Augmented Reality Interfaces for Enabling Fast and Accurate Task Localization

Mengu Sukan

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

COLUMBIA UNIVERSITY
2017

ABSTRACT

Augmented Reality Interfaces for Enabling Fast and Accurate Task Localization

Mengu Sukan

Changing viewpoints is a common technique to gain additional visual information about the spatial relations among the objects contained within an environment. In many cases, all of the necessary visual information is not available from a single vantage point, due to factors such as occlusion, level of detail, and limited field of view. In certain instances, strategic viewpoints may need to be visited multiple times (e.g., after each step of an iterative process), which makes being able to transition between viewpoints precisely and with minimum effort advantageous for improved task performance (e.g., faster completion time, fewer errors, less dependence on memory).

Many augmented reality (AR) applications are designed to make tasks easier to perform by supplementing a user's first-person view with virtual instructions. For those tasks that benefit from being seen from more than a single viewpoint, AR users typically have to physically relocalize (i.e., move a see-through display and typically themselves since those displays are often head-worn or hand-held) to those additional viewpoints. However, this physical motion may be costly or difficult, due to increased distances or obstacles in the environment.

We have developed a set of interaction techniques that enable fast and accurate task localization in AR. Our first technique, SnapAR, allows users to take snapshots of augmented scenes that can be virtually revisited at later times. The system stores still images of scenes along with camera poses, so that augmentations remain dynamic and interac-

tive. Our prototype implementation features a set of interaction techniques specifically designed to enable quick viewpoint switching. A formal evaluation of the capability to manipulate virtual objects within snapshot mode showed significant savings in time spent and gain in accuracy when compared to physically traveling between viewpoints.

For cases when a user has to physically travel to a strategic viewpoint (e.g., to perform maintenance and repair on a large physical piece of equipment), we present ParaFrustum, a geometric construct that represents this set of strategic viewpoints and viewing directions and establishes constraints on a range of acceptable locations for the user's eyes and a range of acceptable angles in which the user's head can be oriented. Providing tolerance in the allowable viewing positions and directions avoids burdening the user with the need to assume a tightly constrained 6DOF pose when it is not required by the task. We describe two visualization techniques, ParaFrustum-InSitu and ParaFrustum-HUD, that guide a user to assume one of the poses defined by a ParaFrustum. A formal user study corroborated that speed improvements increase with larger tolerances and reveals interesting differences in participant trajectories based on the visualization technique.

When the object to be operated on is smaller and can be handheld, instead of being large and stationary, it can be manually rotated instead of the user moving to a strategic viewpoint. Examples of such situations include tasks in which one object must be oriented relative to a second prior to assembly and tasks in which objects must be held in specific ways to inspect them. Researchers have investigated guidance mechanisms for some 6DOF tasks, using wide–field-of-view (FOV), stereoscopic virtual and augmented reality head-worn displays (HWDs). However, there has been relatively little work directed toward smaller FOV lightweight monoscopic HWDs, such as Google Glass, which may remain

more comfortable and less intrusive than stereoscopic HWDs in the near future. In our Orientation Assistance work, we have designed and implemented a novel visualization approach and three additional visualizations representing different paradigms for guiding unconstrained manual 3DOF rotation, targeting these monoscopic HWDs. This chapter includes our exploration of these paradigms and the results of a user study evaluating the relative performance of the visualizations and showing the advantages of our new approach.

In summary, we investigated ways of enabling an AR user to obtain visual information from multiple viewpoints, both physically and virtually. In the virtual case, we showed how one can change viewpoints precisely and with less effort. In the physical case, we explored how we can interactively guide users to obtain strategic viewpoints, either by moving their heads or re-orienting handheld objects. In both cases, we showed that our techniques help users accomplish certain types of tasks more quickly and with fewer errors, compared to when they have to change viewpoints following alternative, previously suggested methods.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First, I would like to thank my advisor Steve Feiner for his relentless support and mentorship throughout this journey. When Steve accepted me to join his Computer Graphics and User Interfaces (CGUI) Lab as a PhD student, I did not have a long working history in UI research, but he took a chance on me and trusted that my performance in his class was a good indicator of my passion for the field and eagerness to learn. During my time in his lab, I witnessed time and again how he tirelessly supported not only me, but all of his students, in every imaginable way; not only by guiding our research, sharing his incomparably deep knowledge of the field, and coaching academic writing, but also by going to great lengths to secure our funding and caring for our general well-being. I will endeavor to emulate his high standards and passion for excellence for the rest of my life.

Next, I would like to recognize Barbara Tversky, who provided such pivotal input to my research for such a long time, that I consider her an unofficial second advisor. We relied on her constantly to guide us in our designs with insights from her groundbreaking work in cognitive science and to help us ensure that our work had a solid foundation in cognitive principles. Barbara also supported me financially for a semester, hiring me as a visiting researcher in her department. For all of her academic and non-academic support, I will be forever in her debt.

I would like to express my gratitude to my committee members, John Kender, Shree Nayar, and Steve Henderson for their invaluable time and input. A lot of the ideas in my thesis were inspired by Steve's earlier work on augmented reality interfaces for procedural tasks and we have continuously used the Rolls-Royce Dart 510 aircraft engine he refurbished and generously donated to our lab as a prop in our demos and user studies. I consider myself lucky to have intersected with his time at CGUI.

My fellow CGUI lab members, Carmine Elvezio, Ohan Oda, Daniel Miau, Brian Smith, Sean White, and Lauren Wilcox, have been a great source of support and inspiration through this entire time and I would like to express my appreciation for their camaraderie. I especially thank Carmine for being my partner and co-author in many of the projects I have worked on and I recognize that without his user interface design and coding expertise, as well as his tireless work ethic, a lot of the work in this dissertation would simply not be possible. I also thank Ohan for spending a substantial amount of his research time developing the Goblin XNA framework, which provided us with the necessary infrastructure to build our projects on.

I thank the students who worked on my projects for research credit for their valuable contributions: Semih Energin, Minhaz Palasara, Yujin Ariza, Bin Shen, Morgan Thompson, Brian Wu, Nora Wixom, Rahul Tewari, and Vaibhav Malpani.

Last but not least, I am eternally grateful to my wife, Eva, and my parents, Sema and Macit, for their unconditional love, unwavering support, and endless patience.

Research in this dissertation was supported in part by NSF Grants IIS-0905569, IIS-0905417, IIS-0855995, IIS-1514429, and IIS-1513841, as well as generous gifts from VTT, Canon U.S.A. Inc., Microsoft Research, and Google.

Dedicated to my wife, Eva, and my parents, Sema and Macit

> Vita brevis, ars longa,
> occasio praeceps,
> experimentum periculosum,
> iudicium difficile.
>
> —— Hippocrates

Chapter 1

Introduction

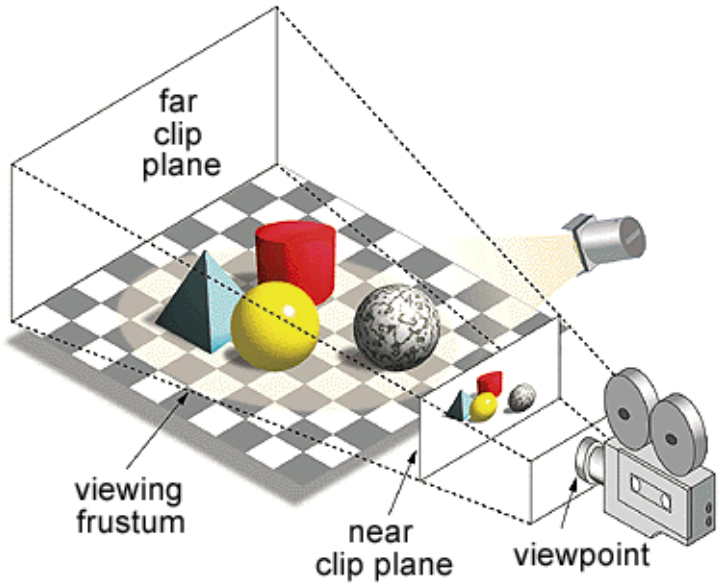


Figure 1.1: Illustration of a viewpoint in 3D computer graphics. Note that the blue pyramid and the red cylinder are partially occluded by the yellow sphere.[1]

In 3D computer graphics, viewpoints are often defined by viewing and projection matrices, which collectively encode the position and orientation of the user, as well as the parameters of the camera, such as viewport dimensions, field of view, and focal length (Figure 1.1). The 2D rendering from such a viewpoint contains visual cues that help users understand the spatial relations among the objects contained in the environment. However, the amount of information that can be obtained from a single viewpoint may be

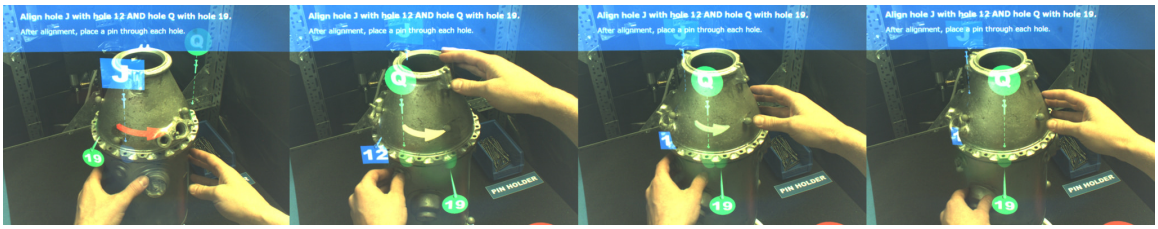
[1] http://www.pcmag.com/encyclopedia/term/61771/viewing-frustum

limited. For example, objects might fall outside the viewing frustum, be occluded by others, or be too small to be seen from a certain distance given the resolution of the rendered image. In these situations, it is crucial for the user to be able to change their viewpoint to obtain one or more additional views of the environment to accomplish a given task.

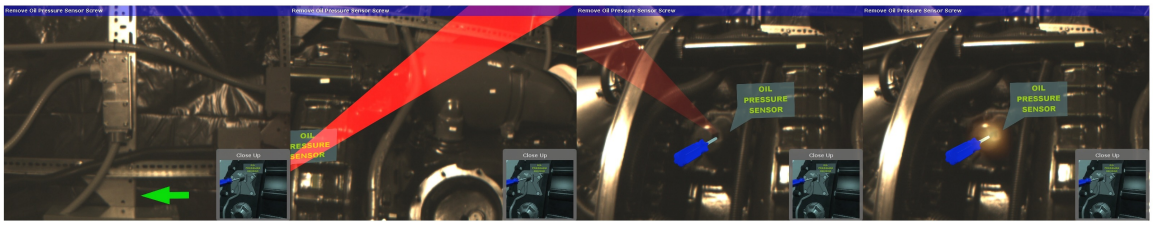
In the physical world, we can intuitively perform the physical motion necessary to position ourselves relative to the environment to change our viewpoint and obtain additional visual information. Similarly, physical motion of the user's body is a direct and natural way to change viewpoints (also known as traveling) in immersive virtual environments (VEs), such as a virtual reality (VR) application. This has the advantage of providing the user with proprioceptive feedback, at the expense of requiring time and effort. Virtually transitioning to an alternative viewpoint without a corresponding physical motion is a well-studied alternative means of travel in VR [Bowman et al. 2005]. While these nonisomorphic "magic" travel techniques are less natural, they allow the user to cover greater distances quickly, and with less effort.

In Augmented Reality (AR), by definition, a user's view of the real world is augmented with virtual content, combining real and virtual objects interactively in a cohesive manner [Azuma et al. 2001]. The combined real and virtual environments can be viewed on a display that is typically either head-worn or hand-held. In another approach, the virtual information can be projected directly on the physical objects to be augmented. With all these display modalities, one common thread is that virtual content is typically displayed from the user's natural, first-person, embedded perspective of the scene and the user has to physically move to different viewpoints to obtain additional visual information.

Having an unobstructed view of an object from a specific angle and distance can be



(a) AR assistance for psychomotor phase of a procedural task using dynamic 3D arrows and labels overlaid on a physical task object [Henderson and Feiner 2011].



(b) A typical localization sequence using an arrow (green) and a "rubberband" (red) [Henderson and Feiner 2009].

Figure 1.2: Examples of previous work on AR task assistance and localization.

particularly important when a physical task (e.g., assembly, inspection, repair) needs to be performed on that object. Providing instructions in AR (i.e., overlaying virtual cues and instructions onto physical objects) has been shown to improve task performance (e.g., Tang et al. [2003], Robertson et al. [2008], and Henderson and Feiner [2011], Figure 1.2a) and continues to be an active area of research. Some AR task assistance systems include attention direction techniques to *localize* the user to a specific object in the environment (e.g., Feiner et al. [1993] and Henderson and Feiner [2009], Figure 1.2b), but these localization techniques focus solely on guiding the user to turn in the direction of the task object.

In this dissertation, we present AR visualizations and user interface (UI) techniques to help users localize quickly and accurately by:

(a) enabling the user to store and virtually revisit previously visited viewpoints,

(b) directing the user to physically travel to strategic viewpoints while allowing a range of acceptable 6DOF poses, and

(c) guiding the user to reorient a handheld task object in 3DOF to obtain a strategic view of that object (instead of moving relative to the object themselves).

1.1 Research Questions and Dissertation Goals

While investigating how we can enable fast and accurate task localization in AR, we focused our efforts on answering the following research questions:

How can we show a user views from alternative viewpoints in AR? Since AR applications need to interface with the physical world by definition, travel is often achieved through physical locomotion by default. In Chapter 3, we start by exploring an alternative method for travel in AR (i.e., virtual travel) and answer several surrounding questions in the process: How can we decouple the viewpoint of the real objects in the scene and the registered virtual content from the physical camera that is controlled by the user? Before we can change a user's viewpoint, how do we obtain alternative views in the first place? If we can successfully obtain and store views from alternative viewpoints, how can the user get information on what viewpoints are available, and where they are relative to their current (physical or virtual) viewpoint? In Chapter 4, we continue our exploration and try to answer how we can effectively guide a user when they have to physically travel to a strategic viewpoint (e.g., to perform a task on a physical object). Finally, in Chapter 5, we take advantage of the fact that sometimes we need to operate on small, handheld objects and explore how we can effectively guide users to manipulate a handheld object to view

that object from a strategic viewpoint.

What visual aids and UI techniques can we provide to help a user who needs to transition to an alternative viewpoint? In Chapter 3, once we show how we can decouple the viewpoint of the user from their physical camera, we try to determine how we can represent the additional viewpoints that are available to the user. What UI techniques would be effective to let them to choose one of those additional viewpoints and trigger a transition to virtually visit that viewpoint? When the user has to move relative to the task object (Chapter 4), how can we guide a user to a specific position and get them to look in a specific direction (i.e., get them to assume a specific 6DOF viewing pose, not necessarily along a certain path)? Noting that increasing the precision required when assuming a 6DOF pose will also likely increase the amount of time and effort required to assume that pose, how can we introduce varying amounts of tolerance for position and orientation, depending on the level of precision required by the task at hand? Suppose that the user's destination is not inside the viewing frustum of their display (e.g., when the destination is outside their current field-of-view (FOV), behind the user, or closer to the user than the near clipping plane). How can we continue to provide the user with visual feedback in such cases? In cases where the user can reorient the task object (Chapter 5), what UI elements provide the most effective assistance for a nontrivial rotation task? What are the best strategies to provide real-time feedback to let the user correct their rotation path interactively?

What are the benefits of being able to transition viewpoints in AR? For all of our work, we strive to quantitatively measure how our UI techniques and visualizations affect user performance (e.g., completion time and task accuracy). In addition, we collect and present

qualitative data on user preference to uncover if there are potential disadvantages to switching among viewpoints or our guidance methods (e.g., disorientation or added workload from switching).

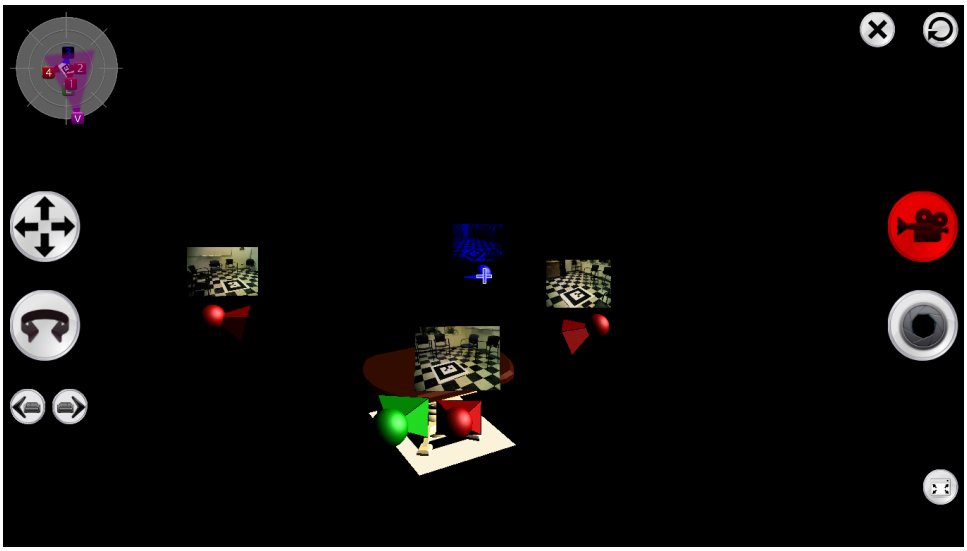
1.2 Contributions

In answering the questions framed above, we make the following contributions:

With SnapAR, we introduce a set of interaction techniques to quickly transition among virtual viewpoints [Sukan et al. 2012]. This is useful for tasks when strategic viewpoints have to be revisited multiple times (e.g., when trying different furniture layouts for interior design, Figure 1.3a). SnapAR allows a user to virtually revisit one of a set of previously saved viewpoints by simply pointing a handheld device at a 3D icon of the viewpoint embedded in the environment (Figure 1.3b). To obtain a collection of strategic viewpoints, we let the user take snapshots (photographs) of environments from strategic viewpoints using the handheld device. To accommodate snapshots that fall outside of the user's FOV, we developed a virtual overview mode, which mimics the motion of the user backing away from the scene until all snapshot icons are captured in the viewing frustum. We allow the user to manipulate virtual content from the perspective of a stored viewpoint while virtually revisiting it. We designed and performed a formal user study that showed participants can accomplish an AR alignment task faster and with fewer errors while using SnapAR than when using just the live AR mode. Moreover, participants preferred manipulating virtual objects using snapshots to the live mode.

With ParaFrustum, we introduce a geometric construct that represents a set of constrained

(a) Live mode



(b) Overview mode

Figure 1.3: Prototype SnapAR furniture layout application lets users view and manipulate virtual furniture in handheld AR using live and snapshot modes.

viewpoints and viewing directions [Sukan et al. 2014]. We present this construct to the user with a set of visualizations that help guide the user to meet those constraints by providing real-time visual feedback (Figure 1.4). This is useful for tasks in augmented or virtual reality that require users to view a target object or location from one of a set of strategic viewpoints to see it in context, avoid occlusions, or view it at an appropriate angle or

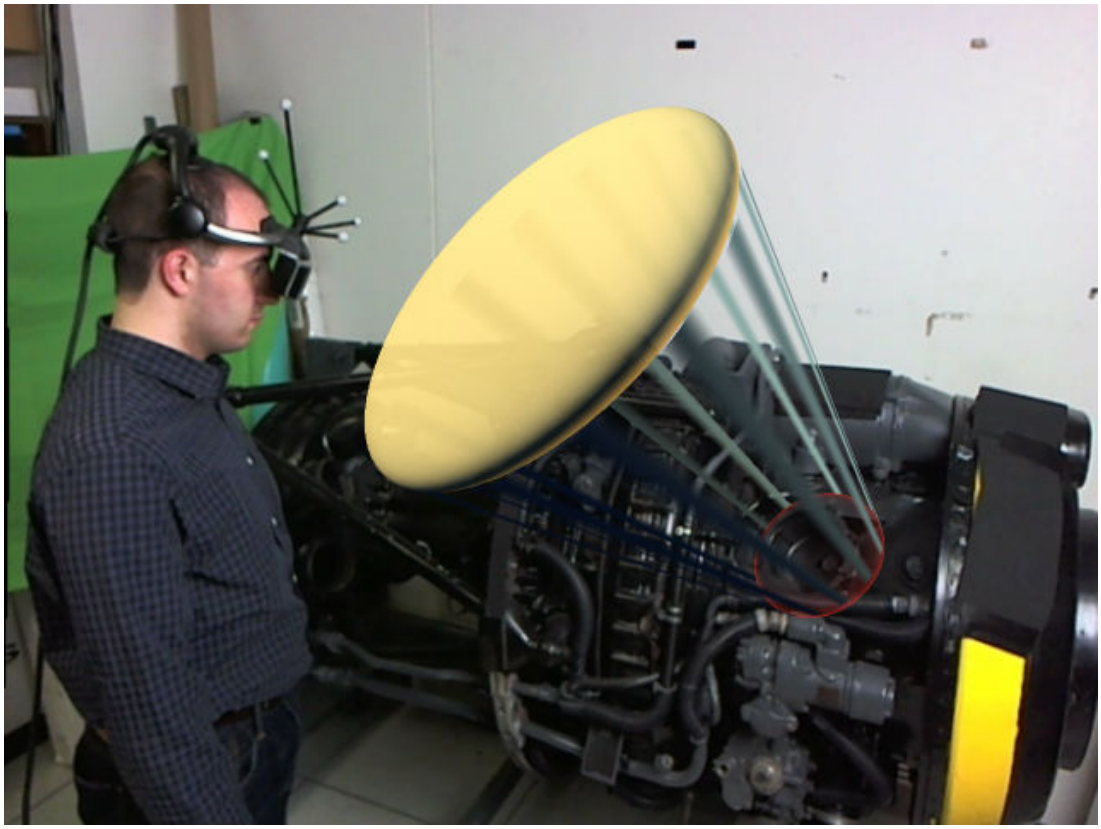


Figure 1.4: ParaFrustum defining a range of acceptable viewing positions and orientations, as visualized by the ParaFrustum-InSitu visualization.

distance. ParaFrustum is inspired by the look-from and look-at points of a computer graphics camera specification, which precisely delineate a location for the camera and a direction in which it looks. We generalize this approach by defining a ParaFrustum in terms of a look-from volume and a look-at volume, which establish constraints on a range of acceptable locations for the user's eyes and a range of acceptable angles in which the user's head can be oriented. Providing tolerance in the allowable viewing positions and directions avoids burdening the user with the need to assume a tightly constrained 6DOF pose when it is not required by the task. We developed two visualization techniques that guide a user to assume one of the poses defined by a ParaFrustum, and measured the performance of these techniques with a user study. The study showed that the constraints

of a ParaFrustum can be satisfied faster than those of a conventional camera frustum, corroborated that these speed improvements increase with larger tolerances, and revealed interesting differences in participant trajectories in response to the two techniques.

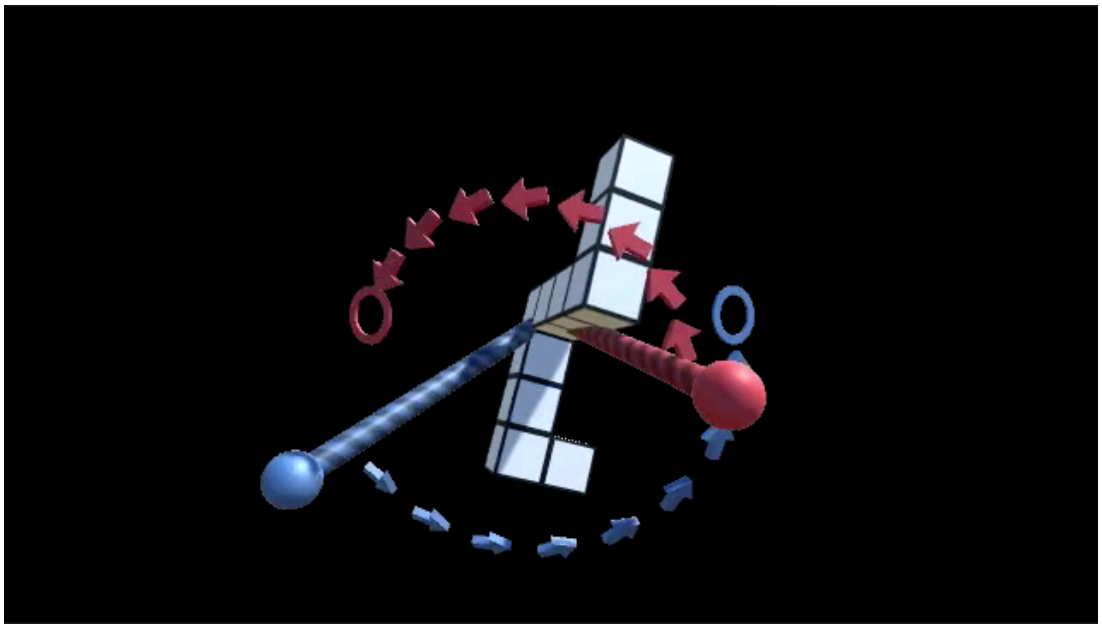


Figure 1.5: Screen capture of HANDLES visualization with persistent, clearly-visible alignment targets.

With Orientation Guidance, we present HANDLES, a novel interaction and visualization approach to provide real-time guidance for unconstrained 3D rotation of hand-held objects (Figure 1.5) [Sukan et al. 2016]. HANDLES overcomes shortcomings of existing approaches by providing persistent, clearly-visible alignment targets, and works well on lightweight, monoscopic, small-FOV HWDs. We show that users guided by HANDLES performed a nontrivial orientation task significantly faster compared to other techniques and tended to prefer it over the other techniques. Additionally, we describe three additional orientation-guidance approaches that are built using variants of common UI elements found in existing orientation-guidance systems (e.g., 3D arrows and animation) and detail how we carefully fine-tuned these approaches to improve their usability. We report results and

analysis from a formal user study that compares these four approaches with an unaided side-by-side representation of the static target orientation and a dynamic virtual proxy of the tracked object, addressing speed of performance (both overall speed, and the breakdown into initial ballistic rotation and subsequent fine-tuning), preference, and task load.

1.3 Structure of Dissertation

We continue this dissertation with Chapter 2, where we provide an overview of previous work on managing multiple viewpoints and providing task assistance in AR. In Chapter 3, we describe the design, implementation, and evaluation of *SnapAR*, a set of interaction techniques that allow users of handheld AR applications to virtually revisit previously stored viewpoints without having to physically travel back. Next, in Chapter 4, we introduce *ParaFrustum*, which generalizes the specification of an acceptable range of 3D viewing positions and orientations, and detail two visualizations along with a formal evaluation. In Chapter 5, we recount our investigation of the usability of various visualization techniques (e.g., arrows, animation) for a nontrivial 3D manual rotation task (rotation of a nearly-symmetric, nearly-featureless object about an arbitrary axis) and report effectiveness of various designs from a formal user study. Finally, in Chapter 6 we present our conclusions and discuss possible focus areas for future research. In the Appendix, we provide copies of the questionnaires given to participants to each of the formal evaluations described in Chapters 3–5.

Chapter 2

Related Work

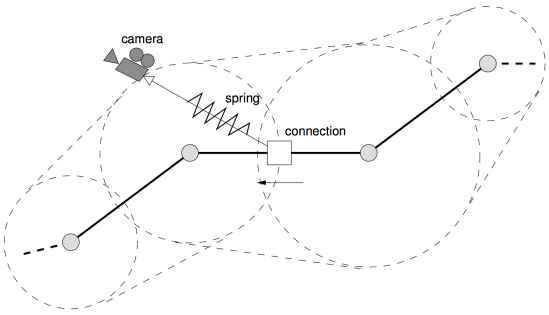
One way to organize previous work related to transitioning among viewpoints in both AR and VR is to partition it into five areas. In Section 2.1, we describe work that focuses on interaction and visualization techniques for switching among multiple viewpoints. Next, we summarize work on presenting multiple viewpoints simultaneously in Section 2.2. Then, we look at interaction and visualization techniques for saving and selecting viewpoints in Section 2.3. In Section 2.4, we summarize significant contributions that involve augmenting static images. In Section 2.5, we review work on guiding a user to physically transition to an alternative viewpoint. Finally, in Section 2.6, we give an outline of the large body of work exploring task assistance using AR.

We note that transitioning among multiple viewpoints is not the only way to gain additional visual information that may not be available from a single vantage point, due to factors such as occlusion, level of detail, and limited field of view. Elmqvist and Tsigas [2008] presented a taxonomy of occlusion management in visualization to uncover five design patterns: multiple views, tour planners, virtual x-ray, projection distorters, and volumetric probes. While we discuss examples of tour planners in Section 2.1 and multiple view and projection distorter designs in Section 2.2, we do not touch on volumetric probe and virtual x-ray designs in this overview.

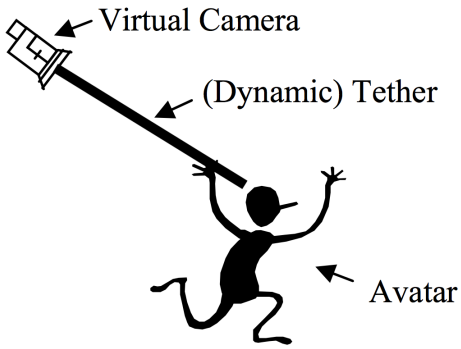
2.1 Switching Among Multiple Viewpoints

Switching viewpoints, especially as a means for locomotion, has long been an active research area in immersive virtual environments (VEs). Bowman et al. [1997] conducted some early evaluation on travel in VEs and concluded that motion techniques which instantly teleport users to new locations are correlated with increased user disorientation compared to techniques where the user moves along a path at a given velocity. It is important to note, however, that having a user move virtually in a VE without corresponding physical motion has been correlated with increased discomfort (i.e., cybersickness) for some users, especially when using wide-FOV displays [LaViola 2000]. Burtnyk et al. [2002] described an approach they called StyleCam that lets content authors determine strategic areas in a 3D scene where users control the virtual camera. When users reach the edge of these interactive areas, they are automatically transitioned to the next interactive area along a preset path without being overburdened with camera control in uninteresting areas.

Elmqvist et al. [2008] evaluated a method to offload some of the cognitive effort of 3D navigation from the users by partially constraining their movement (Figure 2.1a). Their technique used a spring-like tether that connected the viewpoint to a pre-defined path. This allowed the user to locally deviate from the pre-defined path as far as the virtual spring allowed. After the user was done exploring, the spring smoothly returned the user to the path. Compared to "free-flight" traveling techniques, they found that users achieved significantly better results in memory recall and performance when given access to such a guidance method. Chen et al. [2009] built on the guided tour idea by modulating variables



(a) 2D illustration of a constrained 3D motion traveling technique (circles show movable space, free of obstacles) [Elmqvist et al. 2008].



(b) Spatial relationship among dynamic tether, virtual camera, and avatar [Wang and Milgram 2001].

Figure 2.1: Examples of previous work on transitioning between viewpoints in VEs

such as velocity and FOV to improve landmark recognition at decision points along a path (e.g., intersections or turns). Work by Wickens and Prevett [1995] showed that local guidance is supported by greater egocentricism, while global awareness is supported by less egocentricsm, by observing pilots during simulated landings using either egocentric or exocentric viewpoints. Building on this concept, Wang and Milgram [2001] presented the concept of "dynamic viewpoint tethering," where the viewpoint is tethered (i.e., attached) to an avatar controlled by the user. The length of the tether (i.e., level of egocentricity) is adjusted automatically depending on user's speed to find the optimum tradeoff between

local guidance and global awareness (Figure 2.1b).

When studying users' preferences for the ideal visualization when searching for a nearby point of interest on a mobile device, Froehlich et al. [2008] found that users strongly preferred applications where the viewpoint is aligned automatically to the user's orientation compared to orientation-agnostic presentations. Additionally, they reported user preference towards an elevated perspective (as opposed to a first-person perspective), wide FOV, and 3D visualization of their surroundings, but not necessarily realistically textured.

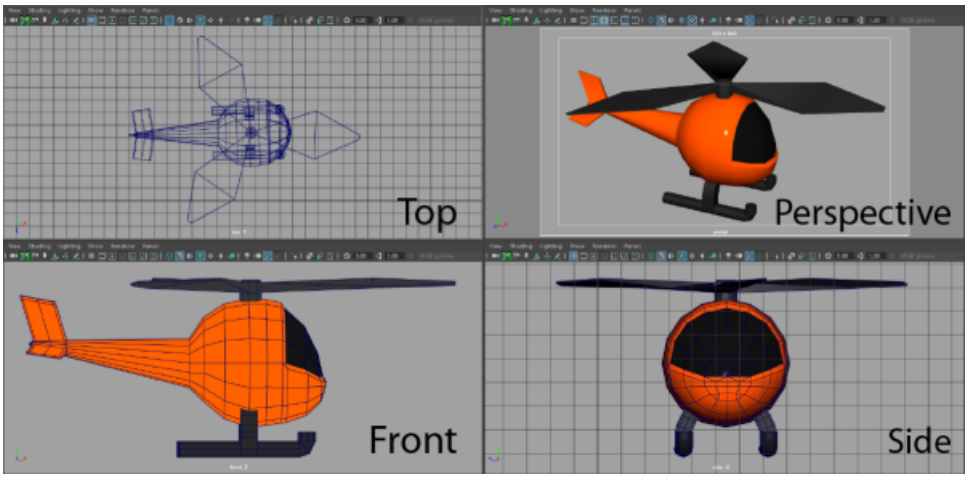
As mentioned in Chapter 1, AR applications need to interface with the physical world by definition and the viewpoint of the real objects in the scene and the registered virtual content is often tightly coupled with a physical camera that is controlled by the user. Switching viewpoints in AR requires either additional virtual content or additional physical cameras to provide context for the additional views. Grasset et al. [2006] describe AR applications that allow natural and continuous transitions between contexts (e.g., across space, scale, viewpoints, and representation) as having a transitional interface. Examples of transitional interfaces in AR can be found in games. Phillips and Piekarski [2005] explored a metaphor in which players could "possess" (occupy) virtual characters, which meant that they could quickly switch to their viewpoints without physically traveling (by transitioning from AR to VR). Cheok et al. [2002] also featured an AR–VR transition in a gaming context—users wearing HWDs are seamlessly transitioned to an immersive first-person VR view of an airplane cockpit after finishing a stage that is presented in a third-person AR view, where a small version of the virtual airplane is attached to and controlled by the user's handheld wand.

An example of a multi-camera setup in AR is provided by Veas et al. [2010], who compared techniques that relate local and remote cameras topologically, via a tunnel, or a via bird's eye viewpoint, in terms of enhancing the spatial awareness of the viewer. Mulloni et al. [2010] experimented with viewpoint switching in the context of browsing geo-referenced information in AR on a handheld device with their zooming interfaces. They present two zooming interfaces that enable users to smoothly zoom between the AR view and an egocentric panoramic view and an exocentric top-down view.

2.2 Presenting Multiple Viewpoints Simultaneously

Sketchpad III [Johnson 1963] (Figure 2.2a), generally accepted as the first interactive 3D computer graphics program, featured four simultaneous views of a 3D object in each quadrant of the screen: top, front, side, and perspective views. This convention is still used by 3D computer aided design software packages today (e.g., "Quad View" in Blender [Blender Online Community 2016], Figure 2.2b). For immersive virtual worlds, Stoakley et al. [1995] introduced the World-in-Miniature (WIM), which augments the HWD with a hand-held miniature copy of the VE that can be used for quick object selection, manipulation, and quick viewpoint switching. Pausch et al. [1995] later added an interaction to their WIM that let users change their viewpoint by grasping and moving a camera icon in the WIM. Viewpoint update was deferred until the user releases the camera icon, at which point the system interpolated the user's viewpoint to that of a doll in the miniature. Lorenz et al. [2008] experimented with the idea of combining multiple perspectives of 3D spatial environments into a single view in real-time to provide both focus and con-

(a) Sketchpad III [Johnson 1963].



(b) "Quad View" in Blender [Blender Online Community 2016].

Figure 2.2: Examples of interactive 3D computer graphics software with four simultaneous views of a 3D object in each quadrant of the screen: top, front, side, and perspective views.

text within the same view by deforming the space (e.g., combining a realistic view of the user's vicinity with a top view of distant areas).

Veas et al. [2012] explored the concept of combining multiple perspectives into a sin-

gle view in outdoor AR, where the augmentations (i.e., models of surrounding buildings) were deformed using a similar technique to the one used by Lorenz et al. Their work also featured a multi-view technique to access a multi-camera setup to allow the user observing the site from multiple perspectives without physically moving around. Bichlmeier et al. [2009] provided surgeons with additional perspectives using a tangible/controllable virtual mirror in specific medical AR applications.

Girgensohn et al. [2007], Ichihara et al. [1999], and Kameda et al. [2004] developed multi-view solutions with specific surveillance problems in mind, trying to help users maintain spatial awareness while switching between the feeds of multiple live cameras. Hoang and Thomas's "augmented viewport" [Hoang and Thomas 2011] is also a multi-viewport system for accessing live feeds from multiple cameras designed to improve manipulation precision of distant virtual objects in outdoor AR.

2.3 Saving and Selecting Viewpoints

The ability to save viewpoints to revisit them has been explored in VR. The X3D (previously VRML97) specification [Web3D Consortium 2014] defines a "viewpoint" node that content authors use to automatically move a user's view. Many X3D browsers (e.g., FreeWRL [Stewart 2014]) allow users to either jump or fly to these viewpoints by choosing from a textual list of author-specified viewpoint descriptions. Many popular 3D mapping (e.g., Google Earth [Google Inc 2016]) and 3D modeling software (e.g., Trimble Sketchup [Trimble Navigation 2016]) feature similar "guided tour" facilities that allow content authors to store a list of interesting views that can be viewed ("toured") by other users.

Elvins et al. [1997] enhanced their VRML browser by introducing 3D thumbnails called "worldlets" that could be interactively viewed in 3D and overlaid atop the main window to indicate their position and orientation, in addition to being used as a bookmark for travel like regular viewpoint nodes.

Schmalstieg et al. [1999] presented UI techniques for a virtual table based on a tracked hand-held pen and a transparent pad. Their system featured a snapshot mode that allowed a particular view of the scene to be locked on the pad, which could later be decoupled from the pad and left floating in the scene at any position. By strategically placing multiple such snapshots in the scene, a user could inspect multiple views at once.

Hirose et al. [2006] presented the tunnel-window technique, which allowed a user to create an arbitrary number of viewing windows at arbitrary positions inside a virtual environment and seamlessly interact with objects through those frames for travel and remote object manipulation.

"Photo tourism" by Snavely et al. [2006] presented an image-based modeling algorithm that can calculate viewpoints from a collection of photographs of a common scene along with a sparse 3D model of the scene. Their work featured a "photo explorer" front end that represented viewpoints as frusta shown relative to the 3D model of the scene and let users smoothly transition between photographs, while also enabling full 3D navigation and exploration of the set of images using their desktop UI. Since those systems are purely VR, however, they do not address physical objects in their environments.

2.4 Augmenting Static Images

Photo-based AR has been explored in industrial settings. For example, Georgel et al. [2009a] presented an interface for visualizing a set of images embedded in CAD software and techniques for navigating within this mixed reality environment. Siltanen and Woodward [2006] described a desktop-based, still-image augmentation system for interior design. For view selection and virtual object manipulation, these photo-based AR systems employ 2D GUIs.

Güven et al. [2006] and Lee et al. [2009] described AR systems that capture "frozen" views of the real world, as seen from strategic vantage points, to create temporary snapshots with which users can more comfortably interact than when viewing the real world directly.

2.5 Guidance for Physically Transitioning to a Viewpoint

There is a body of previous work exploring how to guide a user from a starting position to a specific destination in both real and virtual environments. Darken and Peterson [2001] and Smith and Hart [2006] presented findings on the cognitive impact of different UI elements (e.g., map, trail, compass) during wayfinding in VEs. Modified versions of these UI elements have also been used in AR. For example, Höllerer et al. [1999] described an early mobile AR system that lets a user wearing a HWD view a path outdoors.

However, guiding a user to a specific viewpoint is a 6DOF problem, as it requires a user to not only be at a specific location, but also look in a specific direction. There are many examples of techniques for getting a user to turn their head to look in a specific

direction. Feiner et al. [1993] featured a 3D leader line to help direct a user attention to certain objects in a AR maintenance and repair scenario. Biocca et al. [2006] presented Attention Funnel, a visualization technique that draws a user's attention down a funnel that is drawn along a curve that connects the user's head to a target location. Tönnis and Klinker [2006] showed that an egocentric, screen-fixed 3D arrow in AR combined with spatialized sound cues was faster at drawing a driver's attention than allocentric and visual-only alternatives. Henderson [2011] discussed how an AR maintenance and repair application may need to guide users to specific viewpoints. Similar to drawing a user's attention to off-screen items in 3D, there is a body of work on visualizations to represent off-screen items in 2D (e.g., Baudisch and Rosenholtz [2003], Gustafson et al. [2008], and Miau and Feiner [2016]).

Guiding a user to a specific 6DOF pose has been explored in the context of retaking a photograph (rephotography). Bae et al. [2010] aided users retake a historical photograph by displaying two 2D arrows as feedback while users positioned their camera based on continuously computing relative viewpoint difference using computer vision techniques. Shingu et al. [2010] presented a simple sphere and cone visualization for a tracked camera in AR to help with the task of retaking a photo in an industrial setting (e.g., for inspection of an item before and after a process). Their visualization aimed to ensure that a predetermined target region is within view and not occluded by other objects in the scene. There was no additional feedback once the user is within the specified thresholds, but they processed and distorted the retaken image to make it visually closer to the original photo. Güven and Feiner [2006] let users explore sites by visualizing historic photographs along with their calculated viewpoints registered in situ to enable users to assume the same

camera pose as used to take a historical photo.

2.6 Task Assistance Using Augmented Reality

Instructional manuals have long used arrows to depict rigid body transformations [Mijksenaar and Westendorp 1999]. This approach has been adopted in computer-based documentation systems, including ones targeting AR. For example, arrows can cyclically move in a direction in which an object is to be translated [Feiner et al. 1993] or interactively change in size and color to indicate direction and magnitude of a 1DOF rotation needed to align two tracked workpieces [Henderson 2011]. Ghosting is another common visualization technique used in real-time AR task guidance systems to visualize workpiece placement (e.g., [Gupta et al. 2012]). Ghosting and animation have also been used to provide visual hints on how to move (e.g., reel or shake) hand-held props to activate gestures in an AR system [White et al. 2007]. Oda et al. [2015] used an annotation-based solution to guide a user to match a 6DOF pose specified by a remote subject matter expert.

AR interfaces have been developed to guide users in matching gestures and poses with parts of their body. Freeman et al. [2009] assist users in learning multi-touch gestures on a touchscreen by showing a partial shadow of the user's hands on screen. Sodhi et al. [2012] guide a user in translating a single hand using a 3D arrow, a 3D path, or colored regions indicating movement direction projected directly on the user's hand. Anderson et al. [2013] guide a user in moving their body by displaying augmentations over a mirror image of the user. Their visualization includes both a simple skeletal representation of the user's current and target poses, and a ribbon indicating the path the user should follow to

achieve the target pose. For rotating objects, 3D applications on desktop systems typically use separable rotation control widgets for one or more axes (e.g., [Schmidt et al. 2008]).

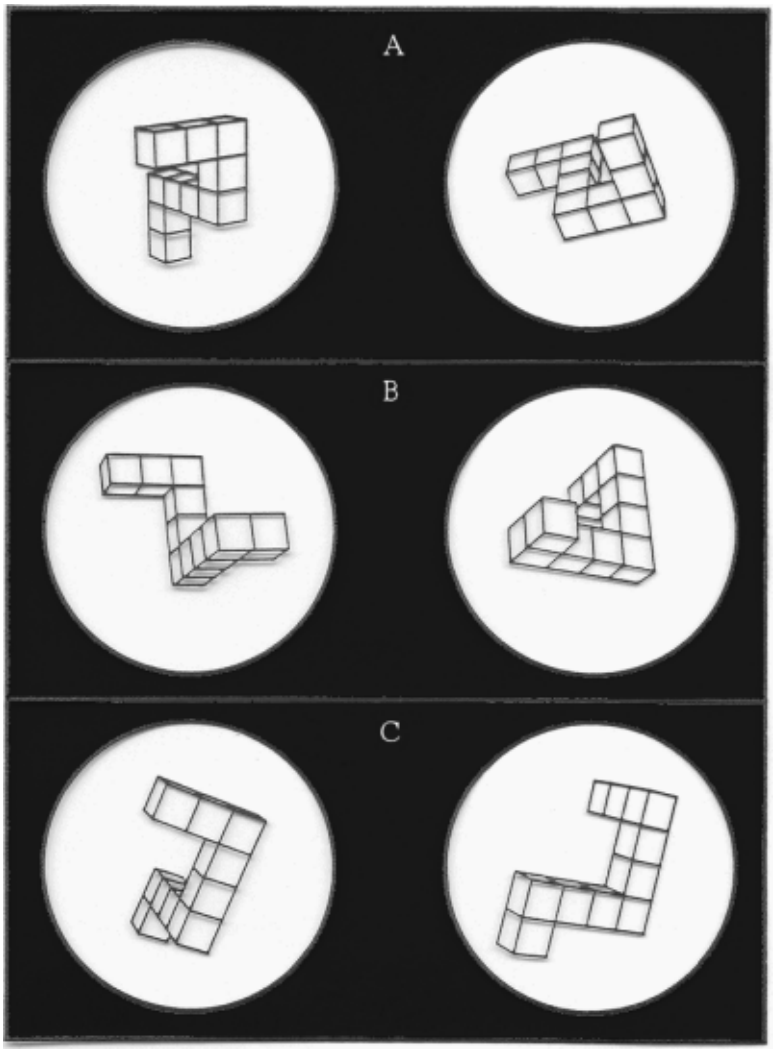


Figure 2.3: Stimulus figure pairs used by Shepard and Metzler [1971]. (a) Identical objects differing by a rotation in the plane of the page. (b) Identical objects differing by a rotation in depth. (c) Mirror-image objects.

There is a large body of research on how quickly and accurately people can imagine 3D rotations. In their seminal work in this area, Shepard and Metzler [1971] showed subjects two perspective drawings of an asymmetric 3D object and asked them to determine whether both drawings showed the same object, only rotated (Figures 2.3a and b), or two distinct, mirror-image objects (Figure 2.3c). Later research showed that people often spon-

taneously rotate a hand when solving mental rotation problems and that when they move their hand in the most efficient direction they perform better, but when forced to move their hand in the opposite direction they perform worse. That is, moving one's hands in a conceptually congruent way helps the user perform a mental transformation [Chu and Kita 2008; Chu and Kita 2011; Wexler et al. 1998; Wohlschläger and Wohlschläger 1998].

In the following chapters, we present our contributions that build on this body of previous work to extend the benefits of being able to change viewpoints precisely, quickly, and with little effort to users of AR applications.

Chapter 3

SnapAR

3.1 Introduction

Figure 3.1: Prototype furniture layout application lets users view and manipulate virtual furniture in handheld AR using live (pictured) and snapshot modes.

Observing an environment from different viewpoints is a common technique used to gain additional visual information about that environment, notably the spatial relations of the objects contained within it. Being able to control camera pose is important in many applications in which all of the necessary visual information is not available from a single vantage point, due to factors such as occlusion and field of view. In certain cases,

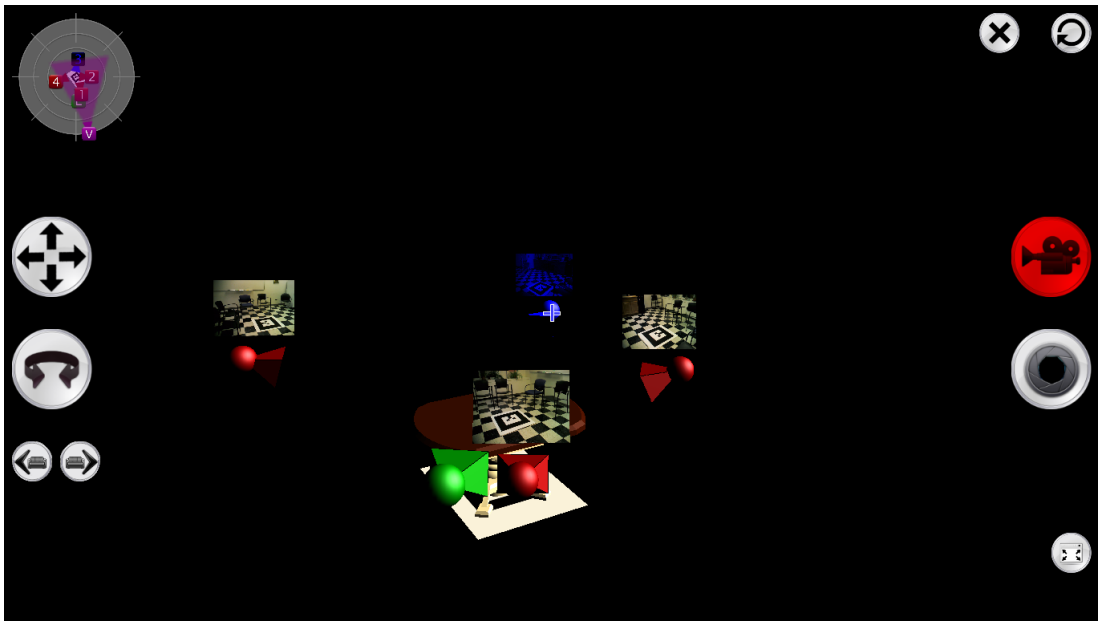


Figure 3.2: Overview mode renders available snapshots for selection.

viewpoints may also need to be visited multiple times. Consider, for instance, an interior designer who is trying to arrange furniture in a room and needs to see the arrangements from many points of view. If virtual furniture is instead being laid out in AR, as shown in Figure 3.1, the need to view the environment from multiple viewpoints is no less compelling; indeed, switching viewpoints may even be more frequent in AR, given the ease with which the designer can swap virtual furniture in and out, in comparison with physical furniture. Examples of other relevant domains include urban design, landscaping, architecture, disaster relief, military operations, and equipment maintenance and assembly tasks. All of these domains can benefit from rapid viewpoint changes, because they require pinpointing objects from different viewpoints. Such precision and speed is especially difficult to achieve in large and complex environments in which viewpoints are distant and/or challenging to reach.

Changing viewpoints in an environment is a well-studied means of travel in 3D user

interfaces. It is generally accepted [Bowman et al. 2005] that physical motion of a user's body is a direct and natural way to travel in virtual environments, with the advantage of providing the user with proprioceptive feedback. However, moving not only takes effort, but also time, which can lead to short-term memory loss. In navigating real environments, people often use maps to avoid unnecessary movement. Nonisomorphic "magic" techniques [Bowman et al. 2005] may be less natural, but allow users to explore greater distances quickly with less effort. Moreover, considerable research in cognitive psychology has shown that people are able to "jump" and reorient from viewpoint to viewpoint in imaginary environments without smooth transitions (e.g., [Tversky 2005]). Since AR applications must interface with the physical world, travel is often achieved through physical locomotion by default with the same advantages and disadvantages as in VR. However, unlike in VR, AR environments are real and may have obstacles that make physical motion even more problematic.

In this chapter, we present *SnapAR*, a quick viewpoint switching approach that extends the benefits of "magic" traveling techniques to AR applications. With our prototype implementation [Sukan et al. 2012], users can use a handheld device to take snapshots (photographs) of environments from different viewpoints, select from previously saved snapshots to virtually revisit those viewpoints without having to physically travel back, and even manipulate virtual content while viewing the world from the current viewpoint or revisiting a previously saved viewpoint (Figure 3.1). In addition to a set of interaction techniques to enable quick viewpoint switching (such as the virtual overview of snapshots shown in Figure 3.2), we also present a formal evaluation of the capability to manipulate virtual objects while viewing previously saved static snapshots. A within-subject user

study shows that participants can accomplish tasks that involve aligning a virtual object with real objects significantly faster using SnapAR than when physically moving between viewpoints, even in a relatively small working space and after taking into account the additional time needed to create the necessary snapshots, and with no loss of accuracy. Furthermore, participants overwhelmingly preferred the quick viewpoint switching approach and found it less demanding.

3.2 Related Work

The ability to save viewpoints to revisit them has been explored in VR. Elvins et al. [1997], Schmalstieg et al. [1999], and Hirose et al. [2006] present snapshot tools that let users manage a collection of 3D views from different viewpoints. As in our case, changes to the VE are reflected in all views simultaneously. Since those systems are purely VR, however, they do not address physical objects in their environments.

Switching viewpoints, especially as a means for locomotion, is an active research area both in VR (e.g., Pausch et al. [1995]) and AR (e.g., [Cheok et al. 2002; Phillips and Piekarski 2005]). In contrast to us, Phillips and Piekarski [2005] decided against using smooth transitions in their possession metaphor between each possession command, citing delay as a concern. We attempt to address that concern by allowing both smooth and instant transitions based on users' comfort with their spatial orientation.

Hoang and Thomas [2011]'s "augmented viewport", a multi-viewport system for accessing live feeds from multiple cameras, allows users precisely manipulate distant virtual objects in outdoor AR. Like us, they cite providing users with novel viewpoints as

their main motivation; however, in addition to architectural differences, they design and evaluate their system to improve manipulation precision, not to save time when quickly changing viewpoints.

Our approach differs from previous work on augmenting static images (e.g., Georgel et al. [2009a], Georgel et al. [2009b], and Siltanen and Woodward [2006]) by enabling users to take and navigate among snapshots while immersed in the physical environment. Being in situ gives users the freedom to obtain novel views on the fly, which may be important for exploratory and iterative tasks such as arrangement and planning. Additionally, for view selection and virtual object manipulation, these photo-based AR systems employ 2D GUIs, whereas our users manipulate and point a handheld device in 3D. In contrast to previous AR systems that capture "frozen" views of the real world (e.g., [Güven et al. 2006; Lee et al. 2009]), we support quick viewpoint switching amongst a set of snapshots, and manipulating objects within any selected snapshot.

3.3 Interaction

We build on the body of related work described above to further explore magic traveling techniques in AR. We started by creating a prototype to test and demonstrate the usefulness and efficiency of our quick viewpoint switching technique.

As we mentioned earlier, SnapAR allows users to take snapshots of the environment from different viewpoints, select from previously saved snapshots to virtually revisit those viewpoints without having to physically travel back, and even manipulate virtual content while revisiting a previously saved viewpoint.

3.3.1 Creating and Storing Snapshots

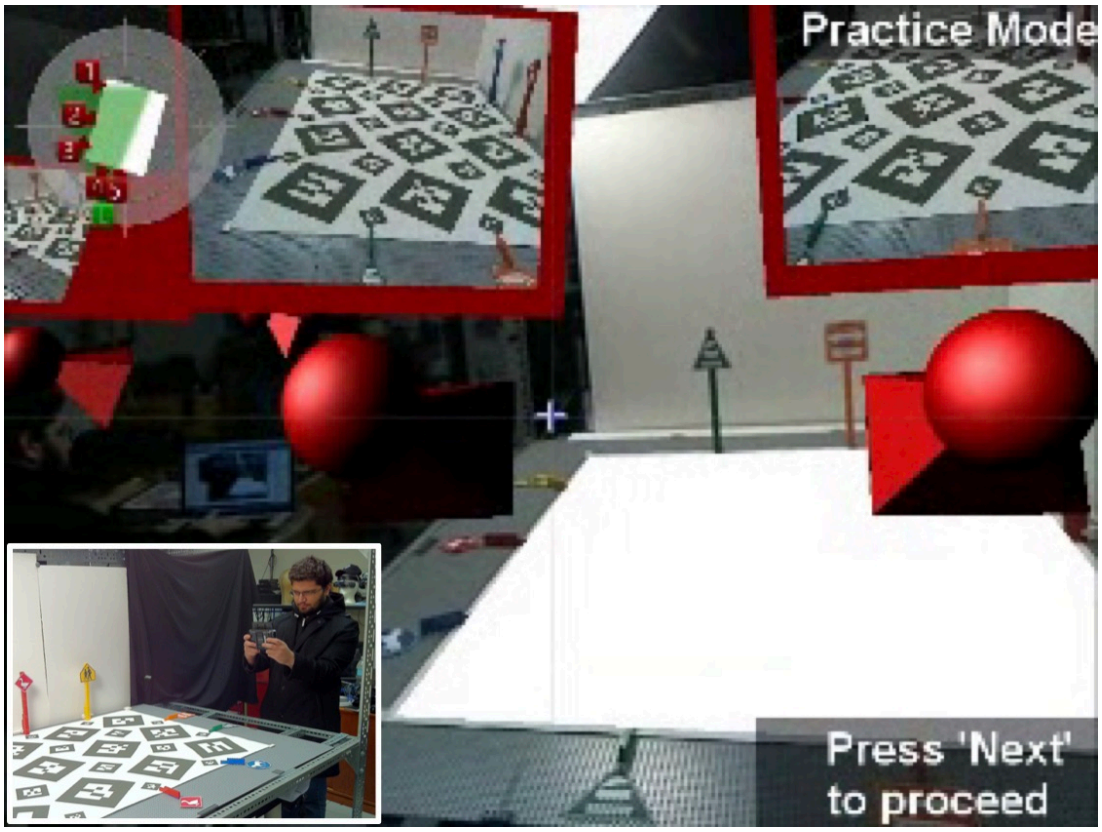


Figure 3.3: Live mode. User study setup includes physical landmarks and virtual snapshot representations. Inset shows user holding device.

Creating snapshots is similar to taking a still photograph. When users click a dedicated button on the handheld device, the current frame of the video feed is stored as a 2D texture. To enable overlaying the picture of the scene (from the snapshot viewpoint) with up-to-date virtual information, the 3D position and orientation of the handheld device, as determined by the tracking software, are also stored. To provide users with visual feedback of the camera position and orientation for each the snapshots, we add a virtual 3D camera icon to the scene for each snapshot, as shown in Figure 3.3.

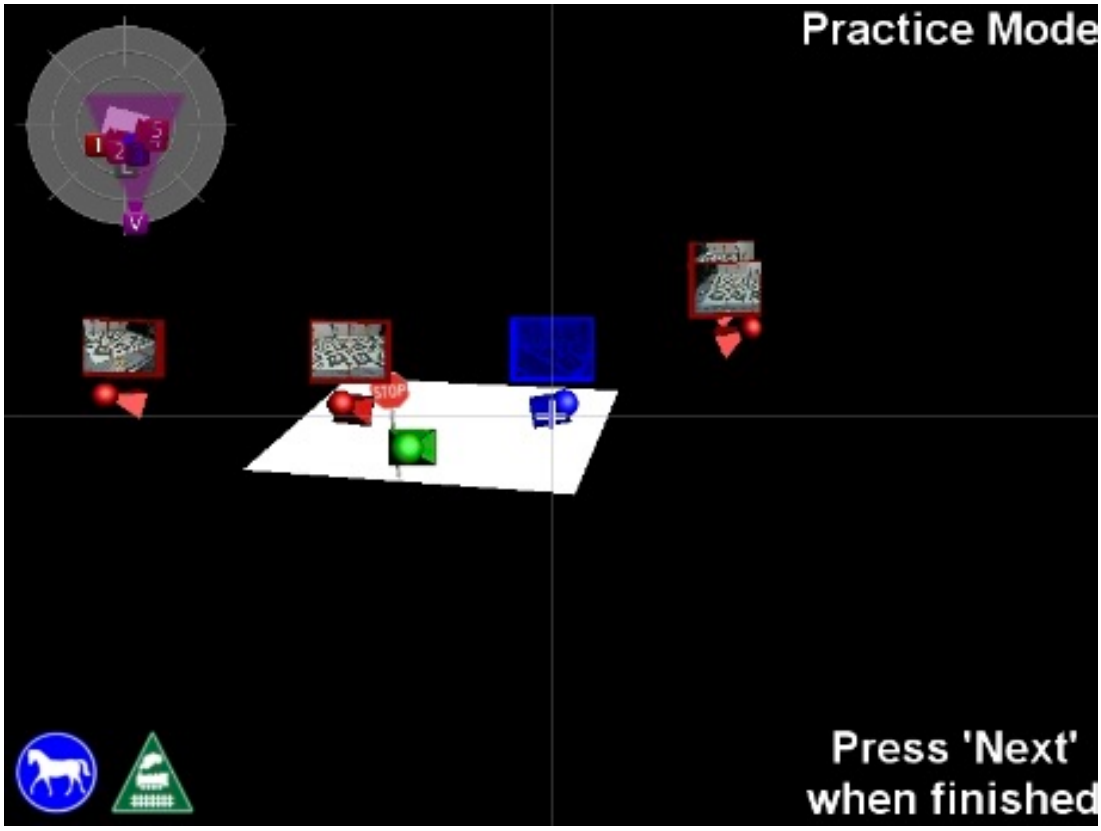


Figure 3.4: Overview mode. Highlighted snapshot (blue) is closest to the crosshairs.

3.3.2 Selecting and Viewing Snapshots

With these virtual snapshot representations attached to the ground marker array, users are able to see the 3D locations and orientations of available snapshots when viewing the environment through the handheld device. During preliminary testing, we noticed that a natural way to gain an overview perspective is to take a few steps back from the ground marker array to capture more of the environment in the viewing frustum. Although intuitive and effective, this method requires additional physical effort and time, both variables that quick viewpoint switching is intended to reduce. Additionally, moving away from the ground marker array poses a challenge to our implementation because of the negative impact of increased distance on optical marker tracking performance.

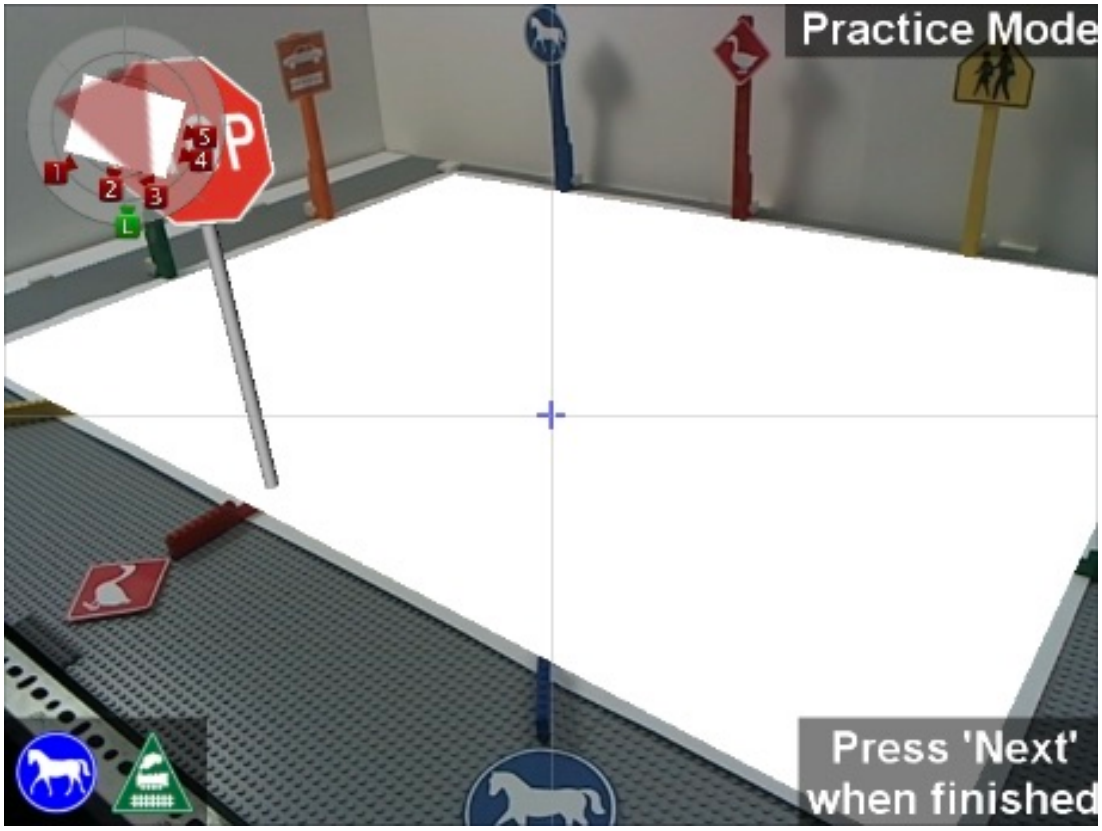


Figure 3.5: Snapshot mode. View after transitioning to selected snapshot.

We address these issues by providing users with a virtual overview mode, as shown in Figure 3.4. In the virtual overview mode, users control a virtual camera that mimics the motion of a user walking away from the ground marker array by translating the virtual camera back from the ground marker until all snapshot icons are captured in the viewing frustum (while maintaining the same orientation). Since the virtual camera is no longer co-located with the physical camera, we fade the live camera image out and show the virtual objects (ground plane and virtual objects, including a 3D camera icon for each virtual snapshot) against a black background. Even though the live camera feed is not shown to users, it is still used for tracking the handheld device, so that users can move the virtual camera by either translating or rotating the handheld device.

To speed the selection process, we designed the interaction so that users press a dedicated button to go into overview mode, align the desired camera with the crosshairs on the screen, and release the button to be taken to the snapshot, as shown in Figure 3.5. Our motivation for choosing this selection technique was two-fold:

1. it can be executed in one quick and fluid motion, contributing to overall time savings and
2. it allows the user to continue holding the device comfortably with both hands (i.e., less strenuously compared to holding it with a single hand) and leaving the thumbs over buttons for further actions

To reduce the likelihood of selecting the wrong snapshot (especially in cases when multiple snapshots project close to each other in screen space), we provide visual feedback to users by changing the color (from red to blue) of the snapshot camera icon that is nearest to the crosshairs in screen space.

Since our technique causes users to change viewpoints without physically moving their heads or bodies, we strive to strike a balance between providing smooth transitions among viewpoints and transitioning the camera at a reasonably fast pace. To calculate a smooth path, our implementation interpolates position and orientation variables separately. An additional translation relative to the origin is applied after translating the virtual camera towards the target snapshot to essentially turn a linear path between snapshots into an orbital one that rotates around the ground marker. This motion mimics a user's physical path around the table ground marker array. All interpolations are implemented as graceful (slow-in, slow-out) transitions based on cubic interpolation.

Because taking a snapshot stores only a single frame of the video feed at that location, we do not have any visual information to display to users about the background during camera transitions. We try to build on the smooth transition concept by fading the background image to black at the beginning of the transition and fading the background image back in as soon as the virtual camera arrives at its destination. By introducing a very brief time delay (.25 sec) before activating overview mode, we allow the same button to also serve as a quick switch button if it is simply clicked instead of being held down. If there is a snapshot icon near the crosshairs in the live mode, the quick switch functionality transitions users to that snapshot; otherwise users are "beamed" (transitioned) to the most recently used snapshot. When beaming, the only visual effect is the fading in and out of the background images. By varying the transition times, we allow advanced users to arrive at places more quickly, while allowing novice users to still enjoy the benefits of smooth transitions.

3.3.3 Heads-Up Display

To address the situation in which users might lose spatial awareness when looking at one of the snapshots and need reminding of the snapshot location in the physical space, we added a virtual heads-up display (HUD) that includes a 2D, top-down "radar" visualization of the locations of the snapshots (Figures 3.3–3.5, top left). In this HUD view, snapshots and their viewing directions are projected to 2D and represented iconically. To help users maintain spatial awareness, the HUD view shows the current location of the handheld device, as well as the active snapshot. Since the HUD view is always oriented forward-

up, when a snapshot is highlighted to the right of the current camera in the HUD view, this indicates that the view on the screen is from a snapshot that is to the right of the user's physical location.

3.3.4 Manipulating Virtual Objects

While exploring how we could use our quick viewpoint switching technique to view and augment an environment from different vantage points, we realized it could be useful and effective for situations in which users not only view the virtual content, but also manipulate it. A practical example is the interior design case mentioned in Section 3.1. After visually evaluating several pieces of virtual furniture from various locations, a designer might wish to try other arrangements and orientations of furniture. To do so, the designer would need to manipulate the furniture while viewing scenes from the different vantage points.

For manipulation, we wanted to maintain some consistency with how users select snapshots to view. Similar to the snapshot selection technique, there is a dedicated button on the handheld device for initiating the virtual object manipulation mode. When users press the manipulation button, the system stores the handheld device's pose matrix and virtual object's pose matrix relative to the ground marker as D_{initial} and O_{initial}, respectively. As long as users hold down the manipulation button, the relative transform between the handheld device's active pose D_{current} and D_{initial} is added to O_{initial}. This technique essentially lets the virtual object mimic the motion of the handheld device. Consequently, when users want to translate the virtual object by some amount and

then rotate it by some other amount, they simply press the manipulate button to "grab" the virtual object, translate and rotate the handheld device in the desired direction by those same amounts, and release the virtual object by releasing the button.

Similar to the motivation for choosing our selection technique, we preferred this manipulation technique because it can be quickly executed and allows both the user to holding the device comfortably with both hands. In the manipulation context, establishing a grabbing metaphor and letting users move a tangible object such as the handheld device has the added benefit that it does not conflict with the notion that the virtual object being manipulated has a meaning within and a connection to the physical environment surrounding the user.

Having the virtual object mimic the motion of the handheld device has certain drawbacks in AR, because the screen and camera also move along with the device. For example, when rotating the handheld device, the camera will eventually point away from the location of the virtual object. However, since our grabbing metaphor is triggered with a button press, a user can easily release (i.e., declutch) the virtual object by releasing the rotation button before the object gets out of view, rotate the device in the opposite direction, and grab the object again by pressing the button. During translation, rigidly attaching the virtual object to the handheld device hinders the user from obtaining depth information due to motion parallax, which could be a disadvantage for tasks where depth plays an important role.

Giving users 6DOF control of a virtual object may be useful in certain cases, but for our prototype virtual furniture placement application, shown in Figures 3.1-3.2, we found it helpful to constrain the motion of the virtual object by disallowing translation along the

up-axis and allowing rotation only about the up-axis (yaw). Thus, the virtual object cannot float in the air or be rotated around any other axis. In the present version, translation was decoupled from rotation by providing separate buttons for each transformation.

Manipulation is performed the same way in both live mode and snapshot mode. However, there is an interesting difference: While the grabbed virtual object moves with the handheld device in both modes, the background image remains static in snapshot mode, rather than updating continuously from the camera feed in live mode. Our formal evaluation described in the next section showed that this mismatch between the static snapshot background and the dynamic motion of the handheld device and virtual objects during manipulation did not affect task performance negatively. We discuss advantages and disadvantages of viewing virtual content on a static image of the scene in Section 3.6.

3.4 User Study

We designed a user study to compare physically walking to new viewing locations (our control condition, Walk) and switching viewpoints virtually in hand-held AR using the quick viewpoint switching technique (the Snap condition). Prior to conducting the formal user study, we performed an informal pilot study with our lab members and nine compensated students to confirm our design, formulate our hypotheses, and test our study procedure.

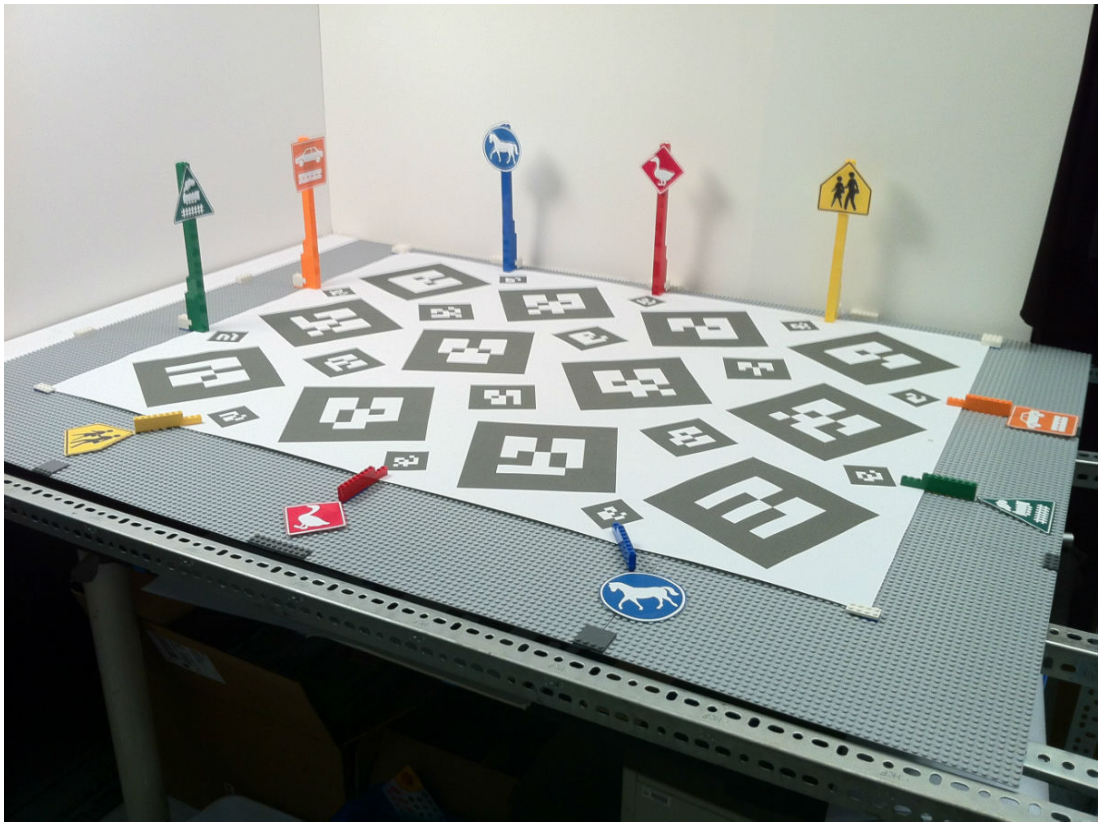


Figure 3.6: Five pairs of physical props around a table. Props are redundantly coded using color, symbol, and shape.

3.4.1 Pilot Study

Nine participants (3 female; ages 19–29, $\bar{X} = 24.8$) were first recruited for a pilot study designed to finalize the design of our approach and of the experiment, and elicit feedback about the usability of our technique. The participants were recruited by mass email to Computer Science students at our university and by flyers distributed throughout campus, and paid $15 for participating. All participants in the pilot study reported using a computer multiple times per day, and all passed the Ishihara Color Test.

The participants' task was to visually align a virtual object at the exact point of intersection of two imaginary lines. Five imaginary lines were defined by placing matching pairs of physical props along the edges of a 6' wide $\times$ 4' deep table, as shown in Figure 3.6.

We picked a relatively small work area to test whether our interface can perform better than walking even when the physical distance to be traveled is short. The maximum walking distance between viewpoint pairings was $56'' + 34'' = 90''$ and the minimum walking distance was $16'' + 20'' = 36''$ (Figure 3.7).

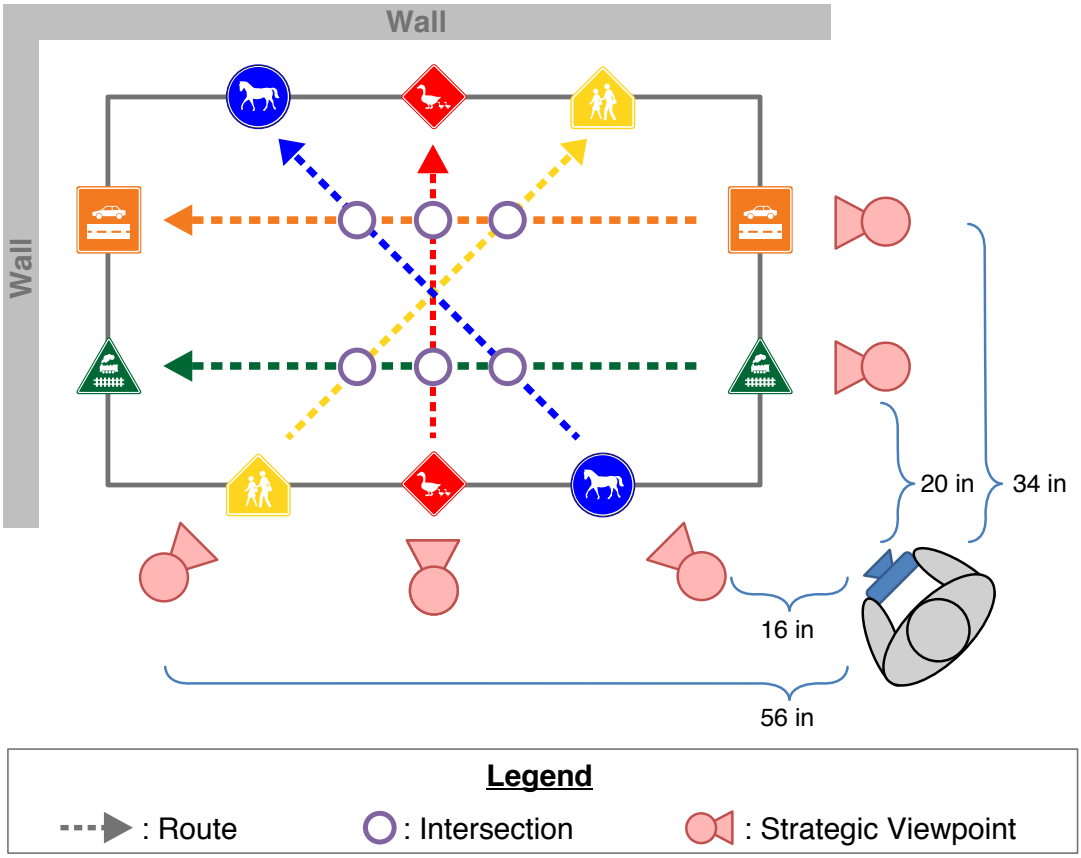


Figure 3.7: Five imaginary routes established by connecting matching physical props around a table.

We chose this task because visual alignment greatly benefits from strategic viewpoints. One plausible and common strategy to confirm that a movable object is placed on a straight line that connects two stationary reference objects is to move to and view the scene from a location that is collinear with the reference objects. Once the viewer is in such a collinear position, the task of aligning is reduced to a one-dimensional translation

problem in the direction perpendicular to the original path. Because alignment needs to be done for each imaginary line, it requires checking two viewpoints. To make the situation more natural, we compared two kinds of alignment tasks, along orthogonal axes (ORTHO intersection type) and along oblique axes (OBLIQUE intersection type).

To measure the efficiency of our user interface in selecting from several available strategic viewpoints, we placed three pairs of reference points along the long edge of a table and two pairs along the short edge, providing a total of five strategic viewpoints from which to choose. The physical reference points on the table were labeled using letters (A, B, C) and numbers (1, 2), and the virtual object to align was an abstract multicolored column. This resulted in sample tasks such as lining up the virtual object with the imaginary lines for B and 2. In addition, the alignment task included a 1D rotational component. Participants were asked to rotate the alignable object around its up axis (perpendicular to the table) until the color on the face of the object visible from the current viewpoint matched the color of the pair of physical objects associated with that viewpoint.

Our pilot study was blocked by condition, with a break between blocks. Each block consisted of two consecutive randomized sequences of the six possible intersections resulting in 12 trials per block. Learning and fatigue effects were controlled by counterbalancing the starting condition.

We computed a 2 (Travel Condition) $\times$ 6 (Intersection Location) repeated-measures ANOVA on the completion times. Travel condition had no significant effect on completion time ($F(1, 8) = 1.26, p = .294$). Intersection Location was significant as a main effect at $\alpha = .05$ ($F(5, 40) = 5.303, p < .001$). Looking further into the data, we noticed that we could simplify our study design by splitting our six intersections into two groups: ones

with orthogonal angles that are easier to align and ones with oblique angles that are more difficult (see Figure 3.7).

An analysis of alignment error (defined as the distance between the alignable virtual object at the time users submitted their answer and the true intersection position), revealed that on several occasions users positioned their alignable object at the wrong intersection. This led us to rethink the way we presented tasks to our users. We hypothesized that users may have been confused by the tasks because they had no recognizable meaning. This led us to develop the "protect the living creature from the vehicle" backstory that we would use in our formal study. Additionally, we explained the task more thoroughly before each trial. We added a live task preview screen before each trial that showed users the starting position and end goal position of the alignable object, as well as overlays of the lines that make up the intersection that is their objective.

A few pilot participants noted that the rotational subtask was confusing, that they had a hard time understanding the exact orientation of the virtual alignable object on the small screen of the handheld device. Based on this feedback, we excluded the rotational subtask from our final study.

The qualitative feedback from the pilot study revealed strong user preference for Snap compared to Walk, with all but one participant preferring Snap. One participant was concerned that the snapshot cameras "would be difficult / confusing to navigate, but actually this method was very user friendly". Another participant responded similarly: "I like [the Snap] approach better overall, although the learning curve was steeper." A different participant noted that "I was surprised that I preferred the Snap method. When both methods were being described and shown to me, I thought Walk seemed very sim-

ple and SNAP was overly complicated. But when actually performing the tasks, SNAP was significantly simpler to operate." Based on the feedback that the SNAP condition appeared complicated, we adjusted the introductory text, and reduced the number of buttons necessary to operate the SNAP mode from 6 to 3.

3.4.2 Hypotheses

Based on an analysis from our pilot study, we formulated the following five hypotheses:

H1. *SNAP will have a faster completion time.*

H2. *SNAP will have improved accuracy.*

H3. *ORTHO intersections will be faster and more accurate than OBLIQUE intersections.*

H4. *SNAP will be preferred over WALK.*

H5. *Participants will report less physical exertion for SNAP in the post-study questionnaire.*

Rationale

H1, H4, H5: Because people are able to reorient quickly to new viewpoints without smoothly transitioning to them and because SNAP saves time and effort by eliminating the need to walk to new viewpoints. Therefore, we expected that participants would perform faster using SNAP, prefer using SNAP, and report less physical exertion in the post-study questionnaire.

H2: Because the still images of the scene stored as snapshots in SNAP are sufficient to perform the task and immune to variability due to human error (e.g., hand tremor), we predicted that participants would be more accurate using SNAP.

H3: It is well known that perception and judgment are superior at recognizing orthogonal axes, rather than oblique ones, and that perception and memory are systematically distorted toward encoding spatial relationships as orthogonal, even when they are not (e.g., Howard and Templeton [1966] and Tversky [1981]). Therefore, we expected Oblique intersections to require more repetition than Ortho ones, which would mean more intuitive, faster view switching should have a pronounced effect on task completion time.

3.4.3 Methods

Participants

To test these hypotheses, we recruited 21 participants (8 female; ages 19–40, $\bar{X} = 23.6$) from the same target population as that of our pilot study (protocol: IRB-AAAF2995). None of the study participants had taken part in the pilot study or had any prior experience with the experimental technique. All but two participants used a computer multiple times per day. Ten participants identified themselves as having some familiarity with AR. Three participants reported playing video games daily, nine reported playing weekly, six reported playing monthly, and two reported never playing video games. Two participants failed the Ishihara Color Test, but were retained because they reported being able to distinguish references using the redundant cues. Each participant experienced both conditions, as described in the Design subsection below.

Equipment

Hardware. Our initial prototype runs on a Sony VAIO UX-VGN-380N Ultra Mobile PC (UMPC), which is a 1.2 lb., $5.9''$(W) $\times$ $3.7''$(H) $\times$ $1.3''$(D) hand-held device with a $4.5''$ diagonal LCD screen and an integral camera in the back of the device. The UMPC runs Windows XP on a 1.33 GHz Core Solo CPU with 1 GB memory, and an Intel 945GMS graphics chip. Setting the backbuffer to SVGA resolution (800×600) and camera to 640×480 resolution, our application performed at 20–25 frames per second (fps). To improve performance, we reduced the camera resolution to 320×240, while leaving the backbuffer at SVGA resolution. This sped up the tracking process considerably, letting us achieve 45–50 fps. Since we were able to control the lighting in our lab (we used two softbox lights), we conducted our user study at the lower resolution camera setting, achieving smooth rendering and animation.

After performing our study, we ported our application to a Samsung Series 7 XE700T1A Slate PC (shown in Figures 3.1 and 3.2), which is a 1.9 lb., $11.7''$(W) $\times$ $7.2''$(H) $\times$ $0.5''$(D) hand-held device with a $11.6''$ diagonal LCD screen and an integral camera in the back of the device. The Slate PC runs Windows 8 on a dual-core 1.6 GHz Intel Core i5-2467M CPU with 4 GB memory, and an Intel HD3000 graphics chip.

Software. Our implementation is developed using Goblin XNA [Oda and Feiner 2014], a managed, DirectX-based framework for constructing AR applications, built on top of Microsoft XNA Game Studio 4.0. 6DOF position and orientation tracking is provided by the ALVAR [VTT 2011] optical tracking library, using a ground marker array containing one or more optical fiducial markers. However, our quick viewpoint switching technique

could also be used on other devices (e.g., a head-worn display or smartphone) or with other tracking technologies.

Design

We designed a within-subject, repeated-measures experiment consisting of two travel conditions (Snap, Walk) and randomized iterations of the virtual object alignment task. The experiment was blocked by condition with a break between blocks. To help our participants understand and remember the task, we introduced a background story that the imaginary lines represent paths of vehicles and animals: the three pairs along the long edge were labeled with signs bearing symbols of living creatures (children, duck, and horse) and the two pairs along the short edge were labeled with signs with symbols of vehicles (car and train).

Each trial was defined as a combination of one of the three living creatures and one of the two vehicle types, resulting in six possible intersections to which the stop sign could be aligned. Each block consisted of two consecutive randomized sequences of the six possible intersections resulting in 12 trials per block. Learning effects were controlled by counterbalancing the start condition.

Before starting the experiment, participants signed consent forms and were screened for color blindness using the Ishihara Color Test. Afterwards, participants were shown slides with instructions for the experiment and for operating the handheld device to accomplish certain tasks, such as moving the virtual stop sign or switching to a particular view. Participants were told to work as quickly and accurately as possible. Participants were then given six practice trials to acclimate them to the handheld device and its func-

tionality. When participants began the timed portion of the following 12-trial block, the software recorded key presses, device motion, and virtual object motion. The handheld device's position and orientation relative to the ground marker array and the position of the virtual object was sampled every 100ms and recorded for post-hoc analysis of user movement and interaction. When the participant pressed the button on the handheld device labeled "Next", completion time and alignment error were recorded before moving on to the next trial.

After the first block, participants completed brief evaluations that requested qualitative feedback on the experience. After a five-minute break, the experimenter instructed participants for the other condition. Participants were given six practice trials for that condition, and then proceeded to complete 12 timed trials for the condition, followed by a second brief evaluation, including questions requesting rankings of the two conditions.

For the Snap condition, participants were shown how to take snapshots and change among them as part of the description of that technique. They practiced taking snapshots during the untimed practice block for the Snap condition, and then took a new set of snapshots, one for each of the five living-creature and vehicle paths, for use in the timed Snap condition block.

Procedure

For ease of recognition and differentiation, we redundantly encoded the signs for each pair of physical reference points using a unique color and shape. A participant's task was essentially to "make way for ducklings" [McCloskey 1941]; that is, to make sure that the vehicles did not collide with the living creatures. The horse and children cross the table

diagonally, resulting in oblique angles, as opposed to the duck, which makes orthogonal angles with the car and train, which cross the table horizontally.

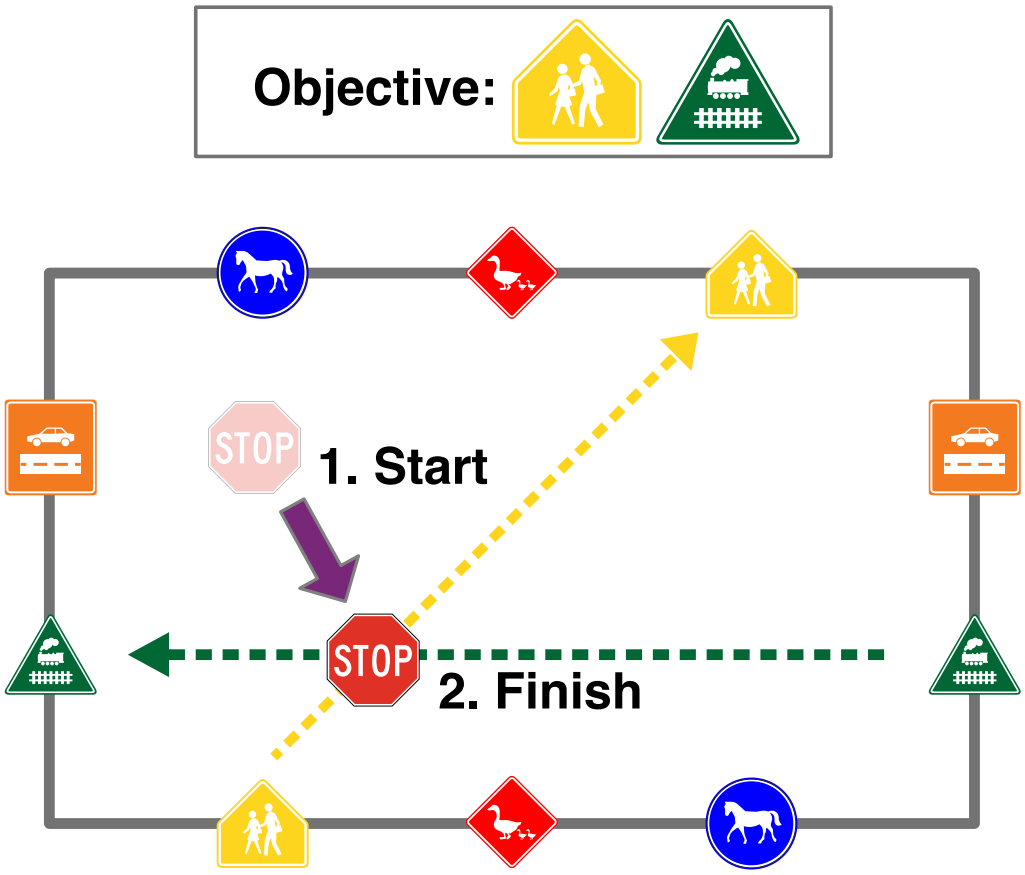


Figure 3.8: Illustration of a sample task: Protect children from train by moving stop sign from starting position to intersection.

At the beginning of each trial in the study, a participant was shown two symbols: one of a living creature, together with one of a vehicle. Participants were told that they needed to ensure the safe crossing of the living creature by placing a virtual stop sign at the exact intersection of the two paths (e.g., protect the children from the train), as shown in Figures 3.7 and 3.8. Completing this task successfully requires participants to travel to a strategic location collinear with the first path, move the virtual stop sign to be aligned with the path, travel to a second strategic location on the adjacent side of the table

collinear with the second path, move the virtual stop sign to be aligned with the second path, and repeat traveling between these strategic viewpoints as necessary to fine-tune the alignment of the stop sign.

We took several steps to ensure that alignment could not be achieved from a single viewpoint (e.g., by using other external references in the scene). To avoid giving users straight lines that could be used as guides for aligning, we varied the sizes, positions, and orientations of the markers in the array placed on our table (Figure 3.6). In addition, we rendered a solid virtual rectangle over the array to obscure it and separated the task area from the rest of the lab by placing solid white cardboard walls and black curtains on two sides of the table (Figures 3.3,3.5,3.6) to ensure that a user viewing the virtual objects on the display could not align it using any real artifacts except for the physical props that define the routes. We also drew crosshairs on the screen to aid with alignment, making it even more advantageous to look through the device when aligning. Careful examination by pilot study participants and us found no loopholes.

Note that the movement operation afforded full 2D motion of the stop sign on the plane of the table, so that moving the stop sign to align it with one pair of physical objects could bring it out of alignment with the other pair of physical objects, requiring that at least one vantage point be revisited. We intentionally did not provide a 1D translation command in order to make this task more similar to one involving aesthetic judgment from multiple vantage points, in which the vantage points would typically be revisited, but with an objective quantitative measure of accuracy.

3.5 Results

As a first step, we looked for potential outliers in our data. One of our participants reported on one occasion making a mistake and submitting his alignment by pressing the wrong button. Several other data points looked suspicious because they were extremely short (0.9 and 9 secs) and did not involve any virtual object movement. One participant reported the disappearance of the virtual stop sign, presumably a tracking error. These outliers, which accounted for 1.04% (1.66% of SNAP; 0.42% of WALK) of all calculated completion times across 2 conditions $\times$ 12 trials, were removed from further analyses. Hypotheses were then evaluated for significance with a Bonferroni-corrected α of $.01$ ($.05/5$).

3.5.1 Completion Time

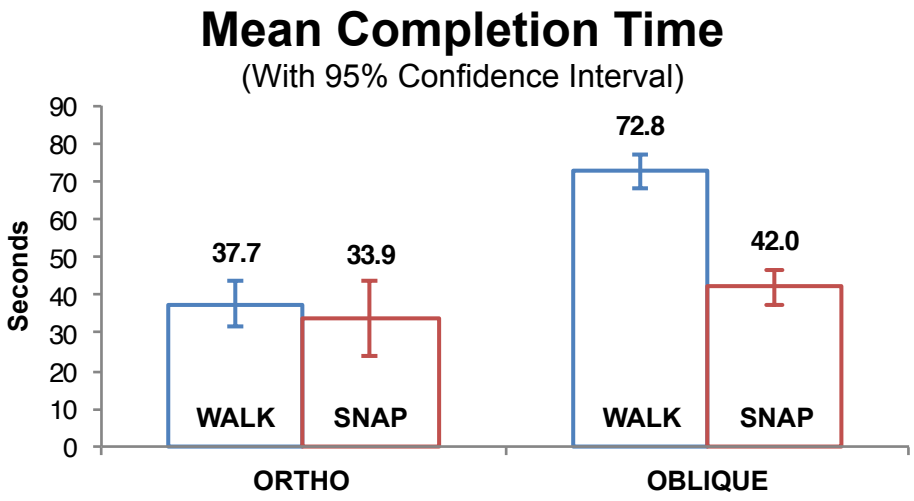


Figure 3.9: Mean completion time (in seconds) across conditions.

We performed a 2 (Travel Condition) $\times$ 2 (Intersection Type) repeated-measures ANOVA on completion times, with participants as the random variable. Results from the ANOVA

showed that Travel Condition was a significant main effect, $F(1, 20) = 21.99, p < .001$, at $\alpha = .01$. Our users completed the alignment task significantly faster switching viewpoints via SNAP (39.30 secs, excluding time to create snapshots) than WALK (61.04 secs), validating H1. Mean completion times for each condition are depicted in Figure 3.9.

It might seem unfair to compare completion times between SNAP and WALK without taking into account the time to create the snapshots, because snapshot creation is a necessary preceding step to using the snapshots in the SNAP condition. Creating snapshots can be seen as akin to automating a manual process. When one automates a task, it usually takes longer than just performing the task manually. Automating the task becomes a sunk cost. The return on investment comes when one can reuse that automation and amortize the sunk cost across many uses. The same is true for snapshots. Participants took 54.15 secs on average to create the five snapshots required for the task (10.83 secs per snapshot). The two snapshots along the short edge (SE) of the table were needed in six trials each and the three snapshots along the long edge (LE) of the table were needed in four trials each. When amortized across uses, the cost of creating a snapshot was 1.81 secs for a SE snapshot and 2.71 secs for a LE snapshot. Since each of the 12 trials required one LE and one SE snapshot, the total amortized cost of creating snapshots amounted to 4.51 secs per trial. Taking this additional time into account, SNAP was still significantly faster ($p < .001$) than WALK (17.23 secs or 28.2%). Obviously, the return on investment in creating the snapshots would scale up with the number of uses for each snapshot, as well as the amount of walking required between snapshots.

Our other main effect, Intersection Type, was also significant, $F(1, 20) = 125.70, p < .001$, at $\alpha = .01$. OBLIQUE intersections took significantly longer on average (57.49 secs)

than ORTHO intersections (35.72 secs), validating the first half of H3.

Finally, our analysis revealed a significant interaction between Travel Condition and Intersection Type, $F(1, 20) = 48.27, p < .001$, at $\alpha = .01$. It turned out that the time savings from traveling via SNAP add up to a much larger impact for OBLIQUE than for ORTHO intersection, presumably because the alignment task requires more repetition for OBLIQUE intersections.

3.5.2 Accuracy

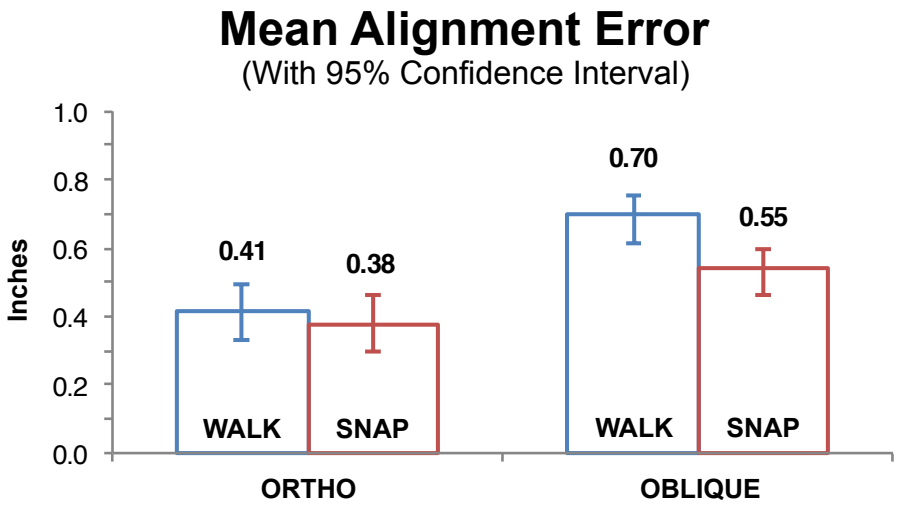


Figure 3.10: Mean alignment error (in inches) across conditions.

We performed a 2 (Travel Condition) $\times$ 2 (Intersection Type) repeated-measures ANOVA on alignment errors, as defined in Section 4.2. Results were very similar to those for our previous dependent variable, completion time. Travel Condition was just above our Bonferroni-adjusted $\alpha = .01$, $F(1, 20) = 6.50, p = .019$. Our users made smaller alignment errors when switching viewpoints via SNAP (0.49 inches) than WALK (0.60 inches),

supporting, but not confirming H2. Mean alignment errors for each condition are depicted in Figure 3.10.

Our other main effect, Intersection Type, was significant, $F(1, 20) = 122.98, p < .001$, at $\alpha = .01$. Participants made significantly less alignment error at ORTHO intersections (0.40 inches) than at OBLIQUE intersections (0.62 inches) validating the second half of H3.

Finally, our analysis revealed a significant interaction between Travel Condition and Intersection Type, $F(1, 20) = 15.89, p = .001$, at $\alpha = .01$. It seems that the ease and speed of travel via SNAP allowed users to iterate many more times when completing an OBLIQUE task, resulting in less error. For ORTHO, the impact is not as pronounced, probably because fewer repetitions are enough to achieve high accuracy.

3.5.3 Questionnaire

After each block, participants completed an unweighted NASA TLX [Hart and Staveland 1988] questionnaire comprising six seven-point Likert-scale questions (1 = most positive, 7 = most negative) to evaluate mental demand, physical demand, temporal demand, performance, effort, and frustration. Additionally, they were asked two ranking questions, first to rank the conditions based on preference for use and second, least overall demand (mental, physical, and temporal). Their responses were analyzed for significance with post-hoc Wilcoxon Signed-Rank comparisons with Bonferroni correction for 8 tests ($\alpha = .05/8 = .006$).

Participants reported perceiving a reduction in mental and temporal demand, as well as an increase in performance using SNAP compared to WALK. However, these effects were

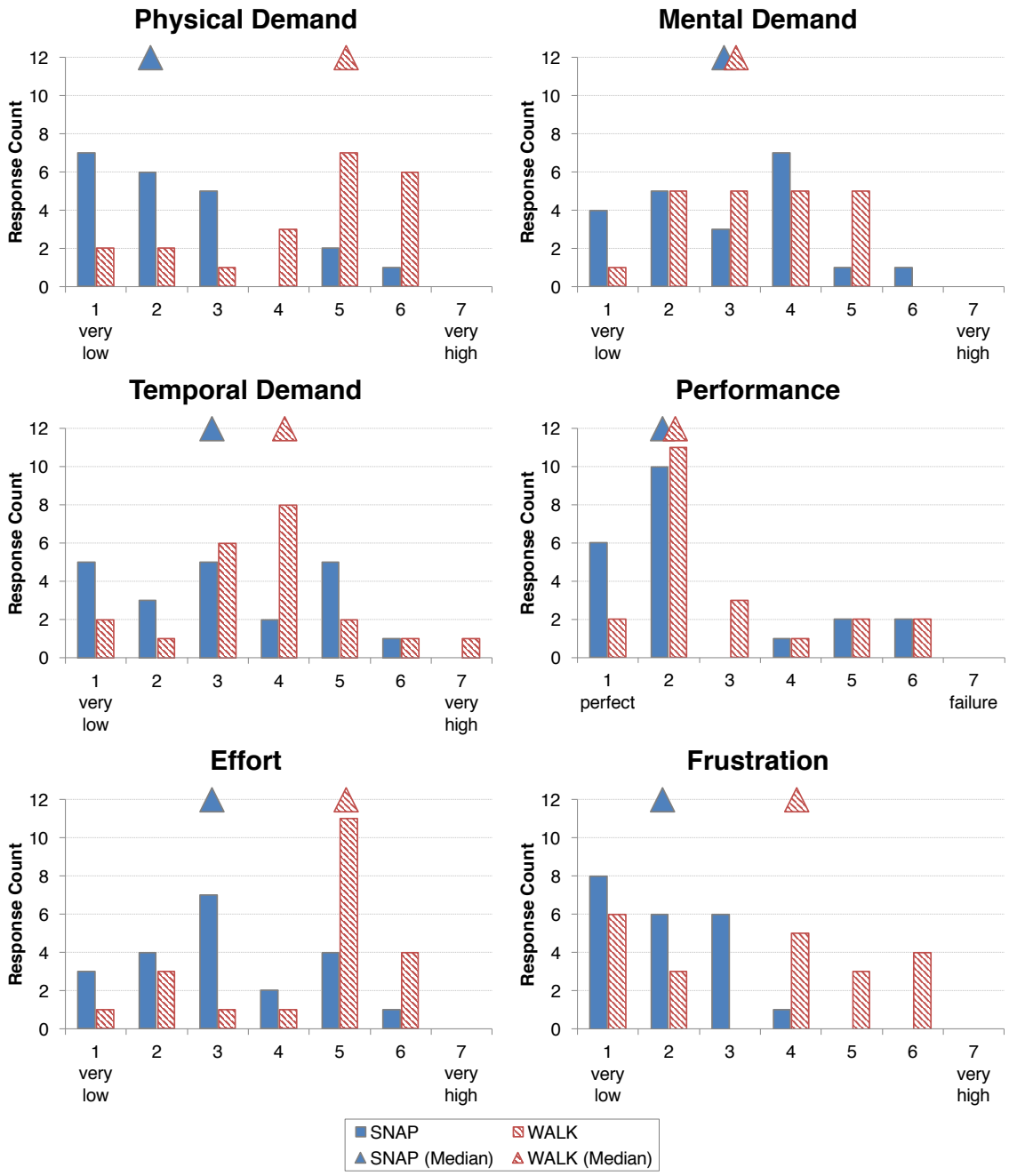


Figure 3.11: Questionnaire response histograms by condition. Median values are displayed as diamonds.

not significant (mental demand $Z = -1.358, p = .174$; temporal demand $Z = -1.536, p = .124$; and performance $Z = -1.658, p = .097$). Participants also reported perceiving a reduction in physical demand, effort, and frustration when using Snap compared to

Walk. These effects were significant, validating H4 (physical demand $Z = -3.598, p < .001$; perceived effort $Z = -3.094, p = .002$; frustration $Z = -2.842, p = .004$). Raw response data from all 21 participants for significant effects are presented in Figure 3.11.

When asked to rank the conditions based on preference for use, a significant proportion of participants (19 of 21) ranked Snap first ($Z = -3.710, p < .001$), validating H4. When asked which form of travel was overall less demanding (mental, physical, and temporal), the proportion of participants who ranked Snap first (17 of 21) was also significant ($Z = -2.837, p = .005$), which was consistent with H4 and H5.

3.5.4 Usage Pattern Analysis

To delve deeper into the usage patterns of our quick snapshot switching technique, we developed an analysis tool to process, analyze, and visualize the log data captured by the handheld device participants used.

	Walk	**Snap**	Total
Ortho	2.5	6.5	4.0
Oblique	6.0	8.9	7.5
Total	4.9	8.1	

Table 3.1: Average number of view switches per trial by condition.

This analysis tool revealed that in the Walk condition, participants switched viewpoints on average 4.9 times per trial, whereas in the Snap condition, participants switched their views, albeit virtually, on average 8.1 times per trial (Table 3.1). This ease of switch-

ing also comes at a slight cost; that is, participants traveled to incorrect snapshots 0.7 times per trial on average when using SNAP, presumably because they were confused about which snapshot to select using the interface. When we subtract these 0.7 spurious switches, we are left with $8.1 - 0.7 = 7.4$ useful switches per trial.

When we looked at the effect of task difficulty on switching behavior, we noticed that people switched considerably more when working on OBLIQUE than on ORTHO. For example, in WALK, they only switched 2.5 times on average when working on ORTHO. Considering that a round trip counts as 2 switches, this means that they felt that an extra iteration is necessary only a quarter of the time. The simplicity of the task combined with the short distance required to travel explains why SNAP does not show a significant improvement in task completion time for ORTHO. Additionally, finding more frequent viewpoint switches for SNAP than WALK suggests that participants prefer to switch viewpoints frequently, but don't do so because of the effort of walking.

We also wanted to analyze our log data to understand how much time our participants spent on each sub-task (i.e., walking, manipulating, and selecting a snapshot). In the SNAP condition, we were able to calculate sub-task timings solely based on button clicks, since all travel was done virtually. However, we needed to analyze our optical tracking data using some heuristics to decide when a participant was traveling vs. working on manipulation. We defined five $15''$ wide rectangular volumes with sufficient height and depth centered at the strategic viewpoints shown in Figure 3.7 (i.e., it does not matter how tall participants are or how close they stood to the table when aligning, only if they were collinear with the route required for alignment). When the tracking data showed that the UMPC was within one of the two volumes relevant to a given task, we labeled

the participant to be in the "work zone", equivalent to looking a snapshot in the SNAP condition, for that point in time.

First, the data showed that it took our participants on average 3.1 secs each time they walked from one work zone to another in the WALK condition (the average for children–car was 4.0 secs and for horse–train was 2.5 secs). In the SNAP condition, the average time spent in virtual overview mode to select a snapshot from the five available snapshots was 2.5 secs. One important note is that after participants selected the second snapshot for the task using the virtual overview mode, they simply used the quick switch functionality (Section 3.3.2) to alternate between the two most recently used snapshots.

3.5.5 Generalization of Findings

We can generalize our findings by making the following statement: Whenever $t_{C,i} < (t_{T,i} - t_{S,i}) \times r_i$, where $t_{C,i}$ is the time to create a snapshot i, $t_{T,i}$ is the time to physically travel to snapshot i's location, $t_{S,i}$ is the time to select and virtually revisit snapshot i, and r_i is the number of repeated visits to snapshot i, then using snapshots will result in time savings. Looking at it another way, we can solve for r_i, and say that whenever $r_i > \frac{t_{C,i}}{(t_{T,i} - t_{S,i})}$, we should use snapshots. As an example, let us take the case when a participant in our study stands at the viewpoint for children and wants to get to the viewpoint for car. We have the time to create a snapshot $t_{C,\text{car}} = 10.8$ secs (recalling that it took our participants 54.0 secs to create five snapshots). Our empirical data showed that the average time to travel from child to car, $t_{T,\text{car}}$, was 4.0 secs, and the average time to select a snapshot, $t_{S,\text{car}}$, was 2.5 secs. Plugging all those values into our equation, we get the

minimum number of repeated visits $r_{car} > \frac{10.8}{4.0-2.5} = 7.2$.

Of course, this number depends on the actual distance between viewpoints, which is relatively short in our case ($90''$) and also does not take into account the quick switch functionality to jump between recently used snapshots. Since each snapshot was needed in 4–6 trials and our users switched their view usefully 7.4 times per trial (i.e., 3.7 per snapshot), our expected number of repeated visits per snapshot r_i for the study comes out to be between 14.8 $(= 3.7 \times 4)$ and 22.2 $(= 3.7 \times 6)$ (i.e., > 7.2), which successfully predicts the significant time savings we observed in our data, even over such a short distance.

3.6 Discussion

Participants were able to align virtual objects faster in quick viewpoint switching mode (Snap), despite the added mental effort from the elimination of continuous transition between viewpoints than when physically moving between locations (Walk). While the time savings from not having to physically move from one place to another may be obvious, the added mental demand of choosing a snapshot and reorienting without movement, as well as the more complex interface with more opportunity for error, did not wipe out the time savings even when the required distance to travel was relatively short. It is not hard to imagine that the time and effort savings would scale up as the distance traveled increases for room-size and larger environments.

The advantage of Snap was observed mainly for the Oblique tasks. With a distance that can be traversed in a few seconds and a task that only needs little over one round

trip on average, there was little room for improvement in terms of time and error for Ortho in our study. The Oblique task is more typical of real world tasks either arranging or checking alignments of multiple objects where the environment controls the viewing angles and multiple objects have to be checked and/or arranged (e.g., surveying, arranging furniture). Encouragingly, participants preferred Snap by an overwhelming majority and a statistically significant number of participants selected the Snap condition as less demanding than Walk.

In Section 3.3.4, we mentioned a few drawbacks of the grabbing metaphor for object manipulation in handheld AR. Once we ported our application to an $11.6''$ multitouch-capable Slate PC after our study, we experimented with conventional touch-based controls, such as tapping on a snapshot icon to select it, or translating a virtual object by dragging a finger across the screen. While touch-based controls don't suffer from the same problems (e.g., the virtual object getting out of view while rotating), there were new challenges, such as fatigue due to holding the Slate PC using a single hand to allow the other hand to operate the touch controls. Based on our experience, we believe that these alternative manipulation methods could have been used in the study without diminishing the benefit gained from quick viewpoint switching.

Chapter 4

ParaFrustum

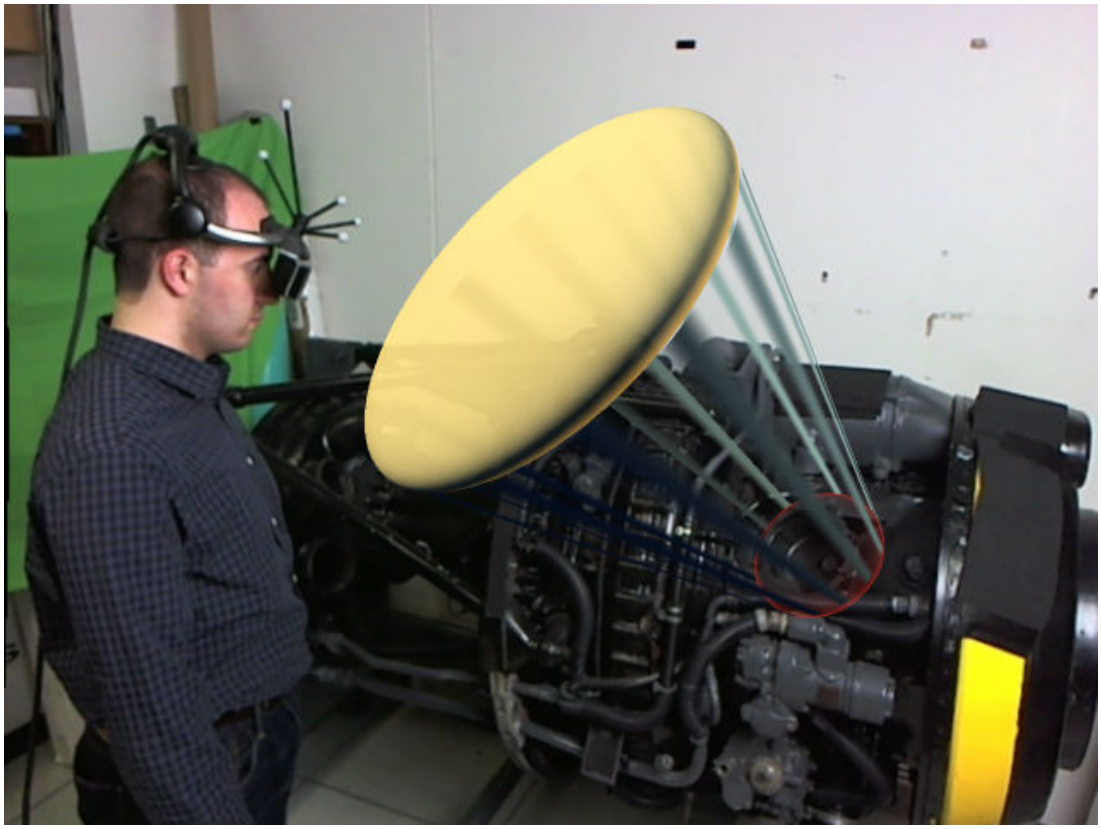


Figure 4.1: ParaFrustum defining a range of acceptable viewing positions and orientations, as visualized by the ParaFrustum-InSitu visualization.

In SnapAR, we focused on a user transitioning from their first-person perspective to other first-person perspectives. To save users time and effort, and enable them to travel to perspectives that might be physically difficult or impossible to get to, we provided them with UI techniques and visualizations to render these transitions and resulting perspectives virtually. In contrast, there are other situations in which it is desirable to physically

travel to a different perspective than one's current perspective. Consider, for example, a maintenance/assembly task, where a technician needs to interact sequentially with parts of a larger system. There are many cases where the technician must also have an appropriate view of that object to perform a task, requiring angle and position constraints. In this work, we make the following contributions:

We introduce the ParaFrustum, which generalizes the specification of a permissible head pose by using two volumes of points that together constrain the sets of acceptable 3D viewing positions and orientations (Figure 4.1).

We describe two different visualization techniques that communicate information about the acceptable positions and orientations, and are designed to interactively guide a user to assume a position and orientation satisfying those constraints.

We present the results of a user study that explores the time and trajectories that users take to reach an acceptable position and orientation using examples of these visualizations that express varying levels of tightness in position and orientation.

4.1 Introduction

When coordinating action with something or someone in the world, such as repairing equipment or sighting a distant object, it can be important to be in the right place and to orient one's body and head in the correct direction. Some tasks require that users view a domain object or location from a precise position and orientation, while others are more flexible, allowing many possible positions and orientations. Using language to communicate spatial location is not straightforward, as everyday spatial language is not precise

(e.g., Clark [1996] and Levelt [1989]). Demonstrating a correct position and orientation may also be problematic, as the user and the helper cannot occupy the same 6DOF pose at the same time. In addition, these solutions require co-location of both participants.

How might this range of possible head poses be represented and communicated to a user? Choosing just one allowable head pose from the set may suffice in some cases. However, requiring that the user assume an overly specific head pose, when others may be just as good, may take longer and be more difficult than necessary. Furthermore, there are tasks in which the most comfortable head position and orientation will differ depending upon the user's height, which may not be known in advance. Therefore, we are interested in exploring how to effectively encode a parameterized set of acceptable head poses and present them to a user.

To address this problem, we introduce the *ParaFrustum*, which loosens the constraints imposed by a conventional computer graphics camera specification. We use the prefix "para" to mean "going beyond" a frustum. A virtual camera in 3D computer graphics can be defined in part by using a look-from point (center of projection) and a look-at point to determine the precise position and orientation (not counting roll) of a camera frustum (Figure 4.2a). In contrast, a ParaFrustum generalizes the concept of a single position and orientation to a set of acceptable positions and orientations (not counting roll) for a frustum. It does this by replacing the look-from point with a look-from volume (the *head volume*) and the look-at point with a look-at volume (the *tail volume*) (Figure 4.2b). The user must place their eyes within the head volume, and orient their head to look in a direction determined by the tail volume. While these volumes may be of arbitrary shape in general, our current implementation uses ellipsoids.

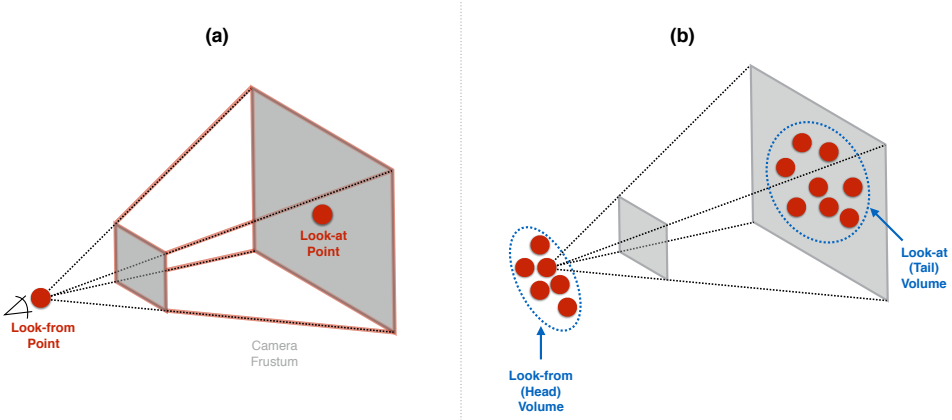


Figure 4.2: (a) A camera frustum defined by a look-from point and a look-at point. (b) A ParaFrustum defined by a look-from (head) volume (a set of look-from points) and a look-at (tail) volume (a set of look-at points).

We assume that the user is ultimately viewing the task domain within which a ParaFrustum resides through one (monoscopic) or two (stereoscopic) display frusta. Each display frustum corresponds to a virtual or real camera frustum—a truncated pyramid whose position and orientation are controlled by the user (e.g., by head motion or hand motion for a head-worn or hand-held display). We define a set of rules, discussed later, that express how the display frusta should be positioned and oriented relative to the ParaFrustum to satisfy its constraints.

To communicate the valid poses allowed by a ParaFrustum and assist a user in assuming one of them, we created two visualization techniques intended for use in AR or VR. One visualization, *ParaFrustum-InSitu*, superimposes in the world coordinate system the head and tail volumes, wrapped by a convex hull, as shown in Figure 4.1. It signals position, orientation, and height information to the user, encompassing the user's eyes, in the process of achieving an acceptable position, orientation, and height. The other visualization, *ParaFrustum-HUD*, superimposes in the coordinate system of the user's head a HUD (head up display) composed of a set of three dials that signal the user's position and orien-

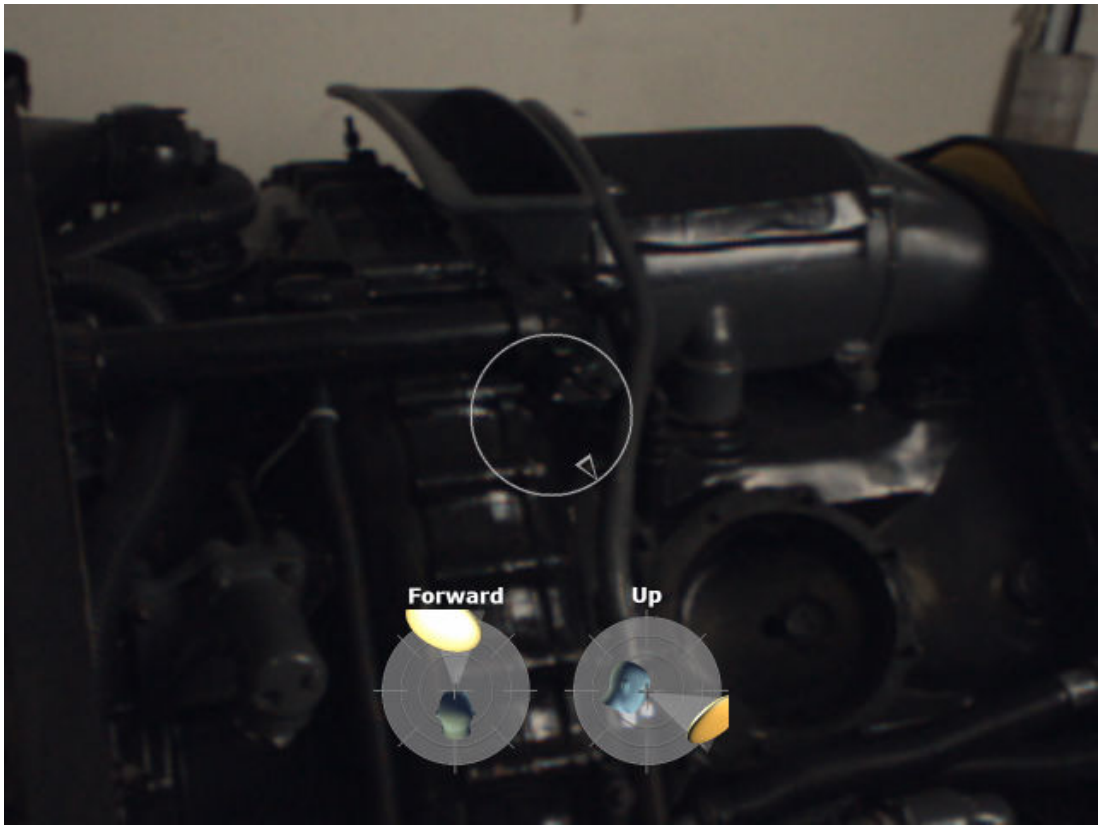


Figure 4.3: ParaFrustum-HUD visualization.

tation relative to the poses encoded by the ParaFrustum, providing continuous feedback to users to guide them as they approach the target (Figure 4.3). Thus, ParaFrustum-InSitu is part of the viewer's world, whereas ParaFrustum-HUD is analogous to a set of diagrams or the dials in a cockpit. ParaFrustum-InSitu integrates the information needed to assume an appropriate head pose, yet presents an unfamiliar way of navigating. ParaFrustum-HUD separates the needed information, allowing users to satisfy constraints in sequence, with some of the familiarity of a videogame HUD.

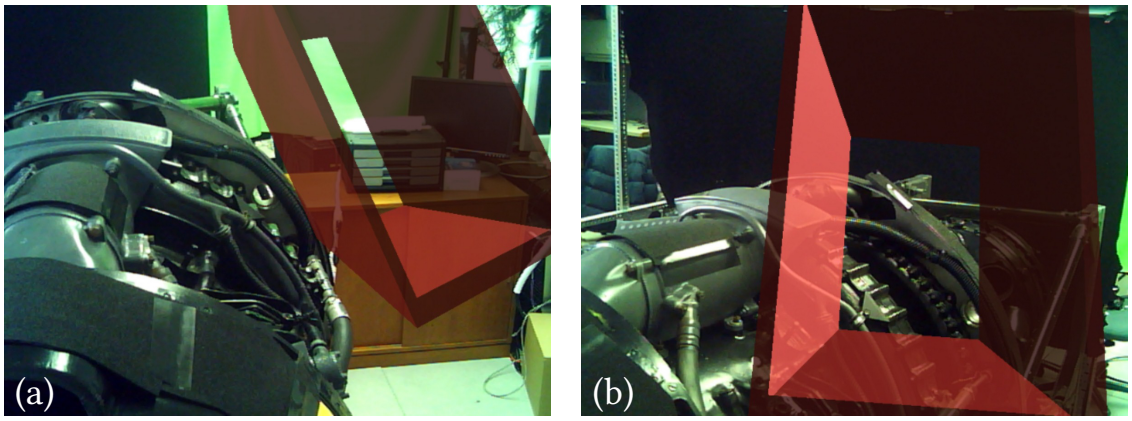


Figure 4.4: Early concept: splayed window-frame to show an ideal viewing pose relative to an aircraft engine for a maintenance task, (a) view from distance, (b) close-up view, before arriving at the viewing position.

ParaFrustum was the result of a collaborative project with Carmine Elvezio and Ohan Oda. I made the following contributions to the project: Inspired by Henderson's "View-pose Management" concept [Henderson 2011], I brought up the need for a visualization to help guide users to physically assume a viewing pose and implemented our first prototypes: a virtual window frame to look through (Figure 4.4) and a virtual mask to place one's head into [Oda et al. 2013]. After extensive piloting and discussions, we recognized the need to allow for tolerance in the underlying representation. To address this, I designed and implemented the generalized ParaFrustum, based on convex hull and CSG operations. Drawing inspiration from our overview visualization in SnapAR, I conceived, designed, and implemented the HUD visualization that features two radar views from non–first-person perspectives. Finally, I helped design the user study, formulate the hypotheses, and took over the responsibility for conducting the quantitative and statistical analyses and plotting the results.

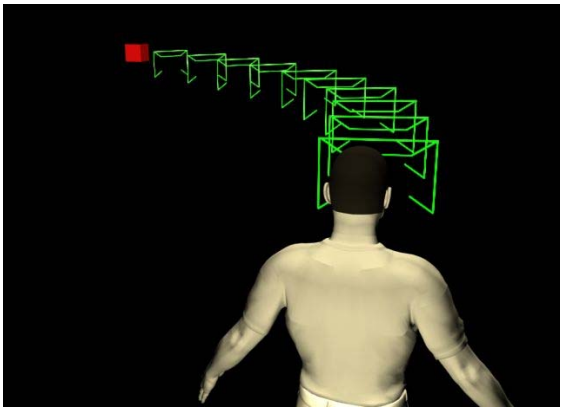
4.2 Related Work

4.2.1 Calling Attention to a 3D Target

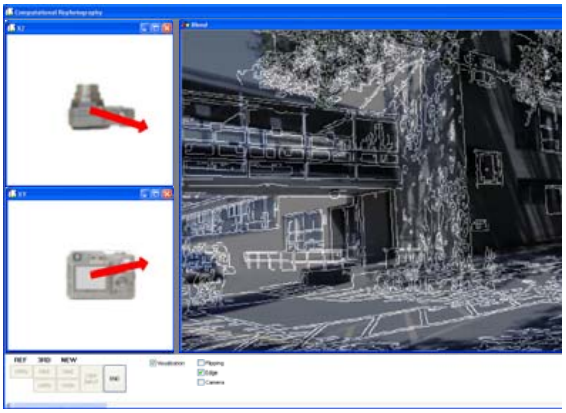
There has been much previous work on how to call a user's attention to a target in a 3D environment, whether real or virtual, when viewed with a 6DoF-tracked display. Perhaps the simplest is highlighting. However, highlighting by itself does not work when the target is occluded or offscreen. One approach to signaling objects that are hard to see uses a leader line anchored onscreen on one end, with the other end terminating on an onscreen target; in the case of an offscreen target, the line is clipped at the screen edge in the direction of the target [Feiner et al. 1993]. The user can then "follow the leader," turning the tracked display toward the clipped portion of the line to bring the target onscreen.

An alternative approach minimizes the portion of the screen devoted to directing the user's attention by using a small conical pointer anchored at its base in the bottom portion of the screen of a head-worn display [Feiner et al. 1997]. Its tip points directly toward an onscreen object, left or right to offscreen objects in front of the user, and down to offscreen objects behind the user. The attention funnel [Biocca et al. 2006] (Figure 4.5a) replaces the simple geometry used in these techniques with a carefully designed set of components that attract the user's attention toward the target, using a funnel-shaped, high-frequency pattern of lines to mark the path toward the object.

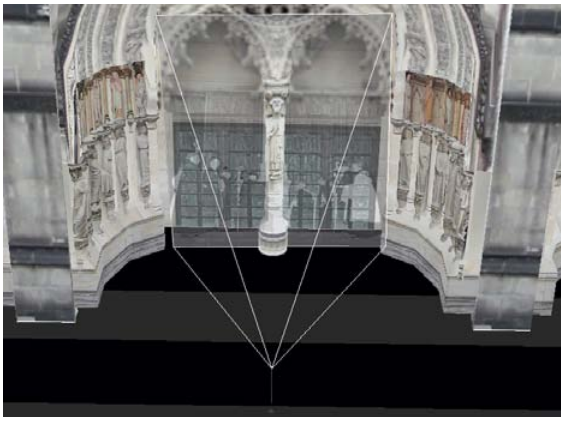
There is also a class of visualizations intended to provide situational awareness, such as so-called "radar views" that are often used in the "heads-up displays" overlaid on a first-person view in games, like the one we implemented for SnapAR (Section 3.3.3). These



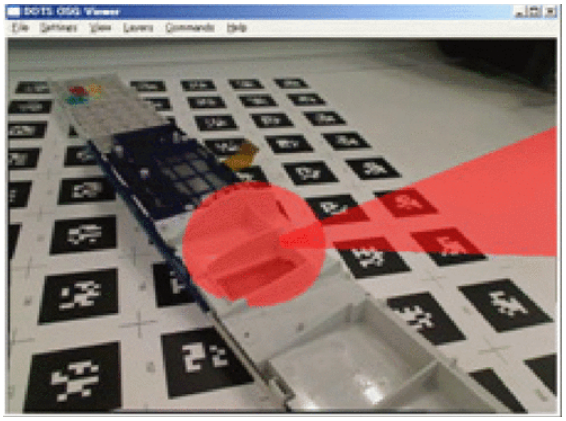
(a) Attention Funnel, Biocca et al. (CHI '06)



(b) Rephotography, Bae et al. (ACM TOG '10)



(c) Situated Media, Güven et al. (3DUI '06)



(d) Camera Pose Navigation, Shingu et al. (ISMAR '10)

Figure 4.5: Sample screenshots from related work.

"radar views" often feature a small circle representing an area around the user located at the center, as seen in an overhead plan view, overlaid by a shaded sector of the circle representing the user's field of view. If a representation of the target is shown in the circle (or marked on the circumference for targets outside the circle), the radar view can guide the user toward the target, by showing the change in position to reach it and orientation to see it. If the altitude of the target is also important, a second radar view can show a side elevation view centered about the user.

4.2.2 Specifying Position and Orientation Relative to a 3D Target

While the techniques mentioned above direct the user's attention to a target, they do not specify how the user should be positioned and oriented relative to that target. One way to do this uses the familiar geometric representation of a camera as a pyramidal frustum, with a center of projection at the pyramid apex (the look-from point) and a base delineating the target, with a look-at point at the center of the base. Snavely et al. [2006] visualize a set of such pyramids, each representing a photograph from which the frustum is derived by the system. The set is displayed at their positions and orientations in a 3D desktop UI, from which the user can choose one to view its photograph.

Güven and Feiner [2006] describe an outdoor AR system in which a line-drawn pyramid view volume is erected over a photograph texture-mapped onto its base and located at the approximate position and orientation from which the photograph was taken, derived from an analysis of the photograph (Figure 4.5c). When users position their heads at the pyramid apex, they can view the image registered with the real world.

Bae et al. [2010] use an alternative approach to guide the user to a desired position and orientation for rephotography (the process of taking a photograph at the same location and orientation as a previous reference photograph). Their system interactively analyzes the current view seen by a camera and, if the camera captures a view sufficiently similar to the reference photograph, the system determines how the camera should be translated to bring it to the correct location. As shown in Figure 4.5b, they present the photographer with an interface showing two views of a camera: An overhead plan view of the camera facing forward is overlaid with a 2D arrow showing the direction parallel to the ground

in which to move the camera. This is complemented with a rear elevation of the camera as seen from the back, overlaid with a 2D arrow showing the direction in the plane of the camera back in which to move the camera. Finally, the view seen by the camera is overlaid with the edges of the reference view to allow the user to adjust the camera orientation.

4.2.3 Specifying a Constrained Set of Positions and Orientations in 3D

The problem we address here differs from that of the work discussed above, in that we do not want to restrict the user to a specific position and orientation, but instead allow a range of possible positions and orientations. Research on automated cinematography (e.g., [Friedman and Feldman 2006; Burtnyk et al. 2002]) has addressed ways of expressing and resolving general constraints on camera specifications; however, this work was not directed toward helping a user to physically realize an acceptable camera pose. The most relevant previous work that we know of is by Shingu et al. [2010], who created an AR visualization to assist with a rephotography task in an industrial setting for inspecting an item before and after a process. A red sphere is positioned around a target of interest and a cone whose apex is at the center of the sphere protrudes from the sphere (Figure 4.5d). The sphere encloses what must be visible and the cone constrains the angle from which it must be viewed. The cone disappears when the camera's viewpoint is inside, and the sphere turns green when it is fully inside the camera frustum, together indicating that an acceptable camera position and orientation have been achieved.

ParaFrustum provides a significant advance over this previous work. ParaFrustum supports a much wider range of shapes and relative sizes of the volumes containing the

look-from and look-at points, makes it possible to constrain the maximum distance the camera can be from the target, and is complemented by a set of visualizations to assist in realizing an acceptable pose that are better suited to the more general geometry of ParaFrustum. ParaFrustum makes a clear distinction between look-from and look-at volumes, allowing the look-at volume to constrain only orientation. In contrast, the sphere of Shingu et al. serves double duty: a look-at volume and an implicit bound on how close the user can get to the sphere. While Shingu et al. do not impose a bound on how far the user can get, providing one by putting a base on the cone would still not produce ParaFrustum's explicit visible head volume, which we use in our visualizations as a target at which users can aim. Because ParaFrustum's tail volume can be asymmetric, it better supports situations in which more leeway is needed in one axis than another. ParaFrustum's rules also allow partial visibility of the tail volume, and explicitly support stereo. Furthermore, the performance of ParaFrustum and its visualizations has been validated in a formal user study investigating a range of sizes and shapes for the head and tail volumes.

4.3 Definition and Rules

The ParaFrustum head volume defines the set of acceptable viewing positions, while the ParaFrustum tail volume defines the set of endpoints for a bundle of viewing vectors that originate in the head end. The head and tail volumes can be thought of as being wrapped by a convex hull that includes all directed lines between any point in the head volume and any point in the tail volume.

We define the set of acceptable viewing locations for the ParaFrustum to require that

the center(s) of projection of the display device's camera frusta (one for a monoscopic camera, two for a stereoscopic camera pair) be fully contained by the head volume. In addition to defining the set of allowable eye positions, the head volume presents a target toward which the user can travel when first approaching the ParaFrustum, as we will describe later.

To define the set of acceptable viewing orientations, we attempt to maximize the portion of the tail volume that is visible to the user. We determine this relative to the intersection of the two display/camera frusta for stereo (the single display/camera frustum for mono), which we will call the *view volume*. As seen from a location within the head volume, the tail volume may either fit fully within the view volume or may exceed it (Figure 4.6). If the tail volume is able to fit fully, then the head orientation will be considered to be correct when it fits. On the other hand, the tail volume may extend past one or more of the left, right, top, or bottom of the view volume. If the tail volume extends past only one of the left and right sides, and/or only one of the top and bottom, the orientation will not be considered correct. The rationale is that in these cases, the user should change their orientation so that tail volume either no longer extends past either one of that pair of sides (fits fully inside the pair) or extends past both sides of that pair.

We note that it is possible to choose pathological combinations of head and tail volumes that make it impossible to establish an acceptable viewing position or orientation under these criteria. For instance, when an ellipsoidal head volume is just big enough to barely contain both eyes of the user at the same time, the tail volume can be positioned in a way that the user would not be able fit it in their viewing frustum without turning. However, since the eyes are tightly constrained by the small head volume, turning even

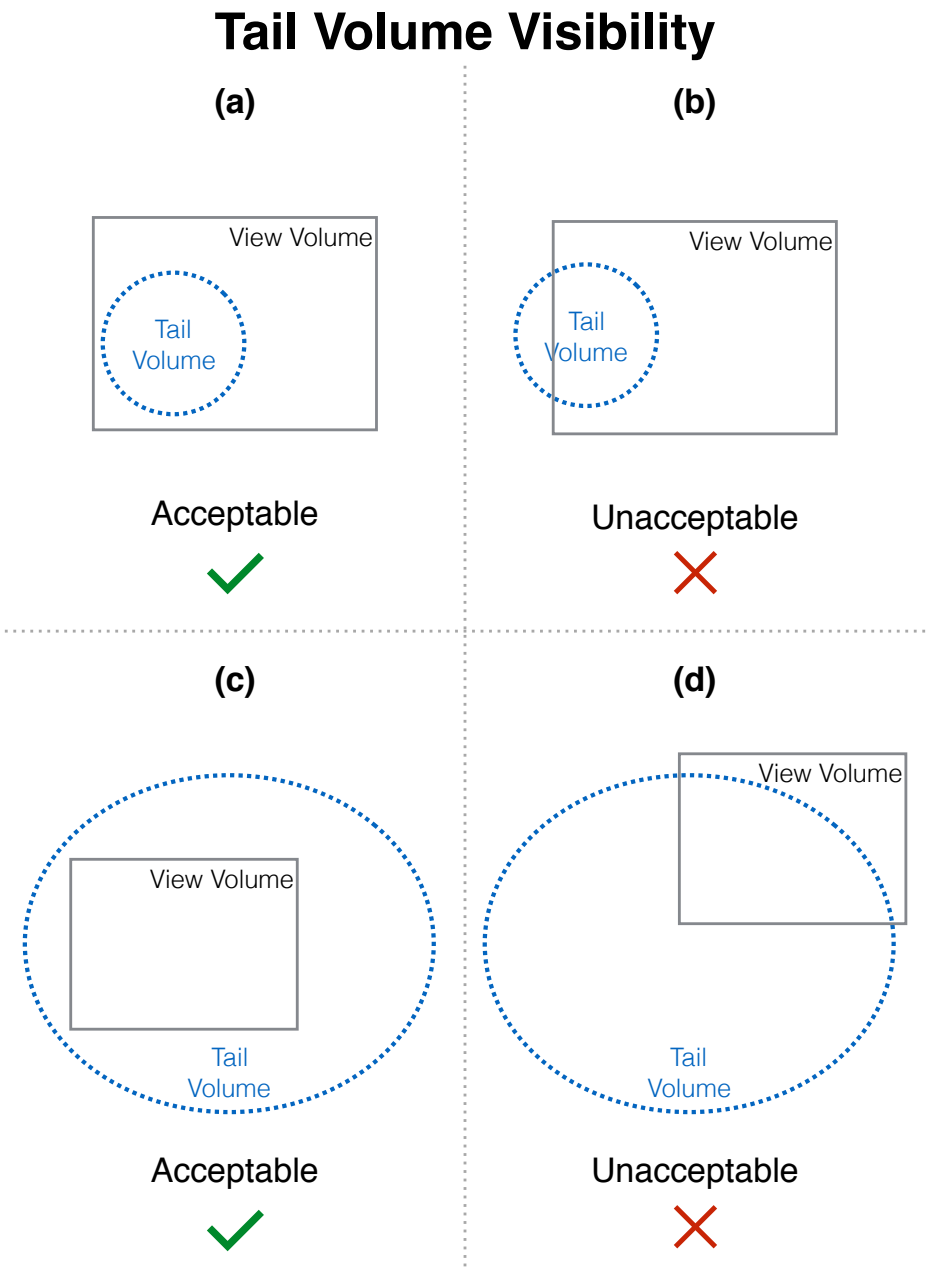


Figure 4.6: 2D projections of tail volume and view volume illustrating visibility rules for satisfying ParaFrustum's orientation constraint: (a, b) Tail volume is small enough to fit fully within the view volume. (c, d) Tail volume is too large to fit within the view volume. (b, d) Unacceptable because portion of tail volume that is visible to user can be increased by rotating camera.

slightly could make one or both eyes leave the head volume, making satisfying the orientation constraint impossible without breaking the position constraint in the process. Nonetheless, we have found it easy in practice to select head and tail volumes that estab-

lish constraints that are satisfiable and make sense for realistic viewing tasks.

It is easy to understand that, in general, smaller head volumes impose tighter constraints on viewing location, while larger head volumes impose looser constraints. The impact of tail volume size is less obvious. Under the definition of acceptable viewing orientation that we use, the tightest constraint on orientation would be imposed by a tail volume that just fits the view volume; for example, this could be an ellipsoid whose projection is circumscribed by the outline of the view volume (Figure 4.7a), or whose projection circumscribes the outline of the view volume (Figure 4.7b). As a tail volume that can be contained by the view volume shrinks in size, the orientation constraint it imposes becomes looser, until the tail volume becomes a point, providing horizontal and vertical leeway corresponding to the horizontal and vertical field of view of the view volume (Figure 4.7c,d). In contrast, as a tail volume whose projection circumscribes the outline of the volume grows in size, it also provides successively more leeway, which can range beyond the horizontal and vertical field of view of the view volume. Note that alternatives to these rules are possible. For example, the tail volume could define a set of points that need only intersect with the view volume or with the view volume's center axis, so that larger tail volumes always result in looser orientation constraints.

4.4 ParaFrustum-InSitu

ParaFrustum-InSitu (InSitu) includes a number of visual aids to assist the user in assuming an acceptable pose. As shown in Figure 4.8a, the head volume is visible only from outside the shape and the tail volume is rendered as a thin red outline. The head and tail volumes

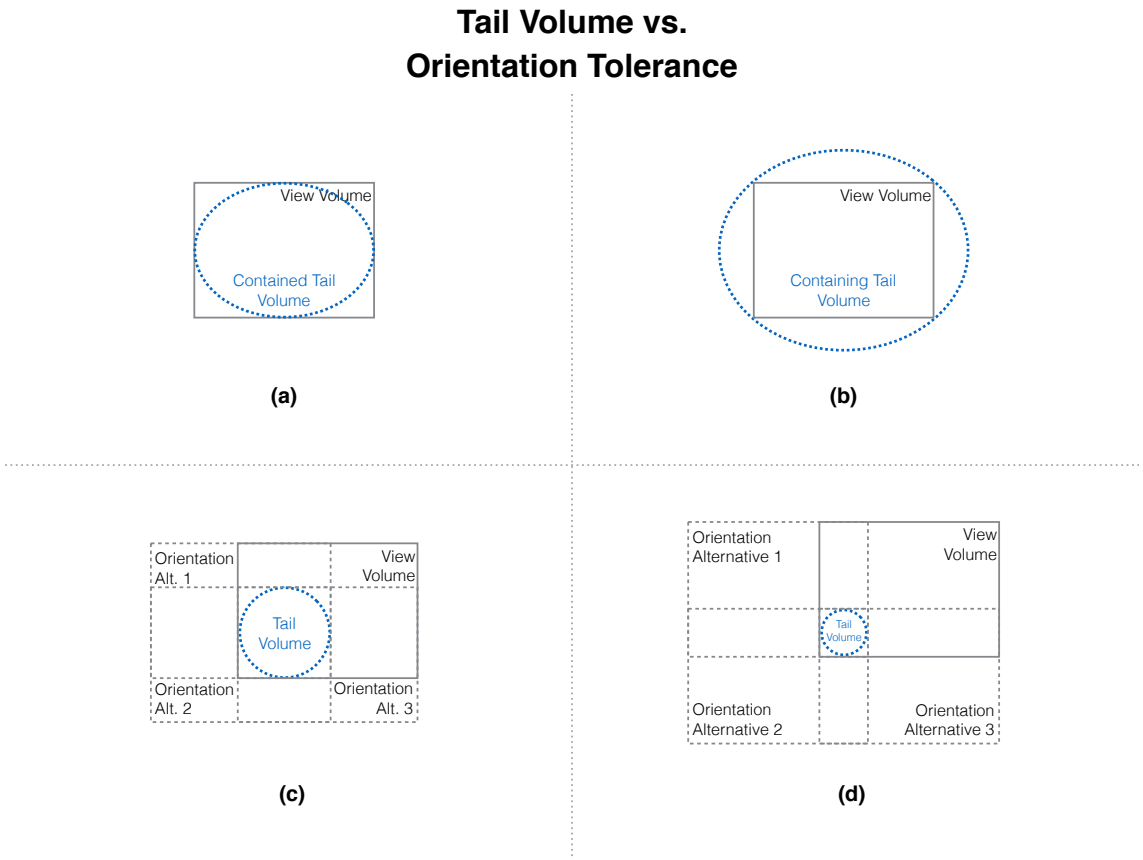


Figure 4.7: (a, b) Tightest possible constraints on orientation. Any change in orientation would result in portion of tail volume that is visible to user to decrease. (a) View volume projection circumscribes the outline of the tail volume. (b) Tail volume projection circumscribes the outline of the view volume. (c, d) A fully contained tail volume is in the bottom-left corner of the viewing volume. Three alternative viewing orientations, which have the fully contained tail volume in each of the three remaining corners of the view volume, are shown with dashed outlines. (d) Has a smaller tail volume compared to (c), allows for larger changes in orientation while still maintaining the tail volume inside the view volume, and is therefore a looser orientation constraint.

are wrapped by a convex hull that is displayed as a series of ribs extending from the head to the tail. As the user approaches the head volume, the head volume becomes more transparent (Figure 4.8b), until a maximum transparency value is reached. Since the head volume is completely invisible from within, we limit the maximum transparency on approach in order to display a discrete jump in color to full transparency when breaching the shell of the head volume (Figure 4.8c). The tail now appear as a faint red ring (defined

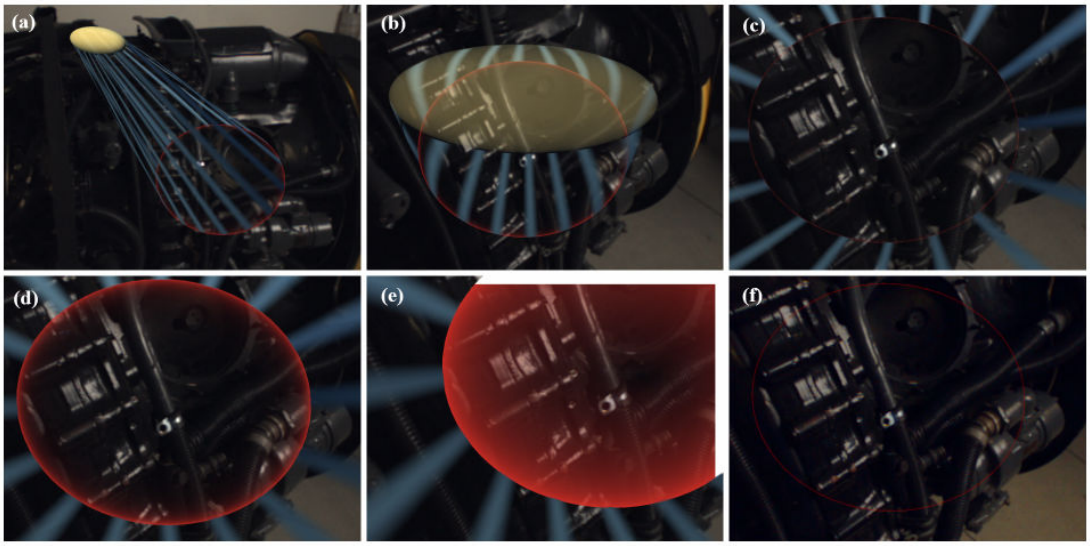


Figure 4.8: ParaFrustum-InSitu visualization. (a) Viewed from a distance, head volume is opaque. (b) Head volume becomes more transparent as user approaches. (c) Looking toward tail volume with eyes inside head volume, red ring and ribs are visible. (d) Eyes have exited rear of head volume, and red ring becomes thicker. (e) Eyes continue further forward and tail volume starts becoming opaque. Tail volume extends beyond top and right edges of view volume, which are highlighted in white. (f) Eyes return back into head volume and orientation is correct, so only faint red ring remains.

by the contour of the tail volume, as viewed from the current perspective).

Inspired by the use of transparency to indicate whether objects are behind, inside or in front of the "Silk Cursor" [Zhai et al. 1994], InSitu's tail shape quickly transitions first to a thicker elliptical ring around the limb of the volume (Figure 4.8d) and then to a nearly opaque ellipsoid (Figure 4.8e), when the user moves forward, exits the head volume, and enters the hull proper. This is done to warn the user that they have exited the head volume, (which cannot be seen from inside the ParaFrustum, when looking in the correct general direction). If the user returns back into the head volume, the tail volume returns to its previous shading (i.e., faint ring). Once the user assumes an acceptable pose, the ribs will shrink until they are completely invisible, and only the light red ring around the tail volume is visible (Figure 4.8f).

Recall that if the tail volume extends past only one of the left and right sides, and/or only one of the top and bottom of the view volume, the user might be able to change their head orientation to make the tail volume fit. In this case, the system shows the offending side(s) of the view volume by highlighting them with a thick white line, as shown at the top and right of Figure 4.8e. If that happens, the user can turn their head in the direction of the line(s) to try to make the tail volume fit completely inside the view volume. If it does not fit, then the tail shape will be cut by both the left and right sides, or both the top and bottom. When this occurs, the thick white line(s) and ribs connecting the ellipsoids disappear, meaning that an acceptable head pose has been achieved.

4.4.1 Implementation

The mesh utilized in the InSitu visualization is generated by passing the meshes for the head and tail volumes to a convex hull library [Sehnal and Campbell 2014] to generate a convex hull wrapping the two end shapes. Once the hull is generated, the two end shapes are subtracted from it using a constructive solid geometry library [Perry and Wallace 2014]. The remaining part of the hull, which does not include the head and tail shapes, is rendered with a shader that applies a sinusoidal function to the alpha value of the hull material's diffuse property to display the ribs that connect the head volume to the tail volume. Finally, the two original end shapes are placed back in the visualization to render the head and tail volumes. In our current implementation, ParaFrusta are built with ellipsoid head and tail volumes, but this process can support arbitrary convex head and tail volumes.

While InSitu uses each visible component for the purpose of visualizing some element of the user's current deviation from an acceptable pose, we recognize that it can potentially obstruct important physical objects in the real world. We have attempted to mitigate this by making the visualization at least partially transparent at all times using a dynamic alpha value between 0.0 (i.e., full transparent, not rendered) when the user is in an acceptable pose and 0.8 (i.e., nearly opaque, but still 20% transparent) when the user's head position is in an incorrect pose and beyond a threshold set to 20cm heuristically for our test cases. With a dynamically adjusted alpha value based on the distance from an acceptable position, there is always an indication of (a) InSitu's presence when the user is not in an acceptable pose and (b) the amount of correction needed to arrive at an acceptable pose, so that the user is always aware of the constraints on pose and can make refinements as needed.

4.5 ParaFrustum-HUD

ParaFrustum-HUD (HUD) uses a system of multiple 2D circular dials, similar to those used in game head-up displays. Three dials were displayed in the user's field of view, as seen in Figure 4.9a. The lower left dial (the "Forward" radar view) indicates the user's position and heading by a top-down orthographic plan view of a mannequin head model and the ParaFrustum head volume, showing the disparity between the heading of the operator and the desired heading. The lower right dial (the "Up" radar) shows an orthographic side elevation view indicating the vertical height of the user's head and the ParaFrustum head volume, information that was not needed in some cases. The middle dial shows the

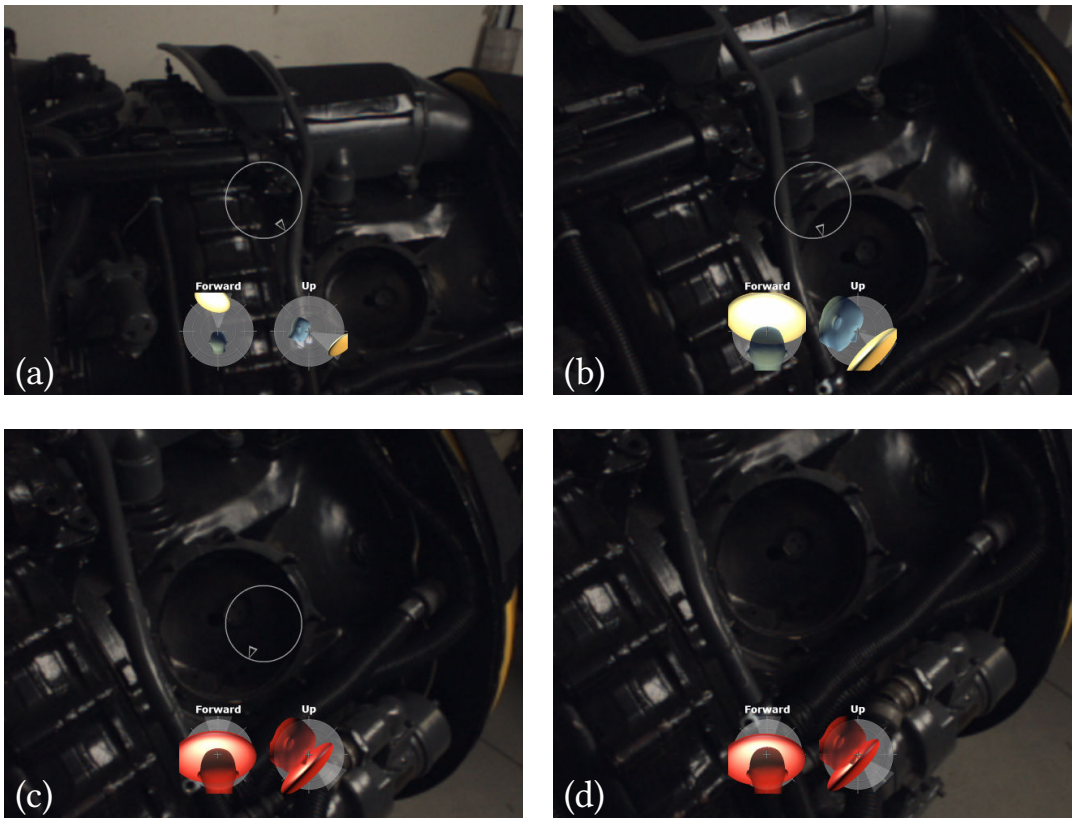


Figure 4.9: ParaFrustum-HUD visualization. (a) Top-down perspective, height offset view, and orientation offset indicator are combined in one visualization. (b) User has approached head shape, which has become larger in Forward and Up radars. (c) User intersects target shape in both Forward and Up radar. (d) User is looking in correct direction and is inside head volume.

user's offset from the final orientation in yaw and pitch, displaying an arrowhead in the direction in which the user must look.

As the user moves closer to the head volume, the scale enlarges (Figure 4.9b). The correct position has been achieved when the projected head volume surrounds the centers of both the Forward and Up radar views, causing the mannequin heads and projected head volumes to turn red (Figure 4.9c). When the user is looking in an acceptable direction, the middle dial disappears (Figure 4.9d) and will become visible again only if the user's orientation is no longer acceptable.

Unlike InSitu, HUD also works when the head is tracked, but the display is not (e.g.,

handheld). Although the user/operator could attempt to integrate the information from all three dials in moving toward the target, visually monitoring three changing objects is challenging (e.g., [Franconeri et al. 2010]). Another strategy that users could use is to follow each dial in a natural sequence, adjusting to each one in turn. The lower left dial could guide users to the right location outside the engine, simply by walking along a path that aligned the nose with the desired orientation. Once in position, users could orient their heads, following the direction indicated in the upper dial. Applying this strategy predicts that users would walk a straight line, then turn and walk straight again.

4.6 Comparison

The dials in HUD are essentially a changing diagram or visualization of the environment superimposed on the user/operator's field of view (in stereo, at a set offset from the user), rather than an integral part of the environment. In contrast, InSitu is part of the user's environment. It is a 3D enclosure that the operator/user must enter. This is a task that people accomplish with exquisite accuracy (e.g., Franchak et al. [2012]). In addition, InSitu combines and integrates the information that is presented in three separate dials for HUD. Using InSitu, users can anticipate the entire sequence of movements needed: the trajectory from the start point to the viewing point and the viewing angle and height. Considerable research has shown that people make anticipatory movements of the body, head, and eyes; that is, they are preparing the next set of movements as they enact the current set (e.g., Bouisset and Zattara [1981], Grasso et al. [1996], Grasso et al. [1998], and Mennie et al. [2007]). Due to anticipation, walkers using InSitu are expected to take a

curved path toward a target. In the present case, walking a curved path toward the position outside the engine would keep the expected position quite central in users' fields of view, as well as allow users to anticipate the position, orientation, and height of the body and head as they reach the viewing point.

In contrast, the straight/turn path that seems optimal for HUD would not keep the expected viewing position in the users' field of view on many occasions. However, because users' movements are guided wholly by the dials, keeping the expected viewing point in the field of view would not necessarily be helpful.

4.7 User Study

We conducted a formal user study to compare the performance of our two visualizations, InSitu vs. HUD, for guiding users to a set of ParaFrusta with varying the amount of tolerance allowed for both position and orientation. Prior to conducting the formal user study, we performed an informal pilot study with our lab members and four compensated students to confirm our design, formulate our hypotheses, and test our study procedure.

4.7.1 Pilot Study

For our pilot study, we defined a set of seven ParaFrusta, representing a range of ParaFrustum position and orientation tolerances. Based on the results, we increased the number of trials for the formal study and adjusted the placement of ParaFrusta locations to standardize the distance the user has to traverse to arrive at a target location when starting from a "home" position. Additionally, the number of ParaFrusta shapes was increased

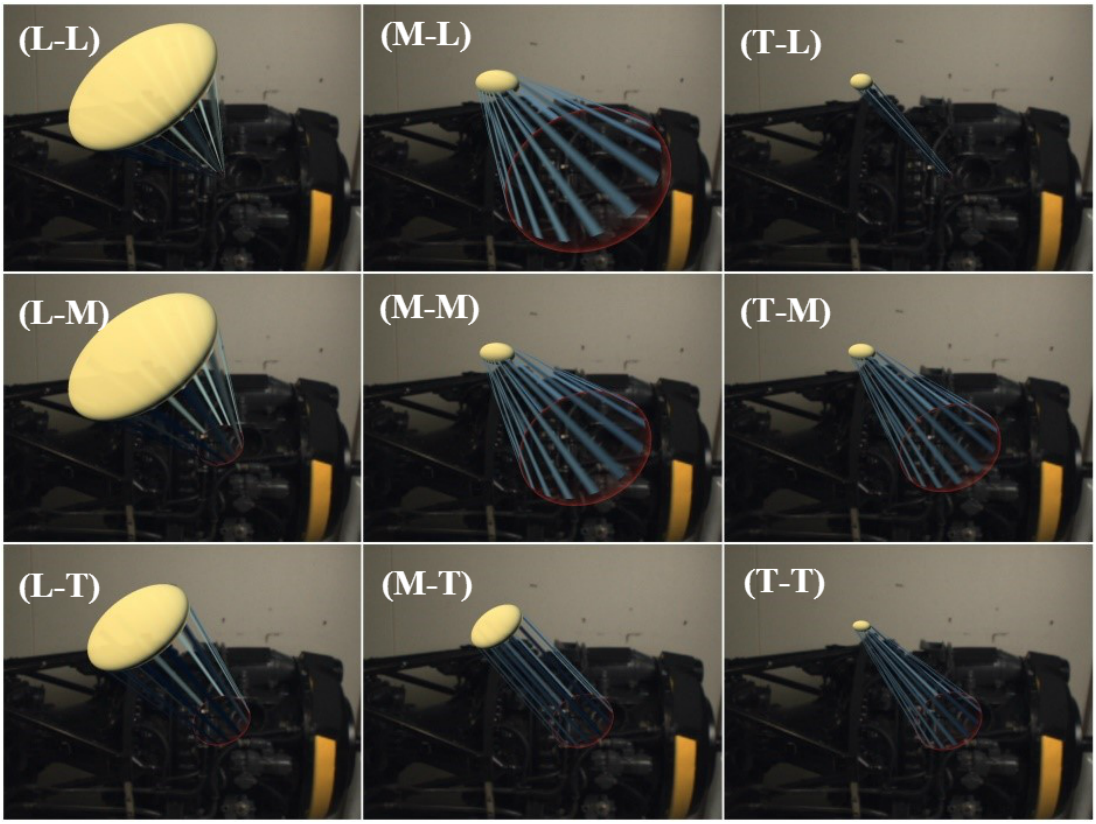


Figure 4.10: ParaFrusta, visualized using ParaFrustum-InSitu. Left-to-right, head position has less tolerance; top-to-bottom, orientation has less tolerance. Labels at upper left of each subimage specify levels of tolerance, and are of form "PositionTolerance–OrientationTolerance", where each of PositionTolerance and OrientationTolerance is one of Loose (L), Medium (M), and Tight (T).

from seven to nine, as shown in Figure 4.10, to collect data for all possible combinations of three levels of tolerances—categorized as loose (L), medium (M), and tight (T)—for both position and orientation.

4.7.2 Hypotheses

The following four hypotheses followed from our analysis and pilot study:

H1. *InSitu will lead to faster task accomplishment than HUD.*

H2. *HUD will increase variability in position and orientation.*

H3. *Both InSitu and HUD will take longer for tighter constraints than looser ones.*

H4. *Participants will prefer InSitu over HUD.*

Rationale

H1: The InSitu condition should lead to faster accomplishment of the task than HUD, because the InSitu visualization is embedded in the surrounding world and thus allows an integrated set of anticipatory actions that people perform well naturally [Franchak et al. 2012].

H2: Due to the difficulty of integrating information from multiple indicators, participants in the HUD condition are expected to follow one indicator at a time sequentially, leading to increased time and variability in position and orientation, because following a single indicator at a time may lead to misalignment on the other indicators. On the other hand, InSitu provides integrated information and allows participants to enact an integrated anticipatory curved trajectory.

H3: Both InSitu and HUD conditions should take longer for tighter constraints than looser ones; in general, this would predict that it should be faster to assume a pose specified by a less constrained ParaFrustum than by a precise frustum.

H4: We anticipated that participants will prefer InSitu over HUD, for the same reasons provided for H1.

4.7.3 Methods

Participants

The participants were 18 students (3 female; ages 19–33, $\bar{X} = 24$) at our institution (protocol: IRB-AAAK6054). They were recruited from lists, email, and websites and each was paid $15 for their participation. All participants used computers on a daily basis and two had experience with AR, though not with the current systems. The experiment took approximately 1 hour.

Equipment

Participants wore a Canon HM-A1 stereo video–see-through HWD (Figure 4.11), tracked by a 12-camera NaturalPoint Optitrack S250E tracking system, and interacted with an application written using Goblin XNA [Oda and Feiner 2014], running on a computer powered by an Intel i7-3770k with 16GB of RAM and using a Nvidia GeForce GTX 780. We positioned the target poses (volumes) around an aircraft engine, looking at various portions of the engine. Participants held a Nintendo Wii remote in their hand and pressed its "A" button to interact with the system.

To minimize the stress placed on the user during the study, we automatically adjusted the height of each target so it would match the user's recorded height. While real world tasks will often require that users assume potentially uncomfortable poses, we decided to eliminate this potential confound.

Figure 4.11: Participants wore a Canon HM-A1 —a stereo, video see-through HWD with 1280 × 960 resolution at 60Hz refresh rate (per eye) and 50° diagonal FOV— during the study.

Design

There were 2 within-subject visualization conditions (InSitu and HUD) × 6 target positions per condition × 9 head–tail shape combinations per target position = 108 timed trials. Trials were blocked by visualization and randomized by position and shape combination. Each block also included an initial nine practice trials. Half the participants experienced InSitu first and half HUD.

In each block, the visualizations were placed at one of three distances from the home position. Each distance was represented by two possible symmetric locations around the engine, and each position had two unique orientations, yielding 12 possible targets. Figure 4.17 uses a plan view of the layout to show for both visualization conditions the three target positions on the right side of the engine in each row, with their two unique orien-

tations (shown as green isosceles triangles with the apex at the center of the head volume and the base oriented towards the tail volume) in each column. The nine head–tail shape combinations prepared during the pilot study were all utilized for both visualizations. Each of the nine unique combinations was placed at all six target positions, and randomly set at one of two possible viewing angles for each position.

Procedure

Participants were welcomed and given the Stereo Optical Co. Inc. Stereo Fly Test to screen for stereo vision. All participants successfully complete the test. They were then introduced to the task, and given exact instructions for playing the part of a technician taking snapshots of the engine. At the beginning of each condition, the participants were allowed to explore the visualizations and position and orientation tolerances they were expected to encounter in the experiment proper. Before starting the practice trials, participants were given an explanation of the study, the details of each visualization, and their role in the study, by the study coordinator. Participants were given a small break in between conditions.

At the start of each trial, the participant was asked to enter a home zone marked on the floor and to look towards the engine (Figure 4.12a). The participant was then presented with a ParaFrustum visualization and asked to assume a pose that it allowed. The participant then proceeded to walk toward the engine (Figures 4.12b and c). Once the participant found a pose they believe to satisfy the visualization, they pressed the "A" button on the Wii remote to "take a photo" and lock in their answer (Figure 4.12d). Head position and orientation were recorded continuously throughout each trial, and the offset

in head position and orientation were also recorded when the "A" button was pressed.

Figure 4.12: (a) A participant starts at "home" zone marked with blue tape on the floor, (b, c) walks towards aircraft engine and approaches a ParaFrustum using one of our visualizations and (d) presses button on Wii remote when she satisfies the constraints of the ParaFrustum.

Participants completed a four-part questionnaire during and after the study, asking them to assess both InSitu (denoted as "X") and HUD (denoted as "Y"). The questionnaire included both an unweighted NASA TLX and questions asking participants to assess the ease, accuracy, and speed of the techniques using seven-point Likert scales (1 = worst, 7 = best).

4.8 Results

We began by eliminating outliers in the data. Several participants took as many as three minutes on trials where median completion times were 6.08 and 9.51 seconds for InSitu and HUD, respectively. Scrutinizing the data revealed that these were a small number of

cases where users were so tightly constrained that it was nearly impossible for them to get their head into an acceptable pose even after the height adjustment. We labeled records as outliers using a conservative version of Tukey's outlier filter—the "outside fence" [Tukey 1977], using three times the interquartile range to determine the cutoff points per 2 (Visualization Condition) $\times$ 9 (Constraint Type) conditions, which we expected to have significant effect on completion time. These outliers, which accounted for 3.14% (3.09% InSitu, 3.19% HUD) of all users across 2 conditions $\times$ 54 trials, were removed from further analyses. Hypotheses were then evaluated for significance with a Bonferroni-corrected α of .0125 ($.05/4$).

4.8.1 Completion Time

We performed a 2 (Visualization Condition) $\times$ 9 (Constraint Type) $\times$ 6 (Target position) repeated-measures ANOVA on completion times, with participants as the random variable using the R Statistical package [R Core Team 2015]. Results from the ANOVA showed that Visualization Condition was a significant main effect, $F(1,3) = 40.509, p < .008$, at $\alpha = .0125$. Participants found an acceptable viewpoint significantly faster using InSitu (7.20 secs) than HUD (12.50 secs), validating H1. Mean completion times for each condition are depicted in Figure 4.13.

The effect of Constraint Type, was also significant, $F(8,24) = 32.044, p < .001$, at $\alpha = .0125$. The most constrained shape T–T took significantly longer on average (21.93 secs) than the rest of the constraints, which have a combined mean time of (8.34 secs), validating H3. There were no effects of target position, nor of any 2-way or 3-way

interactions.

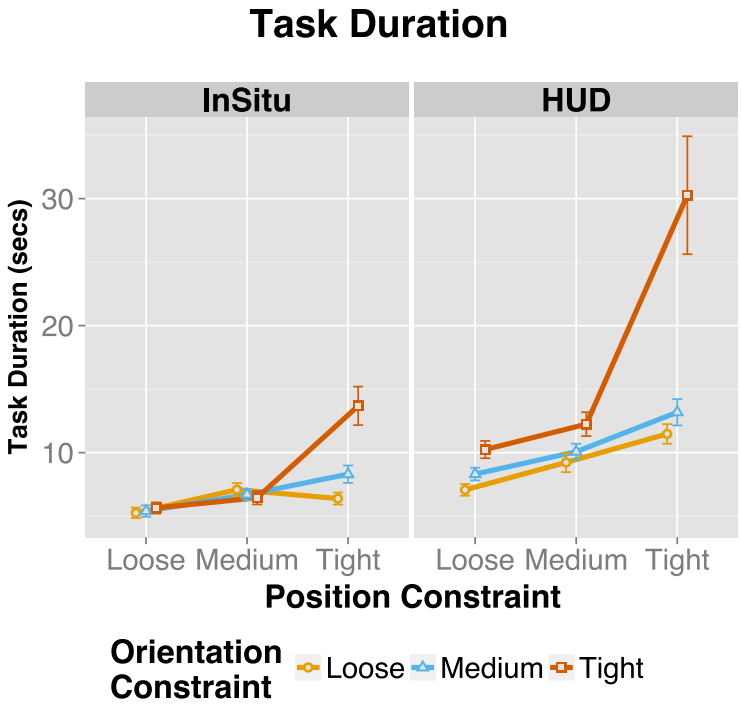
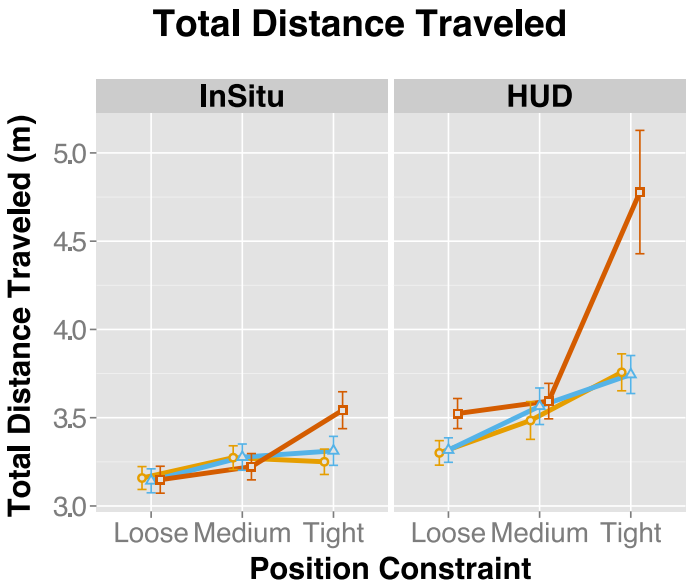


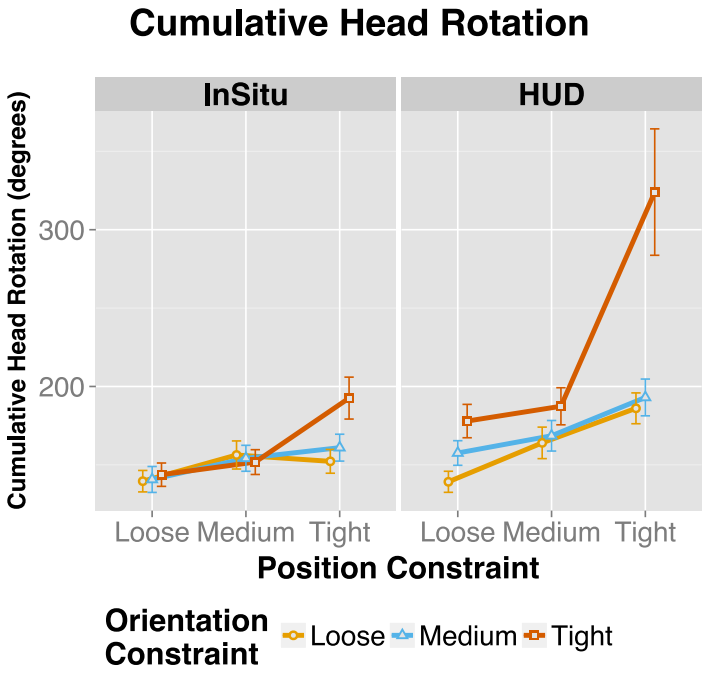
Figure 4.13: Task Duration. InSitu (left panel) vs. HUD (right panel), by position constraint (x-axis), and orientation constraint (color).

4.8.2 Motion Analysis

The greater cumulative head rotation and total distance traveled for HUD over InSitu (Figure 4.14a–b), and the increased positional variability for HUD over InSitu, along with the more uniformly curved trajectories of InSitu relative to HUD (Figure 4.17) support H2.



(a) Cumulative Distance Traveled.



(b) Cumulative Head Rotation.

Figure 4.14: Cumulative motion. InSitu (left panel) vs. HUD (right panel), by position constraint (x-axis), and orientation constraint (color).

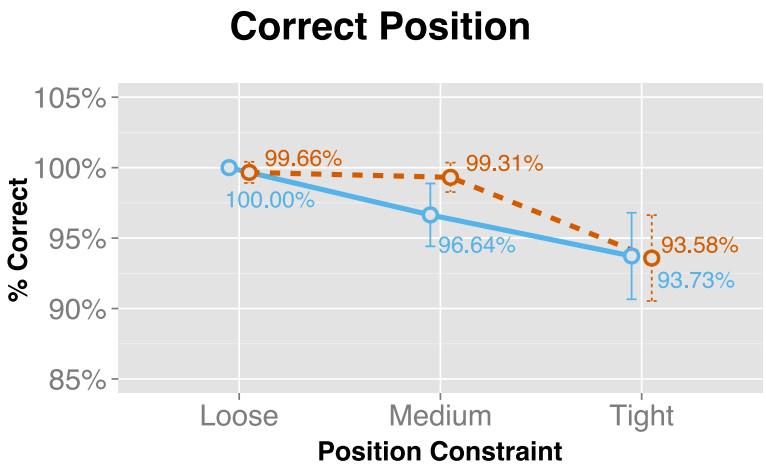
4.8.3 Accuracy

We performed binary accuracy checks for the user's head position and orientation at the time they indicated that they satisfied the constraint on each trial. Overall, users performed very well in terms of both position and orientation accuracy across visualization conditions (Table 4.1).

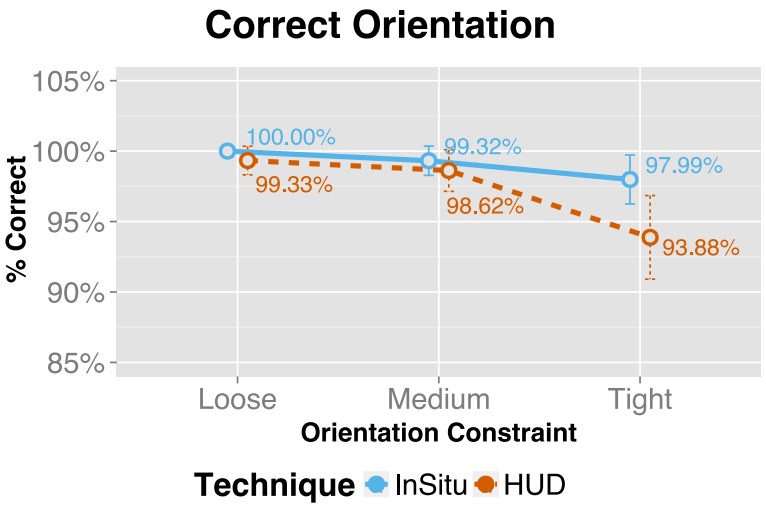
Visualization	Position Accuracy	Orientation Accuracy
InSitu	96.85%	99.10%
HUD	97.51%	97.28%

Table 4.1: Accuracy by Visualization.

A Pearson's Chi-squared test revealed that there was no significant difference between visualizations for position accuracy ($\chi^2_{(1,N=1876)} = 1.629, p = .2018$), but InSitu is significantly more accurate for orientation ($\chi^2_{(1,N=1876)} = 7.2472, p < .01$). Not surprisingly, constraint type had a significant impact on accuracy; that is, users made more mistakes when the constraints were tighter both for position ($\chi^2_{(1,N=1876)} = 40.1464, p < .001$) and for orientation ($\chi^2_{(1,N=1876)} = 26.114, p < .001$) (Figure 4.15).



(a) Position accuracy by Position Constraint.



(b) Orientation accuracy by Orientation Constraint.

Figure 4.15: Accuracy by Technique.

4.8.4 Questionnaire

After each block, participants completed a questionnaire comprising nine seven-point Likert-scale questions that included an unweighted NASA TLX plus three additional questions to evaluate mental demand,physical demand, pace (hurried/rushed or not), perceived success, perceived workload, stress, ease, speed, and accuracy. Their responses were analyzed for significance with post-hoc Wilcoxon Signed-Rank comparisons with Bonferroni correction for 9 tests ($\alpha = .05/9 = .0056$).

The results of the survey showed that participants generally found InSitu less demanding than HUD, both mentally and physically, while perceiving it to be easier, less stressful, and faster (Figure 4.16). Ease was the only pairwise difference that was statistically significant (Table 4.2) at our Bonferroni-corrected $\alpha = .0056$. Fourteen out of 18 participants preferred InSitu over HUD, supporting H4.

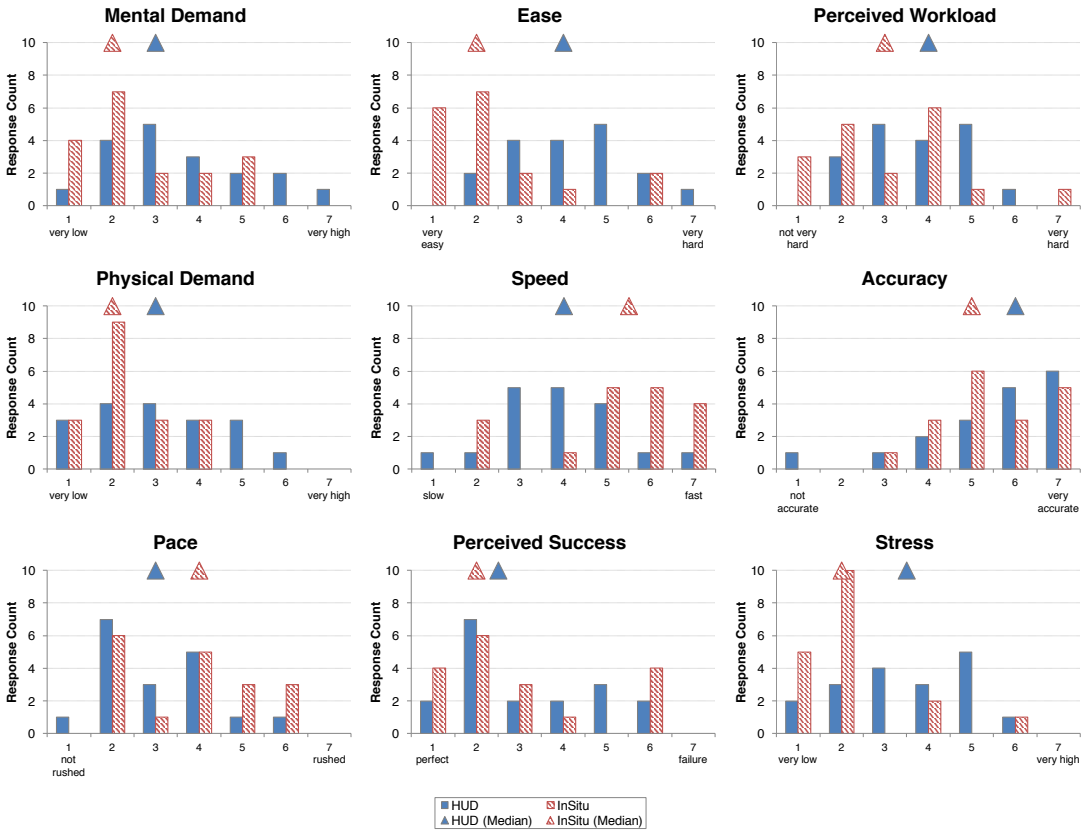


Figure 4.16: Questionnaire Results. Median values for each condition are shown as triangles.

Dimension	Z-value	p-value (two-tailed)
Mental Demand	-2.017	0.0434
Physical Demand	-2.275	0.0232
Pace	-2.045	0.0414
Ease*	-2.840	0.0045^*
Speed	-2.201	0.0278
Perceived Success	-0.706	0.4777
Perceived Workload	-1.525	0.1260
Accuracy	-0.078	0.9362
Stress	-2.471	0.0135

Table 4.2: Questionnaire—Wilcoxon Signed-Rank comparisons. * denotes statistical significance at Bonferroni-corrected $\alpha = .0056$

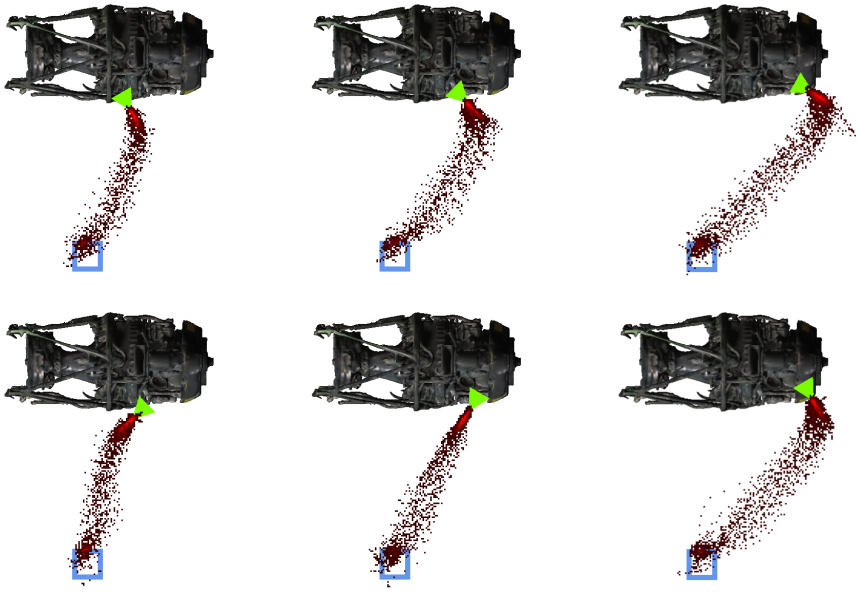
4.9 Discussion

Our post-hoc analysis revealed differences in the speed and shapes of trajectories associated with the two visualizations. Figure 4.17 shows a cumulative view of all participant trajectories as heat maps for both InSitu (Figure 4.17a, top) and HUD (Figure 4.17b, bottom). These are plan views of the layout of our user study area with the aircraft engine towards the top and the blue home zone square towards the bottom of each subfigure. Green isosceles triangles represent six of the possible twelve targets. Only those targets that were on the right half of the engine are shown (the other six targets were mirror images flipped along the y-axis ending up on the left half of the engine). The apex of each triangle is at the center of the head volume for that target and the base is oriented towards the tail volume. A brighter red color for a given location indicates that more participants have visited that location (i.e., that location was along their trajectory).

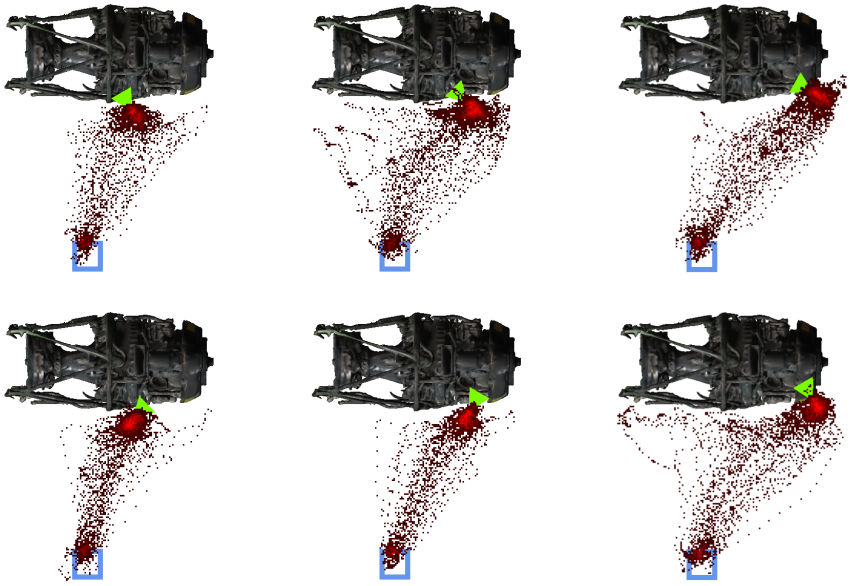
Visual inspection of trajectories indicates less variability in position and orientation for InSitu compared to HUD, supporting H2. The significant increase in cumulative motion and orientation for HUD, especially in tightly constrained situations (Figure 4.14b–c), suggests increased cognitive load for users as predicted by Franconeri et al. [2010]. We speculate that the increased variability and unnecessary motion is due to HUD requiring the users to (1) transform the mediated information into their own frame of reference and (2) integrate information from multiple sources, whereas InSitu presents an integrated visualization embedded directly in the environment. Notably, users corroborated this increased load in the questionnaire.

We acknowledge that ParaFrustum is not fully general. For example, it does not in-

dependently constrain head roll independent of head position and does not account for eye gaze in addition to head orientation. Furthermore, it does not account for situations in which different points within a single head volume should be associated with different tail volumes. However, we believe that it can represent a large family of useful viewing constraints, making it possible to communicate to a user how to assume an acceptable pose more quickly than existing approaches, potentially speeding up a number of tasks.



(a) Using ParaFrustum-InSitu.



(b) Using ParaFrustum-HUD.

Figure 4.17: Heat maps showing a cumulative view of all participant trajectories. These are plan views of the layout of our user study area with the aircraft engine towards the top and the blue home zone square towards the bottom of each subfigure. Green isosceles triangles represent six of the possible twelve targets. Only those targets that were on the right half of the engine are shown (the other six targets were mirror images flipped along the y-axis ending up on the left half of the engine). The apex of each triangle is at the center of the head volume for that target and the base is oriented towards the tail volume. A brighter red color for a given location indicates that more participants have visited that location (i.e., it was along their trajectory).

Chapter 5

Orientation Assistance

Figure 5.1: User's view of a physical, handheld object and one of our visualizations, HANDLES, providing interactive orientation assistance (photographed through Google Glass Explorer Edition).

In ParaFrustum, we explored how to provide AR guidance when a user has to physically travel to a strategic viewpoint (e.g., to perform maintenance and repair on a large physical piece of equipment). When the object to be operated on is smaller and can be handheld, instead of being large and stationary, it can be manually rotated instead of the user moving to a strategic viewpoint. Examples of such situations include tasks in which one object must be oriented relative to a second prior to assembly and tasks in which objects must be held in specific ways to inspect them.

In this chapter, we describe a novel 3D visualization approach, HANDLES (Figure 5.1), and contrast it to three additional visualizations representing different paradigms for guiding unconstrained manual 3DOF rotation. All of our designs in this chapter target smaller FOV, lightweight, monoscopic HWDs, such as Google Glass, which tend to be more comfortable and less intrusive than current generation stereoscopic, larger FOV, see-through HWDs. We conclude with results of a user study evaluating the relative performance of the visualizations and showing the advantages of our new approach.

5.1 Introduction

Many physical tasks require people to hold objects in specific orientations. In some cases, rotation tasks are simplified due to implicit physical constraints (e.g., a knob with discrete steps). However, numerous real-world situations, such as inspecting objects visually or attaching one part to another, require unconstrained manual 3DOF rotation. Further, there are scenarios in which task objects or external references for alignment can be ambiguous; for example, a task object may be symmetric visually, but contain internal sensors, or a hand-held medical imaging device may need to be aligned with internal organs that are not seen directly. In these situations, providing guidance for rotating an object becomes a question of either conveying direction and magnitude explicitly, or annotating the environment to provide additional context for alignment.

Manuals, whether physical or virtual, often show different views of task objects and use annotations (e.g., connectors and arrows) to illustrate the required action [Mijksenaar and Westendorp 1999]. However, these can be difficult to integrate, especially for com-

plex, self-similar, or symmetric shapes, as mentioned above. Systems that present virtual instructions on an HWD have been shown to help in transforming a rigid object to a predetermined position and orientation [Henderson 2011]. Such systems commonly employ basic virtual 3D UI elements such as arrows, animations, or clones of task objects as visual hints that guide users when performing manual operations (e.g., Feiner et al. [1993], Robertson et al. [2008], Miller et al. [2012], Gupta et al. [2012], Oda et al. [2015], and Mohr et al. [2015]). For example, based on this work, we can expect arrows to be suitable for showing a path of movement for a task object or body part. However, displaying paths as 3D arrows can be ambiguous for certain geometric projections, especially when viewed on monoscopic displays.

Even though these basic 3D UI elements have long been used in task guidance systems, we are not aware of any principled exploration of their effectiveness for real-time task assistance. In this chapter, we begin to address this gap by presenting the design and comparative evaluation of a set of UI elements for a nontrivial rotation task, measuring their usability and effectiveness, and attempting to explain their relative effectiveness and trade-offs using cognitive science.

Three of the visualizations described in this chapter, SingleAxis (Section 5.3.2), Euler (Section 5.3.3), and Animate (Section 5.3.4), are refinements of visualizations we proposed in an earlier poster [Elvezio et al. 2015]. As described in detail in their respective sections, to increase their usability, we carefully fine-tuned the parameters and visual appearance of these visualizations based on extensive pilot testing conducted in our lab [Sukan et al. 2016]. While our fourth visualization, Handles (Section 5.3.5), shares the same underlying principle as the *2-Point* visualization from our earlier work, it is a complete redesign

from the ground up, including its underlying logic, as well as its visual manifestation [Sukan et al. 2016].

OrientAssist was the result of a collaborative project with Carmine Elvezio. I made the following contributions to the project: With Carmine's input, I conceived, designed, and implemented the HANDLES visualization, including its handles, tori, the heuristic for torus placement, and the "cookie-crumb" arrows (Figure 5.2). Similarly, I also designed and implemented the visual and interactive elements of the other three visualizations, such as the dynamically-sized, redundantly encoded arrows, described in Section 5.3.1. In addition, I provided Carmine with design input and coding support while he developed a Unity/C# framework [Elvezio et al. 2016] that established an interface and delivered a set of management scripts for building hierarchies of visualizations, which in turn allowed us to re-use and combine visual elements to prototype and iterate on our final visualizations. Finally, I helped design the user study, formulate the hypotheses, and took over the responsibility for conducting the quantitative and statistical analyses and plotting the results.

5.2 Related Work

In this chapter, we focus on a specific subtask—manual orientation of hand-held objects—and aim to improve upon existing techniques by providing users with continuous feedback designed to reduce cognitive load, facilitate corrective action, and provide confirmation once the target orientation is reached.

Unlike 3D applications on desktop systems (e.g., Schmidt et al. [2008]), our techniques

visualize the remaining rotation between a tracked object's current orientation and a target orientation. Further, we assume that our user is holding the tracked object with an unconstrained hand and cannot precisely manipulate the object to rotate only about a given axis as can be done with a desktop widget.

AR interfaces that guide users in matching gestures and poses (e.g., Freeman et al. [2009], Sodhi et al. [2012], and Anderson et al. [2013]) typically focus on hand and body manipulation directly, whereas our techniques focus on guiding users in rotating a hand-held shape to a target orientation, allowing them to use their hands freely as they hold the tracked object.

Oda et al. [2015] used an annotation-based solution to guide a user to manipulate a physical object to match a 6DOF pose specified by a remote subject matter expert (SME). The 6DOF pose of a manipulatable object was constrained by the physical properties of a fixture on which the user placed the object. Two types of orientation guidance were presented. In both, annotations on the manipulatable object and on the fixture provided a complete 6DOF specification for how the manipulatable object should rest on the fixture. In one technique, the user's view of the environment was augmented with a replica of the virtual representation of the manipulatable object that animated in conjunction with the remote SME's control of the manipulatable object's replica in their VE. In their work, unlike what we present in this chapter, rotation guidance was handled completely through matching annotations, with no additional UI elements for rotation guidance.

The large body of research on mental rotation (e.g., [Shepard and Metzler 1971; Chu and Kita 2008; Chu and Kita 2011; Wexler et al. 1998; Wohlschläger and Wohlschläger 1998]) inspired the design of the nontrivial rotation task (rotation of a nearly-symmetric,

nearly-featureless object about an arbitrary axis) underlying our user study described in Section 5.4.

5.3 Visualizations

While our visualizations are device-agnostic (i.e., can be rendered on various screen sizes and modalities—head-worn, hand-held, or desktop; monoscopic or stereoscopic), we developed and tested them on Google Glass Explorer Edition. Figure 5.1 shows a user's view of a physical, handheld object and one of our visualizations, HANDLES, providing interactive orientation assistance photographed through Google Glass' see-through display. Notice how the small, off-center FOV, which is typical of this class of HWDs, means that only a little portion of a relatively small handheld object could be overlaid with virtual annotations at a given time.

5.3.1 Common Components

Virtual Proxy

Instead of overlaying virtual instructions directly onto the physical object, which the user could only see through a tiny, monocular "window", we include a virtual proxy of the handheld object in all of our visualizations, so that annotations can be rendered and registered relative to it instead of the real object (Figure 5.2). We update the orientation of this virtual proxy in real-time as the user manipulated the handheld object, which is tracked using an external camera. The virtual proxy also serves a secondary purpose: we change its color from its default color (white) to green to function as a discrete indicator to signal

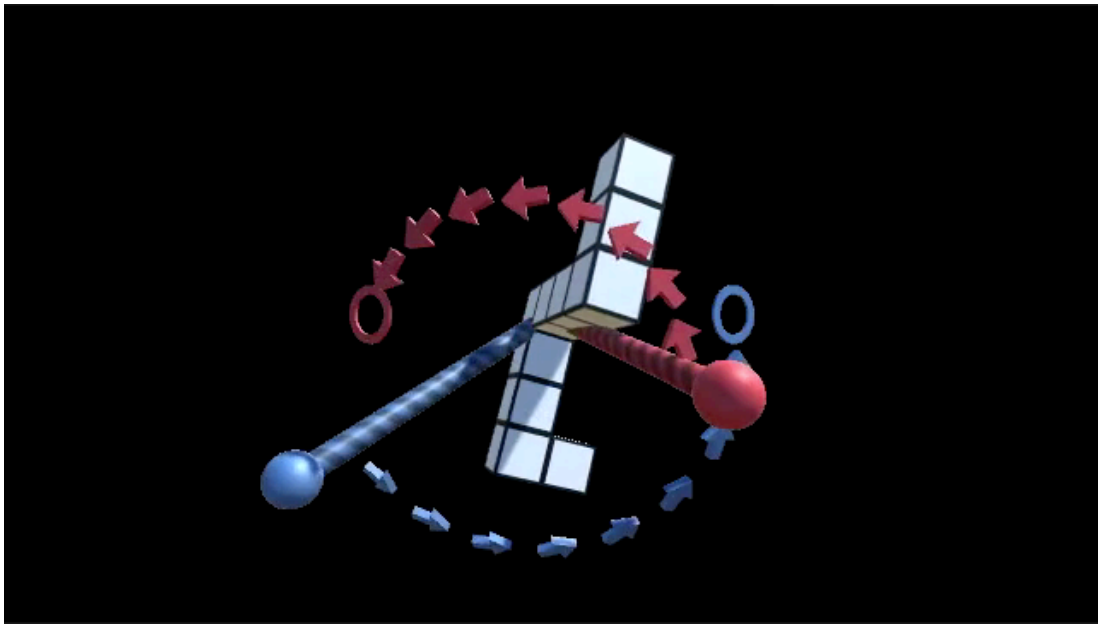


Figure 5.2: Screen capture of HANDLES visualization, rendered on Google Glass Explorer Edition.

when the physical handheld object is within a certain threshold of its target orientation. This proved to be particularly important in the user study we conducted, and allowed us to indicate to participants that they have followed the instructions correctly and can move on to the next trial.

In some of our early prototypes, we also tracked the user's head. The visualizations that we describe in this chapter were developed and tested without head tracking. In early testing, we discovered that head tracking unnecessarily burdened us with keeping our head orientation within a narrow range for an extended period of time, as the graphics would fall outside of the narrow FOV even with relatively small head movements. Instead, we render the visualizations from the perspective of a stationary virtual camera located near the head of the user, who is assumed to be sitting in place, as we ensured in our user study (Section 5.4.4).

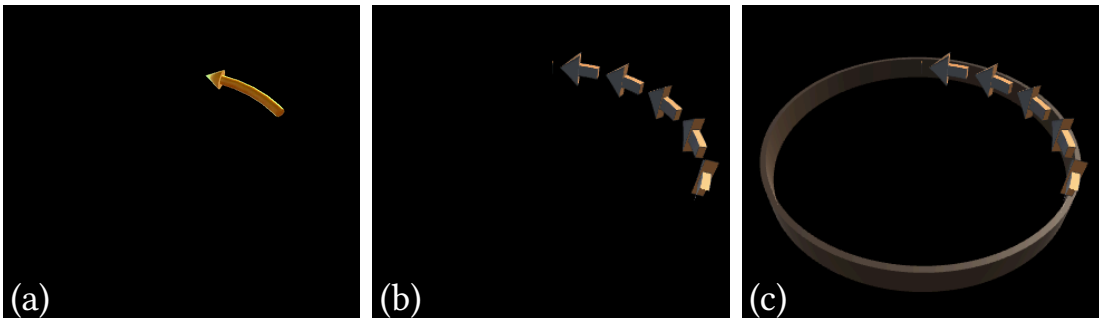


Figure 5.3: Arrow shape evolution. (a) A simple cylindrical, curved 3D arrow. (b) Repeating flattened 3D arrows increase amount of information encoded in arrow body. Walls facing towards the axis of rotation are colored differently to help disambiguate orientation. (c) Repeating flattened 3D arrows with semi-transparent ring to further clarify rotation axis.

Arrows

A number of our visualizations incorporate curved 3D arrows to communicate 3D rotations, whose design evolved as we prototyped and tested our visualizations. Initially, we used simple 3D cylindrical arrows (Figure 5.3a), with a cone for the arrow head and a curved cylinder for the body. During pilot studies, we noticed that the rotation axis implied by these arrows was often difficult to judge, especially when the magnitude of the arrow spanned less than 45°. To improve perceptibility, we switch to a flat, curved 3D arrow that has an extruded triangle for its head and an extruded rectangle for its body, which is essentially a curved version of commonly encountered 2D arrows, slightly thickened. This allows us to provide more visual information about the implied rotation axis by increasing the width of the arrow. To further disambiguate the implied rotation axis, we apply a different color to the walls of the arrow that face towards the rotation axis, as opposed to ones that face away from the rotation axis. The head of a flat 3D arrow can become indistinguishable from its body when the user is viewing the arrow from the side. Therefore, we make the arrow head into a pyramid with a single point at the tip

and base that is wider than the cross-section of the body, to ensure that the head was distinguishable even when viewed from the side.

Even when the axis of rotation is clear, another issue we encounter is that in order to understand the direction of rotation, users have to constantly keep track of where the arrow head is. This problem is compounded when it is occluded by another object in the scene. To increase the amount of information encoded in the arrow body, we break the single curved arrow into smaller ones along the same curve, analogous to a dashed line (Figure 5.3b).

Changing the length of the arrow body to represent the magnitude of remaining rotation creates several issues. First, when the amount of the remaining rotation becomes small (i.e., the task is near completion), the amount of visual information available to the user is also lessened. This is counterproductive, since the user still needs as much information as possible to complete the fine-tuning stage. To address this problem, we add a ring (an extruded annulus) that contains the repeating arrows and does not disappear based on the magnitude of the remaining rotation. The ring is semi-transparent and has the same hue as the arrow (Figure 5.3c). We note that there is a trade-off here between cluttering the scene, especially when multiple arrows are present, and not providing enough information; however, based on our testing, we believe the ring to be worth the visual space it occupies.

Another problem we face with dynamically sized arrows is in their implementation, where we decided not to recalculate the positions of vertices and modify the vertex buffer in each frame. Instead, we leave the full arrow geometry (i.e., 360°, the end of the body touching the tip of the head) untouched and implement a custom pixel shader that takes

the remaining angle as a parameter and paints only those pixels that are within that angle of the tip.

5.3.2 SingleAxis Visualization

SingleAxis is inspired by Euler's rotation theorem [Euler 1775], which dictates that any sequence of one or more rotations of a rigid body in 3D space is equivalent to an optimal rotation about a single axis. (Note that this axis is the unique single axis about which the differential rotation can be performed and therefore cannot, in general, be aligned with a major axis of the shape.)

In this visualization, a ring with small repeating dynamic rectangular 3D arrows is rendered around the virtual proxy, perpendicular to the axis of optimal rotation. In addition, a large cylinder, tied to the axis of optimal rotation, pierces the center of the virtual proxy. As the user rotates the tracked object, the axis and ring update to reflect the new axis and direction for a rotation from the tracked object's current orientation to the target orientation (relative to the world). As the magnitude of rotation gets smaller, the number of arrows decreases (where a single arrow will collapse from head to tail as it disappears), starting from the arrow furthest from the camera, and ending at the arrow closest to the camera (Figure 5.4).

During pilot studies, we observed that while the visualization worked quite well for large ballistic rotations, it became difficult for users to manage as the tracked object neared the target orientation (i.e., the fine-tuning stage). This is due to the rotational error between the current and target orientations changing drastically in direction, even from

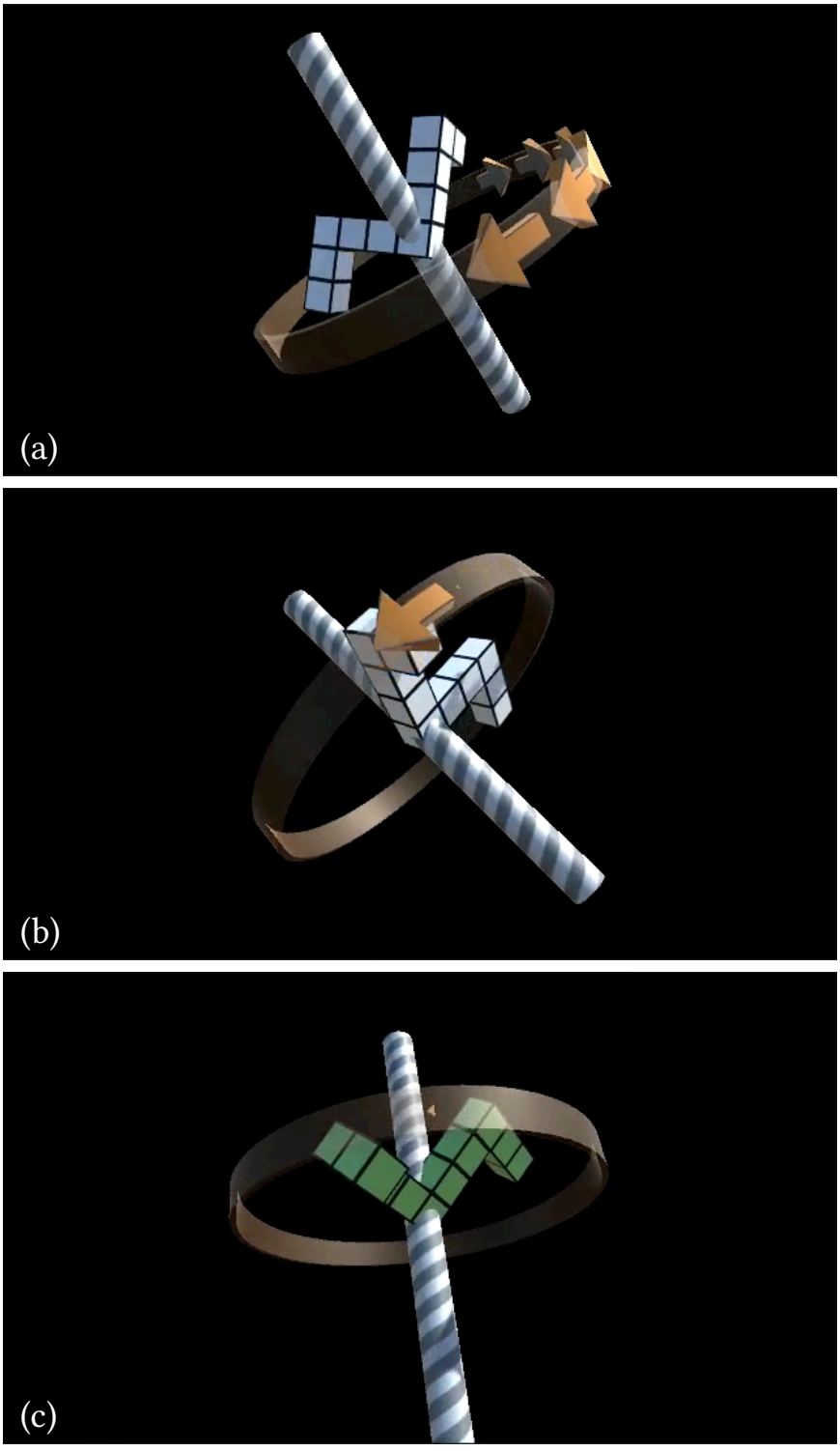


Figure 5.4: SINGLEAXIS. (a) The remaining rotation is represented by a cylinder showing the axis of rotation and a set of dynamic arrows indicating the direction and magnitude of the remaining rotation. (b) As the user follows the visualization, the axis and arrows update to reflect the current optimal rotation from the current pose of the object to the target pose. (c) As the user nears the target pose, the arrows collapse into their arrowheads.

small adjustments made by the user. Visually, this results in the axis and ring swinging wildly around, making it difficult for users to understand how to execute the remaining rotation. To address this issue, we piloted a version of SingleAxis that applied motion smoothing to the cylinder that represents the axis. Surprisingly, we found that smoothing negatively impacted user performance, especially during fine-tuning, where subtle changes to the rotation axis were not immediately represented. As users frequently deviate from the instructed axis during fine-tuning, the smoothed instructions would usually lag behind the user. Thus, we decided not to smooth the visualization during the user study described below.

In another design iteration, we displayed a static version of the original optimal axis of rotation. When the user deviated from this optimal axis, instead of showing the updated axis for the remaining rotation, we displayed a set of arrows that highlighted how to bring their current axis of rotation back to line up with the original. However, this solution was also disliked by pilot users, who now had to mentally resolve two separate rotations, instead of focusing on the single remaining rotation of the original version.

5.3.3 Euler Visualization

Another common way to describe an orientation in 3D space is Euler angles: a sequence of three elemental rotations (rotations about the three axes of an object's local coordinate system). Many objects are easily understood in terms of a particular coordinate system, whose axes can be chosen for the rotations. For the object shown in Figures 5.1 and 5.4–5.9, we use axes perpendicular to the faces of the cubes that make up the object, which

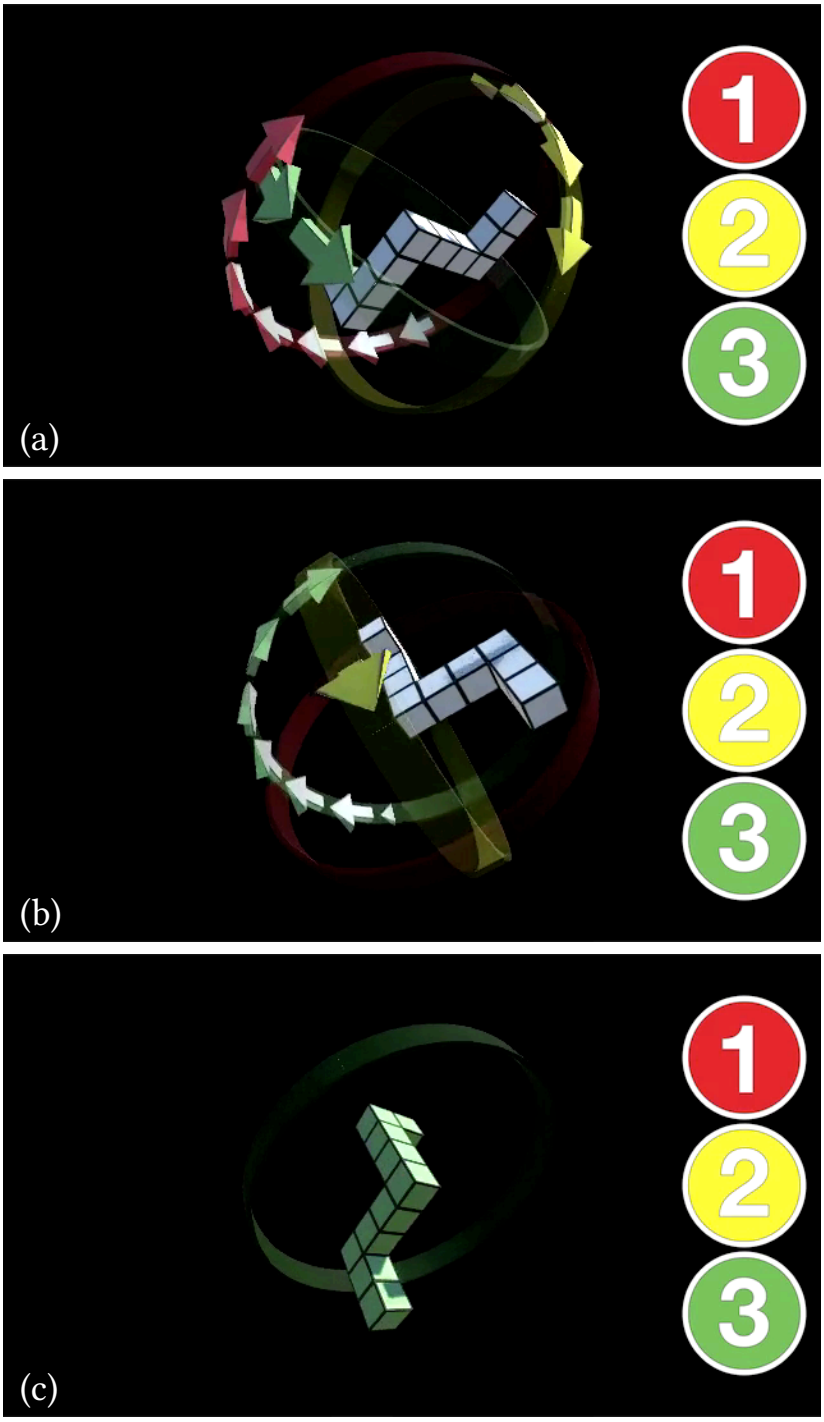


Figure 5.5: EULER. (a) The remaining rotation is represented by a set of three arrows showing the axes, direction, and magnitude of the remaining rotation. (b) As the user follows the visualization, in the order indicated by the colored numbered circles on the side, the arrows update to reflect the remaining rotation, per axis, from the current pose of the object to the target pose. (c) As the user nears the target pose, the arrows collapse into their arrowheads. If the user rotates away from the target about a particular axis, the arrows reappear.

naturally serve as a set of orthogonal axes. Because this visualization decomposes the rotation into three steps, each associated with an easily recognizable axis, it might be easier to enact, especially by people with lower spatial ability (e.g., [Voyer et al. 1995]).

In our early designs, the axes of rotation were described by three cylindrical arrows, color-coded to represent the intended order of rotation based on the decomposition of the quaternion representing the remaining rotation [Elvezio et al. 2015]. When pilot-testing this technique, we discovered that it was often difficult for participants to determine the direction of rotation about each axis, due to the fact that the user needed to search the cylinder that formed the shaft of the arrow for the arrow head. Additionally, it was possible that the virtual proxy itself would obscure the arrowhead, leading to situations where, with an untracked HWD such as Google Glass, it would be impossible to see the direction of the particular arrow without some initial trial and error.

To alleviate this, we use the improved components of Section 5.3.1 to introduce a number of new features. Instead of a single arrow per axis, we render a set of smaller arrows in a ring perpendicular to a particular principle axis (Figure 5.5). The smaller arrows disappear smoothly as described above. In addition, the front of the path is always anchored at the point on the ring closest to the virtual camera. The combination of these changes makes immediately clear, at all times, the intended direction of rotation per axis. Finally, upon nearing the completion threshold for a particular axis, the ring will disappear. It will return if the user breaks from the target orientation about a particular axis.

Since a sequence of 3D rotations is, in general, not commutative, there is a defined order for the axes about which the user should rotate the object when following the instructions. To remove the requirement that the user memorize the axis order, imposed in

our previous work [Elvezio et al. 2015], we render three large icons on the screen showing the rotation order, represented by number and color.

5.3.4 Animate Visualization

Animation is a visualization technique frequently used to communicate motion or action. In our early testing, it quickly became clear that animating the virtual proxy from its tracked (i.e., current) orientation to the target orientation is not ideal because the user has to wait until the animation finished and rewound to get feedback on current orientation. To provide feedback on both current orientation and desired motion simultaneously, our ANIMATE visualization (Figure 5.6) adds a second, animating copy of the virtual proxy to the scene. We also quickly noted that the placement of this animating copy has a significant impact on user performance. Initially, since our task is rotation-only, the animating copy overlapped with the virtual proxy, which makes it difficult to distinguish one from the other. In our following iterations, we tried rendering the animating copy and the virtual proxy side-by-side, similar to our earlier work [Elvezio et al. 2015]. This proved to be suboptimal, especially in the fine-tuning stage, because it requires users to detect differences between two objects that have similar orientations, but are spatially set apart. Going back to a co-located design, we address the occlusion and disambiguation issues caused by overlapping, by modifying the transparency of the animating replica to 50% and changing its outline from solid black lines to dashed grey lines (Figure 5.6). This faded visual is known as *ghosting*, an illustrative technique used in comics [McCloud 1994] where an object is rendered as semitransparent to represent its past or future state,

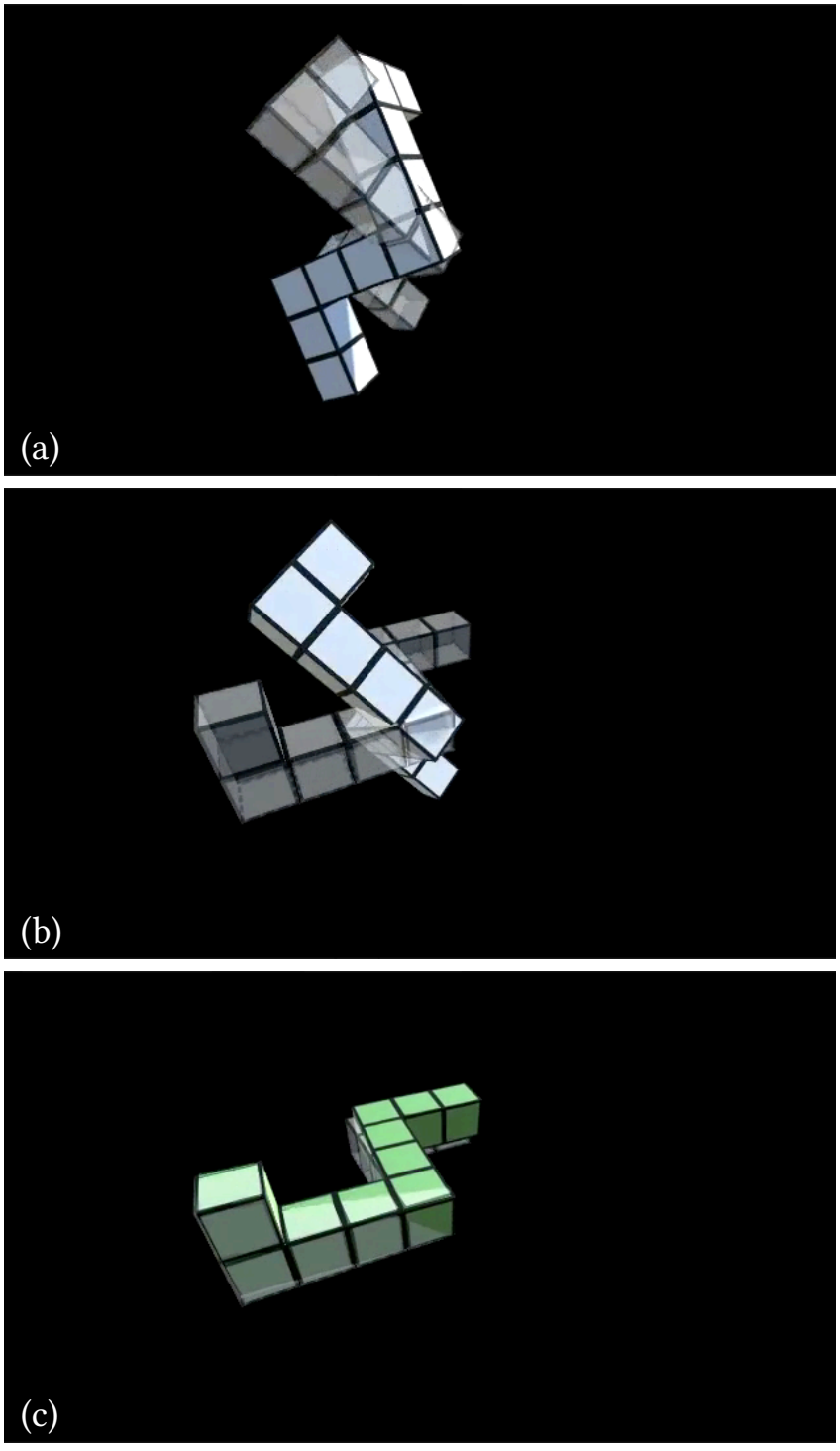


Figure 5.6: ANIMATE. (a) The remaining rotation is represented by an animating clone of the virtual proxy, which rotates from the current orientation of the tracked object, to the destination orientation. (b) As the user follows the visualization, the looping animation will begin from the latest orientation of the tracked object. (c) As the user nears the target pose, the frequency and speed of the animated object will increase, until the task is complete.

and in previous visualizations (e.g., White et al. [2007] and Gupta et al. [2012]).

Another subtle, yet important, design decision is the timing and speed of the animation. We want to provide users with continuous feedback, so it is a natural decision to repeat the animation once the animating copy arrives at the target orientation by rewinding the animating copy to the tracked object's current orientation. We use an ease-in, ease-out interpolator to make the beginning and end of the animation less visually jarring and abrupt for the user. Setting the duration of each animation cycle to a constant value does not make much sense, since that would require the animating copy to move more slowly as the tracked object nears its target, which pilot users found frustrating.

Specifying the speed of the animation turned out to be a better idea and we found a rotational speed of 90° per second to be comfortable based on pilot tests. This ensures that when the tracked object is near its target orientation, the animation takes less time and therefore repeats more frequently. However, when the frequency gets too high, it might become less helpful because it is hard to distinguish between the animation progressing forward and rewinding back to the current orientation. To address that issue, we introduce a 0.5 second gap between animations. Finally, we clamp the total animation duration to be between 0.2 and 2 seconds.

5.3.5 Handles Visualization

The Handles visualization (Figure 5.7) builds on the key insight that orientation in 3D space is commonly parametrized by two different directions (e.g., virtual cameras in computer graphics are often defined by specifying a non-collinear pair of vectors: look-at and

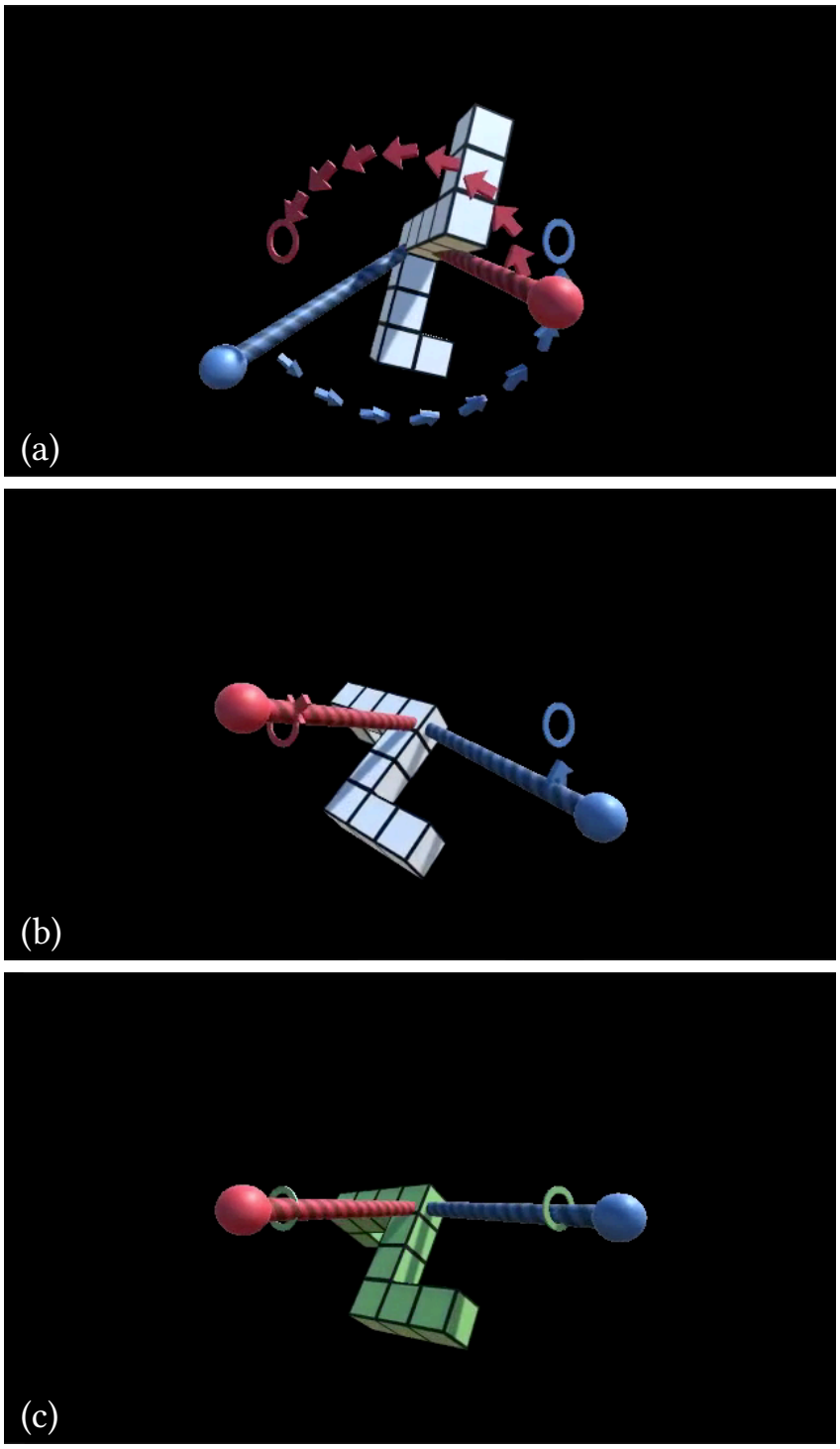


Figure 5.7: HANDLES. (a) The target orientation is directly represented by a set of two colored tori. Two colored poles extend from the center of the virtual proxy, and the user must try to align each pole with its matching torus. A set of arrows show the rotational path from each pole to its corresponding torus. (b) As the user nears the target pose, the arrows update to show the current rotational path from each pole to its corresponding torus. (c) Both handles have been aligned, the tori turn green, and the task is complete.

up). In HANDLES, these non-collinear directions are represented by poles extending from the center of the virtual proxy, which look like physical handles. Each pole's target orientation is represented as a torus whose hole is just wide enough for its pole to go through. These tori are persistently placed in direct view of the user to avoid occlusion. Additionally, each pole is connected to its corresponding torus by a set of color-coded arrows to indicate the direction of rotation necessary to achieve the target orientation (Figure 5.7).

The 2-Point visualization, developed in our earlier work [Elvezio et al. 2015], requires the user to align a pair of points attached to the virtual proxy, represented by cones, with a pair of corresponding points that are fixed in space, represented by target spheres. When piloting an implementation of 2-Point, we found that it had shortcomings that severely limited its effectiveness. Users often tried to translate the shape so that a cone/sphere pair would align, even after being instructed that translation was not being tracked. The locations of the target spheres depended on the specific task in a seemingly arbitrary way. Additionally, users complained that the cones and spheres were too small, making it difficult to distinguish which way the cones were pointing, and the enveloping sphere made it difficult to see the cones and the virtual representation of the main object contained within.

Since we want to make the poles look like physical handles that are rigidly attached to the virtual proxy, we add a spherical knob to the end of each pole to (a) bolster the metaphor that the poles are handles that can be grabbed and moved, and (b) provide occlusion and perspective depth cues, which could be especially beneficial when the handles are near their targets (i.e., during the fine-tuning stage, which is an issue mentioned during pilot tests).

An important design question is where to attach handles to the virtual proxy. Initially, we attached them along the major axes of the shape, but for certain target orientations, this causes the tori to face away from the user, possibly occluded by the virtual proxy. Since the visibility of the tori is crucial for this task, especially in the fine-tuning stage, we want to guarantee that they are always front and center and clearly visible to the user.

To that end, we developed a heuristic in which we start with a vector connecting the centroid of the virtual proxy to the center of projection of the virtual camera, rotate it 30° about the virtual camera's up-vector (clockwise for the first torus, counterclockwise for the second), and pick the intersection of that rotated vector with a spherical hull that contained the virtual proxy (to ensure that the virtual proxy would never touch or occlude the tori in any orientation). Since we have a stationary virtual camera pointed directly at the virtual proxy, this heuristic gives us two locations that are projected to lie on the horizontal centerline of the screen. Picking where the tori end up first means that the poles would have to be attached in different orientations relative to the virtual proxy for each new target pose, which is calculated during initialization by applying the inverse of the rotation between the current orientation to the target orientation, to the tori positions.

To provide users with a sense of which direction to move the object, we add arrows that connect each handle to its corresponding torus. Initially, we used the same curved arrows that we used in other visualizations, which depicted the shortest path along the sphere from handle location to torus location. During pilot tests, we noticed users getting frustrated when following the shortest rotation between one of the handles and its torus worsened the alignment between the other handle and its torus. To alleviate this frustration, we replaced the arrows that traced the shortest path for each individual handle

with “cookie-crumb” arrows that trace the ideal path of the handles when both of them are moved towards their target simultaneously (i.e., by following the single-axis optimal rotation from the current orientation to the target orientation). Similar to our other visualizations, the trail of arrows gets shorter as the user rotates the object and the remaining angle gets smaller.

To provide visual feedback for when the alignment is complete, we rely on color. Specifically, when a handle enters its corresponding torus, that torus turns green to indicate proper alignment for that pair. Once one of the handle–torus pairs is aligned, the user needs to bring the second handle into its corresponding torus while holding the first handle in place, which can be achieved by executing a 1DOF rotation (Figure 5.7b–c).

5.4 User Study

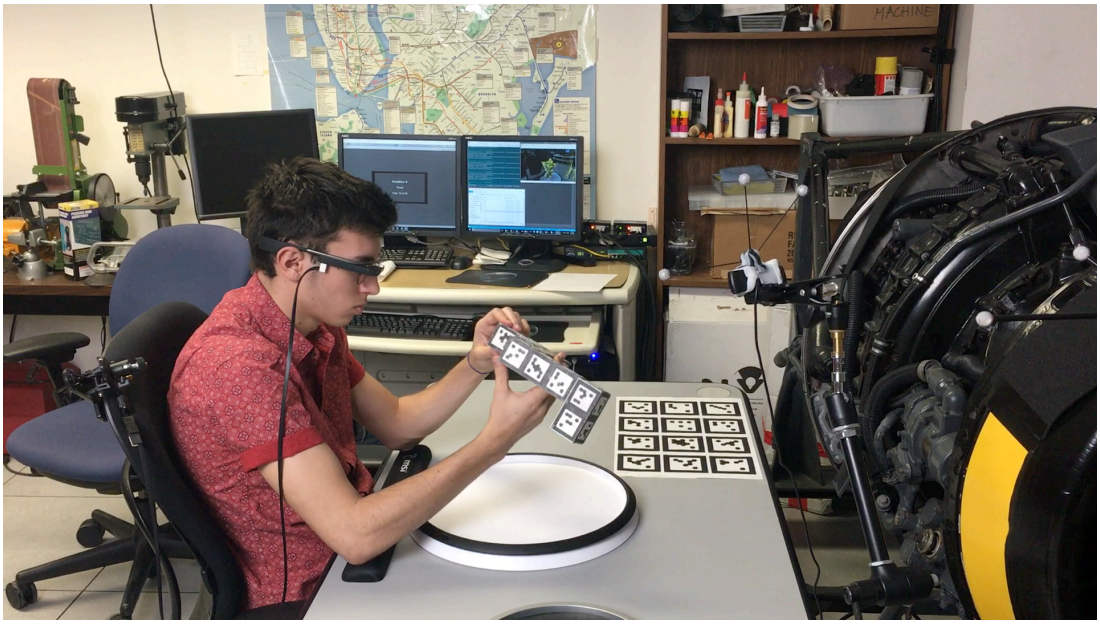


Figure 5.8: Study participant manually orienting task object, guided by our system.

We conducted a formal user study to compare the performance of our new techniques,

in addition to a control condition described below. For our task object, we created an abstract object similar to those used by Shepard and Metzler [1971] in research on mental rotation. Our object consists of ten 1.75-inch wooden cubes attached face-to-face to form a rigid structure with three right-angled "elbows" (Figure 5.8). This type of object is especially suited for rotation tasks because (a) it cannot be transformed into itself by any reflection or rotation (short of 360°) and (b) cognitive science research has shown that it is difficult to mentally rotate [Shepard and Metzler 1971; Vandenberg and Kuse 1978].

We required that the accuracy with which the participant performed each trial be as close to the correct pose as possible for the trial to end; therefore, we compared only time, not accuracy. We settled on a threshold of 8° by incrementally loosening a tighter threshold until pilot users were able to satisfy it consistently. Tighter constraints were especially difficult for visualizations that provide little or no feedback during fine-tuning (e.g., our control condition, STATIC, described in the following section).

5.4.1 Control Condition

To determine the effectiveness of the techniques described above, with respect to a simple baseline, we developed a fifth technique, STATIC, which shows only the static target orientation next to an updating representation of the virtual proxy's current orientation (Figure 5.9). The virtual proxy updates as the user rotated the shape. When the target orientation is achieved, the virtual proxy turns green. This provides a simple control condition to use in the user study described below.

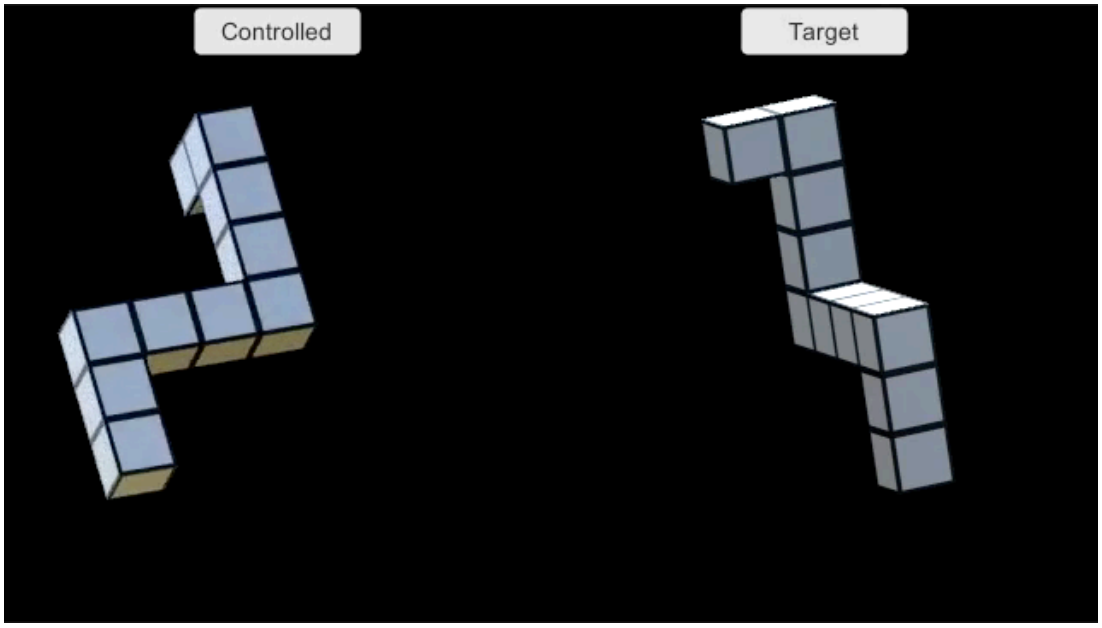


Figure 5.9: STATIC. Control Condition.

5.4.2 Pilot Studies

Pilot studies were instrumental in guiding the evolution of the techniques, as described above, and helped us refine study parameters. In particular, HANDLES is designed to overcome specific shortcomings of existing techniques by providing a persistent target and ensuring high visibility of landmarks and targets.

5.4.3 Hypotheses

Based on an analysis of the tasks and extensive design iterations informed by and tested in pilot studies, we formulated the following five hypotheses:

H1. *HANDLES will be the fastest technique.*

H2. *HANDLES will be the preferred technique.*

H3. *SINGLEAXIS and STATIC will be less preferred compared to HANDLES, ANIMATE, and EULER.*

H4. *Handles will be fastest for fine-tuning.*

H5. *Euler will be preferred by users with low spatial ability.*

Rationale

H1: In presenting the rotation instruction, Handles is the only visualization that presents both a persistent view of the target orientation (relative to the world coordinate system) and a view of the optimal transformation needed to get to the target orientation from the tracked object's current pose. Animate shows the latter transformation, but to avoid cluttering the virtual scene with a third model of the virtual proxy, does not always show the target pose (besides the pause at the end of the animation loop). As a result, it is possible that the user will need to wait for a certain amount of time to comprehend the rotation instruction. SingleAxis also shows the optimal path, but during pilot testing we found that due to limitations inherent to the small monoscopic display, it could be difficult to disambiguate certain rotation instructions, leading to situations where users would lose time trying to understand the direction of the instruction. Euler is similar to SingleAxis, but breaks the transformation into three steps, further lengthening total trial time. Last, during pilot testing, we found that showing the target orientation at all times (and ensuring that the tori are placed at consistent locations relative to the user's line of sight), allowed the user to associate the destination target with a consistent position in the real world. In consideration of all four points above, we hypothesize that Handles will be the fastest condition in total completion time.

H2: Since the core components in Handles, the poles and tori, allow a user to easily determine the target orientation, and the connecting arrows show the remaining rota-

tion, the user should be able to determine their next action by a short glance. A user may need to watch a few cycles of Animate to completely understand the rotation instruction, potentially waiting for the animation to loop back to the beginning to follow along. SingleAxis should work well with ballistic movements, but due to potentially radical motions of the axis in the fine-tuning stage, users may become frustrated as they try to complete a small rotation. Euler works consistently throughout a rotation task, but the required axis completion order makes acceleration difficult; thus, more skilled users could potentially be limited in performance. Consequently, we believe Handles would be the preferred condition.

H3: As the tracked object nears the target orientation, the axis-cylinder is highly sensitive to small movements that change the rotation axis. If the user is only a few degrees from the angular completion threshold, but drifts slightly in following the rotation instruction, it is possible that they stop progressing towards the goal as they try to adjust to the new rotation axis. As a result, users could potentially become confused and frustrated. Secondly, since Static provides no instruction (other than a visualization of the target orientation), users may struggle trying to match the pose in the given threshold, when fine-tuning. As the target object is positionally offset from the virtual proxy, and rendered on a small display, it may be difficult to discern the exact orientation difference between the tracked and target objects. As a result, we expect users will rate either SingleAxis or Static as least preferred.

H4: In the fine-tuning stage, the remaining rotation is within 16° of the tracked object's current orientation, which for many of the visualizations may result in a very small or slight change. As explained in the rationale for H1, H2, and H3, Handles always

shows both the target orientation and the remaining rotation. This gives users two forms of feedback to use in the fine-tuning stage. If one is not clear, the other may still provide enough information to discern the correct instruction. Animate shows the exact motion needed to complete the task, but as the animated object is overlaid on the virtual proxy, and rendered on a small display, it is possible that a user simply may not be able to see the animated object clearly enough to discern the proper action. SingleAxis has the issue of the fast-moving rotation axis cylinder, as described in H3. This makes completing the fine-tuning task potentially difficult. Euler shows the remaining rotation per axis clearly, but since breaking the completion status of a particular axis may require that a user recomplete it before continuing, it is possible that users spend a nontrivial amount of time in fine-tuning dealing with previously completed axes.

H5: Each of the visualizations require that a user be able to mentally map the instruction to a motor action in rotating the tracked object. For SingleAxis, Animate, and Handles, the instructed action may be a rotation about an axis that does not line up with a natural axis of the held object, and that may require an unintuitive motion. Euler instructs the user by presenting the rotation guidance through a set of three transformations about axes fixed to the virtual proxy. This allows the user to focus on one axis per motion, potentially rotating the shape to a more easily manipulated orientation before beginning. As Euler does not require the user to map the rotation axis to one not attached to the tracked object, we expect that users with low spatial ability who may struggle with this particular mapping to prefer Euler.

5.4.4 Methods

Participants

We recruited 17 participants from our institution (9 female; ages 19–32, $\bar{X} = 23$), through email and posted flyers (protocol: IRB-AAAN4100). Participants attended a single-session experiment. Five participants had previous experience with AR, and none had any familiarity with our techniques.

Equipment

Participants wore a Google Glass Explorer Edition running Unity Remote 4. This Android app allows Glass to display visual output provided through USB 2.0 from our software running in Unity 3D 5.3.3 [Unity Technologies 2016] on a computer powered by an Intel i7-3770k with 16GB of RAM and an Nvidia GeForce GTX 780. (Our software can also run in Unity directly on Glass, but causes it to overheat too quickly to complete the study.) The object held by the user was tracked using a Logitech c920 camera (visible at the right of Figure 5.8), using tracking software in the Canon MREAL Platform, running on the same computer. The Logitech camera tracked both the held object and a fiducial array on the table where the user was seated, in order to ground the environment. The software running on Google Glass communicated to the MREAL Platform tracking application through a Unity application server, which ran on the same computer as the MREAL Platform software. Additionally, a foot-pedal was placed under the participant's table, and operated by them to progress through the study.

Design

Since it was possible that certain rotations would be easier to maneuver than others, depending on how the participant was holding the tracked object, the user study was designed to select from one of four possible rotations (80°, 100°, 120°, 140°) that would build on the target orientation in the preceding trial. There were an equal number of each of the possible rotation magnitudes across a single condition. Each trial would also generate a random rotation axis.

Trials were blocked by technique and randomized by rotation axis. Each block included four practice trials and 16 timed trials. The presentation order of the techniques was counterbalanced across participants to minimize bias due to learning.

Procedure

Participants were welcomed by the study coordinators and given the PseudoIsochromatic Plate (PIP) Color vision test to screen for color blindness, the Stereo Optical Co. Inc. Stereo Fly Test (SFT) to screen for stereo vision, and the Vandenberg and Kuse [1978] Mental Rotation Test (MRT) to screen for spatial ability. All participants passed the PIP test. 12 participants passed the SFT, four had weak stereo vision, and one failed the test.

After completing the tests, the participant was seated in a chair pushed up to a table and instructed to rest their elbows on a gel wrist pad (Figure 5.8) while holding the task object with both hands. These constraints ensured that their view of the virtual proxy was consistent with their view of the physical object. The participant was then introduced to the study and given an explanation of each of the techniques, with a small hands-on

demonstration session for each technique (consisting of two practice rotations using the technique). Before the first condition, the participant was given a detailed explanation of each interaction technique.

At the start of each trial, the participant was shown the virtual proxy of the tracked object and the visual components of the current technique. The participant was instructed to match the 3DOF pose demonstrated using the current technique. The participant was also instructed to press a button on a foot-pedal controller when the virtual proxy turned green, indicating that the target orientation had been met. The system prevented the participant from completing the trial by pressing the button if the tracked object had not yet entered the acceptable range for the trial (as explained in Section 5.4). Once the trial was complete, the participant was instructed to hold their pose for 1.5 seconds as they entered the next trial. Throughout the study, the positions and orientations of the tracked object were recorded.

Participants were asked to complete a three-part questionnaire before, during, and after the study, assessing the five techniques. The questionnaire included an unweighted NASA TLX, a request to rank the techniques from 1 ("Most Preferred") to 5 ("Least Preferred"), and room for free-form comments.

5.5 Results

Each participant completed a total of 80 timed trials (5 conditions $\times$ 16 timed trials). We evaluated our hypotheses for significance using a Bonferroni-corrected α of .01 ($.05/5$).

5.5.1 Task Duration

Outliers

We identified outliers in terms of task duration using Tukey's outlier filter [Tukey 1977]. We chose a standard threshold (1.5 times interquartile range per user per condition), resulting in 5.0% (68 of 1,360 trials: 13 ANIMATE, 13 EULER, 16 SINGLEAXIS, 15 STATIC, and 11 HANDLES) of our collected data being excluded from the rest of our analysis. Outliers resulted from occasional unstable tracking, connectivity issues, overheated HWD, or external issues (ringing cellphone, loose contact lens); in a few cases, users could not figure out the right answer and opted out, especially for STATIC.

Analysis

Our task completion metric is similar to reaction time (RT) data commonly analyzed in psychology experiments, in that we measure the time it takes users to react in response to a visual stimulus. Traditional analysis of variance (ANOVA) methods are generally not well-suited to RT data [Whelan 2008], because RT distributions are typically not Gaussian: they often have a long tail on the right, presumably due to confounding factors such as fatigue and external distractions. Before we began our analysis, we quickly confirmed that our task-completion data exhibited similar non-normality by fitting a linear model and visually inspecting the residual plots, which in fact showed obvious deviations from normality.

One widely adopted method for analyzing such heavily skewed RT data is to transform it into a reaction rate (analogous to speed) by taking the reciprocal ($1/x$) and then

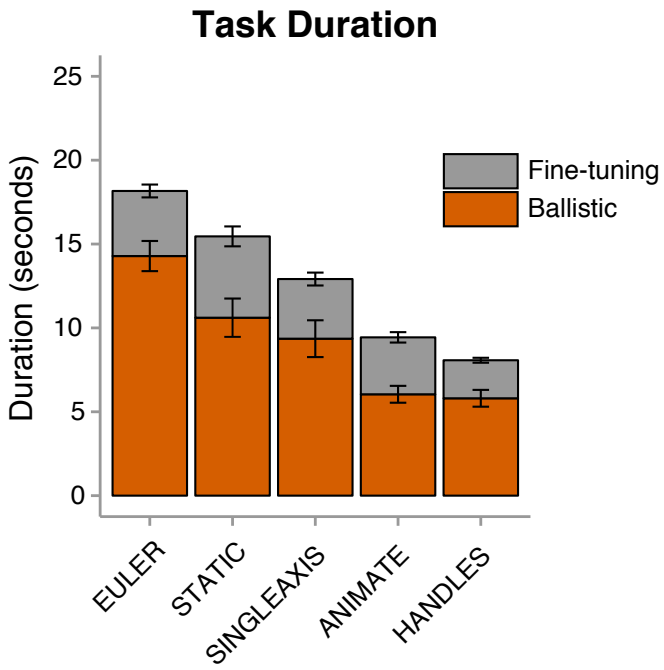


Figure 5.10: User Study: Task duration per technique.

fit it with a linear-mixed-effects (LME) model to identify significant effects by adding and removing factors [Baayen and Milin 2015]. Using R [R Core Team 2015] and its *lme4* package [Bates et al. 2015], we fit an LME model to our task-duration variable as a function of visualization condition (fixed effect) and participant (random effect). Compared to a base model with a random slope and participant as a random effect, a Kenward–Roger corrected F-test showed that visualization condition was significant as a fixed effect ($F_{(1,271)} = 119.16, p < .001$) (Figure 5.10). A pairwise least-squares means comparison revealed that our participants were fastest using HANDLES, followed by ANIMATE, SINGLEAXIS, STATIC, and EULER, in that order, where all pairwise differences were statistically significant (Table 5.1). These findings validated H1.

Pairwise Comparison	t-statistic	p-value
Handles vs. Animate	$t_{(1,271)} = 3.67$	$p < .001$
Animate vs. SingleAxis	$t_{(1,271)} = 4.98$	$p < .001$
SingleAxis vs. Static	$t_{(1,271)} = 4.49$	$p < .001$
Static vs. Euler	$t_{(1,271)} = 6.34$	$p < .001$

Table 5.1: Task duration—Pairwise comparisons

Ballistic Approach vs. Fine-Tuning

In H4, we hypothesized that Handles would lead to faster task completion times compared to other techniques. Subdividing performance into a ballistic phase, followed by a visual feedback phase for "fine tuning" [Gan and Hoffmann 1988], we believed Handles would be faster because of its emphasis on providing visible, persistent feedback to help facilitate fine-tuning. In contrast to other techniques that rely on displaying the difference between current and target orientation, in Handles the difference between current orientation and target orientation is small during fine-tuning.

To confirm this part of our hypothesis, we separated the overall completion time into two subtasks, as shown in Figure 5.10: ballistic approach and fine-tuning, following a simple distance thresholding heuristic. We counted the amount of time for each trial that was spent where the user-tracked object was more than a certain threshold away from the target orientation as *ballistic approach* and the rest of the time (i.e., when the tracked object's orientation was within that threshold) as *fine-tuning*. We chose 16° as our threshold for this analysis, which we arrived at by doubling our completion acceptance threshold of 8°.

Similar to how we analyzed overall task duration, we fitted two separate LME models to model reciprocals of ballistic and fine-tuning durations (i.e., ballistic and fine-tuning rates) as a function of visualization condition (fixed effect) and participant (random effect).

Ballistic Approach: For the ballistic approach, compared to a base model with a random slope and participant as a random effect, a Kenward–Roger corrected *F*-test showed that visualization condition was significant as a fixed effect ($F_{(1,271)} = 111.73, p < .001$). A pairwise least-squares means comparison revealed that there were significant differences between all pairs except Handles–Animate at $p < .01$. In other words, Handles and Animate were fastest, followed by SingleAxis, Static, and Euler, in that order.

Fine-Tuning: For fine-tuning, compared to a base model with a random slope and participant as a random effect, a Kenward–Roger corrected *F*-test showed that visualization condition was significant as a fixed effect ($F_{(1,271)} = 31.13, p < .001$). A pairwise least-squares means comparison revealed that there were significant differences between all pairs except Animate–SingleAxis, Animate–Euler, and Static–Euler at $p < .01$. In other words, Handles was fastest for fine-tuning, followed by Animate and SingleAxis, followed by Static and Euler, which confirms H4.

5.5.2 User Feedback

Technique Rankings

A majority of participants (9, 53%) ranked Handles as their most preferred condition (Figure 5.11), supporting H2. Three participants (18%) chose Animate and Euler each, two participants (12%) chose Static, and none chose SingleAxis as their favorite. On the

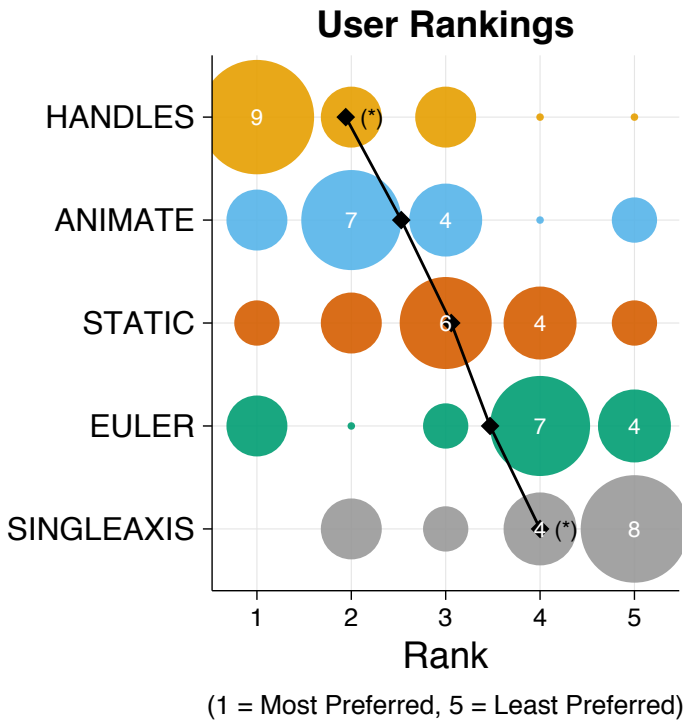


Figure 5.11: User Study: User rankings per technique. (*) denotes significance at $p < .01$. Size of circle and number inside it indicate the number of participants choosing a given rank.

opposite end of the spectrum, SingleAxis was chosen as the least favorite 8 times (53%), followed by Euler with four times (24%). On average, Handles was ranked highest, followed by Animate, SingleAxis, Static, and Euler. A Friedman test confirmed that our participants' differential preference between techniques was statistically significant, $\chi^2_{(4)} = 17.459,\ p < .01$.

In H2, we hypothesized that Handles would be the most preferred technique. A post-hoc pairwise comparison using Nemenyi's procedure showed that the only statistically significant difference in rankings was between Handles vs. SingleAxis, $p < .01$. The differences between Handles vs. Euler and Animate vs. SingleAxis were nearly significant ($p = .039$ and $p = .052$, respectively), but above our Bonferroni-adjusted α of .01, supporting but not confirming H2.

Qualitative user feedback highlighted instances where participants found Handles to be generally more preferred for fine-tuning (e.g., "Handles was very accurate and didn't have changing parameters," "Handles was the best for putting the object in the exact position that the program wanted," "Handles is the best because it really helps with the small movements").

On the opposite end of the spectrum, SingleAxis was generally rated as least preferred and participants reported frustration during fine-tuning (e.g., "SingleAxis was frustrating, since the bar seemed to move very erratically with small movements, so it took a lot of concentration to do the fine movements near the target"). While the first part of H3 (i.e., SingleAxis would be less preferred compared to Handles, Animate, and Euler) was supported in our data, it was not confirmed because not all pairwise differences had $p < .01$.

In H3, we also hypothesized that Static would be less preferred compared to Handles, Animate, and Euler. Surprisingly, Static was generally ranked higher than Euler in terms of overall preference, which meant that the second part of H3 should be rejected. Four participants (24%) ranked Euler as their least preferred technique and another six (35%) participants ranked it as their second least preferred.

In H5, we hypothesized that Euler would be preferred by participants with low spatial ability. The three users who ranked it as their most preferred technique had either above or close to average MRT scores of 8, 12, 14 (out of 20), failing to support H5.

To understand why many users found Euler challenging, we examined the questionnaire comments from participants who did not prefer Euler. Common themes were that the need to perform three sequential rotations along the major axes of the tracked object

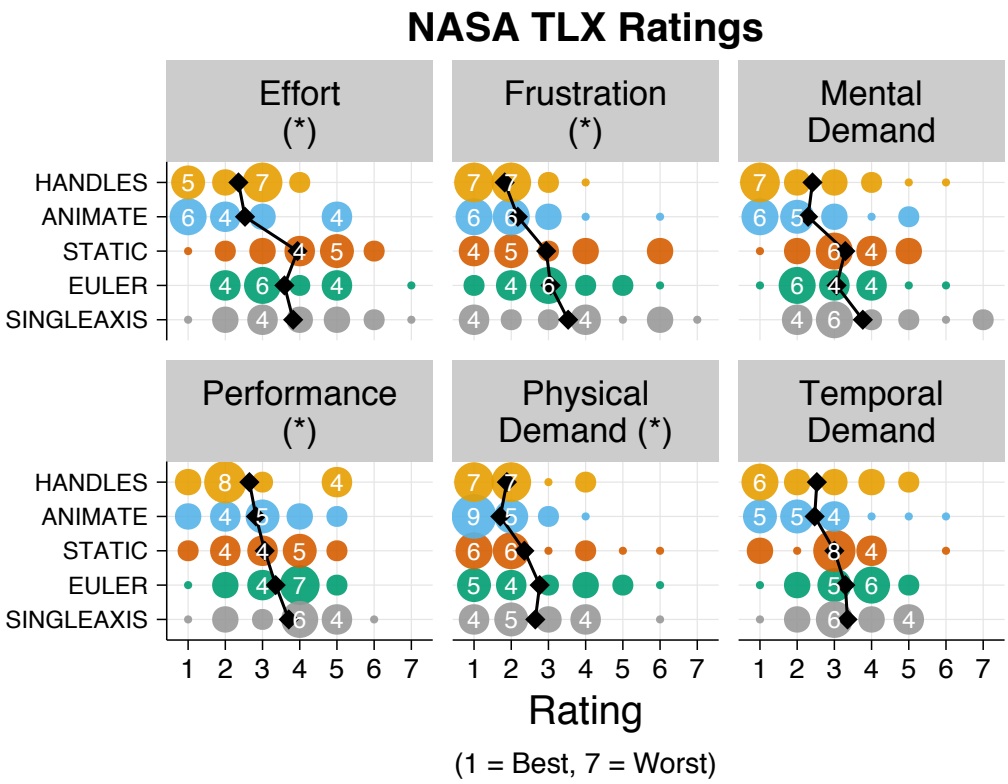


Figure 5.12: User Study: NASA TLX ratings per technique. (*) denotes significance at $p < .01$. Size of circle and number inside it indicate the number of participants choosing a given rating.

was onerous and having to track three arrows at once was challenging (e.g., "in aligning one axis, the other pre-aligned axes may drift and cause some confusion," "Too many rings, and too many changing rotations," "Holding the rotation in one axis constant while rotating the others was challenging. Also, following the order of rotation was not instinctual.").

NASA TLX

When we analyzed the results from the unweighted NASA TLX questionnaire (Figure 5.12), a Friedman test confirmed that technique was a significant factor for Mental Demand, Physical Demand, Effort, and Frustration at $p < .01$. (The p values for Temporal

Demand and Performance were $.081$ and $.051$, respectively.)

Post-hoc pairwise comparison using Nemenyi's procedure indicated that SingleAxis was rated as being significantly more mentally demanding compared to both Handles and Animate ($p < .01$). Handles was perceived to require less effort than both Static and SingleAxis, but the p values for the pairwise comparisons were just above our Bonferroni-adjusted α ($p = .017$ in both cases). Similarly, Handles was rated as less frustrating compared to SingleAxis and Euler, but that difference was also not significant ($p = .028$ and $p = .045$, respectively). There were no significant pairwise differences for Physical Demand.

5.5.3 Discussion

Animate and Static include a representation of the object in the desired orientation, encouraging comparison of the current orientation of the object with the desired orientation. In contrast, SingleAxis, Euler, and Handles provide virtual annotations as guidance (e.g., arrows, handles, and tori), requiring the user to attend only to those and shifting the task from spatial transformation to perceptual tracking. Despite this shift away from spatial thinking, Handles still provides a spatial representation of the final pose via its tori, which might explain why users were able to perform both ballistic and fine-tuning movements quickly. This was highlighted by several participants in their qualitative feedback (e.g., "Having the poles as a guide really helps. Don't have to think, can just get by with spatial intuition," "I just thought about how to put the sticks to the ring," "The two guides were very useful in determining the target position," "With Handles, I did not have to

observe the orientation of the object I was holding to solve the trial").

Chapter 6

Conclusions and Future Work

6.1 Contributions

People often need to perform spatial tasks that require switching viewpoints. Switching viewpoints in real or AR environments can be time-consuming and effortful. Moreover, people do not necessarily need continuous movement between viewpoints to maintain orientation (e.g., [Tversky 2005]).

With SnapAR, we developed and tested a set of interaction techniques for quickly and intuitively switching among viewpoints by using snapshots taken by a tracked camera in AR and manipulating virtual objects within those snapshots. We integrated our techniques into a prototype application for arranging virtual furniture in AR. We designed a representative measurable alignment task and ran a counterbalanced, within-subject user study to compare the performance and evaluation of switching viewpoints virtually versus physically by walking to a new location. The results of the experiment confirmed that quick virtual viewpoint switching was faster and more accurate for alignment activities than physical viewpoint switching, even when accounting for the overhead of making snapshots. The time savings and reduction in error were more pronounced for more difficult (i.e., non-orthogonal) alignment tasks. In addition, virtual viewpoint switching was

overwhelmingly preferred by participants, and regarded as less demanding than physically walking to and using live views. We are encouraged by the positive results and plan to apply this technique to other AR domains involving more complex navigation and manipulation tasks.

We have presented ParaFrustum, a geometric construct that generalizes a computer graphics camera frustum to make it possible to represent a range of positions and orientations associated with acceptable views of a task. We developed two visualizations that can communicate to a user how to achieve one of the views encoded by a ParaFrustum. A user study showed that visualizations of more loosely constrained ParaFrusta can guide a user to an acceptable pose significantly more quickly than visualizations of more tightly constrained ParaFrusta, providing a potential advantage for tasks that can be performed from a range of acceptable positions and orientations. The study showed faster performance with less head rotation and shorter and more direct trajectories with InSitu. Participants also preferred InSitu.

We have described one new visualization (Handles) and three visualizations that improve upon existing approaches for guiding a user in rotating an object to match a specified 3DOF orientation. In addition, we have presented the results of a user study comparing the effectiveness of these visualizations, when viewed on Google Glass, a small FOV, monoscopic, off-center HWD. Our study found that Handles was significantly faster than and trended toward being preferred over the other techniques.

6.2 Lessons Learned

In this section, we present several lessons we have learned along the way as we developed and evaluated our AR localization techniques.

Carefully designed transitional interfaces can provide users with practical benefits, such as improved task performance and reduced error, without increasing cognitive load. Transitional interfaces [Grasset et al. 2006] had been explored in a limited scope before, namely games [Phillips and Piekarski 2005; Cheok et al. 2002] and navigation [Mulloni et al. 2010]. During our initial development of SnapAR (Chapter 3), we argued that nonisomorphic "magic" travel techniques should enable users of a handheld AR application to save time and effort when switching viewpoints, however, we were not sure if the gains would be negated by increased cognitive load and decreased spatial awareness. The qualitative and quantitative feedback from our user study (Section 3.4) showed that users saved time, made fewer errors, and preferred using our transitional interface to switch viewpoints and manipulate virtual content from those viewpoints. This observation suggests that, for applicable tasks and scenarios, AR developers can break from tightly coupling the user's viewpoint to the physical display controlled by the user (i.e., head-worn or handheld) and consider transitional AR interfaces.

When guiding users to follow an action, UI elements that are embedded in the scene and provide real-time feedback seem to be preferred. We hypothesize that embedded UI elements allow users to leverage the spatial context from their surroundings to perform an integrated set of anticipatory actions. For example, in SnapAR, we displayed each stored snapshot as a virtual 3D camera icon in the scene, representing the 6DOF pose of

the handheld device when the snapshot was created (Section 3.3.1). This allowed users to point their handheld device in the direction they wanted to travel to, maintaining and reinforcing their mental model of how the snapshots are connected to the rest of the scene spatially. Similarly, our embedded ParaFrustum-InSitu visualization (Section 4.4) allowed users to save time and effort by following a curved path because they could easily anticipate their target orientation. The importance of real-time feedback became especially apparent in Orientation Assistance, when we noticed that applying smoothing to a critical UI element negatively impacted user performance (Section 5.3.2). While smoothing can be useful in avoiding jarring, unexpected changes, tasks that require precision like the ones we explored in this dissertation (e.g., the fine-tuning stage of an alignment task) seem to benefit greatly from users being able to perceive the impact of their movements immediately, so that they can continuously make subtle adjustments as they approach their target.

Small FOV, lightweight, monoscopic HWDs can have useful applications in interactive task assistance. AR task assistance applications typically rely on projectors or wide–field-of-view (FOV), stereoscopic HWDs to display instructions registered to physical task objects. In Chapter 5, we showed that smaller FOV, lightweight monoscopic HWDs, such as Google Glass, can also be successfully used for interactive task assistance. We believe that the key insight here is to include an interactive virtual proxy of the task object to show the instructions relative to, instead of attempting to overlay instructions on the physical task object seen through the narrow FOV.

6.3 Future Work

6.3.1 SnapAR

Although we were satisfied with the performance of our manipulation method within our prototype SnapAR applications, there are some limitations to the present system. Even though rotation and translation by amounts larger than users' range of motion can be accomplished by successively clutching and declutching, these controls may not appropriate for spaces larger than a tabletop or a small room. In larger settings, it may be worthwhile to explore nonisomorphic controls to allow users to move virtual objects over large distances. The manipulation mechanism used in our study implementation works under the assumption that there is a single virtual object to be manipulated. In contrast, one of our furniture layout prototypes supports object selection.

When users revisit snapshots, the state of the physical environment is presented as the static image captured by the snapshot. This may be a disadvantage for some AR applications that require up-to-date information about the physical world, but it can be sufficient for many other AR applications; for example, ones in which the environment is changed only by users, as presented in this paper. In some cases, viewing virtual content on a static image of the scene can be an advantage. By freezing the scene, we can ensure that multiple objects being compared in that context are compared in identical circumstances. For example, if we could properly render virtual furniture based on the time of day, we could capture a set of views at a particular time and then add the furniture to those views. This would avoid the problem caused by lighting conditions that change over time.

6.3.2 ParaFrustum

Authoring is an important area for future work. Authoring for a specific user and task could be accomplished pragmatically by collecting and processing 6DoF head pose data while the user assumes good (and bad) views for the task. An alternative approach could involve automated analysis of the task and environment to determine head poses that meet constraints on visibility and legibility, as well as user-specific constraints on height, reach, grasp, and general comfort (e.g., as computed by tools such as Siemens PLM Jack). In some cases, a ParaFrustum could be parameterized on user height by pivoting a head volume about a tail volume or changing the ParaFrustum's height. A generalized ParaFrustum might also be constructed using the union of the head volumes of more specific ParaFrusta, when the increased size does not cause problems (e.g., when the larger head volume accommodates taller users, but can be ignored by shorter ones).

While our current implementation uses ellipsoids for head and tail volumes, these volumes may be of arbitrary shape in general. One possible improvement would be to implement ParaFrusta that are based on convex superellipsoids, which include the cube, rounded cube, cylinder, and sphere as special cases.

While our current implementation handles convex head and tail volumes, concave volumes would provide better support for complex viewing constraints (e.g., to rule out points from which important objects are obscured). Concave volumes could be wrapped by a ParaFrustum's convex hull; however, the containment evaluation and user feedback employed in our visualizations would need to change. For example, after exiting a concave head volume, additional portions of it might still lie between the user's head and the tail

volume. Therefore, the visualization would need to determine which way to guide the user to reenter the head volume, while minimizing obscuration of the tail volume.

6.3.3 Orientation Assistance

While the visualizations were designed with small-FOV monoscopic HWDs in mind, they should also work well with wider-FOV stereoscopic HWDs. On a stereoscopic AR display, the virtual proxy could be eliminated and our visualizations could be registered with and rendered directly on the user's view of the task object. However, it is possible that relative performance among the visualizations may change with increased FOV and stereoscopy; for example, EULER might perform better relative to some of the other techniques, when not confined to a small monoscopic display. Further, using tracked, registered AR on a wider-FOV display might also result in different relative performance across the techniques. Thus, we believe it will be useful to run new studies to assess the relative performance of the techniques when used with different display technologies.

Our current version of EULER would be problematic for users with red–green color-blindness (we note that all of our study participants passed the PIP test). It would be a nice usability improvement to build in color profiles that could be applied to accommodate users with color-vision deficiencies.

While this work focused on unconstrained 3DOF rotation, it is possible that a number of the visualizations may work when completing 6DOF transformation tasks. Similarly, looking at specific types of constrained rotations (e.g., camera-plane vs. horizontal or vertical planes; or roll vs. pitch or yaw) could reveal interesting differences between tech-

niques. Thus, it would be interesting to explore how our visualizations may be modified or combined with other visualizations to support translation and constrained rotations, allowing them to address a full range of rigid-body transformations.

6.4 Final Thoughts

With the recent arrival of affordable, consumer-oriented HWDs for VR and the imminent introduction of handheld or head-worn consumer devices tailored for AR, we believe that it is more important than ever for UI designers and 3D interactive content developers to have a toolbox of visualizations for common subtasks (e.g., attention direction, virtual travel, localization) that have been designed and tested following cognitive principles. In this dissertation, we have focused our efforts developing a set of such visualizations that are designed to help users access strategic viewpoints quickly and accurately. While we recognize that our visualizations are not an exhaustive set of all useful visualizations in this domain, we hope that by presenting our exploration of the design space, describing the trade-offs, and providing a detailed account of our evaluations, we can help future researchers and designers to build on our work to fill this toolbox with more visualizations for more efficient, effective, safe, and enjoyable AR interfaces in the future.

Bibliography

Anderson, F., Grossman, T., Matejka, J., and Fitzmaurice, G. (2013). "YouMove: Enhancing Movement Training with an Augmented Reality Mirror." In: *Proc. ACM UIST*, pp. 311–320.
URL: http://doi.acm.org/10.1145/2501988.2502045.

Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., and MacIntyre, B. (2001). "Recent Advances in Augmented Reality." In: *IEEE Computer Graphics and Applications* 21.6, pp. 34–47.

Baayen, R. H. and Milin, P. (2015). "Analyzing Reaction Times." In: *International Journal of Psychological Research* 3.2, pp. 12–28.

Bae, S., Agarwala, A., and Durand, F. (2010). "Computational Rephotography." In: *ACM Trans. Graph.* 29.3, 24:1–24:15.
URL: http://doi.acm.org/10.1145/1805964.1805968.

Bates, D., Mächler, M., Bolker, B., and Walker, S. (2015). "Fitting Linear Mixed-Effects Models Using lme4." In: *Journal of Statistical Software* 67.1, pp. 1–48.

Baudisch, P. and Rosenholtz, R. (2003). "Halo: A Technique for Visualizing Off-Screen Objects." In: *Proc. SIGCHI*, pp. 481–488.
URL: http://doi.acm.org/10.1145/642611.642695.

Bichlmeier, C., Heining, S. M., Feuerstein, M., and Navab, N. (2009). "The Virtual Mirror: A New Interaction Paradigm for Augmented Reality Environments." In: *IEEE Transactions on Medical Imaging* 28.9, pp. 1498–1510.
URL: http://dx.doi.org/10.1109/TMI.2009.2018622.

Biocca, F., Tang, A., Owen, C., and Xiao, F. (2006). "Attention Funnel: Omnidirectional 3D Cursor for Mobile Augmented Reality Platforms." In: *Proc. SIGCHI*, pp. 1115–1122.
URL: http://doi.acm.org/10.1145/1124772.1124939.

Blender Online Community (2016). *Blender - a 3D Modeling and Rendering Package*. Blender Institute, Amsterdam: Blender Foundation.
URL: http://www.blender.org.

Bouisset, S. and Zattara, M. (1981). "A Sequence of Postural Movements Precedes Voluntary Movement." In: *Neuroscience Letters* 22.3, pp. 263–270.
URL: http://www.sciencedirect.com/science/article/pii/0304394081901178.

Bowman, D., Koller, D., and Hodges, L. (1997). "Travel in Immersive Virtual Environments: An Evaluation of Viewpoint Motion Control Techniques." In: *Proc. IEEE VR*, pp. 45–52.
URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=583043.

Bowman, D., Kruijff, E., LaViola, J., and Poupyrev, I. (2005). *3D User Interfaces: Theory and Practice*. Addison-Wesley.

Burtnyk, N., Khan, A., Fitzmaurice, G., Balakrishnan, R., and Kurtenbach, G. (2002). "Stylecam: Interactive Stylized 3d Navigation Using Integrated Spatial & Temporal Controls." In: *Proc. ACM UIST*, pp. 101–110.
URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.5.5382.

Chen, B., Neubert, B., Ofek, E., Deussen, O., and Cohen, M. F. (2009). "Integrated Videos and Maps for Driving Directions." In: *Proc. ACM UIST*, pp. 223–232.
URL: http://portal.acm.org/citation.cfm?id=1622176.1622218.

Cheok, A. D., Yang, X., Ying, Z. Z., Billinghurst, M., and Kato, H. (2002). "Touch-Space: Mixed Reality Game Space Based on Ubiquitous, Tangible, and Social Computing." In: *Personal Ubiquitous Comput.* 6 (5-6), pp. 430–442.
URL: http://portal.acm.org.ezproxy.cul.columbia.edu/citation.cfm?id=592615.

Chu, M. and Kita, S. (2008). "Spontaneous Gestures during Mental Rotation Tasks: Insights into the Microdevelopment of the Motor Strategy." In: *Journal of Experimental Psychology: General* 137.4, p. 706.

Chu, M. and Kita, S. (2011). "The Nature of Gestures' Beneficial Role in Spatial Problem Solving." In: *Journal of Experimental Psychology: General* 140.1, p. 102.

Clark, H. H. (1996). *Using Language*. Cambridge University Press.
URL: http://psycnet.apa.org/doi/10.2277/0521561582.

Darken, R. P. and Peterson, B. (2001). "Spatial Orientation, Wayfinding, and Representation." In: *In K. M. Stanney (ed.), Handbook of Virtual Environments: Design, Implementation, and Applications*, pp. 493–518.
URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.4619.

Elmqvist, N. and Tsigas, P. (2008). "A Taxonomy of 3d Occlusion Management for Visualization." In: *IEEE Transactions on Visualization and Computer Graphics* 14.5, pp. 1095–1109.
URL: http://portal.acm.org/citation.cfm?id=1446259.

Elmqvist, N., Tudoreanu, M. E., and Tsigas, P. (2008). "Evaluating Motion Constraints for 3D Wayfinding in Immersive and Desktop Virtual Environments." In: *Proc. SIGCHI*, pp. 1769–1778.
URL: http://portal.acm.org/citation.cfm?id=1357330.

Elvezio, C., Sukan, M., Feiner, S., and Tversky, B. (2015). "[POSTER] Interactive Visualizations for Monoscopic Eyewear to Assist in Manually Orienting Objects in 3D." In: *Proc. IEEE ISMAR*, pp. 180–181.

Elvezio, C., Sukan, M., and Feiner, S. K. (2016). "A Framework to Facilitate Reusable, Modular Widget Design\\for Real-Time Interactive Systems." In: *Proc. IEEE SEARIS*. Greenville, SC, USA.

Elvins, T. T., Nadeau, D. R., and Kirsh, D. (1997). "Worldlets-3D Thumbnails for Wayfinding in Virtual Environments." In: *Proc. ACM UIST*, pp. 21–30.
URL: http://doi.acm.org/10.1145/263407.263504.

Euler, L. (1775). "Formulae Generales pro Translatione Quacunque Corporum Rigidorum." In: *Novi Acad. Sci. Petrop* 20, pp. 189–207.

Feiner, S., MacIntyre, B., Höllerer, T., and Webster, A. (1997). "A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment." In: *Proc. IEEE ISWC*. ISWC '97, pp. 74–81.
URL: http://dl.acm.org/citation.cfm?id=851036.856454 (visited on 03/29/2014).

Feiner, S., Macintyre, B., and Seligmann, D. (1993). "Knowledge-Based Augmented Reality." In: *Commun. ACM* 36.7, pp. 53–62.
URL: http://doi.acm.org/10.1145/159544.159587.

Franchak, J. M., Celano, E. C., and Adolph, K. E. (2012). "Perception of Passage through Openings Depends on the Size of the Body in Motion." In: *Experimental Brain Research*

223.2, pp. 301–310. PMID: 22990292.

Franconeri, S., Jonathan, S., and Scimeca, J. (2010). "Tracking Multiple Objects Is Limited Only by Object Spacing, Not by Speed, Time, or Capacity." In: *Psychological Science* 21.7, pp. 920–925. PMID: 20534781.

Freeman, D., Benko, H., Morris, M. R., and Wigdor, D. (2009). "ShadowGuides: Visualizations for In-Situ Learning of Multi-Touch and Whole-Hand Gestures." In: *Proc. ACM ITS*, pp. 165–172.
URL: http://doi.acm.org/10.1145/1731903.1731935.

Friedman, D. and Feldman, Y. A. (2006). "Automated Cinematic Reasoning about Camera Behavior." In: *Expert Systems with Applications* 30.4, pp. 694–704.
URL: http://www.sciencedirect.com/science/article/pii/S0957417405001648 (visited on 03/29/2014).

Froehlich, P., Obernberger, G., Simon, R., and Reichl, P. (2008). "Exploring the Design Space of Smart Horizons." In: *Proc. ACM MobileCHI*, pp. 363–366.
URL: http://doi.acm.org/10.1145/1409240.1409289.

Gan, K.-C. and Hoffmann, E. R. (1988). "Geometrical Conditions for Ballistic and Visually Controlled Movements." In: *Ergonomics* 31.5, pp. 829–839. pmid: 3402428.
URL: http://dx.doi.org/10.1080/00140138808966724.

Georgel, P., Benhimane, S., Sotke, J., and Navab, N. (2009a). "Photo-Based Industrial Augmented Reality Application Using a Single Keyframe Registration Procedure." In: *Proc. IEEE ISMAR*, pp. 187–188.
URL: http://dx.doi.org/10.1109/ISMAR.2009.5336468.

Georgel, P., Schroeder, P., and Navab, N. (2009b). "Navigation Tools for Viewing Augmented CAD Models." In: *IEEE CG&A* 29.6, pp. 65–73.
URL: http://ieeexplore.ieee.org/xpl/downloadCitations (visited on 05/31/2010).

Girgensohn, A., Shipman, F., Turner, T., and Wilcox, L. (2007). "Effects of Presenting Geographic Context on Tracking Activity between Cameras." In: *Proc. SIGCHI*, pp. 1167–1176.
URL: http://doi.acm.org/10.1145/1240624.1240801.

Google Inc (2016). *Google Earth*. Version 7.1.7.
URL: https://www.google.com/earth/.

Grasset, R., Looser, J., and Billinghurst, M. (2006). "Transitional Interface: Concept, Issues and Framework." In: *Proc. IEEE ISMAR*, pp. 231–232.
URL: http://dx.doi.org/10.1109/ISMAR.2006.297819.

Grasso, R., Glasauer, S., Takei, Y., and Berthoz, A. (1996). "The Predictive Brain: Anticipatory Control of Head Direction for the Steering of Locomotion." In: *Neuroreport* 7.6, pp. 1170–1174. PMID: 8817526.

Grasso, R., Prévost, P., Ivanenko, Y. P., and Berthoz, A. (1998). "Eye-Head Coordination for the Steering of Locomotion in Humans: An Anticipatory Synergy." In: *Neuroscience Letters* 253.2, pp. 115–118. PMID: 9774163.

Gupta, A., Fox, D., Curless, B., and Cohen, M. (2012). "DuploTrack: A Real-Time System for Authoring and Guiding Duplo Block Assembly." In: *Proc. ACM UIST*, pp. 389–402.
URL: http://doi.acm.org/10.1145/2380116.2380167.

Gustafson, S., Baudisch, P., Gutwin, C., and Irani, P. (2008). "Wedge: Clutter-Free Visualization of Off-Screen Locations." In: *Proc. SIGCHI*, pp. 787–796.
URL: http://doi.acm.org/10.1145/1357054.1357179.

Güven, S., Feiner, S., and Oda, O. (2006). "Mobile Augmented Reality Interaction Techniques for Authoring Situated Media on-Site." In: *Proc. IEEE ISMAR*, pp. 235–236.
URL: http://dl.acm.org/citation.cfm?id=1514243.

Güven, S. and Feiner, S. (2006). "Interaction Techniques for Exploring Historic Sites through Situated Media." In: *Proc. IEEE 3DUI*, pp. 111–118.
URL: http://portal.acm.org/citation.cfm?id=1134820.1130540.

Hart, S. G. and Staveland, L. E. (1988). "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research." In: *Advances in psychology* 52, pp. 139–183.

Henderson, S. J. (2011). "Augmented Reality Interfaces for Procedural Tasks." Ph.D. Columbia University, New York: Dept. of Computer Science. 189 pp.
URL: http://search.proquest.com.ezproxy.cul.columbia.edu/pqdtft/docview/867426891/abstract/1431E2727573F4149B0/9?accountid=10226.

Henderson, S. J. and Feiner, S. (2009). "Evaluating the Benefits of Augmented Reality for Task Localization in Maintenance of an Armored Personnel Carrier Turret." In: *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*. ISMAR '09. Washington, DC, USA: IEEE Computer Society, pp. 135–144.
URL: http://dx.doi.org/10.1109/ISMAR.2009.5336486 (visited on 12/04/2013).

Henderson, S. J. and Feiner, S. K. (2011). "Augmented Reality in the Psychomotor Phase of a Procedural Task." In: *Proc. IEEE ISMAR*. Los Alamitos, CA, USA, pp. 191–200.

Hirose, K., Ogawa, T., Kiyokawa, K., and Takemura, H. (2006). "Interactive Reconfiguration Techniques of Reference Frame Hierarchy in the Multi-Viewport Interface." In: *Proc. IEEE 3DUI*, pp. 73–80.
URL: http://dx.doi.org/10.1109/VR.2006.89.

Hoang, T. N. and Thomas, B. H. (2011). "Multiple Camera Augmented Viewport: An Investigation of Camera Position, Visualizations, and the Effects of Sensor Errors and Head Movement." In: *Proc. ACM ICAT*, pp. 33–40.
URL: http://arrow.unisa.edu.au:8081/1959.8/122082.

Höllerer, T., Feiner, S., Terauchi, T., Rashid, G., and Hallaway, D. (1999). "Exploring MARS: Developing Indoor and Outdoor User Interfaces to a Mobile Augmented Reality System." In: *Computers and Graphics* 23, pp. 779–785.
URL: http://dx.doi.org/10.1016/S0097-8493(99)00103-X.

Howard, I. P. and Templeton, W. B. (1966). *Human Spatial Orientation*. Wiley. 568 pp.
URL: http://psycnet.apa.org/psycinfo/1966-11614-000.

Ichihara, E., Takao, H., and Ohta, Y. (1999). "NaviView: Bird's-Eye View for Highway Drivers Using Roadside Cameras." In: *Proc. IEEE ICMCS*, pp. 559–565.
URL: http://portal.acm.org.ezproxy.cul.columbia.edu/citation.cfm?id=839287.841934.

Johnson, T. E. (1963). "Sketchpad III: A Computer Program for Drawing in Three Dimensions." In: *Proc. Spring Joint Computer Conf.* Pp. 347–353.
URL: http://doi.acm.org/10.1145/1461551.1461592.

Kameda, Y., Takemasa, T., and Ohta, Y. (2004). "Outdoor See-through Vision Utilizing Surveillance Cameras." In: *Proc. IEEE ISMAR*, pp. 151–160.
URL: http://portal.acm.org/citation.cfm?id=1033712.

LaViola Jr., J. J. (2000). "A Discussion of Cybersickness in Virtual Environments." In: *SIGCHI Bull.* 32.1, pp. 47–56.
URL: http://doi.acm.org/10.1145/333329.333344 (visited on 01/16/2017).

Lee, G. A., Yang, U., Kim, Y., Jo, D., Kim, K.-H., Kim, J. H., and Choi, J. S. (2009). "Freeze-Set-Go Interaction Method for Handheld Mobile Augmented Reality Environments." In: *Proc. ACM VRST*, pp. 143–146.
URL: http://doi.acm.org/10.1145/1643928.1643961.

Levelt, W. J. M. (1989). *Speaking: From Intention to Articulation*. MIT Press. 588 pp. URL: http://psycnet.apa.org/psycinfo/1989-97544-000.

Lorenz, H., Trapp, M., Jobst, M., and Döllner, J. (2008). "Interactive Multi-Perspective Views of Virtual 3D Landscape and City Models." In: *11th AGILE Intl. Conf. on GI Science*. Ed. by Bernard, L., Friis-Christensen, A., and Pundt, H., pp. 301–321. URL: http://link.springer.com/chapter/10.1007%2F978-3-540-78946-8_16.

McCloskey, R. (1941). *Make Way for Ducklings*. OCLC: 192241. New York: The Viking Press.

McCloud, S. (1994). *Understanding Comics: The Invisible Art*. William Morrow Paperbacks. 224 pp.

Mennie, N., Hayhoe, M., and Sullivan, B. (2007). "Look-Ahead Fixations: Anticipatory Eye Movements in Natural Tasks." In: *Experimental Brain Research* 179.3, pp. 427–442. PMID: 17171337.

Miau, D. and Feiner, S. (2016). "Personalized Compass: A Demonstration of a Compact Visualization for Direction and Location." In: *Proc. SIGCHI*. ACM, pp. 3731–3734.

Mijksenaar, P. and Westendorp, P. (1999). *Open Here: The Art of Instructional Design*. Joost Elffers Books. 148 pp.

Miller, A., White, B., Charbonneau, E., Kanzler, Z., and LaViola Jr., J. J. (2012). "Interactive 3D Model Acquisition and Tracking of Building Block Structures." In: *IEEE Transactions on Visualization and Computer Graphics* 18.4, pp. 651–659.

Mohr, P., Kerbl, B., Donoser, M., Schmalstieg, D., and Kalkofen, D. (2015). "Retargeting Technical Documentation to Augmented Reality." In: *Proc. ACM CHI*, pp. 3337–3346.

Mulloni, A., Dünser, A., and Schmalstieg, D. (2010). "Zooming Interfaces for Augmented Reality Browsers." In: *Proc. MobileHCI*, pp. 161–170. URL: http://doi.acm.org/10.1145/1851600.1851629.

Oda, O., Elvezio, C., Sukan, M., Feiner, S., and Tversky, B. (2015). "Virtual Replicas for Remote Assistance in Virtual and Augmented Reality." In: *Proc. ACM UIST*, pp. 405–

415.
URL: http://doi.acm.org/10.1145/2807442.2807497.

Oda, O. and Feiner, S. (2014). *Goblin XNA Framework*.
URL: http://goblinxna.codeplex.com/ (visited on 04/16/2014).

Oda, O., Sukan, M., Feiner, S., and Tversky, B. (2013). "Poster: 3D referencing for remote task assistance in augmented reality." In: *3D User Interfaces (3DUI), 2013 IEEE Symposium on*. IEEE, pp. 179–180.

Pausch, R., Burnette, T., Brockway, D., and Weiblen, M. E. (1995). "Navigation and Locomotion in Virtual Worlds via Flight into Hand-Held Miniatures." In: *Proc. ACM SIGGRAPH*, pp. 399–400.
URL: http://doi.acm.org/10.1145/218380.218495.

Perry, A. and Wallace, E. (2014). *Constructive Solid Geometry (CSG) for Unity in C#*.
URL: https://github.com/omgwtfgames/csg.cs (visited on 04/16/2014).

Phillips, K. and Piekarski, W. (2005). "Possession Techniques for Interaction in Real-Time Strategy Augmented Reality Games." In: *Proc. ACM ACE*, p. 10.
URL: http://portal.acm.org/citation.cfm?id=1178584.

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing.
URL: http://www.R-project.org/.

Robertson, C. M., MacIntyre, B., and Walker, B. N. (2008). "An Evaluation of Graphical Context When the Graphics Are Outside of the Task Area." In: *Proc. IEEE ISMAR*, pp. 73–76.
URL: http://dx.doi.org/10.1109/ISMAR.2008.4637328.

Schmalstieg, D., Encarnação, L. M., and Szalavári, Z. (1999). "Using Transparent Props for Interaction with the Virtual Table." In: *Proc. ACM I3D*, pp. 147–153.
URL: http://portal.acm.org/citation.cfm?id=300523.300542.

Schmidt, R., Singh, K., and Balakrishnan, R. (2008). "Sketching and Composing Widgets for 3D Manipulation." In: *Computer Graphics Forum* 27.2, pp. 301–310.
URL: http://dx.doi.org/10.1111/j.1467-8659.2008.01127.x.

Sehnal, D. and Campbell, M. (2014). *MIConvexHull Library*.
URL: http://miconvexhull.codeplex.com/ (visited on 04/16/2014).

Shepard, R. N. and Metzler, J. (1971). "Mental Rotation of Three-Dimensional Objects." In: *Science* 171.3972, pp. 701–703.
URL: http://www.sciencemag.org/content/171/3972/701.abstract.

Shingu, J., Rieffel, E., Kimber, D., Vaughan, J., Qvarfordt, P., and Tuite, K. (2010). "Camera Pose Navigation Using Augmented Reality." In: *Proc. IEEE ISMAR*, pp. 271–272.
URL: http://dx.doi.org/10.1109/ISMAR.2010.5643602.

Siltanen, S. and Woodward, C. (2006). "Augmented Interiors with Digital Camera Images." In: *Proc. IEEE AUIC*, pp. 33–36.
URL: http://dl.acm.org/citation.cfm?id=1151758.1151761.

Smith, S. P. and Hart, J. (2006). "Evaluating Distributed Cognitive Resources for Wayfinding in a Desktop Virtual Environment." In: *Proc. IEEE VR*, p. 115.
URL: http://portal.acm.org/citation.cfm?id=1130523.

Snavely, N., Seitz, S. M., and Szeliski, R. (2006). "Photo Tourism: Exploring Photo Collections in 3D." In: *ACM TOG*. SIGGRAPH '06 25.3, pp. 835–846.
URL: http://dl.acm.org/citation.cfm?id=1141964.

Sodhi, R., Benko, H., and Wilson, A. (2012). "LightGuide: Projected Visualizations for Hand Movement Guidance." In: *Proc. SIGCHI*, pp. 179–188.
URL: http://doi.acm.org/10.1145/2207676.2207702.

Stewart, J. A. (2014). *FreeWRL, an Open Source X3D/VRML Viewer*.
URL: http://freewrl.sourceforge.net (visited on 01/09/2014).

Stoakley, R., Conway, M. J., and Pausch, R. (1995). "Virtual Reality on a WIM: Interactive Worlds in Miniature." In: *Proc. SIGCHI*, pp. 265–272.
URL: http://dx.doi.org/10.1145/223904.223938.

Sukan, M., Feiner, S., Tversky, B., and Energin, S. (2012). "Quick Viewpoint Switching for Manipulating Virtual Objects in Hand-Held Augmented Reality Using Stored Snapshots." In: *Proc. IEEE ISMAR*, pp. 217–226.
URL: http://dx.doi.org/10.1109/ISMAR.2012.6402560.

Sukan, M., Elvezio, C., Feiner, S., and Tversky, B. (2016). "Providing Assistance for Orienting 3D Objects Using Monocular Eyewear." In: *Proc. ACM SUI*, pp. 89–98.
URL: http://doi.acm.org/10.1145/2983310.2985764.

Sukan, M., Elvezio, C., Oda, O., Feiner, S., and Tversky, B. (2014). "ParaFrustum: Visualization Techniques for Guiding a User to a Constrained Set of Viewing Positions and Orientations." In: *Proc. ACM UIST*, pp. 331–340.
URL: http://doi.acm.org/10.1145/2642918.2647417.

Tang, A., Owen, C., Biocca, F., and Mou, W. (2003). "Comparative Effectiveness of Augmented Reality in Object Assembly." In: *Proc. SIGCHI*, pp. 73–80.
URL: http://doi.acm.org/10.1145/642611.642626.

Tönnis, M. and Klinker, G. (2006). "Effective Control of a Car Driver's Attention for Visual and Acoustic Guidance towards the Direction of Imminent Dangers." In: *Proc. IEEE ISMAR*, pp. 13–22.
URL: http://portal.acm.org/citation.cfm?id=1514211.

Trimble Navigation (2016). *SketchUp*. Version 17.1.174.
URL: http://www.sketchup.com/.

Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley.
URL: http://www.getcited.org/pub/101667026.

Tversky, B. (1981). "Distortions in Memory for Maps." In: *Cognitive Psychology* 13.3, pp. 407–433.
URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-0001174077&partnerID=40&md5=6b374f2960433b9d447704404d995c71.

Tversky, B. (2005). "Visuospatial Reasoning." In: *The Cambridge Handbook of Thinking and Reasoning*. Ed. by Holyoak, K. J. and Morrison, R. G. Cambridge University Press.

Unity Technologies (2016). *Unity*. Version 5.5.
URL: http://unity3d.com.

Vandenberg, S. G. and Kuse, A. R. (1978). "Mental Rotations, a Group Test of Three-Dimensional Spatial Visualization." In: *Perceptual and motor skills* 47.2, pp. 599–604.

Veas, E., Grasset, R., Kruijff, E., and Schmalstieg, D. (2012). "Extended Overview Techniques for Outdoor Augmented Reality." In: *IEEE TVCG* 18.4, pp. 565–572.
URL: http://dx.doi.org/10.1109/TVCG.2012.44.

Veas, E., Mulloni, A., Kruijff, E., Regenbrecht, H., and Schmalstieg, D. (2010). "Techniques for View Transition in Multi-Camera Outdoor Environments." In: *Proc. GI*, pp. 193–200.
URL: http://dl.acm.org/citation.cfm?id=1839214.1839248.

Voyer, D., Voyer, S., and Bryden, M. P. (1995). "Magnitude of Sex Differences in Spatial Abilities: A Meta-Analysis and Consideration of Critical Variables." In: *Psychological Bulletin* 117.2, pp. 250–270. pmid: 7724690.

VTT (2011). *ALVAR Tracking Subroutines Library*.
URL: http://www.vtt.fi/multimedia/alvar.html (visited on 11/26/2011).

Wang, W. and Milgram, P. (2001). "Dynamic Viewpoint Tethering for Navigation in Large-Scale Virtual Environments." In: *Proc. HFES*. Vol. 45, pp. 1862–1866.
URL: http://dx.doi.org/10.1177/154193120104502702.

Web3D Consortium (2014). *X3D Specification*.
URL: http://www.web3d.org/x3d/specifications/ (visited on 01/09/2014).

Wexler, M., Kosslyn, S. M., and Berthoz, A. (1998). "Motor Processes in Mental Rotation." In: *Cognition* 68.1, pp. 77–94.

Whelan, R. (2008). "Effective Analysis of Reaction Time Data." In: *The Psychological Record* 58.3, p. 475.

White, S., Lister, L., and Feiner, S. (2007). "Visual Hints for Tangible Gestures in Augmented Reality." In: *Proc. IEEE ISMAR*, pp. 47–50.

Wickens, C. D. and Prevett, T. T. (1995). "Exploring the Dimensions of Egocentricity in Aircraft Navigation Displays." In: *Journal of Experimental Psychology: Applied* 1.2, pp. 110–135.
URL: http://dx.doi.org/10.1037/1076-898X.1.2.110.

Wohlschläger, A. and Wohlschläger, A. (1998). "Mental and Manual Rotation." In: *Journal of Experimental Psychology: Human Perception and Performance* 24.2, p. 397.

Zhai, S., Buxton, W., and Milgram, P. (1994). "The "Silk Cursor": Investigating Transparency for 3D Target Acquisition." In: *Proc. ACM CHI*, pp. 459–464.
URL: http://dx.doi.org/10.1145/191666.191822.

Appendix

SnapAR Questionnaire

Comparative Study of Hand-Held Augmented Reality User Interface Methods

Participant ID: _____
IRB Protocol: IRB-AAAF2995
Principal Investigator: Steven Feiner (skf1)
Co-Investigator: Mengu Sukan (ms3774)

User Experience Survey

Date:

Age: Gender: F / M

I use a computer…
never
monthly
weekly
daily
multiple times per day

I am familiar with Augmented Reality
no
yes

I play computer games…
never
monthly
weekly
daily
multiple times per day

For each question, we would appreciate any additional comments you have in the "Comments" section.

PART I—Rating Methods. For the following questions, please circle a number from 1 through 7 to describe your experience using each method.

A:

Mental Demand: How mentally demanding was the task?

Very Low						Very High
1	2	3	4	5	6	7

Physical Demand: How physically demanding was the task?

Very Low						Very High
1	2	3	4	5	6	7

Temporal Demand: How hurried or rushed was the pace of the task?

Very Low						Very High
1	2	3	4	5	6	7

Performance: How successful were you in accomplishing what you were asked to do?

Perfect						Failure
1	2	3	4	5	6	7

Effort: How hard did you have to work to accomplish your level of performance?

Very Low						Very High
1	2	3	4	5	6	7

Frustration: How insecure, discouraged, irritated, stressed, and annoyed were you?

Very Low						Very High
1	2	3	4	5	6	7

Comments:

B:

Mental Demand: How mentally demanding was the task?

Very Low						Very High
1	2	3	4	5	6	7

Physical Demand: How physically demanding was the task?

Very Low						Very High
1	2	3	4	5	6	7

Temporal Demand: How hurried or rushed was the pace of the task?

Very Low						Very High
1	2	3	4	5	6	7

Performance: How successful were you in accomplishing what you were asked to do?

Perfect						Failure
1	2	3	4	5	6	7

Effort: How hard did you have to work to accomplish your level of performance?

Very Low						Very High
1	2	3	4	5	6	7

Frustration: How insecure, discouraged, irritated, stressed, and annoyed were you?

Very Low						Very High
1	2	3	4	5	6	7

Comments:

C:

Mental Demand: How mentally demanding was the task?

Very Low						Very High
1	2	3	4	5	6	7

Physical Demand: How physically demanding was the task?

Very Low						Very High
1	2	3	4	5	6	7

Temporal Demand: How hurried or rushed was the pace of the task?

Very Low						Very High
1	2	3	4	5	6	7

Performance: How successful were you in accomplishing what you were asked to do?

Perfect						Failure
1	2	3	4	5	6	7

Effort: How hard did you have to work to accomplish your level of performance?

Very Low						Very High
1	2	3	4	5	6	7

Frustration: How insecure, discouraged, irritated, stressed, and annoyed were you?

Very Low						Very High
1	2	3	4	5	6	7

Comments:

D:

Mental Demand: How mentally demanding was the task?

Very Low						Very High
1	2	3	4	5	6	7

Physical Demand: How physically demanding was the task?

Very Low						Very High
1	2	3	4	5	6	7

Temporal Demand: How hurried or rushed was the pace of the task?

Very Low						Very High
1	2	3	4	5	6	7

Performance: How successful were you in accomplishing what you were asked to do?

Perfect						Failure
1	2	3	4	5	6	7

Effort: How hard did you have to work to accomplish your level of performance?

Very Low						Very High
1	2	3	4	5	6	7

Frustration: How insecure, discouraged, irritated, stressed, and annoyed were you?

Very Low						Very High
1	2	3	4	5	6	7

Comments:

PART II—Ranking Methods. In the following questions, please place a 1 through 4 next to each choice. If you feel that multiple methods performed roughly the same, then give them the same ranking.

Rank the methods by how much you would prefer using them, from 1 (most prefer) to 4 (least prefer).

___ A

___ B

___ C

___ D

Comments:

Rank the methods by how demanding (mentally demanding + physically demanding + temporally demanding) you thought they were, from 1 (least demanding) to 4 (most demanding).

___ A

___ B

___ C

___ D

Comments:

Please provide any additional comments about or reactions to any of the methods:

ParaFrustum Questionnaire

ParaFrustum Study

* Required

Participant ID (Ask the study coordinator) *

Age *

Gender *

- Female
- Male

Height in centimeters *
(Conversion Table: http://www.albireo.ch/bodyconverter/table.htm)

I use a computer... *

- Never
- Monthly
- Weekly
- Daily
- Multiple times per day

I have experience using Augmented Reality systems *

- No
- Yes

If you answered "Yes" to the previous question, please explain your experience.

Part 1: X

For the following questions, please choose a number from 1 through 7 to describe your experience with technique X.

For each question, we would appreciate any additional comments you have in the "Comments" section.

How mentally demanding was the task? *

	1 (Very Low)	2	3	4	5	6	7 (Very High)
X	○	○	○	○	○	○	○

How physically demanding was the task? *

	1 (Very Low)	2	3	4	5	6	7 (Very High)
X	○	○	○	○	○	○	○

How hurried or rushed was the pace of the task? *

	1 (Very Low)	2	3	4	5	6	7 (Very High)
X	○	○	○	○	○	○	○

How successful were you in accomplishing what you were asked to do? *

	1 (Perfect)	2	3	4	5	6	7 (Failure)
X	○	○	○	○	○	○	○

How hard did you have to work to accomplish your level of performance? *

	1 (Very Low)	2	3	4	5	6	7 (Very High)
X	○	○	○	○	○	○	○

How insecure, discouraged, irritated, stressed and annoyed were you? *

	1 (Very Low)	2	3	4	5	6	7 (Very High)
X	○	○	○	○	○	○	○

How easy was this technique for getting your head into a specific pose? *

	1 (Very Easy)	2	3	4	5	6	7 (Very Hard)

X							

How fast was this technique for getting your head into a specific pose? *

	1 (Slow)	2	3	4	5	6	7 (Fast)
X							

How accurate was this technique for getting your head into a specific pose? *

	1 (Not Accurate)	2	3	4	5	6	7 (Very Accurate)
X							

Please provide any additional comments about or reactions to techniques X. *

Part 2: Y

For the following questions, please choose a number from 1 through 7 to describe your experience with technique Y.

For each question, we would appreciate any additional comments you have in the "Comments" section.

How mentally demanding was the task? *

	1 (Very Low)	2	3	4	5	6	7 (Very High)
Y							

How physically demanding was the task? *

	1 (Very Low)	2	3	4	5	6	7 (Very High)
Y							

How hurried or rushed was the pace of the task? *

	1 (Very Low)	2	3	4	5	6	7 (Very High)
Y							

How successful were you in accomplishing what you were asked to do? *

	1 (Perfect)	2	3	4	5	6	7 (Failure)
Y							

How hard did you have to work to accomplish your level of performance? *

	1 (Very Low)	2	3	4	5	6	7 (Very High)
Y							

How insecure, discouraged, irritated, stressed and annoyed were you? *

	1 (Very Low)	2	3	4	5	6	7 (Very High)
Y							

How easy was this technique for getting your head into a specific pose? *

	1 (Very Easy)	2	3	4	5	6	7 (Very Hard)
Y							

How fast was this technique for getting your head into a specific pose? *

	1 (Slow)	2	3	4	5	6	7 (Fast)
Y							

How accurate was this technique for getting your head into a specific pose? *

	1 (Not Accurate)	2	3	4	5	6	7 (Very Accurate)
Y							

Please provide any additional comments about or reactions to any of the techniques. *

Part 3: X & Y

Please tell us which one you prefer.

For each question, we would appreciate any additional comments you have in the "Comments" section.

Which technique do you prefer for getting your head into a specific pose? *

- X
- Y

Please provide any additional comments about or reactions to any of the techniques.

Submit

100%: You made it.

Never submit passwords through Google Forms.

Orientation Assistance Questionnaire

Demographic Information

Comparative Study of Task Assistance using Lightweight Mobile Computers

* Required

Participant ID (Ask the study coordinator) *

Your answer

Age *

Your answer

Gender *

- ○ Female
- ○ Male

I use a computer... *

- ○ Never
- ○ Monthly
- ○ Weekly

- ◯ Daily
- ◯ Multiple times per day

I have experience using Augmented Reality systems *

- ◯ No
- ◯ Yes

If you answered "Yes" to the previous question, please explain your experience.

Your answer

Page 1 of 1

SUBMIT

Never submit passwords through Google Forms.

Condition Evaluation

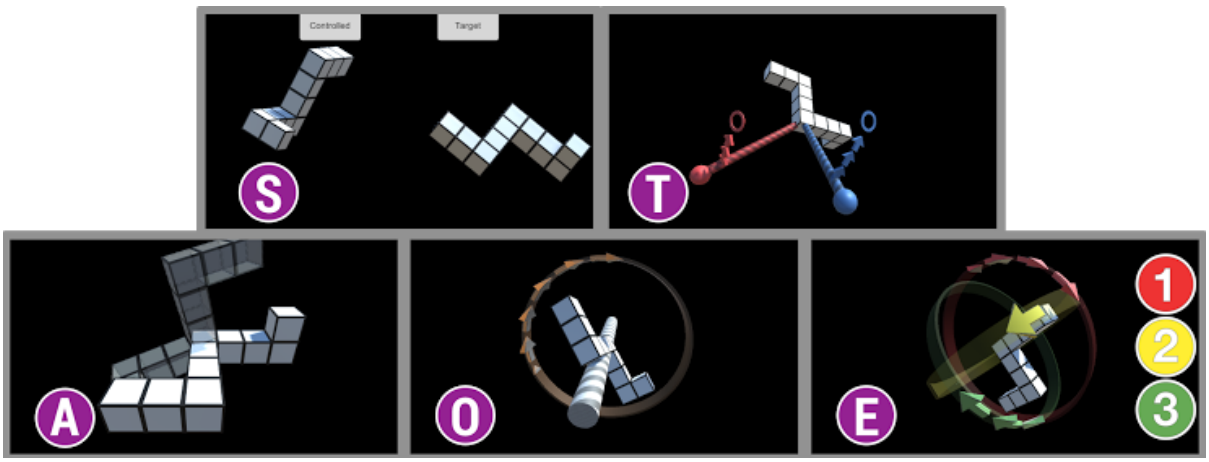
Comparative Study of Task Assistance using Lightweight Mobile Computers

* Required

Participant ID *

Your answer

Conditions



Which condition did you just complete? *

- S
- T
- A

- O
- E

Mental Demand *

How mentally demanding was the task?

	1 (Very Low)	2	3	4	5	6	7 (Very High)
Mental Demand							

Physical Demand *

How physically demanding was the task?

	1 (Very Low)	2	3	4	5	6	7 (Very High)
Physical Demand							

Temporal Demand *

How hurried or rushed was the pace of the task?

	1 (Very Low)	2	3	4	5	6	7 (Very High)
Temporal Demand							

Performance *

How successful were you in accomplishing what you were asked to do?

	1 (Perfect)	2	3	4	5	6	7 (Failure)
Performance							

Effort *

How hard did you have to work to accomplish your level of performance?

	1 (Very Low)	2	3	4	5	6	7 (Very High)
Effort	○	○	○	○	○	○	○

Frustration *

How insecure, discouraged, irritated, stressed and annoyed were you?

	1 (Very Low)	2	3	4	5	6	7 (Very High)
Frustration	○	○	○	○	○	○	○

Please provide any additional comments about or reactions to the condition you have just completed. *

Your answer

SUBMIT

Study Wrap-up

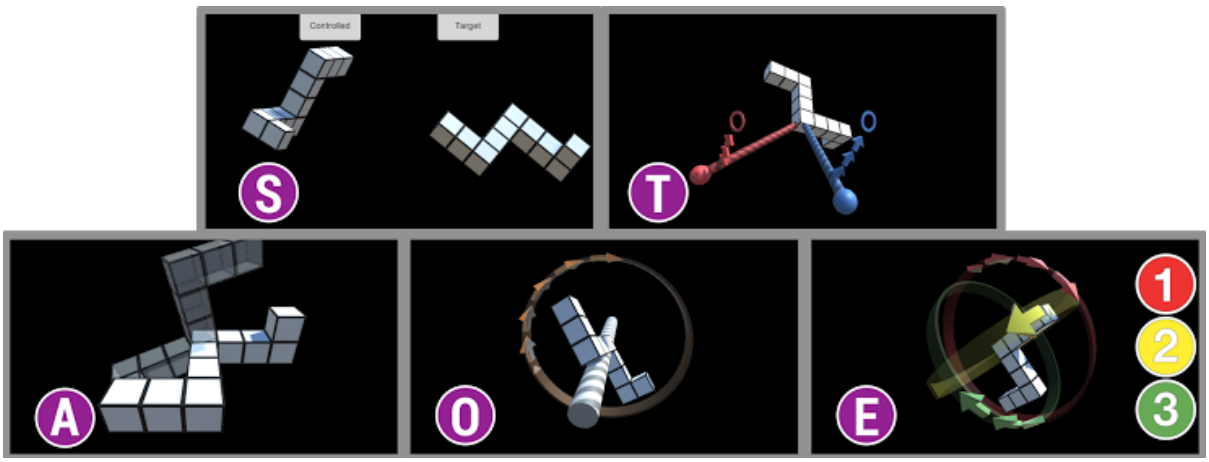
Comparative Study of Task Assistance using Lightweight Mobile Computers

* Required

Participant ID *

Your answer

Conditions



Please rank the conditions from 1 (most preferred) to 5 (least preferred) *

	1 (most preferred)	2	3	4	5 (least preferred)
S	○	○	○	○	○

T	○	○	○	○	○
A	○	○	○	○	○
O	○	○	○	○	○
E	○	○	○	○	○

Please explain why you chose your rankings.

Your answer

Page 1 of 1

SUBMIT

Never submit passwords through Google Forms.
