text (Figure 1.11, left), presented at one of several registered stations on the engine. The participant then used the 3D buttons of the OC interface to select matching text from among entries presented in a screen-fixed 2D menu (Figure 1.11, right). The completion time and accuracy of users performing this selection task using the OC interface was compared to their performance using an undifferentiated baseline condition, depicted in Figure 1.10 (right). This undifferentiated baseline consisted of the same virtual 3D buttons used in the OC interface, but registered on a smooth, flat plastic surface. We hypothesized participants using the OC interface would (H1) perform the selection task more quickly while (H2) making fewer errors than when using the undifferentiated baseline.

Analysis of participant performance under both conditions revealed interface technique had a significant main effect on mean selection times \( F_{(1,28)} = 8.11, p < 0.001 \). On average, users were 16% faster when using our OC prototype than when using the undifferentiated baseline,
which was statistically significant \((t_{14} = 4.98, p < 0.001)\). This result confirmed hypothesis H1. A similar analysis found no evidence of technique influencing error rates \((F_{(1,28)} = 1.94, p = 0.185)\) and thus failed to confirm hypothesis H2. A post-experiment questionnaire revealed strong user support for the OC technique, with 11 of the 15 participants selecting it as the most preferred technique, which was a significant ranking \((p=0.02)\).

1.3 Structure of Dissertation

The remainder of this dissertation provides an overview of previous work, a detailed description of our contributions, and states our conclusions and suggestions for future research. Chapter 2 reviews research related to procedural tasks in general, applicable work from the field of instructional design, as well as work examining various aspects of AR as applied to procedural task problems. Chapter 3 details the design and implementation of a prototype AR interface supporting professional mechanics conducting routine maintenance inside an armored personnel carrier. Discussion of this prototype, which focuses on AR assistance for the informational phase of procedural tasks, includes the results of a formal user study evaluating professional mechanics using our prototype in a field setting. Chapter 4 details the implementation of a second prototype, also implemented using our ARMAR architecture, which provides assistance during a realistic assembly task implemented in a laboratory setting. Discussion of this prototype, which focuses on AR assistance for the psychomotor phase of procedural tasks, also includes the results of a formal user study. Chapter 5 introduces Opportunistic Controls (OCs), describes our implementation of several prototype OC interfaces, and presents the results from two user studies. The first study explores design considerations and user preferences for the use of naturally occurring affordances in OC interfaces. The second study examines the speed and accuracy of an OC interface implementation compared to a baseline condition. In Chapter 6, we state our conclusions
and discuss opportunities for future research. In Appendix A, we provide the details of our AR-MAR architecture, which was used to construct the hardware and software prototypes described in Chapter 3–5.
2 Related Work

There exists a sizable body of work examining how people understand, learn, and perform procedural tasks. This work spans the fields of education, psychology, graphic design, linguistics, computer science and many others. In this chapter, we summarize a portion of these efforts that intersect with the application of AR to procedural tasks.

We first review research into the nature of procedural tasks and what makes them different from other activities (Section 2.1). We then examine pedagogical efforts surrounding procedural tasks (Section 2.2). Next, we explore generalizable design principles for procedural tasks documentation that are not unique to any particular technology (Section 2.3). We then survey notable instances of these principles, first examining non-automated forms of assistance (Section 2.4) and then reviewing efforts to use computers to produce better documentation or assist in its delivery (Section 2.5). Finally, we review a core set of AR works and concepts that span this entire dissertation, and defer complete coverage of applications of AR to subsequent chapters where their review will prove more useful (Section 2.6).

2.1 The Nature of Procedural Tasks

Gagné was among the first to invoke the term procedural task [1977], which he defined as a task requiring the integration of intellectual and motor skills. The emphasis on skills required to perform a task, versus characteristics about the task itself, is key to what distinguishes
procedural tasks from other undertakings. For example, the task of filing an annual tax return is arguably procedural and can require significant intellectual activity. However, for most people, it does not require a sufficient level of motor skill to meet the definition of a procedural task. Similarly, we invoke a procedure when we tie our shoes, but this procedure’s cognitive component is not sufficiently cerebral to categorize shoe tying as a procedural task.

Several works have captured and categorized examples of procedural tasks. Some of these have adopted a task-focused approach to identifying the low-level intellectual activity and motor-skills common to procedural tasks in certain domains. This includes early work by Gilbreth and Gilbreth [1924] who conducted studies examining low-level motions common to various forms of manual labor to optimize the performance of that work. These studies inspired the Gilbreths to propose a taxonomy of major activity types, termed *therbligs*, that collectively define the motions required to complete common tasks. Chenzoff and Loose [1992] provided a list of twenty-three task primitives encountered in aviation maintenance. Drury and colleagues provide a description of the procedural tasks involved in technical inspection of commercial airliners [1990]. Vujosevic and Ianni [1997] provided several motion models encountered in the aviation maintenance domain, similar in spirit to *therbligs*. Guo and Tucker examined procedural tasks performed in the construction domain, and identified a list of 42 common activities.

Other researchers followed a worker-centric approach to categorizing procedural tasks, focusing on human abilities required in their completion. This includes celebrated work by Fleishman, Quaintance, and Broedling [1984] who outlined a comprehensive taxonomy of human abilities matched to work requirements. Bloomfield and colleagues [2003] provided a recent taxonomy that matches procedural tasks in the maintenance and repair domain with required hand and arm motion.
2.2 Teaching, Learning and Assistance for Procedural Tasks

There is much overlap between efforts to teach and efforts to assist with procedural tasks. Teaching is a protracted endeavor concerned with permanent mastery of a task. Assisting is a shorter term endeavor concerned with immediate and successful accomplishment of a task. However, in many instances, these two activities overlap and the same methods used to train a person off the job are often used to assist that person on the job [Chalupsky and Kopf 1967]. Although this dissertation is primarily focused on the use of AR to provide immediate forms of assistance for procedural tasks, we report on notable training-focused work due to its topical intersection with assistance.

Research into the pedagogy of procedural tasks experienced a surge of growth in the late 1960s, reflected in the formation of a new field called instructional psychology [Gagné and Rohwer 1969]. This field enjoyed broad participation, resulting in several important models and insights on how best to teach procedural tasks. This includes work by Fitts and Posner [1967], who suggested that teaching and learning procedures involved three parts: a cognitive phase, an associative phase, and an autonomous phase. Gagné and Rohwer [1969] identified the importance of selecting the appropriate instructional methodology to match a particular set of learning outcomes. Bloom [1976] emphasized the roles of practice and feedback in procedural task instruction.

There are also several important well-cited works focusing on learning and retention of procedural task knowledge. Evidence offered by Vineberg [1975] and confirmed by Schendel [1978] suggested people retain very little classroom task instruction by the time they execute procedures in the field. Wetzel, Konoske, and Monteague [1983] later illuminated many of the factors affecting this lack of retention, such as task complexity, length of non-utilization periods,
and the amount of background knowledge required to perform a task. A study of U.S. Navy recruits by Tannenbaum and colleagues [1993] reported additional factors impacting the retention of procedural task training such as a trainee’s self-confidence and expectations about the training.

### 2.3 Designing Assistance for Procedural Tasks

One of the most important questions associated with the assistance of procedural tasks is: what are the ingredients of high quality documentation? In an effort to answer this question, researchers have examined both structure and content. This has produced several heuristics and principles of interest that apply at various levels of instructional design and which are not necessarily dependent on any one technology.

Moore and Fitz [1993] applied Gestalt theory to instructional design and advocate six guiding principles including proximity, figure-ground segregation, and symmetry. Several efforts have invoked user-centered approaches in an attempt to illuminate desirable characteristics of documentation. This includes work by van der Meij and Carroll [1995], who proposed a set of four design principles emphasizing action, realism, error support, and minimal design. Each of these design principles is supported by its own unique set of heuristics. Efforts by Heiser and colleagues [2004] produced a set of *cognitive design principles* for assembly task instructions. This work featured a comprehensive series of experiments in which the researchers observed, prototyped, and evaluated user-designed documentation associated with a consumer television stand. The results of the study led to the identification of eight cognitive design principles. These principles touch upon the sequence and content of instructions, and were later converted into a computer algorithm capable for generating instructions, which we summarize in Section 2.5.1.

Ganier [2007] reported that a set of user-designed documentation for operating a kitchen pres-
sure cooker outperformed factory-supplied documentation in speed and accuracy. This study also produced a set of design heuristics, several of which corroborate those proposed by Heiser and colleagues.

Several efforts have focused on optimizing specific types of content featured in procedural task documentation. Efforts to improve the use of text include an extensive set of heuristics and suggestions proposed by Wright [1977; 1981]. These govern the structure, typesetting, and writing style of technical prose and instructions. Smith and Goodman [1984] identified the benefits of organizing the subordinate steps of a larger procedure into a hierarchy of functions. This idea gathers a contiguous group of steps under a function that describes the group’s purpose within a larger procedure, which was shown to be a superior structure to traditional linearized instructions. Chervak and colleagues [1996] reported comprehension of instructions improved for a group of maintenance technicians when the instruction used Simplified English, a restricted language specification developed for technical documentation [AECMA 1995].

Several additional works have focused on the integration of text and graphics. Booher [1975] determined that pictures helped subjects complete tasks more quickly, but text was important for ensuring accuracy. He also reported on the merits of combining text with graphics. Ellis, Whitehill, and Irick [1996] extended these findings by noting pictures facilitated learning the task, but did not contribute to performance once the task had been mastered. Young and Wogalter [1990] conducted research on textual warnings and discovered that participants comprehended and recalled warnings better when text was printed conspicuously (e.g., highlighted, with a larger font) and accompanied by icons.
2.4 Non-automated Forms of Assistance

We continue our review of related works by examining several prevalent non-automated* forms of procedural tasks assistance. Chief among these are portable printed materials, which include workcards, checklists, and manuals. Workcards, an example of which is shown in Figure 2.1, are portable paper documents that depict the steps of a procedural task. These documents, which are sometimes referred to as “job cards”, assist in remembering the steps in a procedure while also providing a formal record of work completed that is collected and archived at the end of the workday. Patel, Drury, and Lofgen [1994] provide an excellent analysis of workcards that includes a proposed set of 49 design guidelines.

Checklists are another category of printed material often employed to assist with procedural tasks. Like workcards, they are designed to be portable and accompany the worker during the procedural task. They differ from workcards in that they are designed for reuse and are not collected as a formal record of tasks performed. Degani and Wiener [1993] provide an excellent review of cockpit checklists, discussing many of their characteristics and proposing a set of design guidelines that are applicable to a broad range of checklists used in other domains.

---

* In this dissertation, we define non-automated forms of documentation as systems that do not require computers for day-to-day use, though computers may be involved in their construction.
Figure 2.1: A completed workcard [National Transportation Safety Board 1984].
Workcards and checklists are often extracts from larger books known as manuals. Depending on their intended audience, manuals might also be referred to as technical or user manuals. Manuals are normally rich in content and detail and often contain dozens of procedural tasks in one binding. It is not uncommon to find manuals having over 1000 pages of diagrams, schematics, theories of operation, and checklists for procedural tasks. In many instances, local regulations and best practices mandate that these manuals are placed nearby mechanics performing procedural tasks.

Consumer products feature several examples of well-known, non-automated forms of assistance, and Smith and colleagues [2003] offer an excellent guide to their development. Examples include owner’s manuals, assembly instructions, and quick-reference guides. This assistance often takes the form of single page instruction sets, an example of which is shown in Figure 2.2, where the entire set of instructions is printed on one or both sides of a single piece of paper. This type of documentation is designed for one-time reference before a product is used. Consumer-level manuals include information intended to be referenced beyond the initial use of a product (though they often also contain initial-use instructions). Consumer-level manuals are designed as references and normally accompany systems requiring a certain level of recurring maintenance (e.g., cars, lawn-mowers, and garage-door openers).
Figure 2.2: An example of non-automated, single-sheet, consumer assembly instructions (http://www.alvinit.com).

Another widely employed form of assistance is exemplified by printed posters, charts, and stickers that are displayed, usually permanently, in prominent locations where procedural tasks or performed. These might be used to present documentation to several people performing tasks in a common area. An example lubrication chart, which was mounted on the wall of a garage to assist garage-station attendants, is depicted in Figure 2.3. A variation of this type of doc-
ocumentation involves printing instructions onto stickers or other surfaces that are permanently attached to an object involved in a procedural task (e.g., printing troubleshooting instruction on the inside door of a copy machine). This form of instruction is often represented as a series of wordless, pictorial diagrams for which Rodriguez and Polson [2004] provide a set of design rules.

Figure 2.3: An example job chart used to assist gas-station attendants (http://justacarguy.blogspot.com).
2.5 Computerized Forms of Assistance

2.5.1 Computer-generated documentation

People have leveraged computers significantly to assist with creation and delivery of instruction for procedural tasks. Notable research into the creation of procedural task instructions through the use of computer graphics includes work by Kahn [1979] who examined using computers to automate creation of animated sequences and diagrams. Neiman [1982] developed a system that generated animated instructions for a CAD system. Feiner created the APEX system [1985], which automatically creates sequences of 3D pictorial explanations showing a particular set of actions performed on objects. Seligmann and Feiner [1991] designed an Intent-Based Illustration System (IBIS) that generates illustrations automatically. Feiner and Seligmann later added an algorithm for automatically generating cutaway and ghosting effects supporting procedural tasks [1992]. Feiner and McKeown [1991] leveraged IBIS as part of a Coordinated Multimedia Explanation Testbed (COMET), a system for automatically generating maintenance and repair instructions that coordinated text and graphics, examples of which are shown in Figure 2.4. Driskill and Cohen [1995] proposed a comprehensive system for authoring assembly instructions. This system separated an assembly’s procedure from its component geometry which allowed designers to iterate and test proposed forms of documentation.
Agrawala, Heiser, and colleagues [2001; 2004], developed a system, based on a set of cognitive design principles, that automatically generated procedural steps and illustrations. Li and colleagues built on this work to automatically generate interactive exploded [2004; 2008] and cutaway [2007b] views.

Others have focused on specific aspects of computer-generated content. Tversky and Morrison [2002] reviewed the efficacy of animated graphics in promoting task cognition, including the comprehension of procedural tasks. They concluded that the use of animation should be guided by two overriding principles—the principle of apprehension (animations should be easy to perceive and comprehend) and the principle of congruence (ensuring an animation’s external representation matches a user’s internal mental model). Mayer [2002] proposed nine principles of multimedia learning, many of which can be extended to the use of computers to generate documentation. Ward and Novick [2003] reviewed the use of speech as an input modality in hands-free documentation systems and proposed a list of important design considerations governing its
use. These include the accounting for effects of noise, the importance of feedback, and the tradeoff between number of users and breadth of vocabulary.

2.5.2 Computer-delivered assistance

There has been much interest in using computers to deliver assistance. Booher [1975] implemented a programmable task simulator for exploring information presentation techniques. Although this system was not designed to provide online assistance, it represents an important forerunner of later work. Another early concept that remains the industry standard for computerized assistance today is the Interactive Electronic Technical Manual (IETM). This system, example of which are shown in Figures 1.2 and 2.5, has also been referred to as a Computer-based Maintenance Aid System (CMAS) and a Technical Information Presentation System (TIPS). Originally envisioned in the late 1970s [Connell 1978], an IETM is a technical manual authored specifically for deployment and use on a computer. These aids are designed to leverage a computer’s capacity to store information, as well as its interactive capabilities such as zooming, scrolling, and searching [Rainey 1991]. ITEM capabilities have ranged from simple hyper-linked text documents [Konstantinou and Morse 1992] to systems capable of creating interactive documentation entirely from a database [Boose, Shema, and Baum 2003]. A comprehensive survey by Siegel and colleagues [2002] traces the development of IETMs and captures many of the research directions and salient issues surrounding their design and use. Today, IETMs are typically deployed on laptop computers (such as the one shown in Figures 1.2 and 2.5) and carried to the job site by workers.
A similar form of computer-delivered assistance is reflected in embedded training systems. *Embedded training* refers to instructions that are tightly integrated or packaged with a corresponding system. Examples range from instructions printed directly on a component to systems that merge training stimuli with production technology, such as using a targeting computer LCD to display simulated targets [Zachary et al. 1999]. Kearsley and Seidel [1985] provide a survey of the early use of computers for training that includes a discussion of embedded training. A follow-up survey by Jorgensen [1991] reviewed efforts to quantify the efficacy of embedded training. He concluded that more work was required to answer this question, and proposed his own conceptual procedure for measuring the effectiveness of embedded training. Recent examples of embedded training include a system proposed by Montoya and colleagues [2007] that repurposes the optics used in combat vehicles to display training stimuli.

Figure 2.5: A mechanic references an IETM while performing troubleshooting inside a UH60 helicopter (U.S. Army).
As previewed in our introduction to this dissertation (Section 1.1), task guidance systems [Ockerman and Pritchett 1998] represent an emerging form of computer-delivered assistance for procedural tasks. These systems seek to leverage light-weight, mobile computers to provide the robust assistance of an IETM in a more portable form-factor. Many of the earliest versions of these systems leverage wearable computer technology. This includes a prototype demonstrated by Kraut, Miller, and Siegel [1996] supporting collaborative maintenance tasks on a bicycle. An evaluation showed that participants wearing the system made repairs more rapidly and accurately when assisted by a remote expert. Siegel and colleagues [1997; 2001] extended this work to create wearable IETMs for mechanics working in the aviation maintenance domain, and found strong user support for the system. Other examples of wearable computer systems providing assistance for procedural task include prototypes by Ockerman and Pritchett [1998], Siewiorek and colleagues [1998], Fallman [2002], and Goose and colleagues [2003].
2.6 AR Interfaces

AR represents a technology that holds great potential to assist with procedural tasks, a hypothesis we explore in detail in this dissertation. In the interest of continuity, we have distributed our review of previous AR works focusing on procedural tasks among the remaining chap-
ters, including our review of prior AR interfaces and interaction techniques. In the rest of this chapter, we introduce several fundamental concepts that cut across multiple chapters and serve as a preamble for our own work and contributions.

First among these is a technical definition of AR, for which we will adopt that of Azuma and colleagues [2001]: “An AR system supplements the real world with virtual (computer-generated) objects that appear to coexist in the same space as the real world.” This well-cited definition goes on to identify three prerequisites for an AR system:

1. The system must combine virtual and real objects in a real environment
2. The system must interact with a user in real-time
3. The system must align (register) real and virtual objects with each other

We will apply these three conditions to determine what we consider, and do not consider, AR in follow-on chapters.

Second, we discuss several important works that touch on the cognitive underpinnings of how people use and perceive AR during procedural tasks. In Chapter 1, we described a procedural task model proposed by Neumann and Majoros [1998] that suggests procedural tasks contain two phases of activity—a mostly cognitive portion, known as the informational phase, where a worked is focused understanding and orienting on task and a psychomotor portion, known as the workpiece phase, where the worker is performing mostly kinesthetic activities. This bipartite view of procedural tasks, which is corroborated by findings from Richardson and colleagues [2004] in the case of assembly tasks, reflects our own way of thinking about how and when AR might assist with procedural tasks.

We believe this practical partitioning of procedural tasks into two distinct phases mirrors the partitioning of human spatial cognition into egocentric and allocentric reference systems
[Paillard 1991]. Klatzky [1998] provides a formal definition of these reference systems, and offers robust models for each that include distinguishing primitives and parameters. In general, an egocentric reference frame is dependent on an observer’s current perspective of the surrounding world. The observer is at the center of the frame and all objects are perceived relative to this center. In an allocentric reference frame, which is often referred to as an exocentric reference frame, all objects are perceived relative to an external reference frame that is independent of the observer’s given perspective.

Much work has been conducted examining the various cognitive activities invoked and abilities required when working and conceptualizing within these distinct reference frames. Several notable examples include work by Finke and Shepard [1986], Hinton and Parson [1988], Tversky [1991], Carlson-Radvansky and Irwin [1993], and St. John and colleagues [2001]. Related work by Kozhevnikov and colleagues [2001; 2005] suggests a person’s spatial visualization ability is separable from their object visualization ability and that spatial visualization abilities have an egocentric and allocentric dimensions. Moreover, these dimensions affect how people perform tasks in the real world [Kozhevnikov et al. 2006]. Egocentric spatial visualization is required for activities in the informational phase of a procedural task when a person is orienting on and preparing for the task. Allocentric spatial visualization is required in the workpiece phase of a procedural task where a person is largely concerned with manipulating objects relative to one another. Because egocentric and allocentric visualization abilities are distinct, we believe each might benefit from unique forms of assistance provided by AR. We have structured the next two chapters of this dissertation accordingly.
3 Leveraging Augmented Reality in the Informational Phase of Procedural Tasks

Figure 3.1: A mechanic in our user study, wearing an AR display and wrist-worn control panel, performs a maintenance task inside an LAV-25A1 armored personnel carrier.
One of the central research questions in this thesis is concerned with identifying the efficacy of AR in supporting procedural tasks. To help answer this question, we designed a prototype AR interface (depicted in Figures 3.1 and 3.2) that provides assistance to procedural tasks encountered in the maintenance and repair domain [Henderson and Feiner 2009; 2011]. The application of AR within this domain is intriguing for several reasons. First, navigating and performing maintenance and repair procedures imposes significant physical requirements on a mechanic. For each task within a larger procedure, a mechanic must first move their body, neck, and head to locate and orient to the task. The mechanic must then perform additional physical movement to carry out the task. Assistance optimizing these physical movements, and rendered in AR, can save a mechanic time and energy. Such savings can be significant when performing dozens of potentially unfamiliar tasks distributed across a large, complex system. Second, navigating and performing maintenance and repair procedures imposes cognitive requirements. A mechanic must first spatially frame each task in a presumed model of the larger environment, and map its location to the physical world. The mechanic must then correctly interpret and comprehend the tasks. Effective assistance rendered using AR in these instances can also save the mechanic time, while reducing mental workload.

Of these requirements, we focus this chapter on how AR can fulfill those presented in the informational phase [Neumann and Majoros 1998] of maintenance and repair tasks. As introduced in Section 1.1 and discussed in Section 2.5, activities in this phase are mainly cognitive and involve activities concerned with orienting and preparing a mechanic to conduct physical work performed in a complementary workpiece, or psychomotor phase (which we explore in Chapter 4). Based on our observations of professional mechanics operating in a variety of settings, we believe AR applications designed for maintenance and repair tasks should seek to pro-
vide specific assistance in both phases. Moreover, we suspect that the types of assistance offered in each phase may, in some cases, be distinct.

In this chapter, we demonstrate and evaluate the use of our prototype AR interface to provide informational assistance in standard maintenance scenarios. Our interface provides an enhanced form of task visualization with on-screen instructions, attention-directing symbols, overlaid labels, context-setting 2D and 3D graphics, and animated models. This information is combined with a mechanic’s natural view of the maintenance task in a tracked see-through head-worn display (HWD) and is primarily designed to help the mechanic locate and begin various tasks. We discuss a domain-specific user study examining professional mechanics using our system to perform maintenance tasks on actual equipment in a field setting [Henderson and Feiner 2009; 2011]. Our user study demonstrates how mechanics were able to locate tasks more quickly in the AR condition than when using two baseline conditions. We also document specific instances when the AR condition allowed mechanics to perform tasks with less overall head movement than when using these baselines. We highlight circumstances where AR did not provide significant benefits, and begin to form hypotheses for further testing. Finally, we convey the qualitative insights of these professional mechanics with regard to the intuitiveness, ease of use, and acceptability of our approach.

3.1 Related Work

There has been much interest in applying AR to procedural tasks for maintenance as reflected in the formation of several collaborative industrial and academic research consortia specifically dedicated to the topic—ARVIKA [Friedrich 2002], Service Training through Augmented Reality (STAR) [Raczynski and Gussmann 2004], and ARTESAS [2011]. These and other
efforts have resulted in a sizable body of work, much of which is surveyed by Ong, Yuan, and Nee [2008]. Research exploring the use of AR to support general maintenance tasks includes Feiner, MacIntyre, and Seligmann’s prototype [1993; 1992], which used a tracked monocular optical see-through (OST) HWD to present instructions for servicing a laser printer. Ockermann and Pritchett [1998] studied pilots performing preflight aircraft inspections while following instructions presented on an untracked OST HWD, and demonstrated an undesired over-reliance on computer-generated instructions. Schwald and Laval [2003] proposed a prototype hardware and software framework for supporting a wide range of maintenance categories with AR. Knöpfle and colleagues [2005] developed a prototype AR application and corresponding authoring tool to assist mechanics in removing and installing components, plugs, and fasteners. Platonov and colleagues [2006] developed a similar proof-of-concept system featuring markerless tracking.

The majority of related work focuses on assembly tasks. Caudell and Mizell [1992] proposed a seminal AR prototype to assist in assembling aircraft wire bundles. Subsequent field testing of this system by Curtis and colleagues [1999] found the prototype performed as well as baseline techniques, but faced several practical and acceptance challenges. Reiners and colleagues [1999] demonstrated a prototype AR system that featured a tracked monocular OST HWD, presenting instructions for assembling a car door. Baird and Barfield [1999] showed that users presented with screen-fixed instructions on untracked monocular OST and opaque HWDs completed a computer motherboard assembly task more quickly than when using fixed displays or paper manuals. Tang and colleagues [2003] studied the effectiveness of AR in toy block assemblies and found users made fewer dependent errors when aided by registered instructions displayed with a tracked stereoscopic OST HWD compared to traditional media. An experiment
by Robertson, MacIntyre and Walker [2008] discovered that subjects assembled toy blocks more quickly while viewing registered instructions on a tracked biocular video see-through (VST) HWD than when using non-registered variants. Zauner and colleagues [2003] demonstrated a prototype system for employing AR in a furniture assembly task. Qualitative studies by Nilsson and Johansson involving a medical assembly task [2007] and by Salonen and Sääski involving 3D puzzle assembly [2008] suggest strong user support for AR.

Many notable works address AR interfaces for procedural tasks in domains other than maintenance and repair. In the medical domain, State and colleagues [1996] proposed an AR interface for assisting in the performance of a breast biopsy. Follow-on work by Rosenthal and colleagues [2002] featured a prototype that enabled a trained surgeon to complete a biopsy procedure more quickly and with fewer errors than traditional systems. AR interfaces for procedural tasks also appear in the arts, where Gandy and colleagues [2005] demonstrated AR Karaoke. This application guides users in acting out specified film scripts while interacting with virtual characters. AR interfaces assisting users in the procedural aspects of playing musical instruments appear in works by Cakmakci, Berard and Coutaz [2003], Motokawa and Saito [2006], and Barakonyi and Schmalstieg [2005]. In the food preparation domain, Bonini, Lee, and Selker [2005] created a spatially augmented reality interface for common kitchen tasks. More generalized AR interfaces for assisting in following instructions include work by Asai and Kobayashi [2005], who created a system for operating telecommunications equipment, and Quarles and colleagues [2008] who implemented an interface for teaching the operating principles of an anesthesia machine.

There is also notable work on the general task of localizing a user’s attention in AR, a key activity in the informational phases of procedural tasks. Feiner, MacIntyre, and Seligmann
[1993] used a 3D rubberband line drawn from a screen-fixed label to a possibly offscreen target object or location. Biocca and colleagues developed the “Attention Funnel” [2006], a vector tunnel drawn to a target, similar to “tunnel-in-the-sky” aviation cockpit head-up displays, and showed that it reduced search time compared to world-fixed labels or audible cues. Tönnis and Klinker [2006] demonstrated that an egocentrically aligned screen-fixed 3D arrow projected in AR was faster at directing a car driver’s attention than an exocentric alternative. Wither, DiVerdi, and Höllerer [2007] compared the performance of various displays to support visual search for text in AR (a task supported by localization), but did not detect any significant differences between display conditions. Schwerdtfeger and Klinker [2008; 2009] studied AR attention-directing techniques to help users find and pick objects from stockroom storage bins. Their frame-based technique outperformed static 3D arrows and variants of the Attention Funnel.

Two aspects of our contributions distinguish them from this previous work. First, other than the wire bundle assembly research conducted by Curtis and colleagues [1999], our research is the only project we know of to include a quantitative study of professional users employing AR for maintenance tasks under field conditions. Our work differs from the wire bundle assembly research by examining a more diverse set of maintenance tasks (including inspection, alignment, removal, and installation) in a more restrictive environment using different comparison conditions. Second, our work is the first within the maintenance domain that articulates the potential benefits of AR for reducing head movement.

3.2 Prototype

We developed a hardware and software system for studying AR applications for maintenance and repair. This system, which was built using our ARMAR architecture (Section A.6),
allowed us to implement a prototype focusing on the informational phases of tasks that we evaluated through the user study described in Section 3.3. We note that our prototype is a laboratory proof-of-concept system for exploring the potential benefits of AR for supporting maintenance procedures under field conditions, but is not a production-ready implementation. Therefore, our software and hardware choices did not have to reflect the needs of a production environment.

We have used our prototype to study United States Marine Corps (USMC) mechanics operating inside the turret of an LAV-25A1 armored personnel carrier. The LAV-25 (of which the LAV-25A1 is a variant) is a light-wheeled military vehicle, and the turret portion is a revolving two-person enclosed, cockpit-like station in the middle of the vehicle. The entire turret volume is approximately 1 cubic meter, but much electrical, pneumatic, hydraulic, and mechanical infrastructure encroaches from a myriad of directions and in close proximity to the crew’s operating space. A mechanic servicing the turret works while sitting in one of two seats that are each fixed along the longitudinal axis of the turret. The resulting work area is approximately 0.34 cubic meters and spans the entire area surrounding the mechanic.

Because we did not have regular access to the vehicle, we used an extensive set of 3D laser scans to initially create a virtual mockup of the turret, which we used in our lab during development. We later combined this virtual model with a limited-scope physical mockup of the turret that consisted of turret seats and some structural component extracted from an actual LAV-25A1 turret. We then finalized our design in an actual turret in two separate pilot tests prior to the user study in the real turret. The first pilot test involved prototype testing with users at the Marine Corps Logistics Base in Albany, Georgia. This allowed us to refine our design and gather user feedback about our interface. The second pilot test involved four mechanics from the population recruited for the user study described in Section 3.3. These mechanics experienced nearly the
same test procedure as other participants, but their data was excluded after we modified two
tasks to reduce the overall execution time of our experiment.

Figure 3.2: An example of assistance presented by our prototype during a typical mainte-
nance activity. (Image shows the view through the video see-through display.)

3.2.1 Software Overview

We developed our software prototype as an instance of the ARMAR client, a central
component in the ARMAR architecture. The specific details of this implementation are de-
scribed in Section A.6. At any given point in time, the application assumes a state representing a
particular task (e.g., loosening a fuel line clamp) within a larger maintenance sequence (e.g., removing a combustion chamber). For each task, such as the one depicted in Figure 3.2, the application provides five forms of augmented content to assist the mechanic:

1. Text instructions describing the task and accompanying notes and warnings.
2. Registered labels showing the location of the target component and surrounding context.
3. A close-up view depicting a 3D virtual or real scene centered on the target at close range and rendered on a 2D screen-fixed panel.
4. 3D models of tools (e.g., a screwdriver) and turret components (e.g., fasteners or larger components), if applicable, registered at their current or projected locations in the environment.
5. Attention-directing information in the form of 3D and 2D arrows and highlighting effects.

### 3.2.2 Localization Approach

Attention-directing graphics used for localization follow a general sequence, depicted in Figures 3.3–3.6, that depends on 6DOF user head pose. If the target component is behind the mechanic, a screen-fixed green arrow points the user in the shortest rotational direction to the target. Once the target is within $\pm 90^\circ$ (yaw) of the user’s line of sight (head azimuth), a tapered red semi-transparent 3D arrow appears, directing the user toward the target. The tail of the arrow is smoothly adjusted and placed along the far edge of the display at each frame, based on the vector between the target and the user’s projected line of sight on the near clipping plane. This ensures that the arrow provides a sufficient cross-section for legibility. As the user approaches the target, the arrow increases in transparency and eventually disappears, spawning a highlight-
ing effect for five seconds at the location of the target. Depending on task preferences and settings, the 3D arrow will reengage if the angle between the user’s head azimuth and the direction to target exceeds 30°.

Figure 3.3: Typical localization sequence (1 of 4). When a target task is located behind the user, a screen-fixed arrow indicates the shortest rotation direction to the target. (Image shows the view through the video see-through display.)
Figure 3.4: Typical localization sequence (2 of 4). As the target task enters the user’s field of view, a tapered red semi-transparent 3D arrow appears, directing the user toward the target. In order to promote legibility, the tail of the arrow is smoothly adjusted and placed along the far edge of the display at each frame. (Image shows the view through the video see-through display.)
Figure 3.5: Typical localization sequence (3 of 4). As the user focuses on the target, the red 3D arrow increases in transparency and eventually disappears. The blue screwdriver is animated to show the correct rotation to remove the oil pressure sensor screw. (Image shows the view through the video see-through display.)
Figure 3.6: Typical localization sequence (4 of 4). As the red 3D arrow disengages, the interface spawns a highlighting effect for five seconds at the location of the target. Following this highlight effect, the interface removes the blue animated model of the screwdriver, to avoid interfering with the psychomotor portion of the task. (Image shows the view through the video see-through display.)
3.2.3 Visualization of Tools and Components

For more complex or potentially ambiguous tasks, animated 3D models are added to the user’s view. These animations show the correct movement of tools or components required to accomplish a particular task. For example, when the mechanic is instructed to remove or install hex bolt-type fasteners, an animated socket wrench will depict the correct removal motion relative to the target bolt, as shown in Figure 3.7. By default, animated sequences begin when a task is first presented to the user. The animation continues until five seconds after the 3D attention-directing arrow disengages. At the moment, the interface halts the animation and hides the 3D model. This default behavior is potentially useful for presenting a far-field overview of pending tasks and allows the mechanic to begin planning and preparation before they arrive in the task area. For example, a mechanic might realize she needs to adjust the drive direction on a socket ratchet handle as she views the animation sequence depicted in Figure 3.7. As she transitions to the next task, she can parallelize her work flow by adjusting the wrench as she moves toward the task.

In other scenarios, prolonged or premature animation sequences might confuse or frustrate the mechanic, which we discuss further in Section 3.2.4. In these instances, our prototype has the ability to postpone animations so they begin only after the attention-directing arrow disappears, when the user is presumably focusing on the task. The animation sequence continues for five seconds, and then halts as the interface hides the animated model.
Figure 3.7: An example animated model. (a) As the user localizes on the cable connector, (b) the 3D attention-directing arrow fades and (c) the 3D model of the channel locks begins to animate to show the proper (d-f) motion of the tool. The tool also fades after 5 seconds of animation. (Images show the view through the video see-through display.)
If a mechanic wishes to replay an animated sequence or control its speed, they can use a wireless wrist-worn controller, shown in Figures 3.1 and 3.8, which serves as the primary means for manually interacting with the user interface of our prototype. The controller uses a custom 2D interface application written using the Android SDK, and provides forward and back buttons that allow the mechanic to navigate between maintenance tasks. When viewing tasks with supporting animation, additional buttons and a slider are provided to start, stop, and control the speed of animated sequences. These animation buttons are hidden for non-animated tasks.

![A mechanic uses a wrist-worn controller to cue the next task in a repair sequence. The inset view shows additional features that appear during applicable tasks for controlling animations.](image-url)
3.2.4 Cognitive Design Principles and Heuristics

We leveraged previously published design heuristics in developing the user interface described in Section 3.2.1. One of the main inspirations for our design stemmed from work by Heiser and colleagues [2004], who derived eight principles for visualizing processes over time after observing and surveying users assembling a small television stand:

1. Displaying one diagram for each major step
2. Using arrows and guidelines to indicate attachment
3. Showing stable orientations in a manner that is physically realizable
4. Clear and explicit task ordering
5. Parts added at each step should be visible
6. Mode of attachment should be visible
7. Arrows and guidelines to indicate attachment
8. Action diagrams rather than structural

These heuristics extend earlier work in automated generation of graphics and AR for procedural tasks [Feiner 1985; Seligmann and Feiner 1991; Feiner, MacIntyre, and Seligmann 1993]. We sought to adhere to the entire set of heuristics in designing our prototype. We highlight several individual heuristics here that we feel are particularly crucial to the successful application of AR for maintenance and repair. The first of these is the importance of displaying one diagram for each major step. As shown in earlier work and in our prototype, AR can clutter a mechanic’s natural view of a task with large amounts of virtual content. A potential challenge lies in scoping and organizing this content to preserve the notion of separable and easy to comprehend “major steps.” In our prototype, we sought to maintain a manageable task structure by displaying text instructions and close-up views (as shown in Figure 3.9) similar to what appear in
paper manuals, while also allowing the mechanic to control the progression of steps with the wrist-worn controller.

A second important heuristic highlights the use of arrows and guidelines to indicate action (e.g., attachment, alignment, and removal). While our interface has the ability to display such information using registered 3D models (e.g., the exploded view of fastening hardware shown in Figure 3.9), such models were limited in our prototype to showing the position of tools or major turret components.

Figure 3.9: A representative screenshot that shows the result of applying many of the previously published design heuristics we followed when designing our prototype. (Image shows the view through the video see-through display.)
Two additional and related heuristics governing the design of interfaces for maintenance and repair tasks emphasize showing stable orientations in a manner that is physically realizable, while avoiding changing viewpoints. The use of AR for maintenance and repair can implicitly promote these heuristics by providing a unified in-situ view of both an assigned task environment and its accompanying instructional content. This is demonstrated in our prototype by the use of tracked 3D models and labels depicting components in starting, transient, and target orientations, as specified by certain tasks. Likewise, other 3D models show suggested tool orientations and movements.

Our design also benefited from another set of design principles proposed by Tversky and colleagues governing the use of animated graphics [2002]. The first of these, the principle of congruence, stresses the importance of ensuring an animation’s external representation matches a user’s internal mental model of activity associated with the animation. We uphold this principle by controlling the visibility of animated models so they appear only when depicting a central activity in the current stage of a procedure, as described in Section 3.2.1. Moreover, we limited the use of animated models to complex and ambiguous tasks, which we defined in our prototype as any task requiring use of a tool that can be used to either tighten or loosen an object. Our interface can synchronize model visibility so that the animation is delayed until after task localization when the user is preparing to enter the psychomotor phase of an activity. We also leverage innate characteristics of AR to help maintain congruence. Specifically, the ability to render in-situ 3D animated content to scale and at precise physical locations and orientations allows instructional models to directly match the physical word. The second principle, the principle of apprehension, suggests animations should be easy to perceive and comprehend. Our interface supports this principle by employing realistic, photo-textured, and shaded models in animated sequences that
mimic natural human motion. The ability to control the speed of animated models with the wrist-worn controller can also reduce apprehension by allowing a user to carefully stop, start, and study the animation at their own cadence.

Finally, we adopted the principle of redundant coding throughout our interface design. Redundant coding involves using multiple modes or perceptual channels to present the same information, and has been shown to be effective in comprehending and learning task instructions [Booher 1975; Jubis 1990; Mayer 2002]. This might include varying visual characteristics, such as the size, shape, and color of an object [Christ 1975] or using mixed modes of delivery such as text or verbal instructions augmented by pictures [Mayer 2002]. Our interface combines text and graphics, where text instructions rendered at the top of the screen are complemented by 2D pictorial representations of the task in the lower right-hand corner. Redundancy is also reflected in the main part of the screen, where 3D static models and labels will reinforce information presented in the 2D text and pictorial representations.

3.3 User Study Design

We designed a user study to compare the performance and general acceptance of our prototype (the AR condition) to that of an enhanced version of the electronic documentation system currently used by USMC mechanics (the LCD condition). We also included an untracked version of our prototype in the study as a control for HWD confounds (the HUD condition). Six participants (all male), ages 18–28, were recruited from a recent class of graduates of the USMC Light Wheeled Mechanic Course in Aberdeen Proving Ground, Maryland. Each graduate had minimal experience with maintenance tasks inside the turret of the LAV-25A1, which is featured only in a two-hour introductory block of instruction during the course. Participants categorized their com-
puter experience as having no experience (1 participant), monthly experience (1 participant), weekly experience (1 participant), daily experience (2 participants) or using computers multiple times per day (1 participant). Participants categorized their experience with mechanical systems as either a basic level of experience (2 participants), some experience (2 participants), or very experienced (2 participants). We note that these mechanics’ recent status as students studying under instructors with many years of experience might have led to what may appear to be underreported mechanical experience levels on our qualitative scale. Two participants identified themselves as requiring contact lenses or glasses, and both determined that the separate left and right eye focus adjustments on the HWD provided adequate correction. All participants were right-handed.

3.3.1 Baseline Display Conditions

In our experiment, we wanted to compare our prototype against current techniques used by USMC mechanics while performing maintenance task sequences. These techniques principally involve the use of an Interactive Electronic Technical Manual (IETM) [U.S. Department of Defense 2007], a 2D software application deployed on a portable notebook computer carried and referenced by mechanics while completing tasks. IETM users browse electronic documents in portable document format (PDF) using a specialized reader, an example of which is shown in Figure 3.10.
We felt that a comparison against this system would not be compelling for several reasons. First, the extra time required to navigate this software, which affords less user control than common PDF readers, is significant. Second, the perspective views featured in the software are drawn from arbitrary locations and contain minimal context, which requires users to browse multiple pages with the suboptimal interface. As a result, any task completion or localization metrics would be heavily influenced by the time required to negotiate the IETM interface.

Figure 3.10: Example screen shot from the currently used IETM interface.
**LCD Condition.**

Therefore, we designed an improved version of the IETM interface to use as a baseline in the study. This baseline (the LCD condition) features static 3D scenes presented on a 19” diagonal LCD monitor. The monitor was fixed to the right of the mechanic (who sat in the left seat of the turret during our experiment), on an azimuth of roughly 90° to the mechanic’s forward-facing seated direction. The LCD monitor was positioned and oriented to reflect how mechanics naturally arrange IETM notebook computers while working from the left seat of the LAV-25A1. During each task, the LCD monitor presents a single static 3D rendered scene. Each static scene, such as that shown in Figure 3.11, is rendered using the same graphics component that generates virtual content for the AR condition and depicts identical text instructions, 3D labels, close-up graphics, and animated sequences (if applicable). Additional 3D models are added to the scene to depict the central component of interest, as well as important surrounding context. For each task, static perspective views were chosen that generally correspond to how each scene would naturally appear to a user sitting in the left seat. The diagonal FOV for each scene in the LCD condition was widened to 50° to approximate the projections used in IETMs. When experiencing the LCD condition during the user study, mechanics control the displayed scene by manipulating the system state with the wrist-worn controller.
Head-up Display Condition

To control for the general effects of wearing a HWD, we added a second baseline condition featuring an untracked version of our AR prototype. This HUD (head-up display) condition uses screen-fixed graphics that depict text instructions and close-up views identical to those in the AR condition. However, as shown in Figure 3.12, no localization aids or 3D models were provided. In the HUD condition, participants wear the same VST HWD worn in the AR condi-
tion, and interact with the application using the same wrist-worn controller used in both the AR

![Image: Using a 1/4 inch socket wrench, remove upper Weapon Drive access bolt]

Figure 3.12: HUD Condition. (Image shows the view through the video see-through display.)

and LCD conditions.

3.3.2 Tasks

We selected 18 representative maintenance tasks for inclusion in the user study, from among candidates listed in the LAV-25A1 operator’s manual [U.S. Marine Corps 2003]. Table 3.1 summarizes the selected set of tasks, and Figure 3.13 shows their approximate arrangement inside the turret. These tasks serve as individual steps (e.g., removing a screw, as shown in Fig-
ure 3.1), performed as part of a larger maintenance sequence (e.g., replacing a pump). We specifically avoided adopting an established sequence of tasks whose order would already be familiar to, or make sense to, participants to mitigate experiential influences in the experiment. We selected tasks that a trained mechanic could perform while sitting in the left seat, and which could each be reasonably completed in under five minutes. We also sought to include a diversity of tasks representing various strata within the larger spectrum of maintenance operations [U.S. Army 2007].

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Pitch</th>
<th>Azimuth</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Switch A OFF</td>
<td>31.3</td>
<td>40.2</td>
</tr>
<tr>
<td>T2</td>
<td>Remove Bulb X</td>
<td>26.5</td>
<td>35.9</td>
</tr>
<tr>
<td>T3</td>
<td>Switch B ON</td>
<td>28.2</td>
<td>33.0</td>
</tr>
<tr>
<td>T4</td>
<td>Remove Bolt #1</td>
<td>25.3</td>
<td>15.9</td>
</tr>
<tr>
<td>T5</td>
<td>Switch C OFF</td>
<td>38.5</td>
<td>42.1</td>
</tr>
<tr>
<td>T6</td>
<td>Inspect Assembly #1</td>
<td>30.9</td>
<td>58.0</td>
</tr>
<tr>
<td>T7</td>
<td>Inspect Assembly #2</td>
<td>19.4</td>
<td>-34.4</td>
</tr>
<tr>
<td>T8</td>
<td>Drive Lock to LOCK</td>
<td>-10.1</td>
<td>132.2</td>
</tr>
<tr>
<td>T9</td>
<td>Install Bulb Y</td>
<td>20.2</td>
<td>36.5</td>
</tr>
<tr>
<td>T10</td>
<td>Switch D OFF</td>
<td>23.2</td>
<td>39.5</td>
</tr>
<tr>
<td>T11</td>
<td>Switch E ON</td>
<td>25.2</td>
<td>45.1</td>
</tr>
<tr>
<td>T12</td>
<td>Lever 23 to Manual</td>
<td>11.9</td>
<td>-42.0</td>
</tr>
<tr>
<td>T13</td>
<td>Remove Bolt #2</td>
<td>35.9</td>
<td>17.8</td>
</tr>
<tr>
<td>T14</td>
<td>Remove Screw K</td>
<td>19.0</td>
<td>37.0</td>
</tr>
<tr>
<td>T15</td>
<td>Install Bolt #1</td>
<td>25.3</td>
<td>15.9</td>
</tr>
<tr>
<td>T16</td>
<td>Connect Cable</td>
<td>29.3</td>
<td>19.1</td>
</tr>
<tr>
<td>T17</td>
<td>Install Screw S</td>
<td>19.0</td>
<td>43.9</td>
</tr>
<tr>
<td>T18</td>
<td>Install Bolt #2</td>
<td>35.9</td>
<td>17.8</td>
</tr>
</tbody>
</table>

Table 3.1: Selected tasks (with descriptions expurgated for publication) and corresponding pitch and azimuth measured from 0.7 meters above the center of the left turret seat.
Figure 3.13: Approximate task azimuths and distances as viewed from above the turret and looking down. Neighboring task identifiers are separated by commas.
3.3.3 Procedure

A within-subject, repeated measures design was used, consisting of three conditions (AR, LCD, and HUD) and 18 maintenance tasks. The experiment lasted approximately 75 minutes and was divided into three blocks, one per condition, with a short break between blocks. Each block consisted of all 18 tasks for its condition. Block (condition) order was counterbalanced across participants using a Latin square approach to create a strongly-balanced design. As described above, the task order within blocks was fixed, with the participants experiencing the same tasks in the same location across all three conditions. Before each block, the participant was shown how to wear the equipment used in its condition. In the AR and HUD conditions, this consisted of fitting and focusing the HWD, with an additional brief calibration step for the AR condition. For the LCD condition, participants donned a lightweight headband affixed with IR LEDs to facilitate collecting tracking data. No portion of the tracking apparatus entered the participant’s field of view during the study. We also note that this tracking approach did not use eye-tracking and assumed that the participant’s gaze is coincident with their head orientation.

Before each block, each participant was afforded an opportunity to rehearse the condition using five practice tasks until they felt comfortable. Tools and fasteners required for tasks within the block were arrayed on a flat waist-high structure to the right of the seat and their locations pointed out by the experimenter.

The timed portion of each block consisted of the 18 trial tasks distributed throughout the mechanic’s work area. Each trial began when the mechanic pressed the “next” button on the wrist-worn controller. This started the overall task completion timer, and triggered the presenta-
tion of instructional text, close-up views, and labels associated with the trial task. In the AR condition, cueing information (i.e., the red or green arrow) was simultaneously activated, prompting the participant to locate the target. The localization time was recorded when the participant positioned their head such that the target location entered and remained within a 200 pixel radius of the center of the display for more than one second. In the AR and HUD conditions, a crosshair was displayed to the participant to remind them to center each target. In the LCD condition, which presented static 3D scenes for each task during the experiment, collected tracking data was replayed in a discrete event simulation after the experiment to calculate the localization time. Following target localization, overall task completion timing continued until the mechanic gestured on the wrist-worn controller for the next task. The block then proceeded to the next task until the participant experienced all 18 tasks.

3.3.4 Hypotheses

Prior to analyzing our study results, we formulated six hypotheses:

- **H1: Mean completion time for AR would be less than that for HUD or LCD.**
  The ability to present in-situ instructions in AR should allow participants to remain more focused on the task than the other display conditions, resulting in faster completion times.

- **H2: Mean localization time for AR would be less than that for HUD or LCD.**
  The attention-directing abilities afforded by AR should allow participants to spend less time searching for tasks.

- **H3: Mean error rate for AR would be less than that for HUD or LCD.** The ability to present in-situ instructions in AR should allow participants to make
fewer errors that might result when instructions are read in one location but applied in another location.

- **H4: Mean head rotation for AR would be less than that for HUD or LCD.** Because the attention-directing abilities afforded by AR will actively guide participants to the task, participants will exert less rotational movement performing search activities.

- **H5: Mean head translation for AR would be less than that for HUD or LCD.** Similarly, the direct guidance afforded by AR will result in less translational movement than would otherwise be required during search activities.

- **H6: Mean head velocity for AR would be less than that for HUD or LCD.** Because the participant will follow deliberate attention-directing cues, participants will move their head more slowly than when performing the search activities required while experiencing the other two display conditions.

### 3.4 Quantitative Results

#### 3.4.1 Data Preparation

We performed several preprocessing steps prior to analyzing our results. First, because the tracker coordinate system was centered above the left camera of our VST HWD, we translated tracking data points to a position coincident with the center of rotation for the participant’s head. This was accomplished by adding a small offset vector \( v \) to each reading, where \( v \) was estimated by combining HWD measurements with population-specific anthropometric data from Donelson and Gordon [1996] and supplemented by Paquette and colleagues [1997].
Because the tasks we selected for our experiment differed in difficulty and expected completion times, we organized the 18 tasks into groups of common task types, shown in Table 3.2. These groups facilitated further analysis of our data.

<table>
<thead>
<tr>
<th>Common Task Type</th>
<th>Included Tasks (from Table 3.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspection Tasks</td>
<td>T6, T7</td>
</tr>
<tr>
<td>Aligning Tasks</td>
<td>T8, T12</td>
</tr>
<tr>
<td>Switch Tasks</td>
<td>T1, T3, T5, T10, T11</td>
</tr>
<tr>
<td>Trivial Installation and Removal Tasks</td>
<td>T2, T9, T14, T17</td>
</tr>
<tr>
<td>Non-trivial Installation and Removal Tasks</td>
<td>T4, T13, T15, T16, T18</td>
</tr>
</tbody>
</table>

Table 3.2: Groupings of common task types

We then removed spurious points and outliers in the recorded tracking and completion time datasets. For tracking data, we applied a moving average filter as defined by Law and Kelton [1997]. After some experimenting, we selected a window size of 0.25 secs, which was applied to all six degrees of freedom.

For completion time data, we manually inspected the task completion timestamps that were triggered when the participant gestured for the next task using the wrist-worn controller. We detected several sources of outliers:

- *Interface errors.* In several instances, participants made accidental double gestures, then quickly (usually within two seconds) gestured on the “back” button to reload the appropriate task. We identified and removed 8 of these instances (3 for the AR condition, 2 for the HUD condition, and 3 for the LCD condition) which eliminated 2.5% of our original sample data.
- *Exceptionally long completion times.* We noticed several instances of exceptionally long completion times, which we defined as task completion times exceeding two standard deviations above the mean for a particular task type. These outliers were caused by mechanics dropping tools or components, difficulties experienced when performing certain fine motor skills (e.g., starting screws and bolts into thread holes), adjustments to the head-worn display, and extremely slow performance in one task across all conditions by one mechanic. We identified and removed 17 of these instances (4 for the AR condition, 7 in the HUD condition, and 6 in the LCD condition) which eliminated 5.2% of our original sample data.

Our final data preparation step involved normalizing position and orientation data for each participant. Because the HWD was worn differently by each participant, the relative position and orientation of the tracker to tasks in the experiment varies by participant. To standardize all participants to a common reference frame, we individually normalized each participant’s position and orientation data, as suggested by Axholt, Peterson, and Ellis [2008].

### 3.4.2 Order Effects

We performed an analysis to check for order effects in our study. We applied a 3 (Presentation Order) × 18 (Task) repeated measure ANOVA on both task localization and completion time and with our participants as the random variable. Presentation order failed to exhibit a significant main effect on localization time ($F_{(2,34)}=0.039$, $p = 0.962$) or completion time ($F_{(2,34)}=0.917$, $p = 0.431$).
3.4.3 Completion Time Analysis

Figure 3.14 depicts the average completion time for all tasks, which includes observations from each of the five common tasks types listed in Table 3.2. We applied a 3 (Display Condition) × 18 (Task) repeated measure ANOVA to task completion time with our participants as the random variable. Using $\alpha=0.05$ as our threshold for significance, the display condition failed to produce a significant main effect on completion time ($F_{(2,34)}=9.53, p = 0.095$). The mean task completion times for each condition were 31.1 seconds (AR), 52.9 seconds (HUD), and 26.6 seconds (LCD) and are shown in Figure 3.14. The set of individual maintenance tasks used in the study failed to produce a significant main effect on completion time ($F_{(17,34)}=1.83, p = 0.111$). However, a one-way ANOVA of average completion time for the task types listed in Table 3.2 did reveal a significant main effect ($F_{(4)}=27.65, p < 0.001$) which we expected, given the varying levels of effort required to perform each task type. The mean completion times for each task type are depicted in Figure 3.15.
Figure 3.14: Average task completion times (seconds) across all task types for AR, HUD, and LCD.
Figure 3.15: Average task completion times (seconds) by common task type for AR, HUD, and LCD. The edges of each box represent the 25th and 75th percentiles, and whiskers extend to the most extreme data points not considered outliers. Points are drawn as outliers if they are larger than $q_3 + 1.5(q_3 - q_1)$ or smaller than $q_1 - 1.5(q_3 - q_1)$, where $q_1$ and $q_3$ are the 25th and 75th percentiles, respectively.
3.4.4 Localization Time Analysis

We applied a 3 (Display Condition) × 18 (Task) repeated measure ANOVA on task localization time with our participants as the random variable. Display condition exhibited a significant main effect on localization time ($F_{(2,34)}=42.444$, $p < 0.001$). The mean task localization times were 4.9 seconds (AR), 11.1 seconds (HUD), and 9.2 seconds (LCD), as shown in Figure 3.16. Post-hoc comparison with Bonferroni correction ($\alpha=0.05$) revealed that mean localization time under the AR condition was 44% that of the HUD condition, which was statistically significant ($p = 0.001$) and 53% that of the LCD condition, which was also statistically significant ($p = 0.007$). LCD mean localization time was 83% that of HUD, which was not statistically significant ($p = 0.085$). These results support hypothesis H2.

The particular set of selected maintenance tasks used in the study failed to exhibit a significant main effect on localization time ($F_{(2,34)}=1.533$, $p = 0.103$). We did not analyze localization time by common task type, as we found little evidence during pilot testing to suggest a participant’s ability to locate a task was dependent on the task’s difficulty.

3.4.5 Error Analysis

Errors in our experiment were defined as instances when a participant performed a task to completion on the wrong item, and were logged by the observer during the experiment. Examples of errors included toggling an incorrect switch, removing an incorrect bolt, or inspecting the wrong item. In general, we found mechanics made few errors. Across all tasks performed, our participants collectively made 4 errors when experiencing the AR condition, 3 errors when expe-
riencing the HUD condition, and 6 errors when experiencing the LCD condition. We conducted a 3 (Display Condition) × 18 (Task) repeated measure ANOVA on task errors with our participants as the random variable. Display condition did not exhibit a significant main effect on total errors ($F_{(2,34)}=1.00$, $p=0.410$). This result corroborates earlier findings by Robinson and colleagues [2008] and fails to support hypothesis H3.

3.4.6 Head Movement Analysis

Our analysis of head movement focused on the range of head rotation, rotational exertion and velocity, and translational exertion and velocity. This analysis was confined to only the lo-
calization portion of each task, because it was difficult to isolate head movements from overall body motion during the hands-on portion of some tasks. In these tasks, the participants remained relatively static during localization, but adopted many different body poses once they began the physical portion of the task.

Table 3.3 depicts the descriptive statistics for overall ranges in head rotation about each axis across all tasks. Left and right head rotation about the neck (azimuth or yaw) was the greatest source of rotational movement, and generally conforms to the relative task azimuths shown in Table 3.1. A comparison of ranges by task, shown in Figure 3.17, reveals that the range of head yaw was smaller under the AR condition for many tasks. It should be noted that the range information includes transient head movements between tasks and thus intervals are not necessarily centered on the target task.

<table>
<thead>
<tr>
<th>Axis</th>
<th>Min</th>
<th>Max</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pitch</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AR</td>
<td>-4.0</td>
<td>74.1</td>
<td>78.0</td>
</tr>
<tr>
<td>HUD</td>
<td>-6.3</td>
<td>64.7</td>
<td>71.0</td>
</tr>
<tr>
<td>LCD</td>
<td>1.9</td>
<td>85.9</td>
<td>84.0</td>
</tr>
<tr>
<td>Roll</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AR</td>
<td>-8.3</td>
<td>34.0</td>
<td>42.3</td>
</tr>
<tr>
<td>HUD</td>
<td>-21.8</td>
<td>34.1</td>
<td>55.9</td>
</tr>
<tr>
<td>LCD</td>
<td>-45.3</td>
<td>53.9</td>
<td>99.2</td>
</tr>
<tr>
<td>Yaw</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AR</td>
<td>-56.4</td>
<td>143.2</td>
<td>199.6</td>
</tr>
<tr>
<td>HUD</td>
<td>-67.0</td>
<td>134.2</td>
<td>201.2</td>
</tr>
<tr>
<td>LCD</td>
<td>-39.3</td>
<td>143.3</td>
<td>182.6</td>
</tr>
</tbody>
</table>

Table 3.3: Ranges (in degrees) in head rotation across all tasks.

Rotational head exertion during each task was estimated for each participant by summing the changes in head pitch, yaw, and roll Euler angles at each interval of the recorded data over the time required to locate the task. Rotational velocity during each task was calculated for each
participant by dividing this total rotational exertion for each axis by the time required to locate
the task. Table 3.4 summarizes these statistics. A 3 (Display Condition) × 18 (Task) repeated
measure ANOVA was performed separately for each statistic along each axis, with participants
as the random variable. In this analysis, display condition had a significant effect on pitch exer-
tion ($F_{(2,34)}=12.206$, $p = 0.002$), roll exertion ($F_{(2,34)}=34.496$, $p < 0.001$), and yaw exertion
($F_{(2,34)}=32.529$, $p < 0.001$). Post-hoc comparisons with Bonferroni correction ($\alpha=0.05$) are sum-
marized in Table 3.4. These results provide support for hypothesis H4. During the study, we no-
ticed some mechanics glancing at the LCD from oblique angles, though this activity appeared to
be confined to natural eye motion exhibiting during reading. Future studies should attempt to ac-
count for these finer gaze patterns when comparing display conditions.

For rotational velocity, display condition had a significant main effect on mean pitch ve-
locity ($F_{(2,34)}=12.205$, $p = 0.002$), mean roll velocity ($F_{(2,34)}=48.875$, $p < 0.001$), and mean yaw
velocity ($F_{(2,34)}=44.191$, $p < 0.001$). Table 3.4 captures the post-hoc comparisons of means with
Bonferroni correction ($\alpha=0.05$). As shown in the table, significant differences in rotational ve-
locity among the display types appeared in the roll and yaw axis. The HUD condition produced
the lowest velocities in each of these axes. In the yaw axes, the AR condition was significantly
slower than the LCD condition, but significantly faster than HUD. This mixed result fails to de-
finitively support hypothesis H6.
Figure 3.17: Ranges of head rotation (degrees yaw) for all participants across each task. Tasks are stacked in layers. Each task layer shows ranges for AR (bottom), HUD (middle), and LCD (top).
Translational head exertion during each task was estimated for each participant by summing the change in Euclidean distance exhibited between each interval of the recorded data. The result represents the total Euclidean distances the head traveled during localization. A 3 (Display Condition) × 18 (Task) repeated measure ANOVA test revealed a significant main effect of display condition on translational exertion ($F_{(2,34)}=17.467, p = 0.001$). The mean translational head exertions were 0.25 meters (AR), 0.36 meters (HUD), and 0.68 meters (LCD). Post-hoc comparisons of mean translational exertion with Bonferroni correction revealed that exertion exhibited with the AR display was 69% that of HUD (not statistically significant, $p = 0.432$), and 37% that of LCD, which was statistically significant ($p=0.022$). HUD exertion was 53% that of LCD, which was statistically significant ($p=0.01$). These results provide support for hypothesis H5. This reduction in head exertion is further depicted in the 2D histogram heat maps shown in Figure 3.18. The heat maps depict normalized head positions for all participants across all tasks and reflect a larger overall area required for head movement in the LCD condition.

Translational head velocity was estimated for each participant by dividing total translational head exertion during task localization by the time required to locate the task. A 3 (Display Condition) × 18 (Task) repeated measure ANOVA test revealed a significant main effect of display condition on translational velocity ($F_{(2,34)}=19.907, p < 0.001$). The mean translational head velocities were 0.05 meters/second (AR), 0.03 meters/second (HUD), and 0.08 meters/second (LCD). Post-hoc comparisons of means with Bonferroni correction revealed that the AR display condition maintained a translation velocity 1.6 times that of the HUD condition, which was not statistically significant ($p=0.057$). The LCD translational velocity was 1.7 times that of the AR display condition, which was not statistically significant ($p=0.09$), and 2.7 times that of the HUD condition, which was statistically significant ($p=0.007$).
<table>
<thead>
<tr>
<th>Statistic</th>
<th>AR</th>
<th>HUD</th>
<th>LCD</th>
<th>Comparisons</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mean Rotational Exertion (°)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Pitch | 21.8 | 38.8 | 56.6 | AR exertion 57% that of HUD  
AR exertion 38% that of LCD*  
HUD exertion 68% that of LCD* |
| Roll | 12.7 | 21.2 | 47.2 | AR exertion 60% that of HUD  
AR exertion 25% that of LCD*  
HUD exertion 45% that of LCD* |
| Yaw | 42.9 | 50.2 | 122.3 | AR exertion 85% that of HUD  
AR exertion 35% that of LCD*  
HUD exertion 41% that of LCD* |
| **Mean Rotational Velocity (°/s)** | | | | |
| Pitch | 4.2 | 2.8 | 4.4 | AR velocity 1.54 times that of HUD  
LCD velocity 1.05 times that of AR  
LCD velocity 1.61 times that of HUD |
| Roll | 2.4 | 1.7 | 4.0 | AR velocity 1.48 times that of HUD*  
LCD velocity 1.62 times that of AR  
LCD velocity 2.39 times that of HUD* |
| Yaw | 7.7 | 4.0 | 13.5 | AR velocity 1.95 times that of HUD*  
LCD velocity 1.74 times that of AR*  
LCD velocity 3.41 times that of HUD*  
*statistically significant difference (p < 0.05) |

Table 3.4: Rotational Exertions and Velocities.
We note that it is difficult to generalize our findings that show the AR condition produced less head movement than the LCD condition. While we hope this was entirely a result of the localization information provided by AR, a reduction in head movement might be partially

Figure 3.18: 2D Histograms shown as heatmaps of normalized head positions (viewed from above the turret looking down) across all participants and tasks for AR (upper left), HUD (upper right), and LCD (lower right). The heatmap scale represents the relative number of hits in each bin of a 150×150 grid covering the task area. Bin sizes are 0.004m (X) and 0.003m (Y). The schematic diagram in the lower-left corner depicts the general orientation of the mechanics while data was collected for the heatmaps (also viewed from above the turret looking down).
attributable to mechanics restraining their head movements when wearing the HWD. More work is required to articulate the specific contribution provided by AR in reducing head movements during localization when a user is wearing an HWD with a reduced field of view.

3.4.7 Supporting Task Focus

We employed several methods to analyze how well each condition supported a mechanic’s ability to remain focused on a particular task versus looking elsewhere (e.g., referencing a manual or IETM). Quantifying a mechanic’s ability to sustain physical and cognitive focus on his or her current task is an important research question in the maintenance and repair domain. Breaking this focus can prolong the length of the repair. In addition to incurring more time to move his or her head, the mechanic will also require time to shift their mental model of the task from what they see physically to what they interpret in any referenced documentation. This interpretation process could potentially involve several non-trivial steps: visually searching images to identify features of interest, matching these features to points in the real world, mentally transforming objects from the documentation’s perspective to the real-world, and memorizing supporting information such as warnings or instructions.

The first method we employed to examine support for task focus involved estimating the Distance from Center Point (DFCP) for each task, as defined by Axholt, Peterson, and Ellis [2008]. This measure reflects the average angular distance a tracked body deviates about a reference point. In our experiment, the DFCP reference point is the vector between the participant’s predominant head pose and each of the 18 evaluated tasks. With this definition, DFCP provides an indicator of the level of focus maintained by each mechanic during each assigned task while experiencing each condition. We calculated DFCP for each participant under all combinations of
tasks and display conditions by first defining a center direction. We estimated this center direction due to variations in HWD boresight and because participants viewed tasks from possibly different poses. For the AR and HUD display conditions, we defined this center direction as the mean normalized orientation (pitch and yaw) exhibited by participants during each task. We included data from the entire completion interval in this calculation to provide sufficient sampling for isolation of the task’s principal viewing direction. In the case of the LCD display condition, the mean yaw component of head orientation was not expected to serve as an accurate estimate because each participant alternated between looking at the task and looking at the LCD monitor. Therefore, an additional step was required to identify the principal viewing direction. This involved examining the distribution of normalized yaw angles to estimate the primary direction to each task. This analysis revealed a distinctive bimodal distribution for tasks compared to corresponding distributions in normalized yaw for the AR and HUD conditions. An example of the comparison is shown in Figure 3.19. We isolated the direction to each task in the LCD condition by manually selecting the local optimum in each distribution corresponding to each task’s relative location in the turret. This allowed us to disambiguate the local optimum corresponding to the task from the local optimum corresponding to the LCD monitor.
After defining a center direction to each task, we next summed the distance from this central viewing vector to every pitch/yaw pair in the head tracker data. We approximated these individual distances by calculating the composite vector formed by the intersection of each yaw and pitch angle on a unit sphere. Finally, we calculated DFCP by dividing the sum of each of these approximated distances by the number of samples. We applied a 3 (Display Condition) × 18 (Task) repeated measure ANOVA to DFCP with our participants as the random variable. Display condition exhibited a significant main effect on localization time ($F_{(2,34)}=1043.6$, $p < 0.001$). The mean DFCP values were 0.183 meters (AR), 0.137 meters (HUD), and 0.703 meters (LCD). In comparison, the registration accuracy provided by the OptiTrack tracking system is approximately 0.001 meters. Post-hoc comparison with Bonferroni correction ($\alpha=0.05$) revealed that HUD distance from center point was 0.75 times that of AR, which was not statistically significant ($p = 0.16$). The AR distance from center point was 0.27 times that of LCD, which was significant ($p$)

Figure 3.19: Distribution of normalized yaw angles for AR and LCD for Task T4. In each plot, the value $x=0$ indicates the population’s mean yaw orientation.
The second method we employed in examining the amount of time a mechanic spent looking somewhere other than at the assigned task involved inspecting each participant’s head azimuth trajectory across the entire sequence of 18 tasks. We began by first tracing the ideal head yaw trajectory over the entire sequence. This ideal trajectory assumes a mechanic will begin the repair sequence with his or her head oriented in the forward direction (0° azimuth).

In an ideal repair, a mechanic would move his or her head systematically to each subsequent task azimuth (listed in Table 3.1), stopping at each location to complete the workpiece portion of the task. We next created similar plots for each participant that overlaid their yaw trajectories exhibited while completing the task sequence under each display condition. To synchronize the plots across all participants, we normalized time in each task interval for each participant according to the total time spent localizing and completing each task. The resultant plot, shown in Figure 3.20, offers some interesting insights about potential interruptions in a mechanic’s task focus. Note, we elected to show only the AR and LCD yaw trajectories here, which were the most interesting, in order to promote readability (the characteristics of the omitted HUD trajectories roughly reflected those of the AR trajectory for each participant). An examination of the plot reflects a distinctive aperiodic pulse in the LCD yaw trajectory for each participant. This pulse, as confirmed by a careful review of video recorded during the experiment, reflects the moments during each task where the mechanic glanced at the LCD. We note it is difficult to statistically quantify this motion due to possible variations in the position of each mechanic’s head throughout the task. However, we believe the distinctive signal of the LCD trajectory roughly captures the number of times the mechanic turned his head to glance at the LCD. Visually comparing the
LCD yaw trajectories to those of AR appears to indicate the AR condition allowed mechanics to remain more focused on the task at hand.

Figure 3.20: Head orientation (yaw) trajectories for each participant (S1–S6) under AR and LCD conditions. The x axis shows normalized elapsed time for each task, and the y axis shows rotational head displacement about the forward facing direction.
3.5 Qualitative Results

We asked each participant to complete a post-experiment questionnaire. This questionnaire featured five-point Likert-scale questions (1 = most negative, 5 = most positive) to evaluate ease of use, satisfaction level, and intuitiveness for each display condition. The summary results from these ratings are shown in Figure 3.21. In terms of ease of use, median response for LCD (5) was highest, followed by AR (4.5) and HUD (3.5). For satisfaction, median response to AR (5) was highest, followed by LCD (4) and HUD (4). For intuitiveness, median response to AR (4.5) tied with LCD (4.5), followed by HUD (4). A Friedman test revealed significant rankings for ease of use ($\chi^2(6,2)=4.63, p=0.02$) and intuitiveness ($\chi^2(6,2)=9.82, p=0.007$), but failed to detect significant rankings in the case of Satisfaction ($\chi^2(6,2)=3.1, p=0.212$). Subsequent pair-wise Wilcoxon tests for ease of use revealed that LCD was ranked significantly better than HUD ($p = 0.02$). Subsequent pair-wise Wilcoxon tests for intuitiveness revealed that AR was ranked significantly better than HUD ($p = 0.02$) and LCD was ranked significantly better than HUD ($p = 0.03$).

Figure 3.22 (top) shows the distribution of responses when we asked participants to rank the techniques as to how intuitive they were. This distribution shows 4 of the 6 participants ranked the AR condition first. However, a Friedman test indicated this was not a significant ranking ($\chi^2(6,2)=4.33, p=0.12$). Figure 3.22 (bottom) depicts the distribution of responses when we asked participants to rank the techniques in order of preferred use. The figure shows 4 of the 6 participants ranked the LCD condition first. A Friedman test indicated this was a significant ranking ($\chi^2(6,2)=7.0, p=0.03$). Subsequent pair-wise Wilcoxon tests revealed that LCD was ranked significantly better than HUD ($p = 0.02$).
Figure 3.21: Survey response histograms by condition for ease of use (top), satisfaction (middle), and intuitiveness (bottom). Median values for each condition are shown as triangles.
We also asked participants to comment on each display condition. In reviewing the LCD condition, participants were nearly unanimous in their appreciation for the system. For instance, P1 reported “I liked being able to see what I was doing on the screen. I think the screen idea is good, because it doesn’t restrict your light or head movements.” P2 added “It was a lot easier to
look at a screen than to have your vision blocked by the popups on the screen,” which offers insights into perceived occlusion issues resulting from virtual content in the AR and HUD conditions. Interestingly, none of the participants commented on the disadvantage of having to look back and forth from the target task to the LCD screen. Conversely, several participants actually highlighted the LCD condition’s ability to help them localize. P4 offered “I liked the LCD the most with the program showing me right where the part was, and what tool, without the head-gear getting in the way.” P3 confirmed “things were easy to find” with the LCD.

When commenting on the AR condition, the participants offered useful feedback on our choices of visual assistance. In describing the 3D attention-directing arrow, P1 wrote “I enjoyed this system the most….was easy to navigate with the tracking red line.” P1 also commented on our use of overlaid virtual models, adding “The 3-D image indicators were most satisfying, which allowed for proper item location.” P6 also found the attention-directing graphics of our interface helpful, writing “Prior systems may use over-technical terms that can sometimes be confusing, however this system is directive and simply points. I find that feature extremely helpful.” While echoing these same sentiments about attention-directing graphics, P5 offered additional feedback about the use of animation, “The lines pointing to the objects make it very easy…the animation of the wrench going a different direction, whether tightening or loosening is nice.” P2 found the close-up view helpful in mitigating registration issues when stating “the ‘red line’ takes you right to what your [sic] are looking on…the only problem I had was the arrow sometimes didn’t point to exactly what I was working on but the close-up view helped sort out any confusion.”

Several participants commented on negative aspects of the AR condition. P3 offered “the red line blocked my line of sight.” P4 described the red 3D arrow as “in the way,” but yielded
that the arrow “would help someone who has very little or no experience.” Despite our efforts to control for occlusion by fading the 3D arrow once the mechanic oriented on the target task, these later two comments suggest more careful work is needed to prevent occlusion during localization.

Participant reaction to the HUD condition was overwhelming negative. P1 wrote “My least favorite system because I didn’t know where some things were located in the turret…Identification was more difficult for someone not being completely familiar with the turret.” P3 described this experience with HUD as “It wasn’t hard but it was a little confusing and when it got confusing it got frustrating. It took me awhile to find what I was looking for.” Despite the fact that the HUD condition afforded the same visual acuity and freedom of movement as the AR condition, several participants singled out these characteristics only while experiencing the HUD condition. P2 offered “it restricted some of my head movements in the vehicle and the screen was a little dark and made things kind of hard to see.” P4 cited “picture and head clearance” as drawbacks with the HUD condition.

When we asked the participants to list additional technologies that might assist with their roles as mechanics, we received a number of interesting ideas:

• P1: “Perhaps a device that lets you be able to see using your peripheral vision...maybe just use one eye”

• P2: “I think a voice activated system would make things easier because it would be completely hands free.”

• P4: “Better picture quality and smaller head gear.”

• P5: “Virtual wiring diagram and a hydraulic diagram.”
• P6: “Perhaps an audio track that gives the instructions along with the visual aids. I also think if the software could interpret the movements and actions it could acknowledge when the task was completed and give advice.”

3.6 Discussion

We were pleased that our prototype AR application proved more effective than an enhanced baseline system at reducing time and head movement during the localization portion of maintenance tasks. We were especially encouraged to achieve these results with a population of professionally-trained mechanics working in a field setting, who expressed support for our approach. The AR display condition allowed mechanics to locate tasks significantly more quickly than when using the LCD display condition (representing an improved version of the IETMs currently employed in practice). Because AR was also significantly faster at localization than HUD, we can conclude that the 3D registered localization information overlaid on the mechanic’s view of the task (e.g., arrows, labels, and models) contributed to this result. The ability of an AR interface to save localization time is significant when one extrapolates the average 4.3 seconds of localization time saved in each task of our experiment to a procedure involving hundreds of tasks across a wide area.

The AR display condition also allowed mechanics to incur significantly fewer translational and rotational head movements at lower velocities than the LCD display condition during task localization. Descriptive statistics show that, in general, participants experiencing the AR condition also required smaller ranges of head movement. This result highlights the ability of AR to potentially reduce overall musculoskeletal workloads and strain related to head movement during maintenance tasks. However, more work is required to reconcile strain reductions afford-
ed by reduced head and neck movement with the added strain of wearing a HWD. A technique proposed by Tümler and colleagues [Tümler et al. 2008], which uses heart rate variability to measure strain, could be useful for this analysis. We also note that none of our participants commented on the ability of the AR condition to reduce head and neck movements. Moreover, participants never mentioned head or neck movement or strain during the entire experiment. This suggests that potential benefits provided by AR interfaces in reducing head and neck movements might only be fully realized and appreciated in certain domains.

Our qualitative results provide additional encouragement for the application of AR to maintenance tasks. Despite the disadvantage of wearing a bulky, relatively low-resolution prototype VST HWD with fixed focus cameras, and a narrow FOV, participants rated the AR condition at least as favorably as LCD in terms of satisfaction and intuitiveness. Future AR systems employing lighter, more comfortable displays with wider FOVs and higher resolutions could improve these results. Mechanics also provided some interesting written remarks when responding to our survey. While many participants acknowledged the visibility constraints experienced while using AR, they dismissed this limitation with profuse appreciation for the 3D arrows, labels, and animated sequences. Several participants mentioned the potential utility of the tool in maintaining hydraulic and electrical systems in particular.

We were not surprised by the lack of statistically significant separation between the mean completion times for the AR and LCD display conditions. This can be explained, in part, by examining post-localization information requirements in the workpiece portion of a task [Neumann and Majoros 1998]. While we did provide some alignment and routing information (e.g., content showing the position and movements of select tools and turret components), this was common to both AR and LCD. Moreover, this information was based on an object’s static position in the tur-
ret, as we did not dynamically track tools or components in response to user actions. Therefore, once a mechanic began physically manipulating objects in a task, they tended to not require information provided any display, whether AR, HUD, or LCD. Additionally, because our prototype was not providing assistance to the mechanic during the psychomotor phase, the unrestricted field of view afforded by the LCD condition provided a natural advantage. Contrastingly, our custom VST LCD, with its limited FOV and scaled down VGA resolution became a disadvantage when it was no longer actively assisting the mechanic. We believe a higher quality HWD display, especially an OST display with a wider field of view might mitigate this disadvantage to a certain degree.

In the next chapter of this dissertation, we will build on these observations in order to extend the benefits provided by our interface during task localization to psychomotor activities. First, we will propose novel designs for providing ongoing, dynamic task assistance that use tracking information about the user and the task environment to assist users as they carry out physical manipulations. Second, we will employ a higher-quality HWD to promote a clearer view of the task environment. Finally, we will evaluate these enhancements in a challenging and realistic task as to maximize the relevancy of any findings.
4 Leveraging Augmented Reality in the Psychomotor Phases of Procedural Tasks

Figure 4.1: AR assistance in the psychomotor phase of a procedural task. (left) A user aligns two combustion chamber components while (right) viewing continuous, dynamic feedback about the alignment presented by an optical see-through HWD. (The image on the right was captured by a video camera mounted in a dummy head wearing the HWD. A post-render filter was applied to remove camera distortion and vignetting.)

The research presented in Chapter 3 demonstrated three important benefits of using AR interfaces to assist with procedural tasks. First, the AR interface reduced the time required to locate procedural tasks. This benefit could translate into substantial savings if an AR interface is assisting a person in performing dozens of potentially unfamiliar tasks distributed across large, complex systems. Second, we showed that it is possible for an AR interface to reduce the head
and neck movements of a person during this localization. This benefit could potentially enhance the occupational health of workers by reducing overall musculoskeletal loads and strain related to head and neck movement. Third, we found evidence that AR interfaces allow people to remain more oriented and focused on a task compared to when using a traditional form of documentation. These benefits could potentially reduce a person’s cognitive load by reducing the need to memorize information and eliminating the need to mentally transform objects from instructional models to the physical world.

Although our system clearly helped the LAV-25A1 mechanics during task localization, we did not concentrate on the types of assistance AR could provide once mechanics entered the workpiece (psychomotor) portion of the task, discussed in Section 2.6. Based on our observation of the LAV-25A1 mechanics, we believe the psychomotor phase presents opportunities for AR interfaces to provide novel forms of procedural task assistance.

These opportunities stem from the ability of AR to address two limitations of current documentation systems. The first of these limitations concerns how instructions are synchronized with ongoing user activity. When we reviewed current documentation systems, we noticed most solutions adopted an asynchronous and serialized approach to delivering assistance. In extreme cases, instructions are often intended to be consumed and internalized in their entirety prior to starting a task (e.g. “Please review the operator’s manual prior to use”). In other cases, documentation is intended to accompany a worker as they complete a task. However, if the worker needs to revisit the documentation, she must interrupt the current activity and find, browse, review, and understand the instructions. She must then resume the previous activity while simultaneously integrating and applying new or refreshed information. These interruptions even occur with step-by-step instructions. The advantage of these documentation methods is that interruptions are
structured and help a person to achieve a goal incrementally. This aligns with the way human beings conceive complex, continuous tasks as discrete sequences [Zacks and Tversky 2003]. However, people must still reference instructions separately and asynchronously (i.e., a person cannot focus on the instructions and the task simultaneously). Additionally, workers must consume documentation describing these incremental steps serially (i.e., step-by-step instructions rarely encourage parallelization of activities)

The second limitation of current procedural task documentation is driven by the need for all forms of assistance to make certain assumptions about the task actor and the task environment, the latter of which we define as all physical objects required to complete the task (e.g., subcomponents and tools) and any pertinent objects located in the physical world (e.g., a workbench). These assumptions limit the types of assistance presented and are often the result of limits in technology. For example, paper systems assume a person has correctly paired the documentation with the task, that the documentation is current, and that a person can perceive and comprehend the documentation (i.e., knows how to read, possesses any required assistive technology, etc.). Paper documentation cannot actively query a user to verify progress or gauge understanding and cannot respond to questions. Any pictures and diagrams included with the documentation are drawn in static poses, often emphasizing visibility of components at the expense of practicality. More advanced forms of instruction, such as the IETMs discussed in Chapter 3, also make assumptions. Most systems do not track the user activity relative to the task (e.g., detecting when a mechanic has removed a component from an assembly) and rely on deliberate user input to determine when subordinate steps are achieved. While some advanced systems discussed below in Section 4.1 do track the user and one or more task objects, none use this information to provide continuous, dynamic, and prescriptive assistance.
Our AR prototype demonstrated in Chapter 3 also posed challenges with asynchronous delivery of serialized instructions and makes its share of assumptions. Mechanics were only presented with one task at a time, and incurred task interruptions when interacting with the wrist-worn controller. The system assumed a fixed model of the task environment, and had no knowledge of how the mechanic or other influences altered this model. This was especially problematic in the psychomotor phases of our study which, by definition, involved activities that alter the task environment. In these phases, our system provided minimal assistance, and the tasks were trivial in nature given our population of school-trained mechanics. Furthermore, the low resolution of the VST HWD we used in the study made it difficult to perform relatively simple tasks such as inserting a screwdriver in a slot.

In this chapter, we will explore how AR interfaces can overcome these challenges and deliver novel forms of assistance in the psychomotor phase by presenting synchronized instructions that promote and support parallel activities while continually updating assumptions about the user and task environment. Our exploration will result in two aspects of this chapter’s contribution to the dissertation. First, we present an experimental AR application applied to an assembly task in a manufacturing and maintenance domain. It leverages 6DOF tracking information from both the user and domain objects to provide dynamic assistance on a see-through head-worn display throughout the psychomotor phase of the task. Second, we report on a user study showing that this AR documentation allows participants to complete a physical alignment task more quickly and more accurately than when using 3D-graphics-based documentation displayed on a stationary LCD screen. Qualitative results from this physical alignment task indicate that a majority of participants preferred the AR system, and ranked it as more intuitive.
4.1 Related Work

In Chapter 3, we reviewed previous work involving the use of AR for documenting and assisting with various procedural tasks. In this section, we build upon this review, and focus on work related to providing assistance in the psychomotor phase of procedural tasks. Before proceeding, it is important to note that any AR system providing instructions to a user could potentially impact a future psychomotor activity. For example, mechanics tasked with removing a screw in our LAV-25A1 study (Chapter 3) and presented first with an animated model of a screwdriver might have benefited from this animation when they eventually used the tool. While we acknowledge the crossover of assistance presented in the informational phase into the workpiece phase, in this chapter we are focused on forms of assistance presented during ongoing psychomotor activities. This delineation is readily apparent when examining the implications for tracking requirements. Substantial forms of assistance presented during psychomotor activities require tracking the user and one or more movable objects in the environment. Without both forms of tracking, an AR application is unable to render dynamic information about the task environment that takes into account the positions and orientations of the objects being manipulated, and is limited to rendering information based on assumptions.

Of the many AR systems cited in Chapter 3, all of them track the position and orientation of the user’s head relative to the task environment. However, relatively few of these systems, which we further discuss here, also track one or more movable objects in the environment, as we do. Several assembly [Salonen and Sääski 2008; Zauner et al. 2003] and maintenance [Feiner, MacIntyre, and Seligmann 1993] applications track task objects, but beyond overlaying models of the tracked objects and accompanying precomputed instructions (e.g., directional arrows [Feiner, MacIntyre, and Seligmann 1993]), they use this information solely at the start and end of
psychomotor activities to detect task transitions and verify correct alignment. In contrast, we provide continuous prescriptive feedback for alignment tasks that dynamically reflects the user’s interactions. The needle biopsy systems [Rosenthal et al. 2002; State et al. 1996; Wacker et al. 2006] display virtual representations of tracked biopsy needles within a (simulated) patient’s body. These systems differ from ours in that they intentionally do not provide explicit instructions to the user, but instead rely on the skilled surgeon to make her own decisions based on the AR visualization. Blum, Sielhorst, and Navab [2007] use AR to depict an expert obstetrician’s complex prerecorded actions to a trainee attempting to emulate the expert’s behavior. However, this system was designed as an offline learning tool where AR is used after the psychomotor activity, although the authors indicate plans to implement an online version of the system. Finally we are not aware of any quantitative user studies of either version.

Blum and colleagues’ idea of a student using AR to match an expert’s performance is reminiscent of other systems that invoke AR technology during remote collaboration. These systems provide a form of assistance during psychomotor activities when a local user is guided by a remote expert. Notable work includes a non-AR remote application for bicycle repair proposed by Kraut. Miller and Siegel [1996], an AR-enabled electronics training system demonstrated by Boulanger [2004], and several systems examining the communication of gestures using AR [Fussell et al. 2004; Kirk and Stanton Fraser 2006; Li et al. 2007a]. These systems differ from ours in that we are seeking to automate locally, in real-time, the assistance similar to rendered by a remote expert.
4.2 Selecting a Psychomotor Task Environment

To fully explore the application of AR to psychomotor activities, we needed to identify an appropriate task environment. Ideally, this task environment would involve a realistic maintenance and repair task, include objects that could be easily tracked, and provide enough repetition to support a thorough experiment. Most importantly, a significant portion of the selected task should require psychomotor activities amenable to the use of AR.

We reviewed several procedural task taxonomies, summarized in Chapter 2, to gain a better understanding of potential sources of psychomotor activity. Our review included several works focused on tasks in the maintenance and repair domain, such as Drury and colleagues’ functional decomposition [1990] and Vujosevic and Ianni’s motion models [1997]. However we found these systems lacked sufficient detail, and instead adopted the civil engineering taxonomy proposed by Guo and Tucker [1996]. This taxonomy consists of 42 generic tasks and was first proposed to explore opportunities to apply automation in construction projects. We found the taxonomy provided an ideal level of detail and represented activities also found in the maintenance and repair domain. The taxonomy does not include a quantification of the degree of psychomotor activity required in each task, but rather classifies each general task as “a repetitive single action or movement.” To determine which of the 42 tasks provided a sufficient challenge, we estimated the psychomotor activity required for each task in the taxonomy, and scored each on a five-point scale, where a score of 5 represents a high level of psychomotor activity. These estimations are based on our experience observing or performing each type of task, and could be strengthened with a formal study as discussed in Chapter 6.

This annotated taxonomy is depicted in Table 4.1 and illuminates possible sources of psychomotor activity. Moreover, the tasks in the table present opportunities to develop new
forms of assistance delivered using AR. While AR interfaces have already been applied to some tasks (i.e., Identify, Inspect, and Position), most are unexplored. Ideally, as we propose in Chapter 6, each task in the taxonomy would be supported by one or more AR interface techniques proven to assist in performance of the task. In this chapter, we demonstrate this vision with the Align task.
<table>
<thead>
<tr>
<th>Task</th>
<th>Description*</th>
<th>Estimated Psychomotor Activity Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrange</td>
<td>Put a number of objects in a proper order</td>
<td>3</td>
</tr>
<tr>
<td>Align</td>
<td>Keep objects in a straight line or orientation</td>
<td>5</td>
</tr>
<tr>
<td>Bend</td>
<td>Deform the shape of an object</td>
<td>4</td>
</tr>
<tr>
<td>Caulk</td>
<td>Inject liquid between two adjacent objects*</td>
<td>2</td>
</tr>
<tr>
<td>Clean</td>
<td>Remove unwanted dirt, material or impurities</td>
<td>2</td>
</tr>
<tr>
<td>Coat</td>
<td>Apply a layer of liquid on an object’s surface</td>
<td>2</td>
</tr>
<tr>
<td>Communicate</td>
<td>Talk or use hand signal to transfer information</td>
<td>1</td>
</tr>
<tr>
<td>Compact</td>
<td>Condense material*</td>
<td>2</td>
</tr>
<tr>
<td>Connect</td>
<td>Join or fasten two objects to each other</td>
<td>5</td>
</tr>
<tr>
<td>Cover</td>
<td>Unroll sheet material on an object’s surface</td>
<td>3</td>
</tr>
<tr>
<td>Cut</td>
<td>Divide one object into two or more pieces</td>
<td>3</td>
</tr>
<tr>
<td>Disconnect</td>
<td>Break connections between two objects</td>
<td>4</td>
</tr>
<tr>
<td>Dismantle</td>
<td>Demolish, break down, uninstall</td>
<td>3</td>
</tr>
<tr>
<td>Drill</td>
<td>Make a hole by rotation</td>
<td>4</td>
</tr>
<tr>
<td>Empty*</td>
<td>Remove objects/materials inside another object*</td>
<td>2</td>
</tr>
<tr>
<td>Fill</td>
<td>Place objects/materials inside another object*</td>
<td>2</td>
</tr>
<tr>
<td>Finish</td>
<td>Apply mechanical treatment to a surface</td>
<td>2</td>
</tr>
<tr>
<td>Hit</td>
<td>Strike hardly to push an object</td>
<td>2</td>
</tr>
<tr>
<td>Hold</td>
<td>Keep an object in a position temporarily</td>
<td>2</td>
</tr>
<tr>
<td>Identify</td>
<td>Recognize an appropriate member</td>
<td>1</td>
</tr>
<tr>
<td>Inlay</td>
<td>Set small flat pieces next to each other</td>
<td>2</td>
</tr>
<tr>
<td>Insert</td>
<td>Push an object into another one</td>
<td>2</td>
</tr>
<tr>
<td>Inspect</td>
<td>Examine flaws or verify correctness</td>
<td>1</td>
</tr>
<tr>
<td>Install</td>
<td>Put an object into final position</td>
<td>5</td>
</tr>
<tr>
<td>Level</td>
<td>Keep material on a horizontal plane</td>
<td>4</td>
</tr>
<tr>
<td>Lift</td>
<td>Move an object upward for transporting</td>
<td>1</td>
</tr>
<tr>
<td>Lay</td>
<td>Set objects next to or on top of each other</td>
<td>2</td>
</tr>
<tr>
<td>Measure</td>
<td>Determine or layout correct dimensions</td>
<td>4</td>
</tr>
<tr>
<td>Operate</td>
<td>Control an equipment for work</td>
<td>2</td>
</tr>
<tr>
<td>Position</td>
<td>Move an object to the correct location</td>
<td>5</td>
</tr>
<tr>
<td>Pour</td>
<td>Move liquid between two objects*</td>
<td>3</td>
</tr>
<tr>
<td>Prepare</td>
<td>Make material ready for future use</td>
<td>2</td>
</tr>
<tr>
<td>Pull</td>
<td>Draw cable or wire through channel*</td>
<td>3</td>
</tr>
<tr>
<td>Pump</td>
<td>Transport material by air pressure</td>
<td>2</td>
</tr>
<tr>
<td>Roll</td>
<td>Move an object on wheels along a surface</td>
<td>3</td>
</tr>
<tr>
<td>Shape</td>
<td>Modify the shape of an object to fit in position</td>
<td>5</td>
</tr>
<tr>
<td>Spray</td>
<td>Project liquid or particles without contact*</td>
<td>3</td>
</tr>
<tr>
<td>Spread</td>
<td>Apply semi-liquid material to locations</td>
<td>3</td>
</tr>
<tr>
<td>Tap</td>
<td>Strike or touch an object gently</td>
<td>2</td>
</tr>
<tr>
<td>Transport</td>
<td>Move material to designated location</td>
<td>5</td>
</tr>
<tr>
<td>Vibrate</td>
<td>Shake or tremble to consolidate material</td>
<td>3</td>
</tr>
<tr>
<td>Write</td>
<td>Make notes or marks to indicate purpose</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.1: Guo and Tucker’s [1996] taxonomy of generic tasks with scored values depicting our own estimation of the level of psychomotor activity required in each task. An asterisk (*) indicates text we reworded to generalize the original taxonomy to a broader domain of tasks.
We scanned the tasks listed in Table 4.1 and looked for activities that aligned with procedural tasks involving a Rolls Royce Dart 510 turboprop engine located in our lab (Figure 4.2a), which we selected as an experiment domain. We identified five general activities for exploration—Dismantle, Position, Align, Install, and Connect—all of which involved the engine combustion chambers (Figure 4.2b). The seven combustion chambers are mounted on the aft section of the engine, as shown in Figure 4.2(a), and can be accessed relatively easily. Although they all appear similar at first glance, each chamber is different, and contains a unique arrangement of intake and exhaust ports, in addition to minor ports to power ancillary systems on the engine (e.g., deicing). Some of these features are depicted in Figure 4.2(b–d). The entire set of combustion chambers operates as a single, interconnected thermodynamic system, requiring unique placement of each chamber. We felt these characteristics made tasks involving the combustion chambers excellent examples of potential beneficiaries of an AR interface.

Our original vision for a prototype involved assisting mechanics with installing and removing the combustion chambers. Our experience revealed that these tasks feature significant amounts of psychomotor activity. For example, during a typical installation, each chamber must be carefully turned, pushed, pulled and routed around obstructions. Successful installation requires the precise alignment of each combustion chamber with adjoining chambers and components of the engine. Hoses and wires leading to ancillary systems also need to be reconnected. Removing the combustion chambers poses a similar set of challenges. We hypothesized that an AR interface could assuage these challenges by providing dynamic feedback to the mechanic shortening the time required to install and remove the combustion chambers.
Figure 4.2: Procedural task environment. (a) The Dart 510 engine. (b) Combustion chamber. (c) Upper combustion chamber “cone.” (d) Lower combustion chamber “can.” (Small retroreflective spheres bolted to components are used for optical tracking.)
We implemented our initial vision for the prototype, which is depicted in two variants in Figures 4.3 and 4.4. This prototype provided instructions assisting with the installation and removal of the combustion chambers to and from the Dart aircraft engine. The installation procedure is depicted in Figure 4.4. Unfortunately, during pilot testing, we realized the selected tasks of uninstalling and installing the relatively small number of combustion chambers would not provide enough statistical power to support a robust user study. However, this initial prototype served two important purposes. First, we used the prototype to implement and test the HWD calibration routine described in Section A.7.5. Second, we perfected the use of axial aligned arrows to help guide rotation of various components. Third, we finalized the integration of various tracking technologies and implemented algorithms to fuse their data.

We next examined, and eventually decided to use, the task of assembling an individual combustion chamber. Assembling a chamber requires attaching the chamber’s truncated conical upper section (shown in Figure 4.2c, which we will refer to as a “cone”) to its mostly cylindrical lower section (shown in Figure 4.2d, hereafter referred to as a “can”). Each cone and can has a flange circled by a set of 20 evenly spaced holes, which can be seen in the figure. When installed on a Dart 510 engine, a combustion chamber assembly must satisfy two constraints. First, because each chamber is unique, a valid assembly requires a distinct pairing of a cone with a matching can. Second, the rotation between a combustion chamber can and cone must be correct if the combustion chamber is to fit on the engine and interface with surrounding systems. As we describe in Section 4.4.1, we relax these two constraints to facilitate our design of experiment performed in the user study. This study only involves assembly of the combustion chambers, but not their installation on a Dart 510.
Figure 4.3: An initial prototype for removing and installing combustion chambers (1 of 2). An animated 3D arrow indicates the direction of rotation to achieve a target alignment about the long axis of the chamber. (The image was captured by a video camera mounted inside and looking through the optical–see-through HWD. A post-render filter was applied to remove camera distortion and vignetting.)
Figure 4.4: An initial prototype for removing and installing combustion chambers (2 of 2). Localization information (a) orients the user to the combustion chamber. As the user picks up the chamber, (b) an animated 3D arrow indicates the direction of rotation to achieve longitudinal alignment. As the user turns toward the engine, (c–d) a 3D label is used to suggest the correct lateral orientation. As the user nears the engine with the chamber, (e) a semi-transparent virtual version of the chamber indicates the correct location and orientation for (f) the user to emulate. (Images show view through a Vuzix VR920 video see-through HWD with attached CamAR camera which is used here to promote clarity in the images. An optical-see-through HWD is used in the user study.)
4.3 Psychomotor Phase Assistance

We experimented with the following techniques for assisting the worker in the psychomotor phase of procedural tasks:

- Dynamic 3D arrows. We designed and tested several tracked 3D animated arrows that are rendered over or near movable objects to suggest a certain motion or provide feedback about the current orientation of the movable objects compared to a desired end state. These arrows alter their size, color, animated direction, or visibility in response to user activity. Examples are shown in Figure 4.5.

- Dynamic 3D highlights for connection points. We implemented a series of color-coded, semitransparent highlight effects that are designed to help the user when connecting and/or aligning two rigid bodies in a procedure (e.g., connecting a combustion chamber cone with its can). As the rigid bodies are brought into alignment, the matching color-coded highlight on the receiving component alters its transparency until the connection highlights appear as a single entity. Figure 4.6 depicts an example of this behavior.
Figure 4.5: A large, red, dynamic arrow (a) indicates the direction and magnitude of the motion required to bring the can and cone into alignment. As the can and cone approach alignment (b–c), the arrow reduces size and changes color through yellow to green. If the motion overshoots the alignment, (d) the arrow changes direction to indicate the required correction and (e) slowly fades out when (f) alignment is achieved. (Images show view through a Vuzix VR920 video see-through HWD with attached CamAR camera which is used here to promote clarity in the images. An optical–see-through HWD is used in the user study.)
• Dynamic billboarded labels. We extended the static billboarded labels used in information activities to respond to tracking information collected about the user and the movable object. This includes updating a dynamic occlusion model of any movable objects, as shown in the partial occlusion of can (lower) label 17 by the combustion chamber components in Figure 4.5(b). We also experimented with altering the visibility of the billboarded labels in response to ongoing user activity as a form of feedback. For example, temporarily changing the color of label J in Figure 4.5(e–f) to yellow to indicate achievement of alignment. Another idea involved slowly fading the labels to full transparency once alignment is achieved.

• Our prototype also includes static motion paths that are rendered as Bézier curves depicting a stylized path between an object’s current location and its prescribed destination. Examples are depicted in Figure 4.10 (bottom) and Figure 4.12 (bottom). In our

Figure 4.6: Example of dynamic highlights. Prior to alignment (left), distinct highlights are rendered on the top and bottom holes (indicated by “Q” and “19” leader lines). When the holes are aligned (right), the bottom highlight slowly fades out presenting the appearance of a single hole. (Images show enlarged view through a video see-through display)
current implementation, these techniques rely on a fixed notion of the task environ-
ment, and are not altered in response to ongoing user activity. However, because we
are already tracking the cans and cone, we could readily update the endpoints and
shape specified by the motion path.

We implemented these techniques as part of the combustion chamber assembly procedural task, which we then evaluated with the user study described in Section 4.4. Our prototype displays text instructions in the 2D HUD of the HWD, instructing the user to locate and pick up the prescribed can. The localization technique we featured in Chapter 3 guides the user to the can’s current location. Virtual labels are provided to help the user identify the can and other objects in the task environment. When our prototype detects the user has secured (i.e., begun to move) the can, new text is displayed in the HUD, instructing the user to reposition the can to a prescribed assembly area on a workbench. The prototype also shifts localization cues to this assembly area and presents the virtual motion path leading to this target location. Figures 4.8–4.10 depict an example localization sequence.

After the user successfully places the can in the work area, the process is repeated to locate and reposition the appropriate cone for assembly. Once the user places the cone within 30 cm of the designated location, the prototype begins to display a dynamic 3D curved arrow, centered about the $y$ axis of the cone, representing the optimal direction of rotation to bring the cone and can into the desired alignment. As shown in Figure 4.5, the size and color of the arrow are varied to reflect the magnitude of the motion required to achieve the desired alignment (e.g., a long red arrow indicates a significant correction is required, while a short green arrow indicates a smaller correction is required).
During this psychomotor phase, our prototype also presents assistance in identifying the specific connection points (i.e., holes) on the can and cone where the worker will insert fasteners to secure the assembly after finalizing the alignment. In our task, we use pins as fasteners rather than bolts to save time during the assembly and because we do not consider the act of tightening a bolt a significant source of psychomotor activity (once learned). Virtual labels, redundantly encoded using color and shape, are rendered at locations registered with each connection point. Our prototype also presents dynamic highlights intended to help the worker identify these connection points and determine when they are aligned. When the alignment error between the cone and can is greater than a single hole, highlights are rendered over all four connection points. As depicted in Figure 4.6, when the angular difference between the current and target alignment is less than the size of a single hole, only a single circle highlight remains.

4.4 User Study

We designed a user study to compare the performance and general acceptance of our AR prototype with that of 3D-graphics–based documentation. Twenty-eight participants (7 female, 21 male; age 18–44, \( \bar{X} = 26 \)) were recruited by mass email to the Computer Science students at our university and by flyers distributed throughout campus, and were paid $15 each. Six of these participants served in a pilot test, described in Section 4.4.4, and did not participate in the later formal experiment. All but two participants used a computer multiple times per day. Three users reported having some experience repairing mechanical systems, 14 users reported having a basic level of exposure, and 11 users reported having no experience. Nine participants identified themselves as requiring contact lenses or glasses, and were accommodated by placing the nVisor ST60 optical–see-through HWD over their glasses. We screened for stereopsis by administering
the Stereo Optical Co. Stereo Fly Vision Test to each participant and found all 28 correctly perceived stereo stimuli.

4.4.1 Task

The primary procedural task in our study involves assembling a combustion chamber by aligning a can with a cone. We integrated this task into a workbench setting consisting of three cones and three cans, each positioned in one of six bins arrayed on the workbench, as shown in Figure 4.7. A portion of the workbench was set aside as the work area, where the participant is instructed to assemble a combustion chamber while standing in front of it. A small mechanical turntable was placed in this area to receive the can and facilitate rotation during assembly. We also placed a container of fastening pins, which were required for the task, in the corner of the workbench for easy access by the participant.
Figures 4.8–4.12 depict a typical trial in our experiment. Each trial consists of all the steps of a single assembly task involving one can and one cone. The task begins with the participant pressing the confirm button (Figure 4.8, top), then locating (Figure 4.8, bottom) and picking up a specified can from its bin (Figure 4.9, top). When the participant begins to move the can (Figure 4.9, bottom), the task controller detects the movement and instructs the participant to move the can to the turntable. After the participant successfully places the can on the turntable (as determined by comparing the tracked location of the can to that of the stationary turntable), the task controller instructs the participant to locate and pick up a specified cone from its bin (Figure 4.10, top). When the participant locates and picks up the cone, the system instructs them to place it on top of the can on the turntable (Figure 4.10, bottom). As the cone is placed on top,
the task controller instructs the participant to align two holes on the cone (labeled with letters) with two corresponding holes on the can (labeled with numbers) as shown in Figure 4.11. The same instructions also ask the participant to secure the aligned can and cone by placing one fastening pin (nail) through each of the two sets of holes and signal completion by pressing the “Confirm” button near the turntable (Figure 4.12, top). We used explicit user input, as opposed to detecting the alignment programmatically, to mark the end of this step, because the pins are not tracked by our prototype. After the participant signals completion of the align-and-pin step, the task controller instructs them to place the assembled combustion chamber in one of the bins (Figure 4.12, bottom).

We selected the combustion chamber assembly task because it could offer a large sample of independent, homogeneous tasks that were resistant to experiential effects. We achieved this by modifying the actual combustion chamber assembly procedure to allow arbitrary pairings and alignments of cans and cones across all combustion chambers. This allowed us to present our users with a large number of possible unique tasks, preventing memorization of distinct arrangements. (Note that the single “correct” assignment of properly aligned cones and cans is not visually obvious, would not be known by anyone who was not trained in the maintenance of this specific engine, and would only be important if the combustion chambers were to be installed in the engine.) As described above, to minimize time spent on the task and facilitate disassembling cone–can pairs for subsequent trials, we further modified the task to use two pins instead of the 20 machine bolts normally used to secure the assembly. We believe that these are valid modifications, since our participants had no prior knowledge of the engine, and our prototype was not intended to provide improved documentation for a set of 20 identical bolt-fastening tasks.
Figure 4.8: Experiment trial (1 of 5). (Top) When the participant initiates the trial by pressing the confirm button, the system loads the first step and (bottom) presents assistance in locating the can. (Images show views captured by a camera mounted inside the optical-see-through HWD. A post-render filter was applied to remove camera distortion and vignetting.)
Figure 4.9: Experiment trial (2 of 5). (Top) When the user picks up the can, the system senses this motion and (bottom) loads the next step instructing the user to place the can on the turntable. (Images show views captured by a camera mounted inside the optical–see-through HWD. A post-render filter was applied to remove camera distortion and vignetting.)
Figure 4.10: Experiment trial (3 of 5). (Top) After the can is placed on the turn table, the system instructs the participant to pick up a cone. (Bottom) The system then prompts for the cone to be placed on the can. (Images show views captured by a camera mounted inside the optical–see-through HWD. A post-render filter was applied to remove camera distortion and vignetting.)
Figure 4.11: Experiment trial (4 of 5). (Top) Dynamic alignment instructions are presented after the system detects the cone is atop the can. (Bottom) The participant then completes the alignment and inserts both pins. (Images show views captured by a camera mounted inside the optical–see-through HWD. A post-render filter was applied to remove camera distortion and vignetting.)
Figure 4.12: Experiment trial (5 of 5). (Top) When the participant finishes the alignment and pinning steps, they inform the system by pressing the red confirm button. (Bottom) The system then presents instructions for clearing the turntable. (Images show views captured by a camera mounted inside the optical–see-through HWD. A post-render filter was applied to remove camera distortion and vignetting.)
4.4.2 LCD Control Condition

In addition to the AR prototype described in Sections 4.3 and 4.4 (the AR condition), we created a control condition (herein referred to as LCD) that presents 3D-graphics–based documentation corresponding to material in our AR prototype on a 22" diagonal 1920×1080 LCD screen mounted near the repair area (the LCD condition). Because the documentation available for the Dart (which has been out of production for decades) is limited to printed materials, we developed a significantly enhanced version of the computer-based documentation currently employed by many professional mechanics and discussed in Section 2.5.2. We created an interactive 3D documentation system that employs previously proposed design principles for assembly instructions [Heiser et al. 2004], similar to that used in the work described in Chapter 2. It incorporates many aspects of the documentation used in the AR condition (e.g., text, instructions, labels, and motion paths), but presents them to the user on a fixed display without head tracking. Since this documentation is presented on an opaque display, and not overlaid on the task domain, we render it in conjunction with 3D virtual models corresponding to salient physical objects in the task environment that provide important cues when participants view them directly in the real world in the AR condition. These include a 3D model of the workbench and detailed 3D models of the cans and cones, created from 3D laser scans of the actual components. During localization and movement, this content is used to generate static perspective views rendered from camera poses corresponding to the user’s location at each stage of the task. For the alignment and pinning task, we generate additional close-up views designed to assist the user in identifying the prescribed attachment and pinning points. Figure 4.13 shows a participant performing pinning in the LCD condition and Figure 4.14 depicts a screen-captured example display from a similar task. Finally,
when experiencing the LCD conditions, participants wear a tracked headband used to record head tracking data for analyzing performance.

Figure 4.13: A user pins the cone to the can in the LCD condition. (Used with permission of the participant.)
4.4.3 Experiment Design

We designed a within-subject, repeated measures experiment consisting of two conditions (AR, LCD) and randomized iterations of the combustion chamber assembly task. The experiment was blocked by condition, with a five-minute break between the AR and LCD blocks. Each of these two blocks consisted of a set of trials whose number was established through experimentation during a pilot study, as detailed in Section 4.4.4. Block order was counterbalanced across both conditions to mitigate experiential effects. Each trial was defined as the ordered exe-

Figure 4.14: A screen capture of the documentation provided in our LCD condition.
cution of six steps required for successful assembly of a combustion chamber. We categorized these steps, which are listed in Table 4.2, according to the predominant human activity employed during that step. This categorization, which we created by adapting the taxonomy of major activity types proposed by Gilbreth and Gilbreth [1924] to our task domain, was useful in isolating the steps within the larger procedural task involving psychomotor human abilities. As shown in Table 4.2 (and illustrated in Section 4.4.1), each trial consisted of assembling one of three cones with one of three cans, aligning the cone and can correctly, and inserting two pins through matched pairs of holes on the components. The assignment of cans to cones and the hole pairings were fixed across conditions and participants (i.e., all participants experienced the same combinations in both conditions), and were generated pseudorandomly prior to the experiment.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Activity Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Locate Can $X$ in Bin $W$</td>
<td>Locate</td>
</tr>
<tr>
<td>2</td>
<td>Move Can $X$ to Turntable</td>
<td>Position</td>
</tr>
<tr>
<td>3</td>
<td>Locate Cone $Y$ in Bin $V$</td>
<td>Locate</td>
</tr>
<tr>
<td>4</td>
<td>Place Cone $Y$ on Can $X$</td>
<td>Position</td>
</tr>
<tr>
<td>5</td>
<td>Align Cone $Y$ with Can $X$; Insert pins</td>
<td>Align &amp; Pin</td>
</tr>
<tr>
<td>6</td>
<td>Move assembly $XY$ to Bin $Z$</td>
<td>Position</td>
</tr>
</tbody>
</table>

Table 4.2: Steps of combustion chamber assembly task.

Prior to starting the experiment, each participant was asked to sign a consent form and then watch an instructional video corresponding to their assigned starting condition. Following the video, participants starting with the AR condition were asked to perform the Stereo Fly test and then were assisted with donning the nVisor ST60 optical–see-through HWD. Participants starting with the LCD condition were asked to wear a small, lightweight crown affixed with OptiTrack markers, which was used to collect head movement data. Participants were then given
a short rehearsal period involving five trials to become comfortable with their starting condition. The participant then began the timed portion of the first block, which started with the participant pressing a large button on the workbench near the turntable (Figure 4.7). The participant then performed each of the steps listed in Table 4.2. The completion times for steps 1–4 and step 6 were logged automatically based on state-machine transitions triggered in response to user activity, as described in Section 4.4.1. The completion time for step 5 was measured when the participant pressed the button near the turntable to confirm completion of the assembly. Prior to transitioning to step 6, the system also calculated the mean alignment error between the can and cone by sampling the rotational difference (yaw) between the can and cone, and comparing this sample mean to the angular difference specified by an optimal alignment.

The block proceeded to the next trial after the participant placed the completed assembly in a designated bin at the conclusion of step 6. Because we used only three combustion chambers in the experiment design, some cone and can recycling was required to support multiple trials. This was accomplished by inserting a disassembly task at certain points in the block, which involved the participant locating one of the completed combustion chambers, moving it to the turntable, removing the pins, and then placing the can and cone back into the bins. Following each block, the participant was afforded a 5 minute break in which they were asked to review an instructional video for the subsequent block. No head-worn display or head-bands were worn during the break. After the break was completed, the participant was asked to commence the remaining block. At the completion of both blocks, the participant was asked to complete a post-experiment questionnaire about their experiences with and impressions of the AR and LCD documentation systems.
4.4.4 Pilot Testing

The first six participants from our recruited population participated in a pilot study designed to test our experiment, elicit feedback about our conditions, and form hypotheses. (These subjects did not participate in the formal study.) In this pilot study, each block contained 18 trials. We applied a 2 (Display Condition) × 3 (Activity Type) repeated measures ANOVA on the mean completion time for each of the steps in Table 4.2 and found a significant main effect of display condition on completion time ($F_{(1,5)} = 17.14, p = 0.009$). A similar analysis found a significant main effect of display condition on accuracy of alignment between the can and cone ($F_{(1,5)} = 11.51, p = 0.019$). An analysis of errors accumulated during localization and positioning activities showed participants made very few errors during these activities in either condition. The pilot study also revealed strong user preference for AR compared to LCD, with all six participants ranking AR as more preferred, as well as the more intuitive of the two systems. Finally, our pilot test was also helpful in setting the number of trials in each experiment block. We originally intended for participants to perform 18 trials per condition, which would have allowed two pairings of every can with every cone. However, data collected during our pilot trials indicated this could take up to 90 minutes and we found no statistical evidence to suggest that cone and can combinations had an effect on our dependent variables. Therefore, we settled on 14 trials per condition for each of the 22 participants who would participate in our formal study, which worked out to an average of 50 minutes for the entire experiment, with each cone and can paired at least once.
4.4.5 Hypotheses

Based on our experience with the pilot, and prior to our experiment, we formed several hypotheses.

**H1:** AR would be the fastest technique during psychomotor activities.

**H2:** AR would be the most accurate technique during psychomotor activities.

**H3:** AR would be the most preferred technique.

**H4:** Participants would rank the AR technique as most intuitive.

These hypotheses were based on our belief that the dynamically tracked arrow, labels, and highlights would simplify the task of identifying and matching the attachment points on each cone and can. We expected participants to find the LCD condition more onerous because identifying the attachment points relies on matching salient features on the components.

4.5 Quantitative Results

We began our analysis by looking for potential outliers in our completion-time data. We identified suspicious values by examining Tukey plots of the completion times for each major activity type (Locate, Position, Align) across all trials and participants and cross-checked outlying values against videotaped footage of the participants performing the trials. This led us to establish the following ranges for valid completion times in each major activity: [0.25, 10] secs for Locate activities (both conditions), [0.25,10] secs for Position activities (both conditions), [0.25, 60] secs for AR Align activities, and [0.25,120] secs for LCD Align activities. We discovered that the predominant source of outliers were tracking errors produced when cones or cans were placed in particular arrangements relative to each other that occasionally formed a spurious constellation of retroreflective markers that the OptiTrack software mistakenly identified as a known
rigid body. For example, when cans 1 and 2 are placed close to each other, and both are in a particular orientation, their combined visible configuration of marker may appear to the OptiTrack system as an alias of can 3. This either caused the system to cue the next state prematurely or resulted in the participant stopping until the problem was corrected. We preprocessed our data by removing the outliers we identified, which accounted for 1.49% (1.62% AR; 1.36% LCD) of all calculated completion times across 3 activity types × 14 trials × 2 conditions.

4.5.1 Completion Time Analysis

We performed a 2 (Display Condition) × 3 (Activity Type) repeated measures ANOVA on mean completion times, with our participants as the random variable. Display condition exhibited a significant main effect on completion time ($F_{(1,21)} = 37.09, p < 0.001$). The global mean completion time for all activity types was 9.38 secs for AR and 16.36 secs for LCD. A post-hoc comparison with Bonferroni correction ($\alpha=0.0125$) revealed that AR was 7.01 seconds faster than LCD, which was significant ($p < 0.001$). Pairwise comparisons of mean completion times for each of the three predominant activity types defined in Table 4.2 are summarized in Table 4.3 and are depicted in the Tukey plots in Figure 4.15. As expected, only the aligning and pinning activity (step 5) exhibited a statistically significant difference in means, where the mean completion time for AR was 21.31 seconds (46.79%) faster than that of LCD.
### Table 4.3: Pairwise comparisons of mean completion time by activity type.

<table>
<thead>
<tr>
<th>Activity Type</th>
<th>AR (secs)</th>
<th>LCD (secs)</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locate (Steps 1 &amp; 3)</td>
<td>2.66</td>
<td>2.39</td>
<td>LCD 0.27 secs faster than AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( t_{(21)} = 1.60, p = 0.124 )</td>
</tr>
<tr>
<td>Position (Steps 2, 4, 6)</td>
<td>1.15</td>
<td>1.15</td>
<td>LCD 0.01 secs faster than AR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( t_{(21)} = 0.121, p = 0.905 )</td>
</tr>
<tr>
<td>Align &amp; Pin (Step 5)</td>
<td>24.23</td>
<td>45.55</td>
<td>AR 21.31 secs faster than LCD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( t_{(21)} = 6.27, p &lt; 0.001 )</td>
</tr>
</tbody>
</table>
Figure 4.15: Activity completion times (secs) for AR and LCD. An asterisk marks the mean task completion for each condition. Red lines inside each box represent median values. The edges of each box represent the 25th and 75th percentiles, and whiskers extend to the most extreme data points not considered outliers. Points are drawn as outliers if they are larger than $q_3 + 1.5(q_3 - q_1)$ or smaller than $q_1 - 1.5(q_3 - q_1)$, where $q_1$ and $q_3$ are the 25th and 75th percentiles, respectively.
4.5.2 Accuracy Analysis

We began our analysis of accuracy by looking for potential outliers in our alignment data. We identified those cases where alignment at the end of the trial exceeded 1.57 radians (5 inter-hole widths). We then physically checked each instance by reviewing video footage side-by-side with a virtual recreation of the trial. In five of the cases (1.6%), we discovered tracking errors caused an erroneous alignment calculation at the end of the trial. These instances were excluded from further analysis.

We performed a two (Display Condition) repeated measure ANOVA on mean alignment error measured during the Align activity, as defined in Section 4.4.1, with our participants as the random variable. Display condition exhibited a significant main effect on alignment error ($F_{(1,21)} = 48.754, p < 0.001$). The mean difference between the optimal orientation and that achieved by the user was 0.08 radians for AR (0.25 inter-hole widths) and 0.36 radians (1.15 inter-hole widths) for LCD and is depicted in the Tukey plot in Figure 4.16. A post-hoc comparison with Bonferroni correction ($\alpha=0.0125$) revealed AR was 0.28 radians more accurate than LCD, which was significant ($p < 0.001$).

We also performed a binary accuracy check by counting the number of correct alignments between the can and cone at completion of the task. We defined a correct alignment as a displacement within 0.16 radians (0.5 inter-hole widths) as measured when the participant pressed the confirm button. A McNemar’s chi-squared test with Bonferroni correction ($\alpha=0.0125$) revealed that there was a significant difference ($\chi^2_{(1,N=303)} = 87.94, p < 0.001$) between the binary accuracy rate achieved when experiencing the AR condition and the binary ac-
accuracy rate achieved when experiencing the LCD condition. The mean accuracy rate under the AR condition was 95.3% compared to 61.7% under the LCD condition.

Figure 4.16: Mean alignment error (radians) for the alignment activity in AR and LCDs. An asterisk marks the mean alignment error for each condition. This is a single Tukey plot. The edges of each box represent the 25th and 75th percentiles, and whiskers extend to the most extreme data points not considered outliers. Points are drawn as outliers if they are larger than $q_3 + 1.5(q_3 - q_1)$ or smaller than $q_1 - 1.5(q_3 - q_1)$, where $q_1$ and $q_3$ are the 25th and 75th percentiles, respectively.
4.6 Qualitative Results

Participants filled out a post-hoc questionnaire following their experience with both conditions. The questionnaire consisted of five-point Likert-scale questions (1 = most negative, 5 = most positive) to evaluate ease of use, satisfaction level, and intuitiveness for form of assistance. The results for the 22 participants who experienced AR and LCD are summarized in Figure 4.17. Friedman tests revealed significant rankings in the case of ease of use ($\chi^2_{(22,1)}=8.00, p = 0.005$), satisfaction ($\chi^2_{(22,1)}=11.84, p = 0.001$), and intuitiveness ($\chi^2_{(22,1)}=9.80, p = 0.002$). Subsequent pair-wise Wilcoxon comparisons of AR and LCD revealed AR was ranked significantly better than LCD in terms of ease of use ($p=0.007$), satisfaction ($p=0.005$), and intuitiveness ($p=0.012$).

When asked to rank the two forms of documentation based on preference for use, 20 of 22 participants ranked AR first. A Friedman test indicated this was significant ($\chi^2_{(22,1)} = 11.64, p = 0.001$). When asked which form of documentation was the most intuitive, 19 of 22 participants ranked AR first. A Friedman test indicated this was significant ($\chi^2_{(22,1)}=14.73, p < 0.001$). That participants overwhelmingly preferred AR was especially encouraging, given that the HWD used in the AR condition (Figure 1, left) weighs 1.3Kg (not including the added tracking hardware), and was relatively bulky, in comparison with the lightweight crown used in the LCD condition which weighs 205g.
Figure 4.17: Survey response histograms by condition for ease of use (top), satisfaction (middle), and intuitiveness (bottom). Median values for each condition are displayed as triangles.
4.7 Comparison to Physical Labels

Following up on our experiment, we were interested in how AR would compare to documentation that was physically embedded in the task; for example, by physically labeling or otherwise modifying all components to clearly disambiguate them from each other and clearly distinguish the different ways that the components might be configured. This is a stated goal (although one that is often not achieved) for many manufacturers that design products, such as furniture and toys, designed to be assembled by consumers. We considered implementing this as our original study baseline, but felt it was not ecologically valid for the engine combustion chambers and many other objects that may already have been designed and cannot be modified, that may have shapes and surface treatments dictated by other concerns, or that may be routinely subjected to extreme conditions that would damage or obscure superficial documentation. However, we decided that a pilot-study comparing AR to such a baseline would be useful for situating our results relative to a broader range of task domains.

To accomplish this, we created a modified version of the LCD condition in which we printed and glued small physical labels to all possible connection points on each can and cone (the PRINTED condition). We also added virtual versions of these printed labels to the virtual models displayed on the LCD. Figure 4.18 depicts these modifications to the actual cone and can and to the displayed graphics.
Figure 4.18: (Top) Components labeled for the PRINTED condition. (Bottom) Graphics displayed on the LCD screen.
We recruited eight additional participants, who experienced the same experiment design described in Section 4.4.3, with PRINTED substituted for LCD, and with the printed labels covered in AR. None of these eight recruited individuals had participated in the main experiment described in Section 4.4. One participant failed to perceive stereo during our Stereo Fly test, and we excluded their data from our analysis. We also excluded a second participant’s data after we noticed they failed to achieve the optimal alignment in all 14 trials of PRINTED. An analysis of this participant’s video suggests they were simply placing the cone on the can, and then inserting pins based solely on the cone’s labels (ignoring the can’s labels). For the remaining six participants (all male, age 19–27, $\bar{X}$=23.5), we performed a 2 (Display Condition) × 3 (Activity Type) repeated measures ANOVA on mean completion times, with the participants as the random variable. Display condition failed to show evidence of a significant main effect on completion time ($F_{(1,5)} = 0.67, p = 0.451$). The mean completion time for all activities was 7.87 seconds for AR and 7.29 seconds for PRINTED. The mean completion times for the Align activity (step 5 in Table 4.2) were 20.73 seconds for AR and 19.42 seconds for PRINTED, and a difference in these means was not significant at the $\alpha$=0.0125 level. Our analysis also revealed that display condition failed to show evidence of a significant main effect on accuracy ($F_{(1,5)} = 1.28, p = 0.31$). The mean angular error was 0.034 radians for AR and 0.065 radians for PRINTED, and a difference in these means was also not significant at the $\alpha$=0.0125 level.

The six participants experiencing the PRINTED and AR conditions filled out a post-hoc questionnaire following their experience with both conditions. The questionnaire consisted of five-point Likert-scale questions (1 = most negative, 5 = most positive) to evaluate ease of use, satisfaction level, and intuitiveness for form of assistance. The results are summarized in Figure 4.19. However, Freidman tests did not reveal significant rankings between the PRINTED and
AR in the case of ease of use ($\chi^2_{(6,1)}=0.33$, $p = 0.564$), satisfaction ($\chi^2_{(6,1)}=0.11$, $p = 0.956$), or intuitiveness ($\chi^2_{(6,1)}=2.00$, $p = 0.157$). When asked to rank the two forms of documentation based on preference for use, 5 of 6 participants ranked AR first. However, a Friedman test indicated this was not a significant ranking ($\chi^2_{(6,1)} = 2.667$, $p = 0.102$). When asked which form of documentation was the most intuitive, 5 of 6 participants ranked AR first. Similarly, a Friedman test indicated this was not a significant ranking ($\chi^2_{(6,1)} = 2.667$, $p = 0.102$).
Figure 4.19: Survey response histograms by condition (PRINTED vs. AR) for ease of use (top), satisfaction (middle), and intuitiveness (bottom). Median values for each condition are displayed as triangles.
4.8 Discussion

We presented an AR prototype for providing assistance during procedural tasks with an emphasis on applying AR to support psychomotor phases of these tasks. We applied our prototype to a realistic assembly task encountered in a manufacturing and maintenance domain, and ran a counterbalanced, within-subject, user study comparing the AR prototype with 3D-graphics–based documentation presented on a stationary display. The results of the experiment confirmed that AR was faster and more accurate for psychomotor phase activities, was overwhelmingly preferred by participants, and was considered to be more intuitive, despite the relatively bulky HWD that we used. A small, follow-on pilot study comparing our prototype to an idealized, but often impractical, form of documentation featuring physical labels affixed to components, revealed no statistically significant differences in speed or accuracy.
5 Opportunistic Controls

Figure 5.1: Opportunistic Controls in action. A user manipulates a virtual button to record the results of an inspection task while receiving haptic feedback from the raised geometry of the underlying housing of the engine being inspected. (Screen capture of the imagery presented to the user wearing a video see-through head-worn display)
As discussed in Chapter 1, our research on AR interfaces supporting procedural tasks has sought to identify effective interaction techniques to support these interfaces. Much of this research has focused on the potential role of haptics in procedural task interfaces. This focus was driven partly by a desire to promote interfaces that address the physical nature of procedural tasks and partly by evidence suggesting that haptics promote realism and spatial cognition [Quarles et al. 2008; 2008]. As we considered interface designs combining AR with haptics, we noticed two competing constraints in our observation of procedural tasks: many procedural tasks constrain a user’s head and hands while simultaneously limiting the user’s ability to employ various input devices external to the procedure.

To support these scenarios, we have developed a class of interaction techniques we call Opportunistic Controls, examples of which are shown in Figure 5.1. An Opportunistic Control (OC) represents a form of tangible user interface [Ishii and Ullmer 1997]. A tangible user interface is one in which users employ objects from the physical environment to manipulate digital information. These objects are typically pre-selected and deliberately integrated into the user interface design. In the case of OCs, the physical aspects of the interface are provided by leveraging naturally occurring, tactilely interesting, and otherwise unused affordances—properties of an object that determine how it can be used [Gibson 1986].

These affordances serve as tactile landmarks [Blaskó and Feiner 2004] that provide inherent passive-haptic feedback for hand gestures. Passive haptic feedback [Lindeman, Sibert, and Hahn 1999] is haptic feedback provided to a user by the inherent shape, texture, or other physical characteristic of an object without active system intervention. In an OC interface, this feedback is augmented by visual feedback provided by overlaid 3D widgets presented using AR. 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, a system designed for a mechanic servicing an engine might use fasteners, such as screws and bolts, located on individually serviced components to display documentation specific to each component. A rotating washer on the same component can be used to page through the documentation or select entries from a list. A grooved surface in the vicinity of the component, such as a 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.

In this chapter, we first review related work in Section 5.1. Then, in Section 5.2, we consider alternative user interfaces and contrast their use and applicability to that of an OC interface. In Section 5.3, we formally define OCs and then discuss their design in Section 5.4. Section 5.5 describes a user observation study that revealed important insights about how people think about and use OCs. In Section 5.6 we detail the design and implementation of a prototype OC interface that we evaluated in a second user study (Section 5.7) that showed OCs allowed people to complete task more quickly than when using an undifferentiated baseline. Finally, we conclude the chapter with a discuss in Section 5.8.
5.1 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, a form of feedback proposed by Brooks and colleagues [1990] in which active devices, typically using motors, create forces and torques as part of the user interface. Early examples include a prototype 3D tactile device demonstrated by Noll [1971] that allowed a user to feel aspects of virtual 3D objects. 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.”

Several groups have used tracked hand-held tablets with tracked fingers or stylus to provide a supportive mobile surface on which to operate 2D widgets in AR [Szalavari and Gervautz 1997] or VR [Lindeman, Sibert, and Hahn 1999]. Lindeman, Sibert, and Hahn [1999] referred to this as “passive haptics” or “passive-haptic feedback.” Later work by Insko [2008] 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. 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. Murray-Smith and colleagues [2008] demonstrated a handheld input device that featured 3D-printed passive haptic patterns tightly integrated with embedded sensors. Fails and Olsen [2002] introduced “light widgets” that used optically-tracked hand gestures made on everyday surfaces (e.g., the edge of a bed) to control household appliances. While this is an important forerunner of our work on OCs, light widgets do not present any visual feedback to the user (except in a separate application used during camera configuration), do not allow the widget’s underlying affordance to move, and do not emphasize the use of differentiated surfaces.

All of this work either uses simple naturally occurring surfaces [Weimer and Ganapathy 1989][Olsen, with Fails 2002] or introduces new objects into the environment, whether simple [Szalavari and Gervautz 1997] or more complex [Hinckley et al. 1994]. In contrast, we are interested in the opportunistic use of objects that not only already exist in a particular task domain, but whose possibly complex surface geometry provides affordances that lend themselves well to certain kinds of interactions. Thus, OCs apply the notion of 2D haptically discriminable widgets developed by Buxton and colleagues [1985] to generalize and extend the early use by Weimer and Ganapathy [1989] of a set of 3D widgets laid out on a single undifferentiated existing surface, without adding additional objects.

5.2 Alternative User Interfaces

Prior to designing OCs, we considered 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 they fail to satisfy constraints typical of many procedural tasks that restrict the user’s head and hands while simultaneously prohibiting the use of input devices external to the procedure. For safety reasons, some AR task domains (e.g., aviation maintenance) are not amenable to the introduction of objects that are not indigenous to the domain. These objects can cause damage when they are dropped or come into contact with certain surfaces. Moreover, if these external objects are forgotten, lost, or otherwise left behind, they can interfere with normal mechanical activity and cause catastrophic accidents. Even if mobile devices are made available, they may require a user to shift their hands and eyes away from a specific task. For example, some devices require that the user hold them in one hand (e.g., a Handykey Twiddler [Swartz 1992]) 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 in which 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.
5.3 Opportunistic Controls Definition

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

- $\tau$ 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.
- $\psi$ is a 3D widget satisfying the definition and design specifications provided by Conner and colleagues [1992]. Thus, each instance of $\psi$ consists of a virtual model representing the widget’s geometry and an augmented transition network (ATN) specifying the widget’s behavior.
- $\alpha$ is a function mapping the encapsulated virtual geometry of the widget ($\psi$) to the physical geometry of the affordance region ($\tau$). This function serves to dynamically register the 3D widget’s model at the correct location in $\tau$, 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.
- $\beta$ represents the functional mapping from the grammar of $\Gamma$ to the ATN of $\psi$ and defines how an individual widget responds to gesturing.
- $\rho$ is the 3D transformation required to map locations in the model space of $\Gamma$ to the model space of $\tau$. 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 contiguous physical region paired with a 3D widget, which together respond to one or more gestures.
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 [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, \( \tau \)). Continuing to follow 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, \( \psi \)) is paired to the physical system (the physical affordance region, \( \tau \)). When the user gestures on an OC, the 3D widget and physical affordance region respond and feel as one control.

5.4 Designing Opportunistic Control Interfaces

We followed a deliberate design process when creating our prototype OC interface. This design process involves three activities: affordance design, widget design, and gesture recognition design. We describe each of these activities in the following subsections. In creating our prototype, we executed these activities manually. However, to ensure the practicality of OC interfaces, more research is required to automate the authoring process. Such automated techniques might use real-time computer vision algorithms to extract interesting affordances and map them to predefined gestures and widgets supporting interface requirements.

5.4.1 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. 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 5.2. OCs based on these types of surfaces support bina-
ry 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 of Section 5.7). However, certain types of elastic surfaces might provide additional feedback about the state of the button.

Figure 5.2: Objects that could support button OCs. A rubber doorstop (left); an extruded handle on a chair (center); perforated metal supporting server ventilation.

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, cables, or natural surfaces, as shown in Figure 5.3. Gestures interacting with these types of surfaces require more precise tracking of the user’s hand and 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 the ridged surfaces designed by Murray-Smith and colleagues [2008] to provide haptic feedback).
The third kind of affordance we studied involves surfaces associated with movable objects in the environment. Examples include objects that slide (e.g., the clips on top of a chalk board shown in Figure 5.4, left), objects that bend (e.g., the rubberized tube shown in Figure 5.4, center), and objects that rotate (e.g., the disconnected wiring connector shown in Figure 5.4, right). These objects allow for richer controls whose underlying physical geometry ($\tau$) moves with the 3D widget ($\psi$) in response to the user’s gestures. However, in practice, movement of many these affordances may be assigned an a priori meaning, which can conflict with the OC’s functionality. This limits their potential use in many practical settings.
Throughout this exploration of the space of possible affordances, we adopted the following initial set of heuristic guidelines governing the selection of OCs:

- 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).

- Similarly, OCs should avoid desensitizing the user to a function of an overloaded object (e.g., using switches on a control panel for functions outside their design specification). This includes avoiding the use of objects with strong preconceived purposes and functionality that deviate from the purposes and functionality of the OC. As mentioned above, this guideline especially constrains movable OCs.

- 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 performing its designed purpose).
5.4.2 3D Widget Design

The design of each OC’s 3D widget ($\psi$) involves extending the ATNs proposed by Conner and colleagues [1992]. In an OC ATN, gestures from $\Gamma$ serve as the transitions between each widget state. Figure 5.5 depicts an example ATN for an OC interface with two buttons.

We experimented with several designs for 3D widgets ($\psi$) as part of our prototype development, as described in Section 5.6.1. We share two important lessons learned in these designs here as preliminary heuristics for designing OCs in general. First, 3D widgets accompany-

![Diagram of ATN](image)

<table>
<thead>
<tr>
<th>STATE</th>
<th>WIDGET A MODEL</th>
<th>WIDGET B MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Subdued (Blue) 3D &quot;A&quot; Button</td>
<td>Subdued (Blue) 3D &quot;B&quot; Button</td>
</tr>
<tr>
<td>Button A</td>
<td>Highlighted (Yellow) 3D &quot;A&quot; Button</td>
<td>Subdued (Blue) 3D &quot;B&quot; Button</td>
</tr>
<tr>
<td>Button B</td>
<td>Subdued (Blue) 3D &quot;A&quot; Button</td>
<td>Highlighted (Yellow) 3D &quot;B&quot; Button</td>
</tr>
</tbody>
</table>

Figure 5.5: An example ATN depicting the 3D widget design for a two-button OC interface. Note, the button regions are scanned sequentially implying only one button may be active at any given time.
ing OCs should feature 3D models that match the particular geometry of the OC. Second, increasing the transparency of the 3D models can be useful to allow users to partially view the OC’s underlying physical geometry. The transparency also allows the user to partially view any gestures that might be occluded by the 3D widget.

5.4.3 Gesture Recognition Design

Our gesture recognition algorithm is supported using the hardware design detailed in Section A.6.5.2. This algorithm 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 5.4.4, this relaxation speeds gesture recognition. Although the lack of depth information also restricts our grammar (Γ) to 2D gestures, we have found this sufficient for most interface requirements. 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 mapping from camera coordinates to physical interface coordinates 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, $\rho$ 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 $\rho$ to construct a 2D bounding box encapsulating each OC’s physical geometry (Figure 5.4, left). Adopting Kjeldsen’s approach [1996], 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) the geometry of each OC (Figure 5.6, right).

Figure 5.6: Unsegmented (left) and segmented (right) bounding boxes for a set of OCs as seen from an overhead camera. Graphics are added in debugging interface. (The user does not see the camera’s view)

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 fingertip 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 \( \rho^{-1} \) to translate the point \( p_h \) to the corresponding point \( t_h \) in the physical coordinates of the OC (\( \tau \)). The location of \( t_h \) is used to match each OC’s gesture (\( \Gamma \)).

The gesture parsing phase is accomplished with a finite state machine for each OC that resembles the ATN of the accompanying 3D widget (\( \psi \)). Each state in the finite state machine represents a command (e.g., “BUTTON_1_DOWN” or “SLIDER_2_UP”) in the shared OC grammar \( \Gamma \) and the ATN’s transitions are mapped to the OC gestures \( \gamma \). The gesture algorithm then uses the functional mapping of \( \beta \) to translate the current command to the appropriate state in the corresponding 3D widget \( \psi \). This final step ensures that the widget’s ATN is synchronized with the user’s gestures.

5.4.4 Design Limitations

Our design suffers from several limitations. First, it relies on an optical marker-based tracking scheme to compute the value of \( \rho \). 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 believe that it will be possible to build on recent advances in markerless or feature-based tracking [Bleser and Stricker 2008; Klein and Murray 2007] to replace our current use of markers. Second, our segmentation algorithm’s relaxation of depth information limits the type of interactions one can perform, specifically clutching and hovering. Third, our segmentation algorithm relies on controlled lighting conditions, limiting its practical use in settings outside the laboratory. More work is required to select and incorporate more robust segmentation algorithms into our gesture recognition process. 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.

5.5 OC User Observation Study

OCs enable a wide range of potential surfaces and objects to be used as interface artifacts. This equates to a broad application space for the design process described in Section 5.4, and poses several interesting questions:

- How do users perceive affordances not typically associated with user interfaces?
- What are the best techniques to redirect user thinking to view affordances as OCs?
- What heuristics determine the best affordances to fulfill OC interface requirements?
We designed a user observation study to help answer these questions. Our study rationale was to present participants with 3D interaction tasks that might be encountered when using an AR application within an environment containing a rich set of naturally occurring affordances. Participants would then create hypothetical OCs using any surface or object of their choosing, and we would observe the types of surfaces and corresponding gestures that participants selected to accomplish the assigned tasks.

Fifteen participants (11 male and 4 female), ages 19–35 ($\bar{X} = 25$), were recruited for this study from our university’s Computer Science student population, and were paid $15 each. All participants were frequent computer users, and seven reported experience using 3D interfaces.

5.5.1 Task

During the study, each participant was asked to design and demonstrate a notional OC-based user interface used to perform a series of common 3D interface tasks presented in sample VR and AR applications. These tasks are normally accomplished with common 3D widgets manipulated with traditional input devices. However, in our study, subjects selected any available affordance of their choosing and began gesturing to accomplish the particular task while using a “think out loud” protocol to verbalize expected system responses (e.g., “I’m moving the wiring harness to the left to select the wrench with the 3D cursor”). As they gestured, an observer who was out of direct view of the participant provided “Wizard of Oz” mouse and keyboard inputs to simulate this expected output in our sample applications. This simulated output provided basic visual feedback to the subject.

We used the following seven categories from the 3D widget taxonomy proposed by Dachselt and Hinz [2005] as the basis for the target interaction tasks presented to each user:
• **3D object selection.** Widgets used to manipulate a 3D cursor to select objects in a scene.

• **3D object manipulation.** Widgets used to rotate and translate a 3D object in a scene.

• **3D scene control.** Widgets used to control the position and orientation of a 3D scene’s camera.

• **2D document visualization.** Widgets used to pan and zoom 2D documents in 3D.

• **Discrete valuators.** Widgets modeling a single binary value (e.g., a button).

• **Continuous valuators.** Widgets modeling a continuous range of values (e.g., a slider).

• **Menu selection.** Widgets used to allow selection of items from a list.

We restricted the study to only these seven of the most common (based on our experience) members of the taxonomy’s fourteen widget types in order to limit the scope and duration of the study.

### 5.5.2 Procedure

Each participant experienced two application domains that we selected—performing maintenance on an aircraft engine (herein referred to as MA and depicted in Figure 5.7) and servicing a suite of home entertainment equipment (herein referred to as HE and depicted in Figure 5.8). We selected these particular domains because they are both rich in tactilely interesting affordances and present experiences that our participants would find unfamiliar (the MA domain) and familiar (the HE domain). For each application domain, participants were given individual tasks from our selected set of common 3D user interaction activities. Both the domain and task orderings were randomized, with the participant experiencing all seven tasks from one domain before proceeding to the next. The entire observation lasted approximately 45 minutes. Individ-
ual tasks were presented to the user as part of an untracked AR application, examples of which are shown in Figure 5.9, with the exception of the 2D document visualization task, which was displayed using the Cooliris [2011] application to render 2D images of simulated documentation (Figure 5.9e). Figure 5.9(a) shows an example of a participant receiving “Wizard of Oz feedback” from a 3D cursor (positioned by the observer using a keyboard) as the participant selects a virtual wrench by moving an engine hose.
Figure 5.7: OC aircraft engine maintenance domain (MA).
Figure 5.8: OC home entertainment domain (HE).
Figure 5.9: User observation tasks.
Prior to the study, each participant signed a consent form, read a one-page set of instructions, and then watched a two-minute introductory video of our connector OC, described in Section 5.6.1 and depicted in Figure 5.16. The connector shown in this video is not part of either the MA or HE domain. Following this introduction, the user started the first task in the observation. Each task was displayed to the user using an untracked, hand-held, video see-through “magic lens” display [Billinghurst, Kato, and Poupyrev 2001], described in Section A.7.6.2.

During execution of each task, participants were encouraged to use a “think out loud” protocol to verbalize the type of system output they would expect to result from their gesturing (e.g., “I’m selecting items in the menu by sliding my finger on the back of the television.”), as depicted in Figure 5.10. The observer, who was out of direct view of the participant, simultaneously used mouse and keyboard inputs to simulate this expected output in our Goblin XNA application. Figure 5.9(a) shows an example of this feedback with a cyan-colored 3D cursor (positioned by the observer using a keyboard) moving in response to the user gesture. This simulated output provided basic visual feedback to the subject without requiring specialized tracking equipment.

Data was collected via annotated screen captures taken by the observer through an independent documentation tool not visible to the participant. When the participant indicated a gesture or affordance of interest, the observer snapped a screen shot from the magic lens display, and annotated it with verbal comments from the user and additional commentary of their own. This documentation system was interfaced with the Goblin XNA application via shared memory interprocess communication, and was completely transparent to the participant. This limited extraneous dialogue between the participant and the observer. The documentation tool also tracked the start and finish times of each task.
Figure 5.10: A subject demonstrates a potential OC used during a menu selection task in the HE domain.
Once the participant experienced all tasks from both domains, they completed a questionnaire regarding the perceived usefulness of various affordances and a complete description of their recommended user interface for each task in both domains. Participants were encouraged to provide additional hand-drawn sketches, samples of which are shown in Figure 5.11. To assist the user in this phase of the study, we provided each with an automatically generated, hypertext-based report that contained all screen captures and observer notes from the observed session. A portion of an example report is depicted in Figure 5.12.

Figure 5.11: Selected user concept sketches of proposed OCs.
Figure 5.12: An extract from an example report provided to each subject to aid in their completion of the post-experiment questionnaire. The report includes (top) an image of the original stimulus and (bottom) screenshots, one of which is shown here, captured using the hand-held AR display of the user gesturing during the experiment.
5.5.3 Results

In an attempt to gather insights about user affinities toward possible OC affordances, we examined participant responses to the seven task types using two criteria: OC type and preferred features. We define a participant response as any and all affordances and gestures used to fulfill a particular task. We used the screen captures taken during each session to manually code each participant response as follows. For the OC affordance type criteria, each participant response was examined for presence of affordance types as defined in Section 5.4.1. In cases where the user invoked multiple affordances to complete a task we coded each separately. For example, a user might have elected to control the 3DOF orientation of the 3D camera using one valuator to control pitch, and second valuator to control yaw, and two buttons to control roll (one to increase it and one to decrease it). To determine the participant feature preferences summarized in Section 5.5.3.2, we counted and sorted the appearance of specific affordances across all tasks.

5.5.3.1 Results by OC Affordance Type

Tables 5.1 and 5.2 summarize the distribution of OC types in each domain for each task type across all participant responses. Each column depicts the frequency at which each affordance type appeared in our participants’ notional OC interface. Note, because many participants invoked multiple OC types to accomplish their assigned goals, the column values are not mutually exclusive.
Study participants selected a plurality of valuator-based affordances, which appeared in 57% of tasks in the ME case and 50% of tasks in the HE case. It is difficult to draw conclusions about user affinities from this result due to the influence of task type. The tendency to select valuator-based affordances might easily be the result of participants satisfying device-independent notions about the widgets required to complete a task (e.g., a participant might ex-
pect the ability to control a spinner using a slider). However, the results do provide insights about the general distribution of affordances a typical OC-based application might require.

Examining the results by task type illuminates several interesting findings. In the case of discrete valuators, where one would expect button-based affordances to dominate, our population substituted or incorporated valuator-based affordances 20% of the time in the MA scenario, and 33% of the time in the HE scenario. In these cases, the participants either fully ignored available button-based OCs (e.g., by tapping the individual striations on a grooved wiring harness sleeve rather than pushing a nearby fastener) or opted to incorporate a valuator as part of the task (e.g., using a scroll gesture on the wiring harness sleeve to first select the desired virtual button, then making a select gesture on the nearby fastener). Examining the continuous valuators tasks reveals a similar result. In these cases, participants either exclusively used button-based approaches (e.g., changing a spinner’s value with up and down buttons, then confirming completion with a third button) or used a valuator-based affordance supported by one or more button-based counterparts. This substitution of valuators and buttons is likely the result of variance in general user interface preferences. For example, when confronted with the task of navigating a menu using a tangible user interface, some users may prefer a user interface that mimics a trackpad, while others might prefer one that resembles the buttons on a keyboard.

Further inspection of the results reveals that participant preferences for OC affordance types were not mutually exclusive. In several instances, particularly for the more complex interaction tasks (e.g., 3D scene control), the participants selected multiple affordance types. In most cases, this was due to a combination of convenience and necessity, where the participant was initially drawn to a particular affordance, later found it lacking, and then added the nearest object.
In these instances, it was interesting to watch participants “make their interface work,” rather than start over with a possible more suitable alternative.

Based on these results, one proposed heuristic for the design of OC interfaces, supplementing those of Section 5.4.1, is to include multiple, possibly redundant, OC types in support of a single interface task. Such an approach might include the ability for a user to dynamically select and configure which controls to use for a task.

5.5.3.2 Results by Preferred Physical Feature

Table 5.3 lists the top four physical features used as affordances for each task domain appearing in participant responses, with each feature manually highlighted in a photograph of the domain. Even though we cannot draw conclusions from these results about what specific features make the best OCs in general, the results reveal the wide range of affordances users can envision as supporting OCs. Additionally, we noticed that each of the top four selected features in both scenarios were located roughly at eye level and are within arm’s reach of where participants stood (unprompted) during the study. This supports the importance of location in the selection of affordances for OCs and provides us with an additional OC design heuristic: The affordance underlying an OC should require minimal physical exertion by the user.
<table>
<thead>
<tr>
<th>MA Domain</th>
<th>HE Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
</tr>
<tr>
<td>20%</td>
<td>20%</td>
</tr>
<tr>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
</tr>
<tr>
<td>18%</td>
<td>11%</td>
</tr>
<tr>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
</tr>
<tr>
<td>11%</td>
<td>8%</td>
</tr>
<tr>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
</tr>
<tr>
<td>10%</td>
<td>7%</td>
</tr>
</tbody>
</table>

Table 5.3: Preferred physical features (%) in the MA and HE domains. A filter was applied to each image in this figure (but not used in the experiment) to accentuate the affordance of interest.
5.5.4 Additional Findings

We noted several additional findings as a result of this study. First, it was difficult to inspire participants to imagine objects in the MA and HE environments as being components of a computer interface. When participants voiced confusion about what they were supposed to do, we deliberately avoided demonstrating within the MA or HE environments and instead referred the participant to the connector OC video shown prior to the observation. Many participants verbalized their hesitancy to respond with remarks about how they were unfamiliar with the larger objects in the environment (e.g., “I don’t know how to hook up a VCR” or “I’m not a mechanically inclined person”). This suggests that even though an object (e.g., the back of a television) might contain objects that are individually perceived as meaningful affordances by the user, context from the area surrounding these affordances can cloud perception. Any implemented OC should make full use of virtual content to help mitigate this effect, possibly even removing or hiding real-world objects that are not part of the OC. (For example, although we did not use any kind of highlighting in this study, we believe that techniques such as that used manually in the photographs in Table 5.3, might be useful to emphasize an important feature in a user interface that employed OCs.)

As part of our questionnaire, we asked participants to suggest other objects found in their daily lives that might be used as part of an OC-based interface. Some of the proposed ideas included:

- Using a writing pen to control mobile media players or 3D games.
- Using a chair or sofa arm rest to control a home entertainment system.
- Using the various straps and fasteners on a backpack to control a mobile device (e.g., in the spirit of the wearable communication enabler developed by Mikkonen and col-
leagues [2001]).

- Using rings worn on one’s fingers to control various applications (e.g., in the spirit of the ring-based interface proposed by Ashbrook, Baudisch, and White [2011]).

### 5.6 Prototype Implementation

#### 5.6.1 Implemented OCs

We used our ARMAR architecture to implement a prototype OC interface, and describe this implementation in detail in Appendix A. The interface is designed around a Rolls-Royce Dart 510 turboprop aircraft engine, depicted in Figure 5.13, and features five button-type OCs. We created two variations: one for demonstration purposes (Figure 1.4) and one for evaluation (Figure 5.14) in the performance and acceptance user study described in Section 5.7. Four of these OCs (labeled “A” to “D” in Figure 5.14) map to smooth protrusions on the engine’s compression section and are used to select items in a virtual menu. The fifth button OC (labeled “NEXT” in Figure 5.14) maps to a nearby bolt, and is used as a “next” button to navigate between menus. The menu button widgets were modeled to resemble the underlying protrusions, while the “next” button widget is a semi-transparent circle. We also implemented two other types of OCs. One is a valuator-based OC that maps a grooved wiring harness sleeve to a linear slider (Figure 5.14). This slider is used to control a numeric value recorded in a text box. Gesture matching for this OC follows the algorithm described in Section 5.4.3, and the algorithm approximates the location of the user’s fingertip within the slider’s linear tracked region. This location is then mapped to the slider’s current numerical value.
The other OC we implemented is a rotating OC that maps an antenna connector to a virtual text box and is depicted in Figure 5.16. Gesture matching for this OC generally follows the algorithm described in Section 5.4.3, but the algorithm scans a circular region corresponding to the rotating collar of the antenna connector. This region is scanned in a clockwise fashion from the 12 o’clock position of the tracked connector. The algorithm then locates the largest connected skin component on this circle and uses it to approximate the current rotation of the connector’s collar.
Figure 5.14: Our prototype OCs interface shown (top) without and (bottom) with overlaid graphics. (Images depict view through the video see-through display.)
Figure 5.15: Valuator-based OC. (Image depicts view through the video see-through display.)
Figure 5.16: Rotating OC. (Image depicts view through the video see-through display.)
5.7 OC Interface Technique User Study

We designed a user study to compare the performance and general acceptance of our OC prototype interface technique to that of a more standard tangible user interface technique. This study featured only button-based OCs due to ongoing development of our valuator and button-based prototypes at the commencement of the study. Fifteen participants (11 male and 4 female), ages 20–34 ($\bar{X} = 24$), 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. Only one of these participants also participated in the user study described in Section 5.5. 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 indicated that they required corrective contact lenses or glasses. All participants determined that the separate left and right eye focus adjustments on the HWD provided adequate correction.

5.7.1 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” [Tomasi, Rafii, and Torunoglu 2003]. In order to adapt this technique to our prototype, we installed a 60 cm (width) × 78 cm (height) × 0.3 cm (thickness) panel of PVC plastic over the top of the part of the Dart engine that we used to implement the button-based OC prototype described in Section 5.6.1. The panel, shown in Figure 5.17, 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. This ensured that there was no location advantage afforded by either of the two techniques. The panel was attached with quick release hardware to facilitate a rapid transition between the two techniques during our study.

![Baseline comparison technique (BL). (Image depicts view through the video see-through display.)](image)

**Figure 5.17: Baseline comparison technique (BL).** (Image depicts view through the video see-through display.)

### 5.7.2 Task

Participants were asked to perform a selection task simulating the mechanical inspection of the Rolls Royce Dart 510 turboprop engine. This selection task, demonstrated in Figure 5.18, consisted of matching target text displayed on 3D virtual placards positioned at locations on the engine with a corresponding text entry in a screen-fixed virtual 2D list. The 3D placards are registered to subcomponents of the engine to simulate specific items to be 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 an aviation maintenance manual [U.S. Army 1992].

To successfully complete an individual trial, the user must use OC buttons to highlight and confirm the target condition in the 2D list. The list contains four positions randomly populated with the target and three incorrect alternative conditions. Participants use four OC buttons mapped to each position in the list for the highlight step, and confirm the highlighting with a fifth OC button. Figure 5.18 shows an example 3D placard with target text (b–c) and the accompanying 2D menu (c–e), as seen in the HWD.
Figure 5.18: OC interface technique user study task sequence. (Images depict view through the video see-through display.)
5.7.3 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 block 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. As shown in Figure 5.18, each trial began with the user pressing an OC button to populate the virtual environment with a single virtual placard at one of the five randomly chosen locations. Cueing information was then presented to the participant, prompting them to locate and read the target condition displayed on the placard (Figure 5.18a–b). This portion of the trial was not timed. When the participant 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 (Figure 5.18c). Once the participant used the buttons to highlight and confirm a condition (right or wrong) in the 2D list, the trial ended (Figure 5.18d–e). The experiment logic then logged the overall completion time, the displayed target condition, and the participant’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. The ordering of these ten random target conditions per each of the five locations was randomized between blocks.
5.7.4 Hypotheses

Prior to the experiment, we proposed the following hypotheses:

H1: OC would be faster than BL, as the differentiable tactile landmarks would reduce homing time and facilitate eyes-free manipulation of buttons.

H2: OC would be more accurate than BL, as the tactile landmarks would focus gestures and prevent errant entries.

5.7.5 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. We then analyzed the remaining data set for completion time, error rate, and subjective ratings, with a Bonferroni corrected significance level of $\alpha=0.0125$.

5.7.5.1 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). A repeated measures analysis of variance revealed technique had a significant main effect on selection completion times ($F_{(1,28)}=8.11, p < 0.001$). As depicted in Figure 5.19, the average completion time was 3.67 seconds when using the OC technique, and 4.25 seconds when using the BL technique. A post-hoc comparison of mean completion time revealed task completion time under OC was 86% that of BL, which was statistically significant
(t_{14} = 4.89, p < 0.001). This result confirms H1. Finally, the interaction of Technique and Location did not have a significant main effect on completion time for the selection task.

5.7.5.2 Error Rate Analysis

We applied a 2 (Technique) × 5 (Location) repeated measure ANOVA on mean errors per trial, with our participants as random variables. However, we failed to identify any significant effects of technique on error rates (F_{(1,28)} = 1.94, p = 0.185). As depicted in Figure 5.20, the mean error rates were 2.6 errors for OC, and 1.5 errors for BL, which were not significantly different. Thus, we failed to confirm H2.

![Figure 5.19: Average completion times (seconds) for OC (left) and BL (right). The average completion time when using the OC technique was 86% that of BL, which was a significant speedup.](image)
We attribute the failure to confirm our hypothesis that OC would decrease error rates to two design shortcomings. First, based on our observations of the experiment and user input, 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 participant’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 participant’s selection without allowing time to detect the stray gesture before confirmation. Second, our gesture recognition algorithm does not provide a depth filter. That is, our prototype implementation was unable to determine the height of a user’s hand relative to the surfaces of the OC, and could not distinguish transient movement above the OC from a gesture on the OC. As a result, if the participant’s hand hovers over the top of any buttons while transitioning, the algorithm will detect this hovering as button activation. We believe that including depth information in our gesture recog-
nition algorithm and selecting OC affordances more carefully could decrease the number of these errors.

5.7.5.3 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. The results from these ratings are depicted in Figure 5.21 as a histogram of responses. Collectively, the participants rated the OC technique as better than the baseline in terms of ease of use (4.00), satisfaction (3.87), and intuitiveness (4.67). However, a Friedman test failed to detect a significant difference between the ratings of the two techniques in terms of ease of use ($\chi^2(6,2)=3.00$, $p=0.08$), satisfaction ($\chi^2(6,2)=3.60$, $p=0.06$), or intuitiveness ($\chi^2(6,2)=3.57$, $p=0.06$). When asked to rank the technique they would rather use to perform the task, 11 of 15 participants selected the OC technique. A Wilcoxon signed rank test with Bonferroni correction ($\alpha=0.0125$) revealed this was a significant ranking ($p=0.02$). 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 and the inability of the gesture algorithm to distinguish hovering from selection.
5.8 Discussion

We were encouraged by our two user studies involving OCs, and are excited about the potential of this newly proposed class of user interface techniques. The user observation study demonstrated people are receptive to the idea of leveraging otherwise unused objects in the envi-

Figure 5.21: Survey responses for all participants. Responses are plotted on Likert scales (where 1 is most negative, 5 is most positive), with mean and standard error below the horizontal axes.
ronment as components in a passive haptic user interface, a fact reflected in several of our users offering their own, creative OC designs. This observation study also provided invaluable insights into how people perceive affordances that might be a part of future instances of OCs. For example, the study revealed users invoked multiple affordances types simultaneously when solving a single interface tasks (e.g., combining buttons and movable-OCs to select items in a menu). Additionally, our study suggests people gravitate to using valuator-based OCs even when button-based OCs are sufficient for the task at hand. This suggests a user’s preferences for OC may be linked to their preferences when using traditional interfaces (e.g., a person prefers scrolling a menu by manipulating a valuator with a track pad versus pressing arrow keys on a keyboard).

The results of the user interface technique study comparing OCs to an undifferentiated baseline were also encouraging. We were pleased that our initial prototype implementation of an OC interface was able to support faster completion times compared to the baseline. Moreover, we were encouraged by the level of enthusiasm for our technique expressed by the participants. We 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. The study also revealed several additional findings of interest. First, 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. Second, 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 in the OC condition was achieved by one such participant. Third, although our user study did not explicitly feature tasks mandating eyes-free interaction, several participants tried to select the buttons when they were outside their field of
view during both OC and BL trials. Multiple participants commented on how they felt more comfortable attempting eyes-free interaction in OC, as opposed to BL.

In both studies we noticed the influence of a person’s previous experience with a certain domain and their use of OCs in that domain. As we described in Section 5.5.4, several participants in the user study hesitated to manipulate objects is either the aircraft engine or home entertainment domain, citing their lack of experience in the particular domain as the source of this hesitation. Similarly, in the user interface technique study described in Section 5.7, 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. These results suggest practicality and location alone might not be sufficient criteria when selecting affordances for an OC interface. Future implementations of OC interfaces will need to address such predispositions in order to be fully accepted by users.

In closing, we acknowledge that OC interfaces might not be suitable or necessary for all user interface scenarios involving procedural tasks. In many cases, existing classes of user interfaces such keyboards, keypads, and touch screens will suffice. However, we believe OCs are a good choice for procedural tasks requiring eye and hand focus and which restrict the use of these other interaction techniques.
6 Conclusions and Future Work

In this dissertation, we explored the use of AR interfaces to provide assistance during procedural tasks. The preceding chapters detailed our efforts to quantify the benefits of applying AR to procedural tasks, while identifying a new class of user interaction techniques to support these interfaces. These chapters also described and demonstrated an architecture for constructing AR interfaces for procedural tasks, which was realized in a set of implemented prototypes. In this chapter, we summarize the contributions of this work, and offer some lessons learned governing the use of AR interfaces for procedural task assistance. We then conclude with a discussion of opportunities for future work in the area of AR interfaces for procedural tasks.

6.1 Summary of Contributions

We have sought to answer two research questions: What are the benefits of using an AR interface to support procedural tasks? How can we develop effective user interaction techniques in AR interfaces, while also minimizing interference with the task environment and the worker? This dissertation has made three principal contributions toward answering these questions:

Design, implementation, and evaluation of an AR interface designed to support informational portions of procedural tasks (Chapter 3). We designed a prototype AR user interface sup-
porting professional mechanics conducting maintenance in an LAV-25A1 armored personnel carrier. An evaluation of this prototype under field conditions revealed that the AR prototype allowed mechanics to locate tasks significantly faster than when using an enhanced form of computer-based documentation currently employed in practice. A qualitative survey showed that mechanics found the AR condition intuitive and satisfying for the tested sequence of tasks.

*Design, implementation, and evaluation of an AR interface designed to support psychomotor portions of procedural tasks* (Chapter 4). We developed an AR user interface to provide specific forms of assistance during the psychomotor phases of a challenging and realistic assembly task. This interface tracks both the user and the components being assembled and provides prescriptive and dynamic feedback to the user based on ongoing psychomotor activity. A user study revealed the average completion time when using our AR prototype was 46% that of the average completion time when using an LCD-based control condition, which was a significant speedup. The user study also revealed alignment error under the AR condition was 22% that of the LCD condition, which was a significant improvement. Qualitative results from the study indicated that participants overwhelmingly preferred the AR condition, and ranked it as more intuitive than the LCD condition. A smaller, pilot experiment found no significant differences in performance between our AR condition and an idealized condition in which physical labels were added to the components being assembled.

*Design, implementation, and evaluation of Opportunistic Controls, a novel class of interaction techniques* (Chapter 5). We proposed a new class of AR user interaction techniques, known as Opportunistic Controls (OCs). This interface harvests unused, tactiley interesting affordances in the domain environment as a source of passive haptics for AR interfaces. A user observation study provided insights about how user interfaces featuring OCs might be designed.
A second study examined the effectiveness of an OCs interface and revealed that participants achieved an average completion time that was 84% of the average completion time achieved when using a simpler form of passive haptics.

### 6.2 Lessons Learned

Our research on AR interfaces for assisting with procedural tasks has resulted in several lessons learned, which we offer as practical conclusions supported by our experience designing, developing, and evaluating AR prototypes:

*Effective applications of AR to assist procedures should consider all phases and activities of that procedure.* As we identified in our LAV-25A1 study (Chapter 3), and later proved in our psychomotor study (Chapter 4), effective applications of AR to procedural tasks must consider potential forms of assistance across all activities in those tasks. As we discovered in this dissertation, there are distinct opportunities to apply specific forms of assistance in the various phases of a procedure. Fully leveraging the potential of AR for procedural tasks may require thinking about time-motion studies on a micro scale to reveal opportunities to provide assistance with activities where we do not usually look for help.

*People are willing to temporarily tolerate shortcomings with HWD technology if the HWD is providing value.* This dissertation suggested that people are willing to tolerate the limitations and discomfort associated with current HWD technology if they value the assistance afforded by wearing the HWD. In the LAV-25A1 experiment (Chapter 3), participants rated the AR condition at least as favorably as LCD in terms of satisfaction and intuitiveness despite the disadvantage of wearing a bulky, relatively low-resolution, prototype VST HWD with fixed-focus cameras and a narrow FOV. While some participants acknowledged the visibility constraints experienced while using AR with this hardware, they tempered this critique with appre-
ciation for the assistance it offered. Our study examining the use of AR in the psychomotor phases of procedural tasks (Chapter 4) revealed a similar result. Despite numerous comments on the post-experiment questionnaire about the discomfort associated with wearing the NVIS HWD, 20 of 22 subjects rated it as the preferred technique. While we cannot extrapolate these results to scenarios requiring extended use of current HWDs (e.g., in a full-time workday application), they do provide encouragement for the continued exploration of improved HWD technology in AR applications.

**People are sensitive to occlusions caused by augmented content.** As described in Chapter 3, several of the same mechanics who praised the effectiveness of AR visuals were also bothered when the same content occluded a task being performed. We suspect that this trade-off between the positive and negative aspects of overlaid AR content is likely related to the experience level of the individual person performing the procedure. Future AR applications should tailor AR content to address the needs of each individual. Likewise, applications should adopt a “do no harm” rule of thumb, and limit any assistance to only what is required. Where possible, AR applications should include methods similar to those demonstrated by Bell and colleagues [Bell, Feiner, and Höllerer 2001] that employ view management to avoid occluding more important material with less important material. These methods might be accompanied by interaction techniques that allow people to easily dismiss content once it is no longer needed (e.g., labels) and control the speed of “fade-away” animations (e.g., our red 3D attention-directing arrow).

**A person’s knowledge and experience of a particular procedural task domain shapes their perception about interface objects used inside that domain.** During our observation of participants in our OC user studies (Chapter 5), we learned that a person’s previous experience and thoughts about that domain influenced their ability and willingness to visualize objects in that
domain as OCs. Participants who appeared comfortable in the domain from the onset of the experiment were more enthusiastic about touching and manipulating OCs. Conversely, several of our more timid users cited unfamiliarity with the task domain as a reason to pause or hesitate. We believe this relationship between a person’s experience and knowledge of a task domain might extend to other aspects of AR interfaces for procedural tasks beyond OCs. For example, if a person hesitates to operate an OC button on an aircraft engine because they are not used to touching machinery, might they also hesitate in interpreting a 3D billboard presented in AR anchored on the aircraft engine? That is, does uncertainty about an environment negatively impact perception of AR visualization associated with that environment? We believe it may and any aspect of an AR interface for procedural tasks must manage and address a person’s experiences and preconceived notions about the domain.

*Failure to prove a hypothesis can lead to new ways of thinking.* At the conclusion of our LAV-25A1 study described in Section 3.3, we were disappointed when we failed to find support for our hypothesis that the AR condition would help mechanics complete tasks more quickly than the LCD baseline. However, this caused us to reexamine the nature of procedural tasks and begin to think about specific forms of assistance targeting the psychomotor phase. The end result was an additional application area where we showed AR could provide value during procedural tasks.

### 6.3 Future Work

We are encouraged by the many opportunities to continue exploring the application of AR to help people become more efficient, safer, and less frustrated while conducting procedural tasks. Several of these opportunities are introduced below.
6.3.1 Extending the Benefits of AR Interfaces for Procedural Tasks

We are interested in extending our contributions identified in the maintenance and repair domain to other domains that involve procedural tasks. Potential application areas include manufacturing and construction, where Feiner and colleagues [1995] and Webster and colleagues [1996] have already demonstrated potential. We believe there are more opportunities in the home improvement arena, as demonstrated by Zauner and colleagues [2003]. We are also interested in examining the use of AR to help people perform first-aid, which would allow non-medical personnel to benefit from AR systems [State et al. 1996; Rosenthal et al. 2002; Wacker et al. 2006] designed for medical professionals. We also see opportunities for AR in personal survival planning and crime scene management. Future work in these other domains might lead to the development of exciting new forms of user interaction and visualization patterns.

There are also many opportunities to explore the specific types of assistance provided by AR in both the localization and psychomotor phases of procedural tasks. Our selection of the different types of assistance featured in our prototypes (e.g., the red 3D arrow used for attention directing in Chapter 3 and Chapter 4 or the circular arrow suggesting optimal alignment direction in Chapter 4) was guided by established design heuristics, pilot testing, and intuition. However, more research is needed to validate these design decisions. For example, the function mapping rotational distance to the color and size of the arrow depicting the motion required to achieve target alignments in the psychomotor experiment (Figure 4.5) was established through trial and error, and could benefit from more deliberate research. Likewise, the colors and icons used to label the holes on the cans and cones (Figure 4.5) could also be refined with more deliberate research. These examples represent a small fraction of the types of AR design decisions encountered when designing user interfaces for procedural tasks. Future work could provide more clari-
ty and quantitative support for the specific forms of assistance employed with procedural tasks. For example, a future study might examine many different AR designs for assisting a person in aligning two rigid bodies around a common axis (similar to our combustion chamber assembly task discussed in Chapter 4). We envision this same type of study being repeated for many different types of informational and psychomotor tasks. The end result of these efforts would produce a catalog of preferred AR techniques for all major activities contained in a procedural task taxonomy, such as the one offered by Guo and Tucker [1996]. A user-designed documentation approach similar to the one adopted by Heiser and colleagues [2003; 2004] could prove valuable in this endeavor. This process could begin by articulating the types of AR used to assist with each of the 18 tasks featured in our LAV-25A1 experiment described in Section 3.3.2.

6.3.2 Comparison of AR to Ideal Documentation

We see additional opportunities to compare AR interfaces for procedural tasks to “ideal” documentation. These opportunities build off our analysis of the PRINTED condition described in Section 4.7. For example, future work might create a version of procedural task documentation that uses real physical graphics (presented on thin, flexible display panels) to emulate the graphics presented by an AR interface. This version of the documentation would present the same information as the AR interface, but do so without the need for a HWD or other AR display. Comparing such a condition to a traditional HWD-equipped AR interface would isolate and quantify issues such as vergence-accommodation mismatch, HWD resolution, latency, tracking errors, and HWD weight. Moreover, an ultra-ideal simulated AR condition could identify the upper-bound on the types of improvements we can expect to leverage from AR interfaces as display and tracking technology improves.
6.3.3 Integration of Teaching and Assistance

As we alluded to in Chapter 2, there is much overlap in the literature between methods used to teach a person to perform a procedural task and methods used to assist a person performing the task. We focused our work in this dissertation on the latter. However, there are many opportunities for AR to contribute to the former. These opportunities might adopt the approach of Quarles and colleagues [2008] and focus on articulating the benefits of using AR to teach a person to perform a procedural task. Moreover, if a worker is aided by a reliable AR system, they can transition from a classroom setting into a field environment more quickly. Additionally, we hypothesize that AR could be used to deliberately promote learning while assisting. This might involve an AR system gradually reducing the level of assistance it offers a person as it detects the person has internalized the instructions. After this person masters the task, the system could assume a sentinel role and offer graduated assistance as it detects errors or degradation in performance.

6.3.4 Attention Directing

When we designed the attention-directing graphics used in our prototypes, we leveraged general designed heuristics, such as those proposed by Heiser and colleagues [2004], and pilot testing to guide our approach. However, we feel more deliberate work is needed to explore the basic structure of directing a user’s attention and to identify spatiotemporal parameters that can guide future designs. For example, the attention-directing sequence we employed in our prototypes and introduce in Section 3.2.2 culminates with a large 3D arrow extending from the edge of the screen to a target (as shown in Figure 3.4). Although we showed this localization technique allowed users to locate tasks more quickly than when using an enhanced baseline, feed-
back from users suggests the technique might demand more attention than is needed. Yet, as we learned during pilot testing, our adopted technique appears less visually intrusive than a second technique we implemented that follows the Attention Funnel design proposed by Biocaa and colleagues [2006]. However, there may be instances when more visually-prominent attention-directing techniques are ideal. For example, if an application is trying to quickly orient a user’s attention to an emergency condition, techniques emphasizing coarse and rapid movement of the user’s head might outperform techniques designed to support precision visual search tasks. Future work could examine the requirements for various types and phases of attention directing, and suggest parameters governing the design and use of attention-directing graphics (e.g., color, geometry, and size). This exploration could include evaluating currently accepted attention-directing techniques under various conditions to highlight important differences.

6.3.5 View Pose Management

We see related opportunities in the design and evaluation of AR techniques for view pose management. We use the term view pose management to refer to a combined model for guiding a user’s view direction and head position. This idea is inspired by the point-of-interest motion planning task studied and addressed by Mackinlay, Card, and Robertson [1990]. It is needed especially during certain procedural tasks executed under AR conditions, when workers must adopt precise head positions and orientations when performing physical manipulations. For example, assume we are designing an AR interface for assisting an office worker inserting a device into an unfamiliar desktop computer, where the appropriate receptacle is located at the bottom of the computer’s front cover. Figure 6.1 depicts this example task. The optimal head pose that ensures a clear view of this task involves the user orienting their head on the receptacle, while also posi-
tioning their head near the bottom of the computer. However, if the interface features only the attention-guidance techniques that have typically been employed in AR [Feiner, MacIntyre, and Seligmann 1993; Biocca et al. 2006], which are orientation-centric, the worker might exhibit a response similar to the one depicted in Figure 6.1(b). The end result is an obstructed view of the task.

![Figure 6.1](image)

**Figure 6.1:** (a) A user is given a procedural task involving the receptacle for a computer peripheral (located at the far right of each frame). (b) The user responds to orientation-centric attention guidance directing their attention to the target task. This results in an obstructed view of the target task that requires repositioning of the user's head to view the task.

If the same interface provides only the wayfinding techniques that have typically been employed in AR [Feiner et al. 1997; Holt, Boehm-Davis, and Schultz 1989; Reitmayr and Schmalstieg 2003; Thomas et al. 1998], which are position-centric, the worker might exhibit a response similar to the one depicted in Figure 6.2. In this instance, we hypothesize the worker will progressively orient and move to each node of the path specified by the wayfinding technique. Points W1–W4 in Figure 6.2 show an example wayfinding path presented to the user using an AR display. As the user progressively navigates the path (Figure 6.2a–b), an abrupt and isolated orientation activity is required (transition from 6.2c to 6.2d) for the worker to acquire the optimal view pose for the task (Figure 6.2d).
What is needed is a technique that elegantly and seamlessly integrates both orientation and position. This notion raises several interesting research questions. First, how should we orient the worker once they reach the optimal position? Second, when and where should we change orientation to best promote cognition and awareness? Should we serialize the two activities and possibly induce an abrupt orientation correction at the end of a path (as in the transition from Figure 6.2c to Figure 6.2d)? Or, can we execute both activities simultaneously (e.g., by gradually orienting the user at each step of the path)? If we can combine them, then why not leverage this capability for other purposes, such as guiding the user’s view direction during transitions to provide overview context or other details? Future research might examine combinations of existing attention-directing and wayfinding techniques to guide the precise view pose of the user during procedural tasks. For example, one might use a chain of Attention Funnels [Biocca et al. 2006] to guide the user along a path, and a specially designated “orient only” funnel at the end of the chain.

Figure 6.2: (a) A user responds to position-centric wayfinding guidance presented in AR as a hypothetical path (W1–W4) leading to the receptacle for a peripheral device on a computer (base of CPU at far right of scene). (b) The user progressively positions their head at each node of the path. (c) At node W3, the user requires an abrupt and isolated orientation step to find the (d) final node in the path facing the receptacle.
6.3.6 Improving Opportunistic Controls

There are ample opportunities for future work involving Opportunistic Controls (OCs). Some of these opportunities involve addressing shortcomings in our existing implementation, chief among these being the lack of depth in our gesture recognition. We have done preliminary experiments with a two-camera solution, with one camera mounted parallel to, and just above the dominant plane of our OCs. This allowed our system to suspend the segmentation process of the top camera when the user's hand is not seen gesturing at the same depth as our OCs. However, we are interested in researching more robust options such as leveraging the capabilities of depth cameras, similar to techniques demonstrated by Wilson and Benko [2010]. Another opportunity to improve our implementation involves replacing marker-based tracking with a feature-based approach. We believe many of the same rich features embodied in tactiley interesting OCs could also be leveraged for visual tracking.

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.

Despite the preliminary insights offered by our OC user interface observation study of how users perceive and interact with OCs, more work is required in this area. Specifically, we would like to determine a set of heuristics that govern which mechanical and free-form topological features fit best with various 3D widgets. These heuristics could list possible OC designs for each member of a freeform feature taxonomy, such as the one proposed by Fontana, Giannini,
and Meirana [1999]. These heuristics could be used with the tool proposed above to help automate the creation of OCs.

Finally, future work involving OCs could examine their integration with an AR interface assisting professional mechanics in a field setting, such as the study we describe in Chapter 3. A field study involving professional mechanics would be an interesting and useful extension to the results we achieved with students conducting simulated inspections of a Dart 510 aircraft engine. Moreover, this study could expand the number and types of procedural tasks involving OCs.

6.4 Final Thoughts

This dissertation has examined and enumerated the benefits of using AR to assist with procedural tasks. While we acknowledge that these benefits do not apply to every procedural task in every domain, we remain very encouraged by our results. These results demonstrate significant advantages provided by AR in helping users perform nontrivial tasks in realistic and challenging environments. In one case, our results included the performance data and insights of professional mechanics. We also acknowledge that there is much room to improve the application of AR to procedural tasks, particularly with regard to hardware, especially display and tracking technology. However, we are confident that future industrial and consumer versions of our work will materialize soon, as the necessary technology becomes less bulky and expensive. Finally, we are hopeful our work and ideas will inspire new ways of thinking about how AR can help people perform procedural tasks more efficiently, effectively, safely, and enjoyably in the future.
Appendix A    The ARMAR Architecture

In this appendix, we present a software and hardware architecture we developed for designing AR interfaces for procedural tasks. We refer to this architecture as Augmented Reality for Maintenance and Repair (ARMAR) [Henderson and Feiner 2007], and have used it to build the prototypes described in Chapters 3–5. We designed ARMAR as a reusable, scalable architecture that provides the major functions required for an AR application (e.g., tracking and mixed reality rendering), while also accommodating specific user interface requirements common to electronic documentation systems (e.g., state-based content management). The architecture also supports rapid design of prototype interfaces by providing reusable software objects that separate interface control from procedural content. The ARMAR architecture integrates all hardware required to support AR interfaces for procedural tasks while also providing software dedicated to the creation of procedural assistance. This makes the ARMAR architecture applicable to a variety of procedural task domains.

In our discussion of ARMAR, we first propose the requirements for the architecture (Section A.1), and review related works (Section A.2). Next, we describe the high-level software (Section A.3) and hardware (Section A.4) aspects of ARMAR. We then describe the evolution of ARMAR (Section A.5) which consists of two iterations of the architecture (Sections A.6–A.7) used to construct the various prototype user interfaces in this dissertation.


A.1 Architecture Requirements

As we started building prototype AR interfaces supporting procedural tasks, we noticed several recurring design requirements for these interfaces. Specifically, we found that all AR interfaces supporting procedural tasks should:

- Accommodate the simultaneous use of multiple tracking technologies and combine their data into a single, hybrid model of the task environment.
- Support a wide range of display types including head-worn and hand-held form factors employing optical and video see-through technologies. This support should include the ability to use multiple displays simultaneously. The architecture must also provide functionality to calibrate the displays across a wide user population.
- Provide a reusable model-view-controller design that supports rapid, content-focused authoring of individual steps within a procedural task. This should include a library of reusable 2D and 3D user interface components such as menus, screens, arrows, and guides.
- Include a library of 2D and 3D content common to procedural task domains, particularly to the maintenance and repair domain (e.g., animated and static 3D models and 2D images of tools and other common objects).
- Provide attention-directing graphics and functionality to support localization during procedural tasks.
A.2 Related Work

The idea of a reusable architecture for designing AR applications is not new, as indicated by the citation of 11 AR-specific frameworks in a survey of interactive systems by Endres, Butz, and MacWilliams [2005]. The Distributed Wearable Augmented Reality Framework (DWARF) proposed by Bauer and colleagues [2001] represents one of the earliest AR architectures. Many systems have been constructed using DWARF, ranging from games [MacWilliams et al. 2003] to manufacturing applications [Echtler et al. 2004]. Another early architecture is the Studierstube framework [Schmalstieg et al. 2002] which is currently well-maintained and used in a large number of applications. Other notable examples of generalized AR architectures include Authoring Mixed Reality (AMIRE) [Haller et al. 2002], the Designers Augmented Reality Toolkit (DART) [MacIntyre and Gandy 2003], and OpenSceneGraph ARToolkit (OSGART) [Looser et al. 2006]. Several efforts have focused on creating architectures designed to support a particular class of AR applications. Of particular interest is a system proposed by Knöpfle and colleagues [2005] that facilitated authoring of procedural tasks for automobile maintenance.

However, despite the existence of several capable AR architectures, we could not find a system that met all the requirements specified in Section A.1. The system proposed by Knöpfle and colleagues [2005] is the most similar to our vision for ARMAR, but is primarily concerned with authoring. Published details about the system indicate the authored scenarios are rendered using specific proprietary software and hardware that we did not have an opportunity to evaluate. The generalized AR architectures, such as DWARF and Studierstube, possess many desirable characteristics. However, because these architectures are intended to support a broad range of augmented reality applications, they tend to be high-level tools re-
quiring significant implementation. For example, they lack native implementations of several features required for procedural tasks assistance such as attention-directing graphics and reusable, step-by-step authoring. They also lack substantial amounts of content. While we could have used DWARF or Studierstube to create augmented reality interfaces for procedural tasks, each would still require significant work to meet the precise set of requirements listed in Section A.1. ARMAR differs from these other approaches by explicitly providing the program logic, user interface components, and content required for a wide range of procedural tasks.

In our effort to satisfy our requirements for ARMAR, we also experimented with several game engines, which included Valve Source [Valve Source SDK 2011] and Irrlicht [Irrlicht3D 2011]. We found these useful in providing a rich set of reusable 2D and 3D objects and support for authoring. However, these game engines lacked support for critical AR functions such as object tracking and realistic stereoscopic rendering. We also experimented with several AR-specific software tools, including Goblin XNA [Goblin XNA 2010] and ARTag [Fiala 2005], which we found provided excellent support for constructing AR applications. These solutions were highly generalized and lacked native support for features such as attention-directing graphics and reusable, step-by-step authoring.

Because we could not find a system that met all of our requirements for ARMAR, we designed our own architecture. However, as described in this chapter, we leveraged several existing systems in this endeavor, mainly the Valve Source SDK and Goblin XNA.
A.3 ARMAR Software Architecture

This section presents an abstract view of the ARMAR architecture which is instantiated in two successive versions described in Sections A.6 and A.7. The principal software component of the ARMAR architecture is the *ARMAR Client*, which is depicted in Figure A.1. The purpose of the ARMAR Client is to render tracked and untracked 2D and 3D content to the user while supporting all user interactions with the AR interface. This client is deployed as an executable computer application that operates in a collection of other networked applications and computers that provide it with enhanced forms of interaction, tracking information, and application management. This might include other copies of the ARMAR Client used by workers collaborating on a procedural task. It is this collection of networked applications and computers that comprises the overall ARMAR architecture.
Figure A.1: The ARMAR Client.
As depicted in Figure A.1, the primary inputs to an ARMAR Client consist of user interaction (e.g., keypresses and gestures), tracking data about the user and other tracked objects in the environment, and video (if applicable). External commands from other ARMAR Clients also serve as input. The primary outputs of the ARMAR Client are visual and audio display devices. The conceptual design of the ARMAR Client calls for the integration of several important functional components:

- **Maintenance Procedure Controller.** Loads the maintenance procedure into a finite state machine and manages this state machine throughout the procedure.

- **Virtual Content Controller.** Manages the loading and visibility of all 2D and 3D content.

- **Input-State Controller.** Handles all forms of input from the user or networked sources and converts this input for consumption by the state machine inside the Maintenance Procedure Controller.

- **Scene Controller.** Renders 3D content.

- **Object Tracking Controller.** A collection of classes and interfaces for tracking the user and any objects in the procedural task environment.

- **Data Management Controller.** Manages the storage and access of all data and information required by the client including user information, performance logs, 3D models, and scripts.

Each of these functional components is described in more detail in Sections A.3.1–A.3.6.
A.3.1 Maintenance Procedure Controller

The Maintenance Procedure Controller is an important software class that manages the overall state of the ARMAR Client to match a user-defined procedure. When the user indicates they want to conduct a certain procedure (e.g., remove a generator), the controller will first read the procedure’s specification from the Data Management Controller. In general, this specification is represented as the tuple $M = \{S, O, \nu\}$ where:

- $S$ is a collection of states $S = \{s_1, s_2, \ldots, s_n\}$ matching the steps of the procedure.
- $O$ is a collection of objects $O = \{o_1, o_2, \ldots, o_n\}$ that are part of the procedure’s environment.
- $\nu$ is a mapping, $\nu: S \rightarrow O$, that indicates which objects are germane to each step of the procedure.

The Maintenance Procedure Controller uses the specification to populate a finite state machine that models each member of $S$. The controller also constructs each member of $O$ by instantiating it as one of several core software classes:

- **Static 3D Model.** A static 3D model is a virtual object that is registered and aligned with the physical world in order to augment the procedure environment. It is typically used to show hidden information (e.g. an occluded internal assembly) or indicate a target position for a physical counterpart (e.g., presenting a partially transparent part to indicate where the user should place a physical part). The static model can be accompanied by sounds.

- **Animated 3D Model.** An animated 3D model is an extension of the static 3D model that dynamically alters the virtual model’s position, orientation, visibility, or geometry to communicate important motions and gestures involved with the
procedure.

- **Occluding 3D Model.** An occluding 3D model is a special type of 3D model that is rendered only in the z-Buffer of the computer used to present the interface. It typically represents a real physical object and is rendered at that object’s location. This makes it possible for the physical object to occlude virtual objects that are positioned behind it.

- **3D Billboard.** A 3D Billboard is a special type of animated 3D model that is used to present text and other information as a label or flag aligned with and registered to the physical world. This model is animated to always face the user so that the text or other contained information can be readily viewed.

- **2D HUD Text.** This class of object represents screen-fixed text that is present as a head-up display (HUD) integrated with the user interface. It typically contains a high level description of the current task step and several lines of supporting information.

- **Attention-Directing Object.** Attention Directing Objects are used to direct the attention of the user and are comprised of animated 3D models, 3D Billboards, and 2D HUD text. These objects use tracking information about a user’s position and orientation relative to a target to present enhanced forms of visualization that prompt the user to look at the target.

Once each object in \(O\) is instantiated, the Maintenance Procedure passes the object to the Virtual Content Controller, which manages the object’s visibility over the domain of \(S\).
A.3.2 Virtual Content Controller

The Virtual Content Controller is a software class that implements $\nu$, the mapping of states in $S$ to objects in $O$. This mapping is implemented using an efficient associative data structure (e.g., an associative array, dictionary, hashmap), where the keys of the data structure are the members of $S$, and the contents of the data structure is a collection of members of $O$. After the Maintenance Procedure Controller instantiates a new object $o_i$, the Virtual Content Controller adds $o_i$ to the collection of elements indexed by $s_j$ where $s_j \rightarrow o_i$.

During an ongoing procedure, the Maintenance Procedure Controller will track the current state $s_c$ of the ARMAR Client. When $s_c$ is updated following a state transition, the Virtual Content Controller will enable each object $o_i$ in the mapping $s_c \rightarrow o_i$ and disable all objects that are not specified in this mapping. This act of enabling and disabling of objects is dependent on object type, and is handled by one of four subordinate content managers that manage 2D HUD Text, 3D models, Sound, and Attention-Directing Objects. The type-specific enabling and disabling behavior is summarized in Table A.1.
The Virtual Content Controller includes an Attention Manager component. The Attention Manager creates and presents all 3D content responsible for managing a user’s attentions. Key tasks for the Attention Manager include monitoring the location of the user, determining optimal directions to tasks, and presenting near and far-field cues to influence a user to focus their attention on a particular task or location.

### A.3.3 Input-State Controller

A user interface implemented using the ARMAR architecture passes all user input to the ARMAR Client’s Input-State Controller. The Input-State Controller parses user input using a grammar $C = \{ c_1, c_2, \ldots, c_n \}$, where $c_i$ represents a low-level user command. The commands specified by $C$ fit into one of two categories: those used to trigger state transitions in the Maintenance Procedure Controller (e.g., the user pressing a “next” button), and those used to fine tune aspects of the current state (e.g., pausing an animated model).

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Content Manager</th>
<th>Enable Action</th>
<th>Disable Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D HUD Text</td>
<td>2D Content Manager</td>
<td>Reload HUD with current text</td>
<td>Hide current text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Render current text</td>
<td></td>
</tr>
<tr>
<td>Static 3D Model</td>
<td>3D Content Manager</td>
<td>Render 3D models</td>
<td>Hide 3D models</td>
</tr>
<tr>
<td>Animated 3D Model</td>
<td>3D Content Manager</td>
<td>Render animated models</td>
<td>Hide animated models</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Restart animation</td>
<td>Stop animation</td>
</tr>
<tr>
<td>Occluding 3D Model</td>
<td>3D Content Manager</td>
<td>Render occluding 3D models</td>
<td>Hide occluding 3D models</td>
</tr>
<tr>
<td>3D Billboard</td>
<td>3D Content Manager</td>
<td>Render billboard models</td>
<td>Hide occluding billboard models</td>
</tr>
<tr>
<td>Sound</td>
<td>Sound Manager</td>
<td>Start sounds</td>
<td>Stop sounds</td>
</tr>
<tr>
<td>Attention-</td>
<td>Attention Manager</td>
<td>Render attention directing models</td>
<td>Hide attention directing models</td>
</tr>
<tr>
<td>Directing Object</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table A.1: Content Management by Object Type.**

The Virtual Content Controller includes an Attention Manager component. The Attention Manager creates and presents all 3D content responsible for managing a user’s attentions. Key tasks for the Attention Manager include monitoring the location of the user, determining optimal directions to tasks, and presenting near and far-field cues to influence a user to focus their attention on a particular task or location.
The Input-State Controller is a software class capable of accommodating several forms of user input, which are handled by various subordinate controllers:

- **Keyboard Interface.** The Keyboard Interface is a software component that maps the individual keys of a keyboard (or keypad) onto members of C.

- **Mouse Interface.** The Mouse Interface maps mouse buttons and analog inputs onto members of C.

- **Networked Interaction Interface.** The Networked Interaction Interface processes commands specified in C that originate from external applications networked with the ARMAR Client. For example, one ARMAR Client might remotely control another ARMAR Client.

- **Gesture Interface.** The Gesture Interface maps commands specified by an external gesture application’s grammar to members of C. For example, in one of our implementations described in Section A.6.5.2, an external hand-tracking application is networked with the ARMAR Client through this interface. This application will have its own grammar. The gesture interface translates commands from this external grammar into commands specified by C.

- **Proactive Computing Interface.** Proactive computing [Tennenhouse 2000] is a form of interaction where a computer application proactively responds to user activity not normally considered as a deliberate form of interaction. For example, during an assembly procedure, a computer is used to track each individual component in the assembly. As the user completes each step of the assembly, the computer recognizes these events by analyzing tracking data and automatically presents the next step in the procedure. The ARMAR Client’s Proactive Compu-
ting Interface supports this style of interaction. This software interface allows specification of triggers tied to proactive computing events. The interface then communicates with the Object Tracking Controller to determine when these triggers are activated and translates these events into members of C.

A.3.4 Object Tracking Controller

The Object Tracking Interface provides a common set of methods for accessing and managing all forms of tracking data. This includes tracking data for the user and objects in the procedural task environment, herein referred to as motion tracking data. The centralized Object Tracking Interface standardizes data formats, supports the reuse of various data processing techniques such as filters and smoothing algorithms, and provides a unified model of all tracked entities.

Motion tracking data is read from the Motion Tracking Interface, the ARMAR class that wraps classes, functions, and methods contained in third-party motion tracking libraries. The Motion Tracking Interface is not specific to a particular tracking technology or vendor, and thus allows the ARMAR Client to seamlessly integrate and substitute many different tracking technologies and protocols. The Object Tracking Interface also includes a Network Tracked Object interface. This is an ARMAR software class that represents a virtual tracking device providing data about objects tracked by other applications. For example, an ARMAR Client with its own unique tracking configuration can share tracking data with other networked ARMAR Clients. The Network Tracked Object allows a subscribing ARMAR client to integrate this external tracking information as if the information originated from an attached local tracking device. This class also supports the fusion of tracking information
from multiple devices, either data from locally connected trackers, or data streamed from other ARMAR clients.

A.3.5 Scene Controller

The Scene Controller is a component of the ARMAR architecture that is intended to be implemented using one or more preexisting graphics libraries or game engines. This controller encapsulates all functionality required to render a 3D AR scene, including scene-graph management, virtual lighting, z-buffer rendering, and support for vertex and pixel shading. Moreover, to meet the requirements of ARMAR, the framework must support presenting AR scenes using a broad range of displays. These include monocular and binocular optical see-through (OST) and video see-through (VST) head-worn displays (HWDs), and VST handheld displays. In the case of video see-through displays, the Scene Controller must provide the ability to render real-time video imagery into the back buffer of an AR application. The framework must allow any video rendered into the back buffer to also be used by optical tracking libraries. When rendering to binocular displays, the framework must provide an accurate and highly configurable virtual stereo camera model that can be calibrated to the individual user.

The Scene Controller contains a Display Calibrator element that is a software class for managing the calibration of AR displays used with the ARMAR Client. This calibration is unique to each individual user and consists of collecting data and solving for the intrinsic and extrinsic parameters of the display, whether monocular or binocular. These parameters are then shared with the Scene Controller for inclusion in the virtual camera models. The Display Calibrator provides an interactive user interface designed to guide a user in collect-
ing calibration data and also provides all functionality to estimate the intrinsic and extrinsic parameters.

**A.3.6 Data Management Controller**

The Data Management Controller is a helper software class that handles the modeling, loading and storing of procedural task content used by the ARMAR Client. Ideally, the Data Management Controller affords complete separation of the specific steps and content of a procedural task from the core functionality of the ARMAR Client. This allows the ARMAR Client to support many diverse procedures without the need to hard code content and recompile applications.

The Data Management Controller is backed by several stores or repositories of information. The User Store contains information about a user’s skills, past performance, and assigned tasks. The Environment Store contains detailed information about the procedural task environment including 3D maps and models of the domain and surrounding area. The Maintenance Procedure Store contains a corpus of all procedures including required steps, accompanying text and media, and links to any required 3D models. The 3D models are stored in the Component Model Store which contains models of all possible tools and components featured in any maintenance procedure.

**A.4 ARMAR Hardware Architecture**

The ARMAR architecture integrates the hardware required to support the ARMAR Client and other applications in the ARMAR architecture. The exact configuration of hardware is highly dependent on the functionality required for the implemented procedural tasks
and any physical constraints posed by the task environment. However, ARMAR implementa-
tions generally involve five types of hardware: displays, rendering computers, tracking hard-
ware, interaction devices, and network infrastructure. Several implementations of this hard-
ware are described in Sections A.6.5.1–A.6.5.2 and Sections A.7.6.1–A.7.6.2.

A.5 Evolution of the ARMAR Architecture

Over the course of this dissertation, we created two versions of the ARMAR architec-
ture—one based on the Valve Source Engine Software Development Kit (SDK) [2011] and
one based on Goblin XNA [2011]. These versions represent successive iterations of AR-
MAR, and each provides a core set of software and hardware components fulfilling the func-
tions of the abstract architecture described in Sections A.3–A.4. Our decision to revise the
ARMAR architecture was motivated by limitations identified in the Valve Source SDK, as
well as a desire to adopt AR-centric features offered by Goblin XNA. The two versions of
ARMAR are described in Sections A.6–A.7 below and each section discusses specific im-
plemented prototypes.

A.6 Value Source-based Version of ARMAR

Our first version of the ARMAR Architecture was developed as a game engine
“mod” using the Valve Source Engine Software Development Kit (SDK) [2011]. This ver-
sion of the ARMAR architecture was used to build the LAV-25A1 prototype described in
Chapter 3 and the Opportunistic Controls prototype described in Section 5.6. Each of these
prototypes leverages a core set of software classes that collectively comprises the ARMAR
Client. Table A.2 contains estimates for the number of lines of source code required to im-
plement the various components of this client. Sections A.6.1–A.6.4 highlight important aspects of these components as they exist in the Valve Source SDK version of ARMAR.

<table>
<thead>
<tr>
<th>ARMAR Client Component</th>
<th>Lines of Code (x1000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input-State Controller</td>
<td>1</td>
</tr>
<tr>
<td>Scene Controller</td>
<td>8</td>
</tr>
<tr>
<td>Virtual Content Controller</td>
<td>3</td>
</tr>
<tr>
<td>Maintenance Procedure Controller</td>
<td>2</td>
</tr>
<tr>
<td>Data Management Controller</td>
<td>1</td>
</tr>
<tr>
<td>Object Tracking Controller</td>
<td>2</td>
</tr>
</tbody>
</table>

Table A.2: Source code estimates for the Valve Source-based ARMAR Client. Counts only include lines of code specific to the ARMAR architecture.

A.6.1 Maintenance Procedure Controller

The Maintenance Procedure Controller is implemented as a virtual game object. This causes the Valve Source game engine to call an update method within the Maintenance Procedure Controller once each render cycle. During this call, the Maintenance Procedure Controller polls the Input-State Controller to check for any inputs received since the last rendered frame, and updates the finite state machine and any scene objects accordingly.

A.6.2 Scene Controller

We repurposed the game engine “player” as the virtual camera in our scene, which is positioned by location information from the tracking hardware. The first-person view through this camera comprises the AR scene presented on the prototype’s display. However, we faced three challenges in rendering this scene when working with the game engine. The first of
these stemmed from a lack of support in the Valve Source SDK for placing external video in the back buffer of the scene, which our prototype required for a video see-through HWD. To overcome this limitation, we implemented video rendering at the operating system level by placing the video textures directly into the back buffer outside the purview of the Valve Source game engine. We implemented our own custom-built external dynamic link library (DLL) that intercepts or “hooks” the game engine’s instance of the DirectX graphics interface via the Microsoft Windows Detours library [Hunt and Brubacher 1999]. At the start of the game engine’s render loop, which we detect externally using codes embedded in game textures, software in the DLL captures full resolution stereo video from two Point Grey Firefly MV cameras. The video from both cameras is stretched to the full size of the back buffer and copied directly to the graphics device in side-by-side fashion. The game engine then proceeds to render the 3D scene as it would in a traditional game application. The entire scene is rendered in stereo at 800×600 resolution with an average frame rate of 75 frames per second (fps). However, the effective video frame rate sent to the stereo HWD is approximately 25 fps, due to the software upscaling of the stereo images from 2×640×480 to 2×800×600.

The second challenge we encountered when using the Valve Source SDK was its lack of native support for stereoscopic rendering. To overcome this limitation, we relied on the stereoscopic support provided by the graphics card, specifically the NVIDIA 3D stereo driver [2005]. This driver accomplishes stereoscopy for most DirectX and OpenGL applications by shifting the scene camera left and right in hardware at render time to render the scene twice. When this driver is used to render stereo video, special API functions contained in the NVIDIA Stereo BLT SDK are required. Unfortunately, the driver does not provide sufficient flexibility for adjusting the underlying stereo vision model. Specifically, we were not able to
fully adjust the depth at which objects were verged when using the driver. However, we were able to tune it to accommodate most procedures. The driver also caused an issue when rendering HUD text, which we describe in Section A.6.3.

The Display Calibrator in this implementation consists of manual corrections entered via a keyboard. During calibration, the user viewing the HWD attached to the ARMAR Client is instructed to align two head-tracked crosshairs with two corresponding 3D crosshairs registered at known points in the world and tracked with fiducial markers. An observer using keyboard controls assists with the alignment by adjusting the transformation between the world coordinate system and that of the HWD (as worn by a particular user) while observing video of the AR scene.

A.6.3 Virtual Content Controller

The Valve Source SDK natively supports most functions required for the ARMAR Client’s Virtual Content Controller. Three-dimensional virtual content in the AR scene is provided by native game classes in the Valve Source SDK that support static, animated, and billboard models. Because all virtual content in the AR scene is represented as game objects, standard game mapping tools can be employed to rapidly author an AR scene. We used our own custom-designed 3D models and textures with these classes to create all 3D content in our prototype.

We implemented the state to object mapping $v$ in the procedure specification of the Maintenance Procedure Controller by using a series of files containing scripts interpreted by the game engine. When the client undergoes a state transition, the script for that state is load-
ed and executed. The script contains dozens of commands for altering the visibility and behavior of the 2D and 3D content (as specified in Table A.1)

Although the Valve Source SDK provides support for the 2D HUD layer of the 2D Content Manager, the stereoscopic driver discussed in Section A.6.2 did not render the text at a workable depth. Therefore, we added text support to a 3D static billboard model that was placed at a configurable depth in the user’s field of view.

A.6.4 Input-State Controller

As with the Maintenance State Controller, we used a Valve Source virtual game object class to implement the Input-State Controller in this implementation. This class provides an update method that gets called by the game engine every frame. During this update, the Input-State Controller checks for any local input (e.g., key presses and mouse clicks). The controller also communicates with a gesture server, which was used to capture user inputs made with other interfaces (e.g., an Opportunistic Controls interface) or devices (e.g., a wrist-worn controller). The gesture server buffered any commands from the wrist-worn controller, which were passed to the Input-State Controller upon request.

A.6.5 Hardware

As mentioned in the beginning of Section A.6, we used the Valve-Source version of the ARMAR architecture to construct two of the prototypes described in this dissertation. Each of the prototypes had slightly different hardware requirements, which are detailed in Sections A.6.5.1 and A.6.5.2.
A.6.5.1 LAV-25A1 Prototype

We experimented with two HWDs while developing our LAV-25A1 prototype. The display we eventually used for user trials (Figure A.2) is a custom-built stereo VST HWD, constructed from a Headplay 800×600 resolution, color, stereo, opaque HWD with a 34° diagonal field of view (FOV). We mounted two Point Grey Firefly MV 640×480 resolution color cameras to the front of the HWD, which were connected to a shared IEEE 1394a bus on the PC. The cameras are equipped with 5mm micro lenses and capture at 30 fps. The AR-MAR Client executes on a quad-core 2.66 GHz PC with 4GB of RAM running Windows XP Pro, with an NVIDIA Quadro 4500 graphics card.

Figure A.2: Custom-built video see-through HWD based on a Headplay opaque HWD used in the LAV-25A1 prototype. The HWD is being tracked by an OptiTrack tracker using active IR LEDs mounted to the top of the HWD.
We also experimented with, and initially intended to use, an NVIS nVisor ST color stereo OST HWD, depicted in Figure A.3, and used in the implementation of Section A.7.6.1 and user study of Chapter 4. We selected this display because of its bright 1280×1024 resolution graphics, 60° diagonal FOV, and high transmissivity. However, during pilot testing, we discovered that LAV-25A1 assemblies located directly in front of and behind the seats prevented users from moving their head freely while wearing the relatively large nVisor HWD. This necessitated use of our custom-built HWD in the user study described in Chapter 3.

![Figure A.3: NVIS optical see-through HWD.](image)

Tracking is provided by a NaturalPoint OptiTrack tracking system. The turret’s restricted tracking volume and corresponding occluding structures created a non-convex and limited stand-off tracking volume, which led us to employ 10 tracking cameras to achieve ample coverage. Because we were focused on research, rather than practical deployment, we
were not concerned with the disadvantages of adding a large number of cameras to the existing turret.

The OptiTrack system typically uses passive retroreflective markers illuminated by IR sources in each camera. During pilot testing, we discovered that numerous metallic surfaces inside the turret created spurious reflections. Although we were able to control for all of these with camera exposure settings or by establishing masked regions in each camera, these efforts greatly reduced tracking performance. Therefore, we adopted an active marker setup, using three IR LEDs arranged in an asymmetric triangle on the HWD. Given the confined space inside the turret, we were concerned that a worker’s head position could potentially move closer than the 0.6 meter minimum operating range of the OptiTrack. However, experimentation revealed that, for any point inside our work area, at least four cameras could view a user’s head from beyond this minimum operating range. Moreover, the active marker setup prevented the possibility of IR light from the cameras reflecting off the user’s head at close range. The tracking software streams tracking data at 60 Hz to the PC running the AR application over a dedicated gigabit ethernet connection. The tracking application runs on an Alienware M17x notebook with dual core 2.8 GHz CPU, and 4GB RAM running Windows Vista, with an additional enhanced USB controller PC Card.

We implemented our wrist-worn controller using an Android G1 smartphone (Figure A.4). The device displays a simple set of 2D controls and detects user gestures on its touch screen. Gestures are streamed to the PC running the AR application over an 802.11g link. The user attaches the device to either wrist using a set of Velcro bracelets.
A.6.5.2 Opportunistic Controls Prototype

The Opportunistic Controls (OC) prototype used a similar set of hardware as that of the LAV-25A1 prototype. The prototype includes added hardware support for the recognition of user gestures. Imagery used for gesture recognition is captured using a single Point Grey Firefly MV 640×480 resolution color camera affixed with a 3.6mm microlens and mounted overhead with a clear line of sight to all OCs in our environment. The camera is also tracked by using the ARTag optical marker tracking library [Fiala 2005] to detect a fidu-

Figure A.4: Android wrist-worn controller. The inset image depicts the interface used to control animated models (when applicable).
cial-marker array within the camera’s current frame. This marker array is rigidly positioned relative to the five OC affordances and allows our gesture recognition algorithm to function if the camera is bumped or moved. We use a separate dedicated camera (instead of 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.

Tracking in the OC prototype is provided by two systems. For the user’s head, we used a ceiling-mounted InterSense IS-900 6DOF tracker to track a single station mounted on the custom-built Headplay VST HWD, as shown in Figure A.5. 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.

We experimented with two HWDs in the OC prototype, eventually opting to use the custom-built VST featured in the LAV-25A1 prototype (described in Section A.6.5.1). However, we modified the display to accommodate the IS-900 station as shown in Figure A.5. We also tested with an InnerOptic Vidsee video see through display [State, Keller, and Fuchs 2005], which is depicted in Figure A.6. The Vidsee supplements a Sony Glasstron LDI-D100B with a fixed dual mirror assembly containing a pair of cameras that is designed to provide parallax-free image capture—the cameras are virtually located at the same position and can share the same field of view as the user’s eyes. Using the Vidsee display, we were able to present the user with combined real and virtual stereo content at 800×600 resolution, thirty frames per second, creating a compelling experience. However, initial testing with the display revealed that the interlaced video captured by the display’s cameras did
not support high quality tracking of fiducial markers (which was required for the OC prototype).

Figure A.5: Custom-built video see-through HWD used in the OC prototype. The HWD is being tracked by an InterSense IS-900 tracker as indicated by the tracking sensor mounted on the top-front portion of the HWD.
Our OC prototype included 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-through display, while also supporting hand-gesture recognition. Additionally, we were 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.

The gesture-tracking algorithm runs on a dedicated Dell M1710 XPS laptop (2.16GHz, dual core processor, 4GB of RAM). The gesture recognition application segments the five button-type OCs and parses gestures at 30 fps. The rendering application executes on a PC (2.66GHz, two core processor, 4GB of RAM) running Windows XP Professional, with a single NVIDIA Quadro 4500 graphics card.
A.7 Goblin XNA-based Version of ARMAR

We created a second version of the ARMAR architecture using the Goblin XNA managed, DirectX-based framework for constructing AR applications [Goblin XNA 2011]. We used this version of ARMAR to implement the prototypes used in our study of the psychomotor phase of procedural tasks (Chapter 4) and our OC user observation study (Chapter 5). We modified ARMAR to use Goblin XNA because Goblin XNA natively provides much of the functionality lacking in the Valve Source SDK, which we discussed in Section A.6.2. Moreover, all source code for Goblin XNA is made available to the developer, which enabled us to make small modifications to the framework when necessary. Goblin XNA also uses managed code which we found more stable than when using Valve Source SDK. This migration of ARMAR to Goblin XNA required several thousand lines of code (summarized in Table A.3) and resulted in several important differences between the Goblin XNA-based ARMAR Client and the Valve Source-based client, which we summarize in Sections A.7.1–A.7.2.
Table A.3: Source code estimates for the Goblin XNA-based ARMAR Client. Counts only include lines of code specific to the ARMAR architecture.

<table>
<thead>
<tr>
<th>ARMAR Client Component</th>
<th>Lines of Code (x1000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input-State Controller</td>
<td>1</td>
</tr>
<tr>
<td>Scene Controller</td>
<td>3</td>
</tr>
<tr>
<td>Virtual Content Controller</td>
<td>7</td>
</tr>
<tr>
<td>Maintenance Procedure Controller</td>
<td>1</td>
</tr>
<tr>
<td>Data Management Controller</td>
<td>1</td>
</tr>
<tr>
<td>Object Tracking Controller</td>
<td>2</td>
</tr>
</tbody>
</table>

A.7.1 Maintenance Procedure Controller

The interface uses a finite state machine to manage visibility of content in the HUD and scene graph, where each state represents a single step in the procedure. This finite state machine is implemented in the main method of the application, and is updated by events originating from the Input-State Controller, as opposed to polling the Input-State Controller for changes. During state transitions, the Maintenance Procedure Controller invokes methods in the Virtual Content Controller to update content to match the new state of the application.

A.7.2 Virtual Content Controller

We implemented the state-to-object mapping \( v \) in the procedure specification using a multimap data structure located inside the client’s Virtual Content Controller. Each step in the procedure is assigned a state from the state-space of the finite state machine modeled in
the Maintenance Procedure Controller. These individual states become the keys of the multimap, where each key can be associated with many 3D objects. Prior to starting the application, the keys of the multimap are populated with states representing each step of a procedure. The ARMAR Client then instantiates all 3D content used during the entire procedure, and associates these objects with multimap keys representing procedure steps requiring the 3D content. The text displayed in the HUD for each step is modeled in a similar manner, but uses a distinct multimap.

We created a library of software classes to assist in authoring procedural tasks. This library includes a ContentMap class, which contains the multmaps for the 3D content and text. The ContentMap extends the Goblin XNA transform node; when a new object is added to the multimap, it becomes a child node of a ContentMap node for its procedure. We attach this ContentMap as a child node of a single common Goblin XNA SwitchNode at the top of the application scene graph. This allows the ARMAR Client to accommodate multiple procedures and use the SwitchNode to select the ContentMap associated with a particular procedure. The ContentMap also includes methods to manipulate content in a particular step of the procedure without having to update the finite state machine. These methods are used when the interface needs to modify minor details (e.g., changing the color of text in the HUD) in a way that is not significant enough to warrant a distinct state in the procedure.

Our library also added several methods that aggregate common low-level Goblin XNA methods and design patterns. For example, Goblin XNA required five lines of code to create one of our Static 3D Models. We condensed this to a single method in our library. These aggregate methods greatly facilitate authoring new scenarios.
A.7.3 Input-State Controller

State transitions are either manually cued by using an input device, or triggered automatically using a proactive computing model [Tennenhouse 2000] similar to the non-AR system demonstrated by Antifakos and colleagues [2002]. That is, in certain steps where the worker must reposition an object in order to complete the task, our system uses tracking data to monitor this activity and either automatically transitions to the next step or displays an error message. Our Proactive Computing Interface supports the following types of events:

- **Proximity Trigger.** Activated when the Euclidean distance between two tracked objects is less than a specified value for a specified time. This trigger can also be used to detect when an object is close to an arbitrary location.
- **Negative Proximity Trigger.** Activated when the Euclidean distance between two tracked objects is greater than a specified value for a specified time.
- **Moving Object Trigger.** Activated when the velocity of a tracked object exceeds a specified value. Only applies to object having a previous velocity equal to zero.
- **Stationary Object Trigger.** Activated when the velocity of a previously moving object is less than a specified value.

We also created a remote control interface to allow control of the ARMAR Client from another computer. This application, which is depicted in Figure A.7, is a 2D GUI that provides an administrative interface that controls the client via the Networked Interaction Interface. This remote interface is useful for debugging, assisting with calibration, and navigating through the steps in a procedure.
A.7.4 Scene Controller

We used the StereoCamera class of the Goblin XNA framework as the virtual camera in our scene. This camera is used to generate views of a 2D HUD layer and 3D scene graph maintained by the system. At the start of each render cycle in the application, the location and orientation of this camera are updated by information from the tracking hardware.

When the ARMAR Client is used with an OST HWD, the views through each half of the virtual stereo pair are rendered side-by-side against a black backbuffer (for optical see-
through display) within a single 2560 × 1024 viewport at a combined framerate of 60 fps. This side-by-side frame is sent to the computer’s display device, which sends the left half of the image to the left eye of the NVIS and the right half of the image to the right eye of the NVIS.

When the ARMAR Client is used with a VST HWD, video captured from video cameras attached to the display is placed in the backbuffer of the 3D scene using Goblin XNA’s native support for video rendering.

A.7.5 Display Calibrator

We implemented the HWD calibration technique developed by Fuhrmann, Splechtna, and Přikryl [2001]. In our implementation, the user holds a 10cm×10cm patterned target, attached to a Wii Remote that is tracked by the OptiTrack configuration. The user is requested to align the real target with a series of virtual targets, each of which is projected at eight 3D locations within each of the left and right view frusta of the display. These correspondence points are then used to solve the extrinsic calibration between each eye and the tracked HWD. Figure A.8 depicts a visualization of the stereo calibration model yielded by our implementation. The calibrated camera matrices for the left and right eyes are used as the corresponding view matrices in the left and right cameras in the Goblin XNA StereoCamera object with which the AR scene is rendered. We have tested this technique with dozens of users and found it to provide excellent results over a large tracking volume (~5 m³).

We use a modified version of this technique to calibrate the camera used for fiducial marker tracking (described in Section A.7.6). After calibrating the intrinsic parameters of the Point Grey camera [Tsai 1987], we substitute the video from the camera as the real-world
reference and repeat the extrinsic procedure as if calibrating for a single eye. This allows us to find the extrinsic calibration between the focal point of the camera and the origin of the tracked HWD.

Figure A.8: An example visualization of the extrinsic stereo camera model yielded by the Display Calibrator. The model shows the estimated locations (yellow spheres) of a user’s left and right eyes relative to the tracked origin of the NVIS HWD.
A.7.6 Hardware

A.7.6.1 Psychomotor Phase Prototype Implementation

This prototype uses an NVIS nVisor ST60 color, stereo, optical–see-through HWD (Figure A.3), which a user wears while completing procedural tasks. The display has a 60° diagonal field of view per eye, 40% optical transmissivity (of light from the natural task environment), and provides a $1280 \times 1024$ resolution image to each eye. The display is connected to an ATI Radeon HD5770 Eyefinity graphics card installed in a quad-core, 3.4GHz AMD Phenom II 965 powered desktop computer with 8GB RAM, running a 64-bit version of Windows 7. The side-by-side stereo pair rendered by our software interface (described in Section A.7.4) is mapped by the graphics card display manager to the separate left and right DVI channels of the NVIS HWD.

We support two kinds of optical tracking. The 3D position and orientation of the HWD and combustion chamber components are optically tracked at 100Hz using a cluster of 11 NaturalPoint OptiTrack FLEX:V100R2 and FLEX:V100 infrared cameras, mounted around the work area. Three retroreflective sphere markers are fixed to the back of the HWD to create a single rigid body as defined by the NaturalPoint TrackingTools application. The combustion chamber cones and cans are each outfitted with four markers per component, so that they are continually tracked within their range of motion.

We also experimented with the ALVAR optical tracking library [2011], which uses printed fiducial markers that we attached to each combustion chamber. We mounted a $640 \times 480$ Point Grey Firefly MV IEEE 1394a color camera on a fabricated ledge we attached to the NVIS nVisor ST60 (Figure A.3) and use this camera as input to the ALVAR library. We
found the tracking quality provided by ALVAR to be quite sufficient for tracking objects held in the user’s hands.

The implementation includes several input devices for supporting user interaction during procedural tasks, including a mouse, keyboard, Wiimote controller, and the wrist-worn smartphone described in Section A.6.5. The implementation also includes a single 4”-diameter button, connected via USB, which supports navigating forward to the next step in the task sequence and was used in the user study described in Chapter 4. This single-button user interface works well to simplify user interaction with the series of tasks chosen for our user study.

A.7.6.2 Opportunistic Controls User Observation Prototype

We leveraged portions of the Goblin XNA-based version of ARMAR to implement the prototype AR interface used in our user observation study described in Section 5.5. Because the prototype features untracked AR, this implementation was simpler than the other prototypes described in this chapter. The 3D scene and background video were rendered at 800×600 resolution. The Virtual Content Controller consisted of multiple 3D scenes for each of the hypothetical interfaces presented to subjects in the user study. These scenes were created using Goblin XNA Geometry and Transform Nodes. Each screen was then attached to the scene graph using a single Switch Node. The switch node is attached to a single transform node that is positioned and oriented by keyboard controls. This allowed the study observer to provide the “Wizard of Oz” feedback to the user.

The magic lens display described in Section 5.5.2 was implemented using a Xenearc 700TSV LCD panel with a back-mounted, untracked, Point Grey Firefly MV 640×480 reso-
olution color camera equipped with a 3.6mm microlens. A stand was provided nearby to hold the display, allowing the user to use both hands for any interaction that they demonstrated. No tracking hardware was used in this prototype because the 3D models rendered in this study were overlaid on, but not geometrically registered with, the physical environment.

Figure A.9: Users look through a hand-held VST AR magic lens display. The display presents unregistered 3D graphics that are manually aligned, in Wizard of Oz fashion, with back-ground video. The video is captured from a back-mounted video camera (visible in right image).


U.S. Army (2007). *Maintenance operations and procedures (Field Manual 4-30.3)*.


