UNIVERSITY OF CALIFORNIA, SAN DIEGO

RealityFlythrough: A System for Ubiquitous Video

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in
Computer Science

by

Neil James McCurdy

Committee in charge:

Professor William G. Griswold, Chair
Professor James D. Hollan
Professor Leslie A. Lenert
Professor Stefan Savage
Professor Mohan M. Trivedi

2007

The dissertation of Neil James McCurdy is approved, and it is acceptable in quality and form for publication on microfilm:

_____

_____

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2007

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank the open source community for the many components that have been integrated into my work. This project would not have been possible without their tireless effort and their willingness to share their code with the world. I am standing on their shoulders as much as I am standing on the shoulders of those cited in my bibliography. I would specifically like to acknowledge the contributors to the following projects: boost.org, openh323.org, FFmpeg, imagemagick.org, the Cartographic Projections Library at remotesensing.org, and the Spatial Index written by Marios Hadjieleftheriou at AT&T.

I would also like to thank my colleagues in Bill Griswold's Ubiquitous Computing/Software Engineering lab, the members of the WIISARD project, and many individuals in the Department of Computer Science and CalIt2 for their insightful comments, help with exasperating problems, and help running or setting up experiments. I list their names here at the risk of leaving someone out: Patricia Shanahan, Adriene Jenik, Steve Brown, Bob Boyer, Jennifer Carlisle, Ted Chan, James Kileen, Doug Palmer, Colleen Burono, Barry Demchak, Tim Sohn, Kevin Li, Alan Calvitti, Fang Liu, Ricky Huang, Neil Jones, Charles Lucas, MacNeil Shonle, Tod Ferguson, Julie Conner, and the San Diego MMST.

I am particularly grateful to my committee members: Stefan Savage, Leslie Lenert, Jim Hollan, and Mohan Trivedi for their help in framing and focusing my research. I am also indebted to Leslie Lenert for including me on the WIISARD team and giving me access to the real-world scenario of disaster response which gave my research a driving theme.

My chair, advisor, and very good friend, Bill Griswold, played a much bigger role in the completion of this dissertation than he will ever know. He gave me confidence when I doubted myself, he gave me comfort in the face of rejection, and his enthusiasm for my work and his incredibly insightful observations drove my research. I cannot thank him enough, and this work is as much his as it is mine.

And finally I would like to thank my family for putting up with me for the last

six years. My wife, Michelle, embraced this experience as much as I did, relishing the cut in salary and the move into humble graduate housing. Collette, my daughter, was our ticket into graduate housing, and without her I would not have been able to afford grad school. Ian, my son, was a big enough addition to the family to create enough claustrophobia in our little apartment to compel us to move and force me to finish. My mother is quite possibly the only person to read this dissertation in full, and I thank her immensely for that. That is only a small, but very representative example of the support that she and my father have given me my whole life. Collin, my older brother, was my real inspiration. He opened my eyes to graduate school and showed me that a Ph.D. was within my reach. I am returning the favor by finishing first. The love of my family not only made this experience possible but also enjoyable. Thank you.

Chapter 3 is, in part, a reprint of the material as it appears in A systems architecture for ubiquitous video. Neil J. McCurdy and William G. Griswold. In Mobisys 2005: Proceedings of the Third International Conference on Mobile Systems, Applications, and Services, pages 114. Usenix, 2005. The dissertation author was the primary investigator and author of this paper.

Chapter 5 is, in part, a reprint of the material as it appears in A robust abstraction for first-person video streaming: Techniques, applications, and experiments. Neil J. McCurdy, William G. Griswold, and Leslie A. Lenert. In ISM 06: Proceedings of the Eighth IEEE International Symposium on Multimedia, pages 235-244, Washington, DC, USA, 2006. IEEE Computer Society. The dissertation author was the primary investigator and author of this paper.

Section 7.2 is, in part, a reprint of the material as it appears in Harnessing mobile ubiquitous video. Neil J. McCurdy, Jennifer N. Carlisle, and William G. Griswold. In CHI 05: CHI 05 extended abstracts on Human factors in computing systems, pages 1645-1648, New York, NY, USA, 2005. ACM Press. The dissertation author was the primary investigator and author of this paper.

This work was supported in part by a gift from Microsoft Research and contract N01-LM-3-3511 from the National Library of Medicine.

VITA

2007	Doctor of Philosophy in Computer Science University of California, San Diego San Diego, CA
2005	Master of Science in Computer Science University of California, San Diego San Diego, CA
2002-2007	Graduate Student Researcher Department of Computer Science and Engineering University of California, San Diego

PUBLICATIONS

Neil J. McCurdy and William G. Griswold, "A Systems Architecture for Ubiquitous Video", *Mobisys 2005: Proceedings of the Third International Conference on Mobile Systems, Applications, and Services*, Usenix, 2005.

Neil J. McCurdy, William G. Griswold, and Leslie A. Lenert. "A robust abstraction for first-person video streaming: Techniques, applications, and experiments", *In ISM 06: Proceedings of the Eighth IEEE International Symposium on Multimedia*, pages 235-244, Washington, DC, USA, 2006. IEEE Computer Society.

ABSTRACT OF THE DISSERTATION

RealityFlythrough: A System for Ubiquitous Video

by

Neil James McCurdy
Doctor of Philosophy in Computer Science
University of California, San Diego, 2007
Professor William G. Griswold, Chair

We are rapidly moving toward a world of ubiquitous video where personal networked video cameras are everywhere. Already, camera-equipped cell phones are becoming commonplace. Imagine being able to tap into all of these live video feeds to remotely explore the world in real-time. We introduce RealityFlythrough, a telepresence system that makes this vision possible. By situating live 2d video feeds in a 3d model of the world, RealityFlythrough allows any space to be explored remotely. No special cameras, tripods, rigs, scaffolding, or lighting is required to create the model, and no lengthy preprocessing of images is necessary. Rather than try to achieve photorealism at every point in space, we instead focus on providing the user with a sense of how the video streams relate to one another spatially. By providing cues in the form of dynamic transitions and by stitching together live and archived views of the scene, we can approximate photorealistic telepresence while harnessing cameras "in the wild."

This dissertation describes how to construct a system like RealityFlythrough and explores the issues with deploying such a system in a real-world setting where limits in wireless bandwidth place constraints on the quality and number of mobile video feeds.

The primary contribution of this dissertation is a demonstration that with the appropriate division of labor between the human brain and the computer, some computationally intractable problems can be solved. In particular, this dissertation demonstrates that the computationally intensive task of stitching multiple video feeds into a cohesive

whole can be avoided by having the computer perform a relatively simple rough stitch that is accurate enough to allow the human brain to complete the process.

This dissertation also makes the following contributions to the field of telepresence: 1) A functioning system is presented and studied in a series of user experiments, 2) a robust system architecture is described that has successfully adapted to a series of unanticipated changes, 3) a novel visualization technique is presented that reduces the computational requirements that have so far posed a barrier to live, real-time image synthesis in real-world environments, and 4) this same visualization technique is shown to reduce the negative effects of low-frame-rate video.

Chapter 1

Introduction

Ubiquitous computing is often described as computers fading into the woodwork [Wei95]. Ubiquitous video, then, is cameras fading into the woodwork, a notion captured by the expression, "the walls have eyes." Ubiquitous video is characterized by wireless networked video cameras located in every conceivable environment. The data is transmitted either to a central server or simply into the ether for all to view. Although many believe that such an environment is inevitable [Bri98], we do not have to wait for the future to take advantage of ubiquitous video. There are a number of scenarios that could benefit from having live, situated access to ubiquitous video streams using today's technology.

Consider, for example, scenarios where it would be useful to attach head-mounted cameras to personnel entering dangerous, restricted, or remote sites. The video feeds can be streamed to a control "room" where commanders can navigate through the remote environment using the information acquired from the cameras. There are numerous such scenarios: In a disaster response setting, the failure to achieve adequate situational awareness can have catastrophic outcomes [MC02]. Live video situated in the disaster scene may be of benefit. Police Special Weapons and Tactics (SWAT) teams [JH02] that are routinely involved in high risk tactical situations may derive a similar benefit from live video. Other examples are: Hazardous Materials (HazMat) teams securing and decontaminating dangerous sites, police monitoring of events that

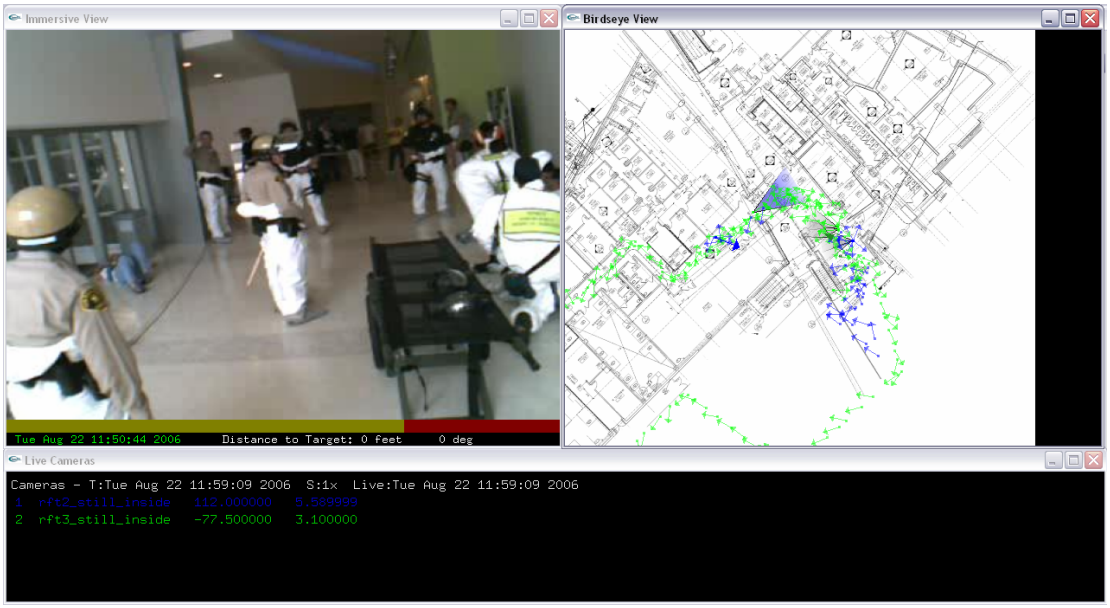


Figure 1.1: A screenshot of a typical RealityFlythrough session. The left window is the immersive view that shows the video feeds, camera images, or transitions. The right window is the birdseye view that shows all of the images that can be viewed. Each arrow represents a camera view, and the arrows are colored to indicate which camera they came from. The bottom window provides some information about the current state of the system.

attract large numbers of people such as holiday celebrations or protest marches, security personnel monitoring a remote site, and scientists studying a remote environment—one as benign as a nursery school or as dangerous as a volcano.

The common thread through this class of applications is that the harsh conditions of the real world need to be accommodated, and live, real-time access to the video is a requirement. Also, true, though, is that the accuracy of the data is far more critical than aesthetics.

The key to harnessing ubiquitous video is in managing the incoming video streams. A naive approach would display the video on an array of monitors similar to those used in many building security systems today. An ideal solution would have infinite cameras in the field, and allow the user to move seamlessly through the environment choosing any desired vantage point. A more practical solution provides the illusion of the ideal system while operating under the constraints imposed by the real environment.

Figure 1.2: Snapshots of a transition. The transition uses two "filler" images to provide additional contextual information. During this transition the viewpoint moves roughly 20 meters to the right of the starting image and rotates 135 degrees to the right. Figure 1.4 shows a transition between the same two images, but uses more advanced techniques described later that make the experience more aesthetically pleasing. The simpler transition shown here provides the user with a similar level of contentful information, however, and is a better illustration of the core RealityFlythrough concepts.

We have created RealityFlythrough, a system that uses video feeds obtained from mobile ubiquitous cameras to present the illusion of an environment that has infinite camera coverage. Stitching the multiple video streams together into a single scene is a straightforwardly sensible abstraction of numerous video streams. With such an abstraction, the user need only understand one integrated scene, as in a video game, rather than multiple feeds, as in a building security system. However, the limited number of cameras as well as the untamed elements of ubiquitous video make such an abstraction non-trivial to construct.

The key *limitation* of ubiquitous video is the incomplete coverage of the live video streams–every square meter of a space cannot be viewed from every angle with a live video stream at any chosen moment. For two cameras pointing in two rather different directions, when the user switches from viewing one camera to another, it is often not obvious how the subject matter in the two views relate to each other, nor is it obvious what is in the intervening space between the two cameras.

To address this limitation, RealityFlythrough fills the intervening space between two cameras with older imagery (captured from the live camera feeds), and provides segues (i.e., transitions) between the two live cameras that sequences and blends the imagery in a way that provides the sensation of a human performing a walking camera pan. In certain scenarios the display of older imagery may be undesirable or impossible due to a lack of camera coverage. While not ideal, transitions without back-

ground imagery are still sensible because the motion and timing of the transition and background spherical and floor grids convey the angle and distance traveled. The user has complete control over how older imagery is displayed—whether it is displayed at all, in a sepia tone, or with an age-indicator-bar.

The key *untamed element* of ubiquitous video is the imprecision of the sensed location and orientation of a camera (due to both sensor latency and sensor inaccuracy). Such imprecision gives misleading cues to the user about how the subject matter seen in one camera relates to the subject matter in another. For example, the images might appear farther apart than they really are.

Under certain assumptions, offline vision techniques could perform seamless stitching [Sze94]. To achieve real-time flythrough, this problem is instead handled by taking advantage of a property of the human visual system called *closure* [McC93]. *Closure* describes the brain's ability to fill in gaps when given incomplete information. It is a constant in our lives; *closure*, for example, conceals from us the blind spots that are present in all of our eyes but it also works at higher levels of cognition allowing us to make sense of two disjointed frames in a comic strip. RealityFlythrough embraces *closure* by presenting the user with an approximate model of the relationships between two camera views, and having the user's visual system infer the relationships between the objects in the views. Dynamic transitions between still-images and live video feeds reveal the misregistrations in overlapping images (with an alpha blend), rather than hiding them through blending or clipping. Although this sacrifices aesthetics, the benefits obtained due to *closure* increase sensibility. For this technique to work, images must overlap. This property is sought by the mechanism that captures the older still-images for filling. When no intervening images are available, however, the spherical and planar grids provide cues to the translational and rotational motion required to get from one camera's view to another, aiding another form of closure that McCloud describes as *Scene-to-Scene Closure* [McC93]. More information about *closure* and why the RealityFlythrough illusion works can be found in chapter 4.

1.1 How does RealityFlythrough Work?

A significant part of the user experience in RealityFlythrough is dynamic and does not translate well to the written word or still-photographs. We encourage the reader to watch a short video [MG05a] that presents an earlier version of RealityFlythrough, but we do our best to convey the subtlety of the experience in this section. When observing the images in Fig. 1.2, keep in mind that the transformation between the images is occurring within about one second, and the transitional frames represent only about 1/10th of the transition sequence.

The user's display is typically filled with either a video stream or a still-image taken directly from a camera. When the user is "hitchhiking" on a camera in this way, the experience is similar to watching a home-video where the camera operator is walking around while filming. A still-image, then, is simply the home-video paused. When a new vantage point is desired, a short transition sequence is displayed that helps the user correlate objects in the source image stream with objects in the destination image stream. These transitions are shown in a first-person view and provide the users with the sensation that they are walking from one location to another. The illusion is imperfect, but the result is sensible and natural enough that it provides the necessary contextual information without requiring much conscious thought from the users.

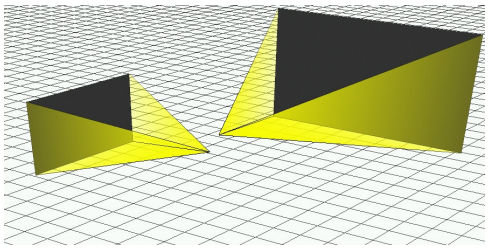


Figure 1.3: An illustration of how the virtual cameras project their images onto a wall.

RealityFlythrough works by situating 2d images in 3d space. Because the position and orientation of every camera is known, a representation of the camera can be placed at the corresponding position and orientation in virtual space. The camera's image is then projected onto a virtual wall (see Fig. 1.3). When the user is looking

at the image of a particular camera, the user's position and direction of view in virtual space is identical to the position and direction of the camera. As a result, the entire screen is filled with the image. Referring to Fig. 1.2, a *transition* between camera A (the left-most image) and camera B (the right-most image) is achieved by smoothly moving the user's position and view from camera A to camera B while still projecting their images in perspective onto the corresponding virtual walls. By using OpenGL's standard perspective projection matrix to render the images during the transition, the rendered view situates the images with respect to each other and the viewer's position in the environment. Overlapping portions of the images are blended using an alpha-blend. By the end of the transition, the user's position and direction of view are the same as camera B's, and camera B's image fills the screen. As shown in Fig. 1.2, additional images are displayed (if available and if desired) to help provide contextual information.

It may be easier to understand how RealityFlythrough works by envisioning the following concrete example. Imagine standing in an empty room that has a different photograph projected onto each of its walls. Each image covers an entire wall. The four photographs are of a 360 degree landscape with one photo taken every 90 degrees. Position yourself in the center of the room looking squarely at one of the walls. As you slowly rotate to the left your gaze will shift from one wall to the other. The first image will appear to slide off to your right, and the second image will move in from the left. Distortions and object misalignment will occur at the seam between the photos, but it will be clear that a rotation to the left occurred, and the images will be similar enough that sense can be made of the transition. RealityFlythrough operates in a much more forgiving environment: the virtual walls are not necessarily at right angles, and they do not all have to be the same distance away from the viewer.

1.2 Why does RealityFlythrough Work?

RealityFlythrough works in the wild because the human visual system has a high tolerance for error. The computationally complex pre-processing that would be

required to create perfect alignment of images during transitions is not necessary, and the position sensors which often can only record a rough estimate of position are good enough. The only information RealityFlythrough needs to know about each camera is its position in space, its angle of view, and its field of view. The position of the camera can be obtained from whatever locationing technology is desired (we use WAAS-enabled consumer GPS's for outdoor tests), and the tilt, roll, and yaw can be determined with a tilt sensor that has a magnetic compass (we use an AOSI EZ-Compass). We have found that an accuracy of 6-9 meters is adequate in outdoor settings where there is a wide field of view. Much higher accuracy would be necessary indoors—room-level accuracy, at minimum. Orientation accuracy is much more important because a camera that has less than a 40 degree field of view (typical of most web cameras) cannot be off by many degrees before images do not overlap at all. Magnetic compasses have provided good results, but sometimes have trouble in areas of high magnetism.

Since the transitions between images are modeled by moving the user's gaze from one virtual wall to another, the processing requirements are minimal once the virtual walls and corresponding texture maps (the images from the cameras) are loaded into the system. The rendering requirements of modern 3d games are far greater since the complete 3d structure of every object in view (each of which is usually made up of thousands of polygons) must be projected into 2d space. The RealityFlythrough virtual walls, in contrast are simple rectangles, and there are usually no more than five of them visible at any time. Since modern graphics cards can render far more complex environments at 60 frames per second, RealityFlythrough can provide a live, real-time, immersive view of a remote scene using commodity hardware.

The experience of moving through a RealityFlythrough environment is similar to the experience of moving through a 3d environment in modern computer games. The major difference, however, is that we cannot recover the 3d geometry of the scene, which would allow the user to get an accurate rendering of the scene from any arbitrary position. Instead, we are forced to simplify 3d space to a finite set of 2d projections (the actual images of the environment that have been captured by the cameras), where

near objects are pushed out to the camera plane and distant objects are pulled in. This simplification is what allows RealityFlythrough to work in the wild, but it is also the source of a lot of the problems that we have had to overcome. We have found that people are quite capable of resolving the ambiguities that arise during simple transitions where only a few images are involved and the misregistrations between them are minimal, but once users are allowed to move between arbitrary locations, the number of potential images involved in the transition and the potentially greater misregistrations that occur across larger distances make the transitions much less sensible. The real challenge with creating this system was in determining how to best simplify complex movements into a series of simple movements that were readily understandable by the user, while at the same time giving the user as much freedom as possible in their exploration of the space.

1.3 Other Forms of Telepresence

Now that the reader has a basic understanding of what RealityFlythrough is, it is worth taking a step back to briefly look at what other forms of telepresence are out there to better understand the approach that we took. We will limit the discussion in this section to the forms of telepresence that allow exploration of a real-world environment. We will take a broader view of presence (including its manifestation in artificial environments) and more thorougly discuss related work in chapter 2.

There have been several approaches to telepresence with each operating under a different set of assumptions: telepresence [KIN+95], tele-existence [Tac98a], and tele-reality [KRVS99a, Sze94]. Telepresence and tele-existence both generally describe a remote existence facilitated by some form of robotic device or vehicle. There is typically only one such device per user. Tele-reality constructs a model by analyzing the images acquired from multiple cameras, and attempts to synthesize photo-realistic novel views from locations that are not covered by those cameras.

RealityFlythrough contains elements of both telepresence and tele-reality. It is like telepresence in that the primary view is through a real video camera, and it is like

tele-reality in that it combines multiple video feeds to construct a more complete view of the environment. RealityFlythrough is unlike telepresence in that the cameras are likely attached to people instead of robots, there are many more cameras, and the location and orientation of the cameras is not as easily controlled. It is unlike tele-reality in that the primary focus is not to create photo-realistic novel views, but to help users to internalize the spatial relationships between the views that are available.

All of this work (including RealityFlythrough) is differentiated by the assumptions that are made and the problems being solved. Telepresence assumes an environment where robots can maneuver, and has a specific benefit in environments that would typically be unreachable by humans (Mars, for example). Tele-reality assumes high density camera coverage, a lot of time to process the images, and extremely precise calibration of the equipment. The result is photorealism that is good enough for movie special effects ("The Matrix Revolutions" made ample use of this technology). An alternative tele-reality approach assumes a-priori acquisition of a model of the space [NYH+03a], with the benefit of generating near photo-realistic live texturing of static structures.

There is another category of work that has its foundations in tele-reality but is more geared toward static images. Photosynth is the premiere example of this, creating spatial relationships between large collections of photos and organizing them in the form of large panoramic images [SSS06]. The navigation metaphor is similar to that of RealityFlythrough's in that the user is expected to look at actual images and not view the scene from novel points of view. Google's Streetview (`http://maps.google.com/help/maps/streetview`) is another impressive example of the spatial synthesis of archived imagery that is the culmination of work dating all the way back to the 1978 Aspen Movie Map [Cla] by way of the 1995 Quick Time Vr [Che95].

The primary difference between RealityFlythrough and the non-telepresence related work cited above is that RealityFlythrough creates a *live* representation of any real-world environment that is being explored, and it can do this with live *video streams* in conjunction with still photographs.

The difference between RealityFlythrough and traditional telepresence solutions that do allow live exploration of an environment is the use of humans as camera operators instead of robots. Robots are effective in some situations, but also have some limitations: (1) robots are physical devices that need to move through the environment and are limited by their physicality, (2) robots either require some form of artificial intelligence to maneuver successfully through the environment or they need a reliable tether (wired or wireless) to a human operator, and (3) the cost of such a device is usually high. RealityFlythrough instead can make use of one or more human camera operators that are comparatively agile and intelligent. The video feeds (or perhaps only images) can be transmitted across an unreliable network asynchronously, and the camera operator can make reasonably intelligent decisions about what to film and where to travel even with no communication with the people viewing the feeds. And finally, the ability to aggregate multiple feeds allows the people viewing the feeds to move across impassable areas freely by moving from one camera to the next.

1.4 How do you Build RealityFlythrough?

The architecture of RealityFlythrough can be described at two levels of detail. There is the system architecture that describes how all of the various client devices, sensors, server software, and viewing stations interact and communicate with one another, and then there is the architecture of the RealityFlythrough engine which sits on the server. The latter is the focus of this dissertation, but the former needs to be briefly discussed to give the reader a better understanding of the system.

1.4.1 System Architecture

The RealityFlythrough system is made up of one or more camera units and a server. The camera units contain a camera for recording video, a GPS device for recording the position of the camera, a tilt-sensor for recording pitch, roll, and yaw, and a handheld computer for synthesizing this data. As the reader will soon learn, the client

software was later enhanced to also perform point-matching on consecutive frames to augment the data retrieved from the position sensors.

The camera units transmit the video and sensor data to the server using a wireless protocol such as 802.11 or 1xEVDO Cellular. We used the H323 video conferencing protocol for handshake and video codec negotiation.

The server consists of an H323 Multipoint Control Unit which collects the incoming streams from the various camera units and the RealityFlythrough Engine which creates the user experience. The RealityFlythrough engine is discussed in more detail in the next section and throughout this dissertation. The viewing station is currently integrated with the server allowing only a single user of the system, but the system architecture can be extended to support multiple viewing stations. Details of this modification are discussed in section 3.6.3.

1.4.2 Engine Architecture

The purpose of the RealityFlythrough engine is to create the user experience that simulates a first-person immersive free-walk through a remote environment. The engine must decide where in a virtual 3D space to place the images recovered from the cameras, and which images to display at any given time. The simplicity of this description hints at the power of the architecture, but belies the complexity of doing this well enough to create the appropriate illusion of walking freely through space. As an example of this complexity, consider that even though each camera image is accompanied by sensor data that records the image's location in real space, the sensors are not always accurate and the data they collect may contradict information that point-matching algorithms provide. The point-matching results are not always accurate, either, though, so which data should be trusted, and more importantly, where should the image be placed in the virtual 3D space? The complexity of this decision will be discussed in detail in chapter 6, but we will briefly touch on it later in this section once the architectural components that support the abstraction of infinite camera coverage have been introduced.

As alluded to earlier, the engine architecture greatly reduces the complexity of

the system, replacing complicated algorithms with concepts as simple as fitness functions. The architecture has two unique qualities. First, it uniformly represents all image sources and outputs as *Cameras*, supporting a rich yet simple set of operations over those elements in achieving the desired abstractions. And, second, it employs a separate *Transition Planner* to translate the user's navigation commands into a sensible sequence of camera transitions and accompanying image blends.

A *camera* in RealityFlythrough is simply a position in space (x, y, z), an orientation (pitch, roll, and yaw), a field of view, and an image that represents that field of view. With such a loose definition, many RealityFlythrough components can be described as cameras. Obviously, the video cameras that provide our source input are *cameras*, but so too is each one of the frames of those video feeds. The user's view of the environment is also represented as a *camera* since the user has a position, orientation, and field of view. The camera that represents the user's view is a *Virtual Camera* which because of its generality has many other uses in the RealityFlythrough engine. A *Virtual Camera* is a *camera* that contains one or more other cameras. A transition from one image to another is modeled as adding *cameras* as appropriate to the *Virtual Camera*, and simply moving from one position and orientation to another while doing a cross-fade between the images.

Two enhancements to the system that we will learn about a little later are also represented as *cameras*. A *Smart Camera* is a *Virtual Camera* that applies RealityFlythrough-style transitions between consecutive frames of a video feed, creating a more sensible view of low-frame rate video, and a *Composite Camera* is a *Virtual Camera* that creates a panoramic, stitched, composite of multiple frames. The *Composite Camera* is basically just a *Virtual Camera* that has more accurate placement of its constituent images.

The role of the *Transition Planner* is to select which images to display at any given time. Recall that the effectiveness of a transition is dependent on the display of archived imagery when there is no overlap between the source and destination images. The *Transition Planner* decides which imagery to display. It may not be obvious why

we cannot just display all intervening images. For one thing, RealityFlythrough is designed to be effective in dynamic environments where scenes may change quite a bit from moment to moment. Images of the same scene may look quite different and confuse the user who is trying to use *closure* to make sense of the environment. Another reason for not showing all images has to do with the length of time it takes for most people to commit *closure*. We have found that most users do well with a *closure* interval of roughly one second. Anything less creates confusion. The *Transition Planner*, then, must choose the images that will give the best representation of travel through the environment while ensuring that each image is in view for a period of time long enough to facilitate *closure*.

1.4.3 Handling Dynamic Environments

The architecture described above is well suited to handle dynamic environments where not only do the objects in the scene move, but so too do the cameras that are capturing the scene. The details behind this can be found in chapter 3, but we highlight the important concepts in this section.

A challenge of real-world dynamic environments is that the number of camera feeds that can be supported is limited by the bandwidth of current wireless networks. One way to overcome this limitation is to move the cameras throughout the scene, thereby increasing the view of the cameras. The older imagery (the previous frames) can be stored and used as a reference for what a portion of the scene looked like "not too long ago". By continually refreshing these archived views with new images whenever one of the mobile cameras pans over the same scene again, it is possible to create the illusion that there are far more cameras viewing the scene than there really are. The goal of RealityFlythrough is to create the illusion of infinite camera coverage, and the use of archived imagery is one way that we create this illusion.

The use of mobile video cameras creates a number of further challenges, however. There is the obvious challenge of determining the location and orientation of the cameras, but for the purpose of this dissertation, we assume that GPS and commodity

orientation sensors solve that problem. A more surprising challenge is how to move to a mobile camera. Since the mobile camera may be moving, the challenge becomes how to zero in on a moving target. The solution is not as simple as making constant course corrections similar to how a missile acquires a target because the course corrections can be rather nauseating to the user who must watch as the images of the scene sway from side. The motion can actually be quite severe because while the actual position of the mobile cameras does not change that much during the one or two seconds it may take to move to the destination camera, the camera's orientation could swing back and forth by 90 or more degrees several times during that interval. Remember, these cameras may be attached to peoples' heads.

The solution that we arrived at was to move to where the camera was located, and then to quickly zero in on the camera's orientation once we get there. Since the camera probably will not move much during the one or two second transition, the bulk of the final zeroing-in is rotational motion only and we can quickly zero-in on the orientation without causing too much confusion to the user.

A further challenge is that the archived imagery that is displayed to the user to fill in gaps may also consist of live views from mobile cameras. In fact, it is actually desirable for this to happen because the live views would provide the most consistent and temporally-sound view of the scene. Since the cameras are moving, we cannot know where the camera will be or in which direction it will be facing until the user's viewpoint actually intersects with the camera's viewpoint. This suggests that we cannot pre-compute the user's path as the user moves from one camera to another. We have to determine *just-in-time* what camera to display next so that we do not accidentally choose a mobile camera that is in fact now facing in the opposite direction.

All of these challenges were handled well by the RealityFlythrough engine architecture. The use of the *camera* abstraction allowed us to model mobile cameras and the archived images from the cameras in the same way. This allowed us to easily interchange the two when planning transitions. As far as the *Transition Planner* was concerned, all of these *cameras* were the same, but some may have had more desirable

properties such as being live or more recent. Having the *Transition Planner* be a separate entity allowed us to create the dynamic, just-in-time transition planning that was necessary to avoid the challenges outlined above.

1.4.4 Using Transitions to Compensate for Low Frame Rates

A further challenge of using mobile cameras in real-world environments is that the bandwidth provided by current wireless networks is so limited and so variable that even if the number of cameras in an environment are limited, it is still difficult to transmit video at more than five or six frames per second. If operating in a radio-congested location, using 1xEVDO Cellular, or using an adhoc mesh network [Ari05] in regions where no infrastructure can be assumed, the available bandwidth drops further and it may be difficult to achieve more than one or two frames per second. More problematic than the low frame rates, however are the lost or delayed network packets that create ugly artifacts in the video streams. To accommodate these problems and to create a new asynchronous communication paradigm for video, we elected to intentionally drop the frame rates from our cameras to one frame per second, but to guarantee the delivery of every frame by using a deliver-when-possible TCP based networking strategy. When network conditions are poor, images may be delivered every three or four seconds (or perhaps not at all if conditions are really poor), but when conditions improve, the baseline one frame per second can be achieved with enough bandwidth left over to also allow for the delivery of the older images that have been stored on the client device. The older images are interleaved during transmission, but not during viewing. They are sent straight to the archival store in RealityFlythrough so that the user's perception of streaming video is not affected.

The problem with low frame rate video is that as the frame rate drops, the video feed looks less like video and more like a sequence of still images. Worse, the sequence of images is difficult to watch without getting disoriented during intervals of high camera pan. Since the cameras used with RealityFlythrough are attached to the camera operators' heads, this first-person-video is full of disorienting camera pans. To

restore some of the sensibility to low frame rate first-person-video we created a new camera abstraction called the *Smart Camera* which augments a traditional video feed with RealityFlythrough-style transitions whenever position sensors indicate that a large amount of movement has occurred between frames. A *Smart Camera* reproduces (to some approximation) the motion that the camera may have taken between frames. If appropriate archived imagery exists, the archived imagery can be used to fill in the spatial gaps just as we do when moving *between* cameras in the traditional RealityFlythrough transitions.

The key insight, from an architectural standpoint, is that the same RealityFlythrough architecture that was designed to support inter-camera transitions was able to adapt to be able to also support inter-frame transitions. All of the same *Transition Planner* logic that is used to create the best sequence of images to display during a transition from one camera to another is now used to create the best sequence of images to display during a transition from one frame to another. The way that the *Smart Camera* fits into the existing architecture is quite elegant. The *Smart Camera* is just another *camera* and as such, it is possible to do transitions from one *Smart Camera* to another. The entire RealityFlythrough engine is being logically replicated for each *Smart Camera* so that we effectively have RealityFlythroughs within RealityFlythroughs.

Chapter 5 discusses the *Smart Camera* in more detail and also details a user study that indicates that RealityFlythrough-augmented-video may actually be preferred over traditional first-person-video in some situations – even when the first-person-video is presented at high frame rates. The calming nature of smoothly transitioning one frame per second video contrasts starkly with the jumpy, nauseating video that normally comes from head-mounted cameras.

1.4.5 Using Point-Matching to Augment Position Sensors

One problem with the *Smart Camera* was that while it helped users make sense of consecutive frames that did not overlap, the sometimes poor alignment of overlapping frames (a result of sensor error), while still sensible, was not aesthetically pleasing.

Figure 1.4: Snapshots of a point-matched transition. This is a transition between the same two images as was shown in figure 1.2 except that this one uses point-matching to improve image placement.

Since we had already dropped the frame rate to one frame per second in order to handle bandwidth limitations, we had a one second window in which to run a point-matching algorithm to help with the alignment of the images used during the transitions. We were able to get an implementation of the SIFT point-matching algorithm to run in the allotted time by decimating the images to a quarter of their original size [Low04]. Each pair of consecutive frames is point-matched on the client devices, and the matching points are transmitted to the server along with the image data. We then used this same technique on the server to augment inter-camera transitions as well. The ideal would be to do an all-pairs point-match on all of the images on the server, but the processing time required to do this would prevent us from providing the semi-live, real-time view of a real-world space that prompted the creation of RealityFlythrough in the first place. Instead, we recursively run the point-matching algorithm on likely targets from the user's current position. The longer the user waits at any location, the better chance he or she has that the next transition will use point-matching and thus be more aesthetically pleasing.

We are indebted to David Lowe for his scale and orientation-invariant point-matching algorithm (SIFT) that finds point matches despite sometimes complex motion around an object. It is the alignment of the point-matched images and the sometimes rocky marriage of point-matched positioning with sensor-based positioning that is the

contribution of this dissertation.

The integration of point-matching into the RealityFlythrough architecture is once again rather elegant due to the *camera* abstraction. A *Composite Camera* is simply a *camera* that has a wider field of view and is made up of a panorama of point-matched images. The *Composite Camera* is given a position and orientation in space that matches the position and orientation of the last image that was added to the panorama (most likely the current image being viewed). Because a *Composite Camera* is just a *camera* (with a position and orientation) it is possible to move from a *Composite Camera* to any other camera using the traditional RealityFlythrough transitions. This kind of transition may be necessary if neither the destination image nor the next image selected for the transition has any matching points with any of the images that are already members of the composite.

The other central architectural component, the *Transition Planner*, is also heavily involved with making this point-matching illusion work. For one thing, point-matched transitions are desirable and so should be selected for whenever possible. The fitness logic that is applied to each image that is being considered during transitions weighs point-matched candidates more highly. In addition, when transitioning to a point-matched image, a *Composite Camera* either needs to be created or, if one already exists, the new image needs to be added to it. The standard transition that typically combines motion and blending to show the spatial relationships between images is augmented to also include a morph that further illuminates these relationships. Even though we are doing a morph, however, and even though all of the images that are in the panorama are contained in a single *Composite Camera*, the mechanics behind the transition are the same. Each image inside the composite is really just a *camera*, and the transition occurs by moving the user's point of view from one of these cameras to another. It just so happens that while the viewpoint is moving, the images that are displayed by these *cameras* are each warped over time in a way that creates the illusion of a morph.

1.4.6 Walking Through Hallways

The system as presented so far functions well in certain conditions, but it falters when sensor inaccuracies are too great or when the layout of an environment limits the wide fields of view that help both the user and the point-matching algorithm make sense of the images. Indoor scenes proved to be particularly troublesome especially with buildings that have many small rooms connected by hallways. Rarely is it desirable to move through a building "as the crow flies", but that was exactly what RealityFlythrough did on every transition (see fig. 1.5). Short transitions worked fine, but anytime the user tried to move from one room to another or move from one side of the building to another, the transitions would rarely contain filler images because photos of these inaccessible spaces simply did not exist. On rare occasions, filler images were available, but the resulting transitions were still difficult to understand. The paths were unnatural and thus unfamiliar, and the walls that usually form barriers to us physical beings, also served as barriers to *closure* and point-matching. With no image overlap, and thus no shared objects between images, it is very difficult to register how the images relate to one another.

Indoor scenes were not the only problem, however. Errors in the sensor data can also cause problems, especially when the readings are temporally distant. It only takes one falsely registered image in a transition sequence to confuse the user. GPS errors are rarely realized as sudden large jumps. Instead, the readings tend to drift slowly around within a certain error range of the devices true position. The relative positions between temporal neighbors tend to be much more accurate than those of readings that are temporally distant. The readings between multiple devices tend to differ quite a bit, as well.

These two problems seem to be unrelated, but the solution is the same, and fortunately quite simple. The idea is to have the transitions meander along the same paths that the camera operators walked. The users are basically hitchhiking on one or more historic video feeds until they reach their destination or intersect with another feed that will take them closer to their destination (see fig. 1.6). It is very much like the way

subway passengers jump from line to line as they transit through a city.

Not only does this constrain the paths to hallways, but it also naturally selects for temporally and spatially neighboring images making the bulk of the transitions very sensible and also quite aesthetically pleasing especially since many of the immediate temporal neighbors have been point-matched. Chapter 6 has further details about how this logic was implemented.

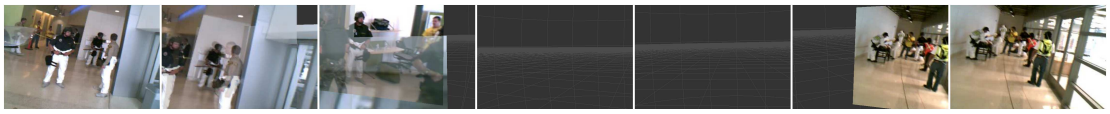


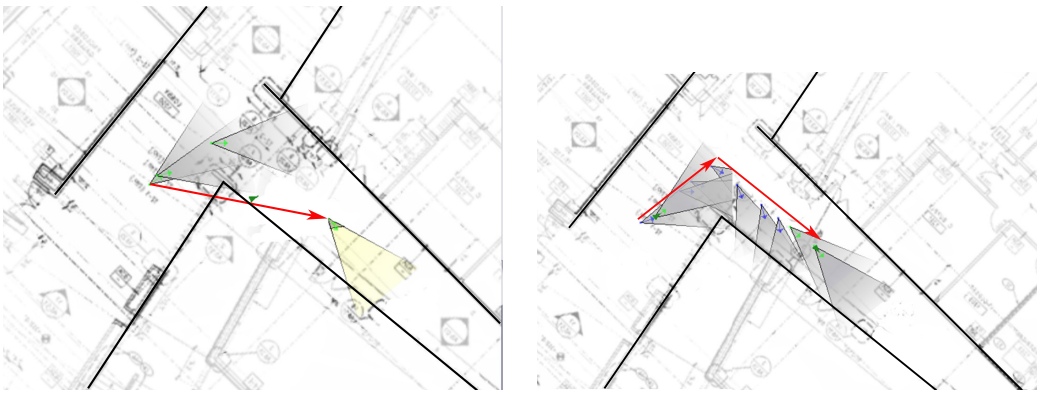
Figure 1.5: This figure shows a transition that takes a less than ideal path through two walls. As figure 1.7a shows, the problem is that the user is traveling as the crow flies.

Figure 1.6: This figure shows a transition that is similar to the one shown in figure 1.5 only this time the user does not walk through walls. Figure 1.7b shows the path that was taken.

1.5 Organization of the Dissertation

This dissertation is organized into eight chapters. The next chapter, "Presence", discusses the concept of presence by exploring its manifestation in various systems, including RealityFlythrough. This chapter defines the qualities of presence that are important and provides guidelines for how a presence-based system should be developed.

This chapter is followed by a chapter entitled, "An Abstraction for Ubiquitous Video", which is the heart of the thesis. It discusses in detail how the core RealityFly-



(a) (b)

Figure 1.7: These two figures show the the paths that were taken when completing the transitions shown in figures 1.5 and 1.6. The small arrows represent the camera views that were displayed and the cones emanating from them indicate the field of view. Notice that in figure (a) the user moves to the destination as the crow flies, and in figure (b) the user walks down the hallway. If you are viewing this figure in color, you will also notice that there are green arrows and blue arrows. The colors represent different physical cameras, so this indicates that the transition show in figure 1.6 was not a consecutive set of images taken from a single camera.

through system works, and it describes the system architecture that proved to be so robust to change.

The chapter "Closure: Why the Illusion Works", discusses the concept of closure in detail and provides a better understanding of the mental processes that may be contributing to a user's comprehension of RealityFlythrough transitions. We explore both McCloud's definition of closure and cognitive film theory to arrive at this understanding.

The "Smart Camera" chapter motivates the need for a *Smart Camera* that aids comprehension of low frame-rate video, and it presents two user studies that assess the effectiveness of the *Smart Camera*. The *Smart Camera* is found to not only increase comprehension of low frame-rate video, but also to be a possible substitute for higher quality first-person video in certain situations because of its aesthetic qualities.

In "Putting it All Together", we discuss how the *Smart Camera* and *Composite Camera* concepts that were introduced in the previous chapter are integrated into the

RealityFlythrough system. We also discuss some other key components to the system that help tame the wild elements of the real world.

And we conclude by presenting four user studies, two of which investigate the basic elements of the system, one which explore how users comprehend transitions in more complicated environments, and one which explores the usage of the system as a whole. The results of these studies quite convincingly demonstrate the effectiveness of RealityFlythrough as a tool for live remote exploration.

Chapter 2

Presence

Presence (the sense of being in a place) is often perceived as the ideal for virtual reality and tele-reality researchers, but is it an ideal that we should strive to achieve or do the technologies that support mediated presence offer other benefits?

In this chapter we will explore what it means to be present and consider the various technologies that encourage mediated presence. We will drill into specific applications and determine if the goals of the projects align with the notion of presence. The user's needs and the task to be completed are what drive the requirements for an application. Is a user interface that evokes presence necessary, just beneficial, or may it even be a distraction?

2.1 Introduction

Presence is the sense of being in an environment—sensing it with full fidelity, and having the ability to control and modify elements of it [Ste95]. It can also be thought of as being both physically and temporally proximate to something (whether it be a place, a thing, or a group of individuals). We will expand on these definitions throughout the chapter, but they serve well for now. Mediated presence is the experience of presence through a medium such as the telephone, television, or the computer. Mediated presence supports telepresence (presence at a distance), artificially generated presence as created by Virtual Reality researchers, or some combination of the two. Lombard and Ditton

describe mediated presence as "the perceptual illusion of non-mediation" [LD97]. In other words, the mediated environment is perceived as if it were not mediated. The illusion is so convincing that the mediated environment appears to be real.

These definitions seem reasonable at first blush, but presence is not what it seems. Even when considering non-mediated presence, what it means to be present is not clear [Hee03]. How much consciousness is required to be present? We are all guilty of daydreaming at times when we were supposed to be present. This is often described as being lost in thought. The use of the word "lost" is used in other expressions as well (lost in a book, lost in music), and indicates that we perceive these mind states as places; places in which we can be present. If we are physically and temporally proximate to reality, but mentally distant, are we really present? If not, how often are we truly present? We constantly rehash the past and plan for the future. Are we really only present when we are "living in the moment" and consciously aware of the present? Do we have to be paying attention to appreciate presence?

The convergence of multiple presences and the resulting split-attention complicates things further. The convergence is nothing new, but the increasing intrusion of the cell phone into public spaces has made it more apparent. A person communicating on a cell phone while walking down the street is present in two worlds. Many would argue that the mediated world dominates, and the reported increase in muggings because of cell phone use supports this claim [Ros]. It is interesting that the telephone, which mediates only a single sense at a fairly low fidelity, is able to at times invoke a larger sense of presence in people than the real world with all of its sensory glory.

That a low fidelity telephone can invoke so much presence hints at the conclusion that Jim Hollan and Scott Stornetta draw in "Beyond Being There" [HS92]. Their work focuses on communication, but can be extended to all forms of mediated presence. Face to face communication, they argue, should not be the standard by which all tele-conferencing systems are judged. Face to face communication should be viewed instead as using just another medium (physically proximate reality), a medium which has advantages and disadvantages just like any other. For example, some disadvantages

of face to face communication are its reliance on proximity, its lack of anonymity, and its lack of archivability. The simple telephone trumps face to face communication in all of these areas. This is not to say that the telephone is better than face to face communication; it is just different; it has different strengths and weaknesses. Extending the *Beyond Being There* argument to all forms of mediated presence suggests that presence should not be the goal nor the benchmark. Applications should embrace the affordances of the medium, and ultimately, they should be judged by how effectively they facilitate the tasks they were designed to support. The quantity and quality of presence should be commensurate with the task being attempted.

In this chapter, we will explore the concepts of presence and mediated presence to determine what they are and to discover the advantages and disadvantages they offer. We identify four applications that cover four broad areas of research, and determine the degree to which presence is used, the reason presence is used, reasons why presence may not be the best solution, and the *Beyond Being There* qualities of the medium that can be exploited. The Virtual Window is a communications solution, Visual Eyes is an example of Virtual Reality, the Space Browser is an example of telepresence, and Augmented Virtual Environments is an example of tele-reality. We will close by investigating how RealityFlythrough relates to these other presence-based systems, and evaluating RealityFlythrough as a tool for mediating presence.

2.2 Presence

In this section we will explore what it means to be present and determine the properties of presence that should and should not be transferred to mediated presence. We also identify *Beyond Being There* qualities that may be exploited.

2.2.1 Properties of Presence

Presence is Personal

Jonathan Steuer defines presence as the sense of being in an environment [Ste95]. Presence is not the environment itself, but a person's sensing of the environment using the five main senses: sight, hearing, touch, smell, and taste. The sensing of an environment is a personal experience, so presence cannot be separated from the individual. Every individual senses the world in a different way, and has different requirements for what the environment should present. The reproduction of some sensory input may be important to some, distracting to others, and literally nauseating to a few. Even if all sensory input could be duplicated with exact fidelity, it may still be desirable to adjust the input to match the needs of the user to maximize performance on a certain task. We often attempt to deaden our senses and thus reduce the impact of the environment. This may be done to increase attention on a task or a particular sense, or to reduce the discomfort caused by stimuli. For example, we wear headphones to drown out distracting background noise, we wear sunglasses to reduce the intensity of visual imagery, and we plug our noses when encountering an offensive smell. Each individual has a different tolerance for "noise"; a mechanism for tuning the quantity and quality of sensory input should be considered in any application attempting to mediate presence.

Consciousness

The example described earlier of a cell phone user having split attention between the mediated and non-mediated environments suggests that attention plays a role in presence. Sensory input is not only perceived by consciousness, though. Regions of the brain that operate independent of consciousness are always monitoring the senses and making judgments based on the input [Joh04]. A region of the brain called the amygdala, for example, is continually observing the eyes of people and making instant judgments about their mental state. A slight twitch in the muscles around someone's eyes can trigger a "this guy is lying" alert. The twitch likely goes unnoticed, but a feel-

ing of mistrust sweeps through consciousness. This suggests that sensory data may be important even if users are not aware that they are using it. At a minimum, this example indicates that the addition of video of sufficient quality to reveal subtle eye movements would support at least some of the nonverbal dialogue that is typical of face to face meetings.

Missing or incongruent sensory stimuli may draw attention away from a task, so even if there is evidence that a user is not using the extra sensory data, providing it may still be of benefit. For example, when opening a filing cabinet in virtual space, a user may think, "huh, that's strange that it didn't make a noise." The sound is probably irrelevant, but the absence of the sound can be distracting. Of course there is danger that the converse situation can occur; the user may marvel that the sound effects are so good that even the filing cabinet is accurately modeled.

A typical task requires focused attention, and it is the responsibility of the application to provide the focus. Distractions caused by either the absence or presence of sensory input should be avoided. It may be necessary to continually adjust the level of presence to keep the user engaged [Hee03]. This technique is used in movies to make clear distinctions between the highs and lows. Full sensory engagement during a movie would inure the senses to the presence invoking stimuli; anomalies stand out much more than expected and repetitive sensory input. Too much stimulation can also distract the user from interpreting and analyzing the input. Sensory down time is usually required to reflect on what has been experienced. The slower (less stimulating) parts of a movie allow time for reflection. To provide similar variations of presence in a mediated environment, the application needs to incite states of non-presence and not states of real presence. Simply turning off the mediated sensory input may re-engage the user in the real world. A user can always close her eyes or stare off into the distance to invoke a moment of reflection, but the system may need to be complicit in deadening other sensory input.

2.2.2 Dimensions of Mediated Presence

Sensory Breadth

Sensory breadth refers to the quantity of the senses that are involved in the experience. Breadth often provides redundant information to the various senses which serves to reinforce the interpretation of an experience. The redundancy also acts as insurance against lost information; if one sense misses an important stimulus, another may be able to detect it. For example, the vision system may be fooled by camouflage, but a cracking twig or unusual smell may reveal the location of a hidden subject. Once cued to look for something unusual, the camouflage is easily revealed.

The reinforcing and redundancy qualities of sensory breadth suggest that mediated presence systems should strive for sensory coverage. The individual user preferences cited earlier, though, should be weighed in, as should the interaction between the mediated world and the real world. For each sense that becomes dedicated to the mediated world, a sense is lost (or impaired) in the real world. This may not be an issue if the task requires complete immersion in mediated space, but there are probably more examples where dual citizenship is a requirement. Consider again the case of the telephone. Full sensory immersion in the phone call would remove some of the benefits of the telephone. Anonymity, privacy, and multi-tasking behavior that is typical of phone users may be compromised, and access to real-world artifacts such as notes, filing cabinets, paper and even people would be lost. Clearly some kind of dual presence is necessary. The equipment required to support greater sensory breadth may make it difficult to act in the real world. What is involved with "jacking in" to a mediated space? How long does it take to engage and disengage in the mediated environment?

Sensory Depth

Sensory depth refers to the fidelity of a particular sensory input. What makes up sensory depth varies from sense to sense. Unfamiliar with *Beyond Being There*, Steuer sees the full fidelity sensory input of reality as the ideal; after all, bandwidth

is not usually seen as a limiting factor in human perception. Faced with bandwidth and processing limitations, mediated presence is forced to limit the quantity of sensory depth.

Lombard and Ditton identify some characteristics of displays where tradeoffs are made [LD97]: The perceived image quality depends on the number of pixels and colors used to represent an image, on the quality of the optics used to capture the image, and on the degree of quantization and lossy compression used on the captured images (if digitized). To a point, the higher the resolution and the higher the color depth, the more the visual stimuli will mimic reality. The size of the images and the viewing distance also impact the amount of presence that is experienced. This has to do with how much of the visual field is consumed by the mediated experience, thus obstructing the competing real environment. The dimensionality of the rendered scene also has a large impact. Humans use stereo vision and parallax to infer three dimensionality from the real world. If the mediated world can generate output that supports the same 3d inference mechanism, the experience will appear to be more real to the user. And finally, camera techniques can generate a feeling of immersion. Having the display bob slightly when moving through a scene provides the sensation of walking.

The sound produced in a mediated environment is also subject to variability ranging from mono to 5.1 or greater surround sound, and from coarse quantization that clips the lowest and highest frequencies to finer quantization that gives a truer representation of the dynamic range. Sound is extremely important for conveying a sense of presence. One of the lessons learned by Frederick Brooks after reviewing the state of the art in virtual reality was how important sound is in reproducing the overall environment [FPB99]. He cites examples where the only VR environment is aural. Anyone who has experienced a good surround sound system in a home theater would concur with this observation. The crisp and immersive sound alone can draw you into the action, regardless of the size of the display.

There are a number of *Beyond Being There* qualities associated with each sense that can increase the sensory depth beyond what is available in the real world.

With vision, why stop at 20/20? Crisp detail at various zoom levels can give the user incredibly acute vision. Augmented reality type approaches can highlight important objects speeding recognition. For example, a book on a bookshelf could be highlighted [RS01], or if the target book was not known ahead of time, the contrast of the display could be modified to enhance the readability of the titles making search faster. Lighting conditions can be modified to enhance the user's vision. Brightness and contrast can be adjusted to either make a scene more pleasing or to draw out details that might otherwise be lost. Vision can also be augmented by allowing sight to occur at frequencies above and below the visible spectrum. And finally, historical events or perhaps future predictions can be overlayed on the current display allowing a user to see into the past and future.

The sense of sound is equally ripe for *Beyond Being There* enhancements. Sound can be amplified or it can be sent through a wide range of digital signal processing filters to enhance or distort it. Sounds can be masked to support anonymity. Sounds can be targeted to a particular person so that even in the presence of loud ambient noise, normal conversations can be supported.

There are numerous possibilities and they do not necessarily require bandwidth above and beyond that which is required to mimic the real environment. In many cases more targeted sensory data consumes less bandwidth. It is not the case that we need to first get to "being there" before we can go to "beyond being there".

Control

Merely sensing the environment is not sufficient for achieving presence. A person must also have some control over it—to be able to move around in it and to manipulate it. Lombard and Ditton identify some dimensions along which interactivity can vary across systems: (1) the number of inputs from the user to which the system responds, (2) the number of characteristics that can be modified by the user, (3) the amount of modification that is allowed, (4) the mapping between the user action and the system response (i.e. moving head changes view vs. using the keyboard to change

views), and (5) the response speed to user inputs [LD97].

Control of the environment highlights both the greatest strengths and greatest challenges of mediated presence. Humans have evolved to become very adept at moving through the real world, and it will be difficult to replicate this agility in mediated space. On the other hand, humans are also grounded by gravity and other physical forces that may be nonexistent in mediated space. People may be able to walk through walls, fly, travel at incredible speeds, or move so slowly and with such precision that they can explore microscopic space.

As *Beyond Being There* components get added to mediated systems, the mapping between user actions and system response become less clear. How do people fly, for example? A common refrain is that controls should mimic what is natural, but perhaps "unnatural", yet learned, behavior is what is preferred. The keyboard, the mouse, and the Nintendo style thumb controllers are fast becoming the interface for navigating through virtual spaces in game environments. A whole generation has grown up using these (what they might consider natural) interfaces. Perhaps walking is not the most natural means of locomotion in mediated space.

Support for manipulation of the environment varies by equipment and technology. Artificial worlds can be manipulated in whatever way the designer allows. Telepresence solutions (Section 2.5) are limited only by the capabilities of the robotic device, while tele-reality solutions (Section 2.6) usually offer no support for manipulation.

Content

The content of the environment affects the degree of presence experienced. Environments based on reality have a better chance of being believable. Lombard and Ditton note that live environments may be even better. They note that people react differently to television news broadcasts depending on whether they are live or prerecorded. Live broadcasts of any kind feel more exciting. Sports broadcasts, in particular, are unscripted and spontaneous; no one knows what the outcome may be. Despite the prevalance of PVRs (personal video recorders), many people still watch certain live

broadcasts live with everyone else. Hearing the cheers or moans erupting from neighboring apartments conveys a sense of common humanity. We are all in this together. This sentiment is echoed by the suggestion that interacting with people in a mediated space can also effect presence [LD97]. The telephone is so effective at conveying presence because it supports interaction with people whom we care about.

The realness of the content is not essential for conveying presence, but without it the user has to be more willing to "suspend disbelief" [LD97]. People are often engaged by fantasy tales, but the author of the tale has to convince the reader to come along for the ride, and the reader must be a willing participant. The more real the story, the more real the content in a mediated experience, the less the participant has to be willing to suspend disbelief. The more relevant the content is to the user, the more real the characters and events will feel. Reality is usually pretty relevant, but other stories, events, imagery, artwork, or lighting can resonate with certain individuals.

The imagination is extremely powerful; it can be stimulated inexpensively, it has unlimited bandwidth, and it can engage all of the senses. The more a mediated environment captures the imagination, the less it will have to do and the more effective it will be at conveying presence. Less may indeed be more [Hee03].

2.3 The Virtual Window

The first mediated presence application we will study is an example of a media space—a form of always-connected video conferencing. A media space is a window into another location that serves to support an informal peripheral awareness. The window can of course be used for collaboration, as well. The requirements for such a system are that it provide a natural view into a remote space without being obtrusive. It must remain on the periphery, and it cannot distract from the primary tasks.

Gaver, et al. have attempted to make media spaces behave more like windows than monitors by adjusting the view based on the position of the user's head [GSO95]. This is a very natural interface for camera panning. The Virtual Window, as they call

it, does not just do camera panning, however. Since the camera can also slide from side to side, it allows the view to pivot around a focal point some distance away from the camera.

Successes

The authors identify a number of affordances of this medium. (1) Movement parallax is supported giving the viewer a good sense of depth, even though the scene is viewed on a two-dimensional screen. No special glasses or equipment needs to be carried by the user to achieve this effect. (2) The viewer has an increased field of view since the camera can move and pan. (3) The increased field of view does not come at the expense of resolution. (4) The view of the remote scene is continuous since the camera moves smoothly. This contrasts with other systems that jump between different camera views. (5) There is continuity between the local and remote scenes. Actions (i.e. head movements) in the local scene correlate with similar visual consequences in both the local and remote scene. The spaces share the same physics. And, (6) the local control of the remote camera promotes active exploration and engagement in the remote environment.

The most striking contribution of this paper from a presence standpoint is the elegant use of the window metaphor to seamlessly blend a mediated and non-mediated space. The window metaphor captures not only the physics of the space but also the psychological detachment. A window provides a narrow view of another physical space that can be broadened by looking through the window from different angles. The window pane serves as a barrier to other senses. Sound is muffled, smells usually cannot penetrate, and touch is blocked by the glass. The Virtual Window mimics these attributes almost exactly, and thus succeeds at presenting a very natural user interface.

A key feature of the Virtual Window's blending of mediated and non-mediated space is the clean separation between the two environments. No special equipment needs to be worn by the user to engage the mediated space, and as a result nothing from the mediated experience encroaches on the real world. Assuming the sound is turned off and

the user is not looking at the window, there are no lingering effects of the mediated space. Contrast this with other possible solutions which might require the user to wear goggles to see into the mediated environment and/or sensors to facilitate tracking. Using such artifacts, the user has to make a conscious effort to engage or disengage the mediated space.

The Virtual Window interface is very suited for collaboration across distance where the subtle interactions between participants that the system supports are most appreciated. The authors describe an interaction between collaborators where one holds up a piece of paper for the other to see, and the viewpoint has to be shifted a little in order to get a good look at the document. With fixed cameras, this subtle and very natural interaction would be much more complex: "I can't see it, can you move it a little to the left. No, your left. Too far." By eliminating this source of frustration along with many others, this medium can go a long way towards making the participants feel co-present.

Drawbacks

The increase in presence achieved by the Virtual Window may actually reduce its utility as a peripheral awareness display. What people want to see may not always be visible in the window. For example, a viewer sitting at her desk may have something specific that she wants to observe in the remote scene. If the desk is not positioned correctly, and the natural gaze direction does not cover the desired scene, the viewer would have to reorganize the office to correct this. This problem is easily corrected by allowing the user to manually control the view. What this situation reveals about presence is much more interesting, though. A natural presence-oriented user interface can be very successful in one context, but inadequate in another.

The Virtual Window has some limitations even when used in a collaborative environment. Only one user can control the viewpoint, so the Virtual Window is only a natural interface for that one participant. If other users are present, they will have to reconcile the arbitrarily shifting camera angles. Deficiencies in the head tracking algo-

rithms limit the naturalness for the primary user as well, though. Eye gaze direction is not tracked, and neither is the tilt angle of the head. Without this information, the user's focal point cannot be determined automatically, and must therefore be set manually. This problem could potentially be resolved with a better tracking mechanism, but care would need to be taken to make sure that the clean separation between the mediated and non-mediated environments is maintained. These limitations are ones that plague even the most sophisticated systems. We will see shortly that the CAVE displays that represent the state of the art in Virtual Reality have similar shortcomings.

Beyond Being There

There are a number of *Beyond Being There* elements of this medium that can be exploited. The window can be virtually shaded to support privacy. While the simple light filter of a standard window shade may provide some benefit, the digital environment supports more sophisticated shades that can control the quality and quantity of information that is transmitted. The shade can reveal general information about the state of a person, while concealing detailed information that may be unimportant. For example, the shade may reveal that a person is in the office and talking on the telephone, but the details of the phone call that might be inferred from body posture and other clues would be concealed. This does more than protect privacy; it may also reduce the discomfort associated with knowing that someone may be watching you.

Archiving the Virtual Window content would be valuable. This has obvious benefit for the person doing the observing, but it is also useful for the observed. It introduces deterrents to prevent an observer from abusing the system. *I know when you are watching me, so you better be watching me for a reason*. Unfortunately it is not possible to archive the Virtual Window in its entirety. A particular view through the window is recordable, but there is no way that a user viewing a replay of the stream could benefit from the affordances of the system. The field of view would be narrow and fixed. A high resolution panoramic camera can solve the replayability problem, although substantially more bandwidth would be required to transmit the much larger

stream.

Reflections

The Virtual Window is a well focused technology solution that enables a very useful form of mediated presence. Sensory breadth and control of the environment is somewhat limited, but the depth of the visual sense is high. This depth is achieved in spite of the very natural physical division between the mediated and real worlds that allows a participant to engage fully in both environments.

The Virtual Window's deficiencies when used as a peripheral display support the claim that presence-based user interfaces may not always be the best solutions. There are certain situations where it is better for the Virtual Window to not mimic the physics of a window. One of the benefits of the Virtual Window medium is that the physics can be modified depending on the task being supported.

2.4 Visual Eyes

The next mediated presence application that we will examine, Visual Eyes [LJDB99], represents the state of the art in Virtual Reality (VR). The term Virtual Reality has been used to describe everything from completely immersive computer-generated virtual environments to text-only "adventure"-style computer games [MTUK]. Jonathan Steuer defines VR "as a real or simulated environment in which a perceiver experiences telepresence" [Ste95]—a catch-all definition that subsumes all of the categories of presence that we are examining in this paper. A more typical definition of VR involves the notion of immersion and is thus tied to the equipment that is used to generate the experience: "goggles 'n' gloves" definitions, as Steuer describes them. The definition we use is the one suggested by Milgram et al: Virtual Reality is one pole on a Reality-Virtuality continuum [MTUK]. It is the "completely virtual environment" which contains only computer-generated virtual objects.

The key feature of VR is that the environment is synthesized. The purpose

of using synthesized imagery is to (1) allow for visualization of things that do not exist, and (2) allow the environments to be interactive. There are a number of situations where people may want to view things that do not yet exist (buildings before they are constructed, for example), but there are also many situations where VR is used simply because it is the only way to allow for interactivity in a remote environment. As we will see in Section 2.5, tele-presence has a number of limitations, and in some cases Virtual Reality can produce an increased sense of presence with synthetic imagery by allowing more interactivity. For example, a realistic architectural flythrough of an already constructed building may be more suitable than a movie-based flythrough of the real building. This tradeoff is certainly desirable with video games where the playability of the game is often tied to the amount of interactivity allowed.

2.4.1 Goggles'n'Gloves

The "goggles 'n' gloves" variety of VR immerses users in the virtual environment. Rather than viewing the environment through a screen or a virtual window, users are actually drawn into it. The video and audio wraps around them completely replacing real stimuli with those of the virtual world. Special gloves may allow for manipulation of virtual objects and some even provide haptic feedback.

Tracking of the user's position and gaze direction is crucial in VR environments because the imagery needs to be rendered using the correct perspective. The requirement for tracking limits where VR can be performed because tracking equipment usually requires infrastructure installation. Less accurate tracking allows for mobile VR [TP03], but it impacts the realism of the experience because the imagery may not always be displayed in the correct perspective. Not only does the tracking need to be accurate, but it must also be fast. Lag between the user's movement and the re-rendering of the display is manifested as a disconcerting "swimming" effect that interferes with the illusion of immersiveness.

These conditions dictate the kinds of applications that VR can support. The experience is completely immersive, leaving little room for dual existence in reality. To

do VR a user must go to a special room or at a minimum don special equipment. There is little chance of working with real artifacts at the same time unless they are carried into the room, but even then the artifacts may not tie into the virtual world without harming the illusion of the virtual reality. These limitations do not cripple VR; they simply dictate the kinds of applications that this medium can support.

2.4.2 CAVE's

A CAVE [CNSD93] represents the state of the art in VR. It is a square room that has stereo video projected onto six of its surfaces including the floor. Speakers located in each corner of the room immerse the user in sound. The primary advantage of a CAVE over systems that use head mounted displays is that multiple people can be in the environment at the same time allowing a natural mix of reality and virtuality in the same space. Only one viewer controls the viewpoint, though, so as noted in Section 2.3 some elements of presence are lost for the other participants. The viewers need only wear lightweight shutter glasses—which contribute to the stereoscopic effect—and carry a wand for 3d interaction with the environment. Communication and interaction with other local participants is therefore relatively natural.

CAVE-based VR has inspired research in *teleimmersive* environments which interconnect multiple remote CAVE's [LJDB99]. Multiple sites can participate in the same virtual environment with remote users represented as avatars locally. The virtual environment is designed to persist, becoming a permanent location that participants can travel to. The driving force behind teleimmersion is collaboration among teams residing on different continents who need to share a visual experience. A typical example is the global workforce of an automobile manufacturer collaborating on the design of a new vehicle. The time-differences between locales suggests the need for a persistent environment that can be visited at convenient times for all participants.

2.4.3 Visual Eyes

Visual Eyes is a teleimmersive application developed by General Motors Research and Development Center to facilitate design reviews of CAD models. The CAD models are imported into the CAVE environment and are rendered full-size and in 3d.

Successes

Viewers can walk around the rendered object in a natural way, and can even modify the object by making verbal requests to a remote user running a 3d modeling program. Local viewers have a truly enhanced collaboration experience because they are able to interact with one another while visualizing and modifying a new design. Remote viewers get the same collaborative experience with other local viewers at their site. It is only cross-site collaboration that must be mediated, but the primary advantage of teleimmersion over other communication solutions is that the central focus of the event is a virtual object or scene that is present and modifiable at both locations. The virtual world is an artifact that is shared between the two (or more) spaces acting as another communication device. Words may not be necessary because the world being experienced is nearly the same. It is similar to (but obviously much better than) a telephone conversation involving two parties who are watching the same TV channel; the conversation shifts to one of shared experience.

It is difficult to imagine a better solution for viewing CAD models. Improvements to the system come down to a matter of degree. Clearly, better realism would be better. The more natural a car or car part looks, for example, the better reviewers will be able to judge the aesthetics or perhaps even the performance of it. Adding accurate tactile sensations to the object would contribute even more to the realism. Being able to use the object would be better, still. Sitting in the seat, feeling the position of the lumbar support, and actually driving the vehicle would lead to better design reviews. All of these enhancements are simply adding more realism; adding more presence. The CAVE is the best option we have for bringing life to virtual objects, so it appears VisualEyes is taking the correct approach.

Drawbacks

The VisualEyes approach is not perfect, however. CAVEs are extremely expensive, costing on the order of 100s of thousands of dollars. Is the use of a CAVE perhaps overkill? How much does a full size rendering of a CAD model actually add to the experience? Stereoscopic rendering is probably important, but perhaps a comparable experience could be achieved by using a simple projector in a standard conference room. Do people really need to physically walk around an object to get an appreciation for it? It may be just as easy to maneuver through the environment using an alternate input device since movements like flying do not map naturally to standard human actions.

There are times when having a consistent view among collaborators is desirable, but individuals may prefer to explore the object on their own and only collaborate when necessary. This suggests that multiple individual screens with a mechanism for synching them all to a particular view when collaboration is needed may be better. One result of simplifying the application to run on individual displays is that it allows the system to work from offices, from home, or perhaps even on the move. If designed appropriately, the user would have access to both the real and virtual worlds simultaneously, giving them full use of real world artifacts while engaged in the design review. The CAVE would have to dramatically increase the usefulness of the application to warrant the extra effort required to use it.

Beyond Being There

Jason Leigh, et al. have made a start at identifying some *Beyond Being There* elements of teleimmersion [LJDB99]. They note that teleimmersion may be so effective that "collaborators may choose to work virtually even if more traditional face-to-face meetings are possible." The primary reason for this is the medium's ability to display objects that otherwise do not exist. This is clearly better than reality.

Looking specifically at VisualEyes, there are several *Beyond Being There* attributes of the system that jump out. We already alluded to the movement possibilities when discussing the limitations of the "walking around" interface. Flying around the

object and viewing it from *any* angle is a clear benefit of VR. Users should be able to view the inner workings of an object, as well. While viewing a model of a car, for example, it should be possible to go through the hood and watch the engine as it runs, removing visual obstructions like hoses without altering the "behavior" of the engine. Engine components that might otherwise not be visible could be explored, as well. Users could walk around inside the combustion chamber while pistons crash down. This is a perfect example of where it is not always desirable to mimic all elements of presence; the tactile and aural senses are best left muted when exploring the inside of an engine.

Reflections

CAVE-based VR solutions provide users with a strong degree of immersion. There is little doubt that being surrounded by sensory stimuli induces a sense of presence in the virtual environment. What is less clear is if the increased sense of presence helps users perform a task. One study concludes that immersion can help with certain tasks [PPW97]; the natural and practiced behavior of looking around by rotating and tilting the head appears to be more efficient than alternate user interfaces. This is a reasonable claim. What needs to be weighed, however, is how much additional benefit this interface provides given the consuming nature of it. VR solutions generally do not play well with the complexities of the real world. Unlike with the Virtual Window seen in section 2.3, a decision usually needs to be made whether to engage the virtual world or the real world. Living in both is usually not an option.

2.5 Space Browser

The Space Browser [PC97] is an example of the telepresence approach to exploration of a remote space. Telepresence involves the remote control of a robotic device that can move through the environment. The robot becomes an extension of the operator, transmitting sensations acquired from the environment. Tele-reality, another approach to remote exploration, will be discussed in the following section.

Telepresence takes many forms depending on the kind of application that is being supported. Robotic vehicles may be used to explore planets or hazardous volcanic terrain. The Mars Rovers are recent examples of this kind of telepresence. The use of telepresence to operate remote heavy machinery has also been investigated [Tac98b, SH01]; one benefit being that a single operator can control more than one machine simultaneously.

It is a blimp-based form of telepresence, the Space Browser, that we will examine in detail in this section, however. Eric Paulos and John Canny attached a camera, a microphone, and a speaker to a person-sized blimp creating what was in affect a real-life avatar [PC97].

Successes

An operator can control the movement of the blimp, navigating it through most walkable areas including up and down stairs. A blimp has the advantage, though, that it can also fly high into the air to look down on the scene below. The blimp is light and soft and moves slowly enough that it does not pose a danger to others. Even an out of control blimp can do little harm. As a result, the blimp can operate in crowded spaces allowing it to behave just like any other person. The large stature of the blimp gives it an obvious physical presence in the environment, making it easier for the blimp to engage in conversations, and reducing the discomfort nearby people may have about an invasion of privacy.

The key success of the blimp is that it acts as an avatar for a person in the real environment. The blimp is a physical presence in the environment; it is something that people can interact with and identify as a stand-in for a person. People may actually think of it as a person wearing a blimp costume. Having a presence in the environment is a key feature of telepresence, and distinguishes it from other solutions where the operator is just a passive viewer of the environment.

Drawbacks

The blimp has some disadvantages, however. The most serious one is that the payload is severely limited because a reasonably sized blimp can only carry 500 grams. There is little opportunity to correct the problems that we will identify because the solutions would increase the weight.

One problem with the blimp is that it cannot manipulate the environment. It cannot open doors or press elevator buttons so even its movement is restricted. This seems at first to be a major disadvantage, but it can be thought of as a handicap similar to those endured by the many disabled citizens who still engage the environment. The blimp operator simply needs to ask for help or use special blimp accessible entryways. *Would you mind getting the door for me?* Manipulation of the environment has already been identified as a desirable property of mediated presence, but perhaps more important is having an identity in the environment that others can ascribe to you. The ability to manipulate the environment is much more critical in telepresence settings where help cannot be proffered—on the surface of Mars, for example.

A more serious problem is that the blimp can only operate indoors because the motors are not strong enough to counter the effects of wind. Stronger motors would increase the payload weight requiring a much larger blimp. The authors do not even consider outdoor conditions, worrying instead about the increase in motor strength required to compensate for the drafts produced by air-conditioning vents.

Beyond Being There

The physicality of telepresence solutions places limits on the *Beyond Being There* elements of the medium. The senses can be enhanced, but movement is limited to what the robotic vehicle can provide. The blimp, for example, can only travel at walking speeds, is mainly restricted to paths accessible to people (unless it can fly over a barrier), and cannot go outside. Other robotic vehicles may have trouble negotiating stairs, may be too bulky to go indoors, and probably lack the ability to fly. Instant travel between locations is not possible; barriers cannot be crossed; and physical conditions of

the environment may have an effect on the equipment. Sand may clog gears, wind may blow a blimp off course, and extreme heat may destroy sensitive equipment.

The obvious *Beyond Being There* element of the blimp is its ability to fly. Except for the restrictions noted earlier, the blimp can travel anywhere and can look at a scene from any vantage point. Of course, the physical presence of the blimp does force it to adhere to social and propriety customs, placing some limits on mobility. A blimp observing a theater production would not be welcome on stage, for example.

Reflections

Telepresence solutions allow users to explore territories inaccessible to humans, but they can also provide a good interface for exploring populated areas. The physical presence of the robotic device in the environment can act as an avatar for the operator, leading to an engagement in the environment through interactions with the people of the environment. The realness and liveness of the setting, the engagement with the environment, and the high fidelity of video and audio stimuli (they are real) lead to a strong degree of presence. The primary disadvantage of telepresence as a mediated presence solution are the physical restrictions on mobility.

2.6 Augmented Virtual Environments

The final mediated presence solution we will explore is Augmented Virtual Environments [NYH+03b], a tele-reality approach to remote exploration of a space. The goal of tele-reality is to construct novel views (views from positions not captured by a real image) that are as realistic as possible. The views are novel because they are not real views captured by a camera; they are representations of what the scene may look like from that particular angle. There are several approaches to synthesizing novel views from existing images. The challenge with these techniques is to recover information about the geometry of the scene so that objects in the scene can be displayed in their correct proportions when viewed from different angles. An additional

challenge involves handling holes and overlapping seams that result from looking at the scene from a vantage point that is not covered by a single camera. There are three categories of image-based rendering: mosaicking (stitching multiple images into one high resolution image), interpolation from dense samples (using the intensity and direction of rays emitted from a scene to generate a new view of the scene), and geometrically-valid pixel reprojection (reprojecting images onto a geometrically accurate model of the scene) [Kan97].

We will focus our study on geometrically-valid pixel reprojection because it is the technique that holds the most promise for supporting real-time exploration of a space. Mosaicking usually works only with static imagery, and interpolation from dense samples (plenoptic modelling) requires very dense camera coverage and a tremendous amount of processing time, making plenoptic modelling useful only for exploration of static scenes.

There are two main approaches to doing geometrically-valid pixel reprojection. The key requirement is the acquisition of the geometry of the scene so that the objects in the scene can be projected in the correct perspective when observed from a novel viewing angle. The geometry can either be pre-acquired using range finders [NYH+03b], or it can be acquired from the existing images using vision techniques such as structure from motion or stereoscopic reconstruction.

Virtualized Reality [KRVS99b] is a recent example of an approach that uses stereoscopic reconstruction. The images captured from the cameras are projected back onto the 3d reconstructed model allowing the scene to be viewed from any angle. Virtualized Reality uses dynamic video as its input, but it does not work in real-time. A real-time version of such a system should be possible in the near future (if not now) as processors get faster and algorithms improve.

A more serious challenge of Virtualized Reality is that in order to do stereoscopic reconstruction the cameras must be accurately calibrated and the distances and angles between cameras must be precisely measured. The result is a fairly dense array of cameras attached to scaffolding that surrounds a small stage-like area. The kind of

action that can be captured by Virtualized Reality is quite restricted, limiting the system to laboratory use. All current vision techniques should be susceptible to the same problems because uncalibrated cameras create too large a search space for finding the correspondences between images that provide insight into the scene geometry.

Systems such as Photosynth [SSS06] can reconstruct the geometry of a scene by discovering matching points on uncalibrated images, but this is a processor-intensive task and is not currently a viable solution for live tele-reality. The applications of Photosynth are nonetheless far reaching and the current technology gives a clear vision of what the future will hold.

Because of the limitations of the vision-based approaches to tele-reality, we instead explore geometrically-valid pixel reprojection. Using a system called Augmented Virtual Environments (AVE), Neumann, et al. project mobile live video onto modeled elements of the USC campus. The mixture of virtual reality with reality has often been explored. Some systems overlay video on a 2d map of the environment [HHT01].

The model was acquired via an aircraft-borne Light Detection and Ranging system (LiDAR), and was refined semi-automatically to a building-level resolution. Camera positions are tracked using differential GPS devices accurate to within 2- to 10-cm and inertial sensors that capture tilt, roll, and yaw. With the position and orientation of the cameras known, the images captured from each camera can be back-projected onto the 3d model of the scene in real-time. Registration between the image and the model must be very accurate for the effect to be useful, and since errors in the position and orientation sensors result in misalignments, real-time vision processing techniques are used to correct these errors.

AVE is presented as a technology solution, and is not motivated by an actual application or application domain. However, when discussing why an AVE visualization is desirable, the authors compare it to other visualizing techniques which use an array of monitors or windows to present independent video streams to the user. This suggests that the authors may have had a security application in mind when designing the system. We proceed with this assumption.

Successes. The AVE system provides a very natural interface for a security application. Rather than having to make sense of a series of disjoint images, security personnel could fly around a scene as if viewing it from a helicopter. If anything looked suspicious, the user could sweep down to the ground where a full resolution image may provide additional detail. When following a suspect, the interface is again very natural. Security personnel need not be aware of what camera is covering a suspect. They can very easily follow the suspect as he or she moves through the scene transitioning from one camera's view to the next. Movement from one camera-covered scene to another is continuous and almost natural even when the intervening space is not covered by live video. This continuity gives the user a very real understanding of the relationships between the live scenes providing additional contextual information that situates the scenes.

Since the cameras in the AVE system can be mobile, head-mounted cameras attached to security guards in the field could provide additional imagery, literally shedding light on the areas devoid of security cameras.

As described, AVE is very close to a perfect presence solution. With dense camera coverage, a user could explore a remote environment live and in real-time viewing it from whatever angle desired. It degrades gracefully with sparser camera coverage, allowing users to explore a model of the environment at minimum. Unlike the blimp telepresence solution, there are no limits to where the user can travel, and there are no speed restrictions.

Drawbacks. Unfortunately, the AVE security application just described was idealized. There are some problems that limit the utility of the system. The primary problem is with the defining characteristic of AVE: the use of a pre-acquired model. Such a model can only capture static structures, and there is an upper bound on the resolution of the model because smaller objects tend to be dynamic and there is no way to know at the time of model acquisition if an object is static or dynamic. For example, the model may include a parked car that is no longer there. The projection of an object on a surface that is not consistent with the object results in gross distortions when it is

observed from a novel view. The projection of a person standing in front of the wall of a building may result in the person appearing splayed out across the ground, for example.

The authors attempt to resolve this problem by automatically detecting dynamic objects and rendering them on surfaces that are approximately the correct size and in the correct position. The problem with this technique is in the assumption that all dynamic objects are moving. There are many dynamic structures that remain static for long periods of time. Parked cars and seated people are just two examples.

The resolution of the model used in the current AVE system is not good enough to identify objects such as trees, parked cars, lamp posts, and the myriad other small static structures that typically fill our field of view when walking through an area. All of these objects would be uncomfortably projected onto the surface of the ground or a nearby wall. This condition poses an even more serious problem when multiple cameras cover the same view—a situation that is necessary to provide the seamless exploration described earlier. The authors do not discuss how images from multiple cameras are fused, but it is safe to assume that they are either blended or a dominant image is selected. In either situation there will be times when multiple cameras are contributing content to the same scene. Since there is no geometry available for small objects, they will be splayed out across different surfaces depending on the original view of the camera. Objects may be duplicated and if there are enough dynamic elements in the scene the result will be a jumbled mess. Reducing camera density ameliorates this problem, but also increases the number of shadows and raw, uncovered regions.

A further problem with AVE is that it only works in places where a model has been acquired. Model acquisition is expensive and time consuming, so many places would not be covered, and those that are would probably not be updated frequently. Indoor spaces would need to be modeled using some other technology and would be similarly plagued by the dynamic object problem cited above.

In contrast with the telepresence blimp, AVE does not generate an avatar that people on the ground can interact with. An AVE user therefore has less ability to manipulate the environment. Camera operators could potentially act as surrogates, but the

camera positions do not necessarily relate to user positions. Also, people on the ground may be more comfortable interacting with an inanimate blimp because it more easily assumes the personality of a tele-operator. It is easier to imagine going to a movie with a friend who is traveling as a blimp than with a hired cameraman who is playing the part of your friend.

These problems do not render AVE ineffective, but they do impact the quality of the presence experience. The sparse camera coverage that is a requirement given the problems cited above reduces the degree of visual sensory depth.

Beyond Being There. Besides the obvious *Beyond Being There* qualities of instantaneous movement, flight, etc., the medium offers the opportunity to give the illusion of increased camera coverage by maintaining a history of the most recently captured images for every given camera position. This would have the effect of painting an almost live texture over everything that a camera's view covers. The system could also support travel through history allowing for exploration of the space as it was at an earlier time. This would be especially useful for the security application where the activities of a criminal recorded the night before could be seamlessly monitored across the entire campus.

Reflections. The primary issue with tele-reality solutions is the complexity of the processing required to do it well. Done well it has the potential to create very compelling mediated experiences. There would be few limits to what could be explored, and the exploration would look and feel natural. The problem is that such an experience is not yet possible. It is possible to explore real static environments with high visual fidelity and highly constrained pre-recorded environments with reasonable fidelity, but live environments are crippled by the assumptions that are made to make the computations tractable.

2.7 Presence in RealityFlythrough

Now that we have a better understanding of presence, we will re-examine the user interface of RealityFlythrough to determine if it is an effective tool for the tasks it was designed to handle. We begin first with a review of the key characteristics of RealityFlythrough, using our knowledge of the other presence solutions to clearly explain and differentiate RealityFlythrough from its related work.

RealityFlythrough combines elements of telepresence and tele-reality. It harnesses ubiquitous video to allow remote exploration of any environment that supports network connectivity. Ubiquitous video is characterized by wireless networked video cameras of all varieties located in every conceivable situation. Ubiquitous cameras are everywhere, or at a minimum can go anywhere. They are inside, outside, carried by people, attached to cars, on city streets, and in parks. Ubiquity moves cameras from the quiet simplicity of the laboratory to the harsh reality of the wild. The wild is dynamic—with people and objects constantly on the move, and with uncontrolled lighting conditions; it is uncalibrated—with the locations of objects and cameras imprecisely measured; and it is variable—with video stream quality, and location accuracy varying by equipment being used, and the quantity of video streams varying by location and wireless coverage. Static surveillance-style cameras may be available, but it is more likely that cameras will be carried by people. Mobile cameras that tilt and sway with their operators present their own unique challenges. Not only may the position of the camera be inaccurately measured, but sampling latency can lead to additional errors.

As the study of the AVE tele-reality system in Section 2.6 revealed, it is a non-trivial challenge to support live and real-time remote exploration of the world. The ideal is to have a camera lens at every possible vantage point so that a photorealistic view can be realized from anywhere. Given the pragmatic limits to ubiquity, this will not be an option in the near term. The solution, then, is to take advantage of the camera lenses that are available, and to either attempt to synthesize a novel view from the available images, or to provide a mechanism for the user's view to transition from one image to

another. As we have seen, the synthesis of photorealistic novel views in real-time is not possible with today's technology given the conditions of the wild, but it is possible to generate sensible transitions between camera feeds.

RealityFlythrough uses these transitions to convey spatial context. Transitions are a dynamic, real-time blend from the point of view of one camera to the point of view of another, and are designed to help the user generate an internal conceptual model of the space. Transitions provide a first-person immersion that is natural and comfortable. Other interfaces could be used to display the relationships between the cameras (a bird-seye map, for example), but these have the effect of cognitively removing the user from the scene. There is an inherent tension between the uncalibrated nature of the environment and the first person immersion. Because the true relationships between images are not known, the transitions can only provide a hint of how the images are related to one another. This hint is enough to allow the human visual system to piece together the relationship between the images.

Unlike the applications discussed in Section 2.6, RealityFlythrough does not use geometrically valid pixel projection to display novel views. Instead, the geometry is simplified by flattening space to the image plane, pulling in distant objects and pushing out nearby ones. Since the location and orientation of each camera is known, the image from a camera is projected onto a virtual wall (the image plane) that is some distance away. As long as the images are observed from the point of view of the cameras, this simplification produces geometrically accurate renderings. During a transition from one image to another, however, there are brief intervals where novel views are constructed. These novel views are not geometrically accurate, but there is sufficient information contained in the transition to provide the user with the necessary contextual hints to make sense of the relationships between the images. These hints primarily come in the form of movement and image overlap.

As we have done throughout this chapter, we will use a specific application of RealityFlythrough to motivate the discussion on presence. Police Special Weapons and Tactics (SWAT) teams [JH02] are routinely involved in high risk tactical situations

in which the Incident Command Post (command and control) is situated some distance from the incident site. It is the responsibility of the command post, and specifically the team commander to direct the field operations, but this activity is often done "blind", without the aid of visuals from the scene. The commander forms an internal spatial model of the scene generated from either prior knowledge, maps, or reports from the officers in the field, and must update and reference this model throughout the event. Commands must be issued to field officers from their point of view, further straining the commander's conceptual model [JH02]. RealityFlythrough can help. Head-mounted cameras attached to field officers can provide the commanders with live video feeds, and RealityFlythrough can provide a navigation mechanism that gives them the desired situational awareness.

The following are the minimal requirements of a system that supports SWAT: It must work at novel sites with minimal configuration; cameras should be mobile and therefore wireless; the system needs to handle very incomplete camera coverage with fewer than 25 cameras in the field; and the system must work in unforgiving environments with intermittent network connectivity.

2.7.1 Properties of Presence

We will now step through each of the properties of presence and mediated presence that were identified in Section 2.2, and discuss how these properties are supported in RealityFlythrough. This will serve as a summary of the key points made concrete by their application to a specific example.

Presence is Personal

We determined that presence is not the environment itself, but an individual's sensing of the environment. This makes presence a very personal experience, suggesting the need for personalization in any application that hopes to mediate presence. RealityFlythrough was designed to be highly customizable, allowing users to adjust the feel of the experience to suit their needs and their requirements for presence. This is nec-

essary in RealityFlythrough because of the variability in users' abilities to internalize and quickly recognize the information that is conveyed by transitions. Comprehending transitions appears to be a learned skill, and as such, the experience must be adjusted as users acquire more experience.

The experience also needs to be adjusted based on the current conditions imposed by the user's task. A SWAT commander, for example, will be constantly engaging and disengaging RealityFlythrough depending on the external information and tools that are available. When disengaged from RealityFlythrough, the commander may wish to treat the system as a peripheral display, viewing two or three key video streams rather than using the first-person immersive mode.

Consciousness

Part of experiencing presence is paying attention to the environment. Simply being in an environment is not sufficient. It is also important to allow attention to be split between environments if that is a requirement of the application. The nature of a SWAT commander's task suggests that motivating the commander to pay attention will not be difficult. What is critical, though, is to not divert attention from the task. The interface should be as natural as possible, and any additional information that is presented should be unobtrusive though clearly visible. RealityFlythrough presents some challenges for the creation of a natural user interface. On the one hand free-movement should be supported, but on the other, live-camera views should be favored. The user interface is still being refined. An example of an interface that is unobtrusive but natural is an age indicator bar displayed at the bottom of every video feed that reveals the age of the imagery—an important feature when connection drop-outs are common or when historical content is being viewed. The user need only be peripherally aware of the bar to acquire the necessary information.

Since the SWAT scenario requires attention to be split between multiple environments, it is essential that RealityFlythrough be a tool that can be picked up and put down as necessary. It should not impede communication with others or exclude

the use of communication devices like cell phones and walkie talkies. Engagement and disengagement with the environment needs to be instantaneous.

Sensory Breadth

Sensory breadth refers to the number of senses that are involved in the experience. RealityFlythrough's focus is on the sense of vision, but supporting the sense of sound has always been a goal for future work. Support for the other senses (smell, taste, and touch) is unlikely to provide enough additional benefit for applications like SWAT to justify the investment.

Sensory Depth

Sensory depth refers to the fidelity of each sensory input. Since RealityFlythrough displays images acquired from real cameras, the depth for the sense of vision is quite pronounced. However, the depth is limited by the tradeoffs that need to be made due to the bandwidth of the wireless network and the speed of the video compression algorithms. The user must decide whether resolution, frame rate, low latency, or image fidelity is most important. When ignoring image fidelity, the complete RealityFlythrough system was able to sustain frame rates of 6-7 fps at CIF (352x288) resolution. Because of reasons cited in chapter 5, we have chosen to emphasize image fidelity and instead transit frames at 1 fps but with a quality guarantee. Users spend most of their time viewing the action from the vantage point of a real camera and thus view unmodified high fidelity video feeds or images. It is only during the short transitions between images that geometrically inaccurate renderings of the scene occur. Enough additional information is presented during these transitions, however, that the geometry can be inferred.

Sensory depth is necessarily limited because of the conditions at a typical SWAT environment. The wireless network limits the quantity and quality of the video and audio feeds, the commander needs to be able to quickly engage and disengage the environment, and the equipment used must be mobile since the Incident Command Post is only a short distance away from the incident site. These conditions limit the effective-

ness of goggles'n'gloves type interfaces, and suggest that laptop-based solutions may be preferable.

Control

Providing users with control over a virtual environment helps convey a sense of presence. RealityFlythrough allows free movement throughout the environment, but with the caveat that imagery will only be displayed if a video feed or image covers the field of view. To increase the chance that there will be a visible image at the desired point of view, the most recent snapshots of the live video feeds are stored at regular intervals. The age indicator bar described earlier reveals how relevant the image is.

Like AVE presented in Section 2.6, RealityFlythrough has no mechanism for allowing users to directly manipulate the environment. In a SWAT scenario, though, the commander can effect change by communicating with the field officers who are wearing the cameras. A SWAT scene is an example of a setting where having a physical avatar is undesirable. Invisibly moving from camera location to camera location is a great advantage when stealth is a priority.

2.7.2 Content

The content of the environment affects the degree of presence experienced. The more believable the scenario presented, the easier it will be for a user to suspend disbelief and engage in the experience. A live experience presented through live video feeds is eminently believable. Liveness, itself, adds to the sense of attachment to an experience. It is real. It is happening now. It is unscripted.

We have observed that a good story and a user's willingness to suspend disbelief can also have a positive effect on the amount of presence experienced. There are few stories as compelling as a real-life SWAT operation where the lives of citizens and fellow officers are at risk.

There are some situations where believability is not desirable, however. Hollywood is good at using special effects to make something that is not real appear real.

A SWAT commander would not want to sacrifice reality for believability. A conscious decision was made in the design of RealityFlythrough to not conceal defects that occur during image blending. Hiding the defects would modify the images that are being displayed and may conceal crucial evidence that the commander needs to make a decision. The defects also help a user make sense of how two images are related to one another

2.7.3 Beyond Being There

Throughout our exploration, we have identified a number of *Beyond Being There* elements of the various media involved in presence. These attributes contribute to the sense that an accurate replication of reality should not be the goal of applications that mediate presence. There are things that the media may be able to do better than physically proximate reality. The properties of presence should be exploited where necessary or where helpful as a user interface. Ultimately, however, the task should dictate the degree to which the experiences of reality are reproduced.

What follows is a categorization of *Beyond Being There* properties that have been identified so far and a discussion of how these properties might be supported in RealityFlythrough.

Freedom of Movement

Virtual space allows for movement without the constraints imposed by physical forces. RealityFlythrough users can move as fast as they desire and can breeze through walls and other barriers. They can also observe a scene from whatever angle is covered by a camera. This means that a user can fly to the fourth story of a nearby building to look down on the plaza below, for example.

Heightened Senses

The sensors monitoring an environment can be augmented to capture stimuli that would ordinarily be missed. There is nothing that prevents RealityFlythrough from using these augmented sensors since the images that are displayed to the user are not

processed using vision techniques. Infrared, ultraviolet, and night-vision can all be supported with ease.

Augmented Reality

Scene content can also be augmented with abstract data that is not directly derivable from the environment. RealityFlythrough, for example, can overlay meta-data on buildings and objects visible in the scene. A building might be tagged with a name, a description, and even a URL, for example.

History

Mediated presence provides the opportunity for exploration through history. RealityFlythrough maintains a history of all images that were captured during a session and allows the user to explore the scene both temporally and spatially. For every image that is viewed, the user can choose to temporarily jump to that time. The video feed associated with that time can be viewed, or the space can be explored as it looked at that time. The typical controls that are present on all video players are available (pause, rewind, fastforward) with some extras. Time can progress at any speed and in any direction. Users also have the ability to single-step through a video feed in either direction to allow viewing of temporally nearby images without affecting the progression of real time.

Privacy

We identified the opportunity for mediated presence to preserve elements of privacy by introducing filters on the content. These filters can be as simple as a mute button on a telephone or as complex as the "shade" suggested for the Virtual Window that only allows certain content to filter through. Privacy is a large concern for remote presence systems that allow clandestine exploration of a space. The problem could be ameliorated if each camera had a "shading" capability that could be controlled by users who are in the field of view. The privacy preserving shading functions could be managed

by the client-side camera software. The actual content of the image makes no difference to the RealityFlythrough engine, but the experience could be partially degraded if the majority of cameras used such a privacy mode. The benefit of targeting applications such as SWAT is that the participants are all collaborators (except for the targets, of course), and privacy is not a primary concern.

2.7.4 Reflections

RealityFlythrough is successful in environments where other solutions fail primarily because its design was application driven. RealityFlythrough is a remote presence solution, but it only uses a presence-like interface where applicable and where possible. Physically proximate reality inspired the first-person immersive user interface, but the images displayed during transitions diverge somewhat from what might be expected in reality. This divergence is a necessary result of the conditions RealityFlythrough is designed to work in. What is remarkable is that even with imprecise input, a human visual system primed with a 3d immersive environment and given cues in the form of motion, zooming, and object overlap during transitions, can infer the 3d geometry of a scene. The computer need only provide the necessary stimuli, and the human brain can work out the rest. The natural presence-like interface takes advantage of the brain's ability to do automatic processing, thus freeing up consciousness to work on the tasks that the user is trying to perform.

2.8 Conclusion

We have explored five mediated presence applications and highlighted the key properties of presence. I have argued that presence should not be the goal of applications that mediate presence. Instead, the requirements of the user and the task should be what drive application design and determine the degree of presence that is needed. We have discovered three qualities of mediated presence that support this claim: (1) We can do better than presence; we can go *Beyond Being There*. We have seen that we can

go to remote locations without traveling—dangerous locations like SWAT scenes and volcanoes. We have seen that we can move through the environments without physical constraints, have heightened senses, and have access to Augmented-Reality style meta-data. (2) There are times when presence is simply the wrong interface. Gaver's Virtual Window showed us this when it was incorrectly used as a peripheral display. SWAT commanders may find themselves in similar situations. Despite having access to mediated-presence they may prefer instead to use the system as a peripheral display if there are a few key locations that they want to pay attention to—the entrances and exits to buildings, for example. And, (3) people often need to be present in more than one place at a time, and an all-consuming presence would prevent this.

This is not to say that the use of presence as an interface for applications is the wrong design. In many situations, the naturalness of a presence-like interface is the best design. We have seen a number of examples of this: The Virtual Window showed us how natural and seamless a presence-like interface is. Visual Eyes showed us how effective a presence-like interface is at facilitating communication. The common reference replaces a whole class of communication. The Space Browser showed us that having a physical presence and being able to interact with the people in an environment is a powerful means of exercising control. And, AVE showed us that having a natural interface for viewing a live environment from any angle is compelling and desirable. For these reasons, RealityFlythrough uses a presence-like interface for navigating through live video streams. There are other ways that the same information could have been conveyed to the user, but the use of presence is very natural and allows many of the navigation and orienting tasks to be done by automatic brain processes, thus freeing up consciousness to work on the tasks that the user is trying to perform.

When presence is a desirable property of an application, there are a number of ways that the sense of presence can be enhanced. More realistic, more believable, and more imagination inducing environments enhance presence. Allowing the user to personalize the experience by controlling the sensory content of the environment is also presence enhancing. Including more senses or increasing the fidelity of the sensory

input can help, and finally, the more the users are able to manipulate or control the environment, the more present they will feel.

Chapter 3

An Abstraction for Ubiquitous Video

Note that this chapter is a reprint with minor changes of *A Systems Approach to Ubiquitous Video* [MG05c], a paper co-authored by Neil McCurdy and William Griswold. This chapter describes the architecture of RealityFlythrough as it was in 2005, an architecture that has largely remained the same despite the addition of two major components, the *Smart Camera* that uses RealityFlythrough transitions to augment low frame-rate video, and the *Composite Camera* that uses point-matching meta data to improve image placement. Both of these concepts are introduced in chapter 5, and expanded upon in chapter 6.

3.1 Introduction

The key to harnessing ubiquitous video is in managing the incoming video streams. A naive approach would display the video on an array of monitors similar to those used in many building security systems today. An ideal solution would have infinite cameras in the field, and allow the user to move seamlessly through the environment choosing any desired vantage point. The more practical solution employed by RealityFlythrough provides the illusion of the ideal system while operating under the constraints imposed by the real environment, including the constraint that the resulting displays should not be misleading.

The key *limitation* of ubiquitous video is the incomplete coverage of the live

video streams–every square meter of a space cannot be viewed from every angle with a live video stream at any chosen moment. For two cameras pointing in two rather different directions, when the user switches from viewing one camera to another, it is often not obvious how the subject matter in the two views relate to each other, nor is it obvious what is in the intervening space between the two cameras.

To address this limitation, RealityFlythrough fills the intervening space between two cameras with older imagery (captured from the live camera feeds), and provides segues (i.e., transitions) between the two live cameras that sequences and blends the imagery in a way that provides the sensation of a human performing a walking camera pan. In certain scenarios the display of older imagery may be undesirable. While not ideal, transitions without background imagery are still sensible because the motion and timing of the transition and a background floor grid convey the distance and angle traveled. The user has complete control over how older imagery is displayed—whether it is displayed at all, in a sepia tone, or with an age-indicator-bar.

The key *untamed element* of ubiquitous video is the imprecision of the sensed location and orientation of a camera (due to both sensor latency and sensor inaccuracy). Such imprecision gives misleading cues to the user about how the subject matter seen in one camera relates to the subject matter in another. For example, the images might appear farther apart than they really are.

The contributions of this paper are the RealityFlythrough architecture, and its evaluation along three dimensions: (1) its support for the desired abstractions for ubiquitous video, (2) its scalability, and (3) its robustness to changing user requirements that is the measure of every good architecture.

The emphasis is on the architectural components that support the abstraction of infinite camera coverage. As will be shown throughout the paper, the architecture greatly reduces the complexity of the system, replacing complicated algorithms with concepts as simple as fitness functions. The design of a large-scale system that can accommodate thousands of cameras across multiple locations is considered in Section 3.6.3, but is not the focus of this paper. In many scenarios (most disaster response

and SWAT scenarios), the size of the site and the availability of network bandwidth will limit the number of cameras that can be deployed. The architecture, as described, can easily handle these situations.

The architecture has two unique qualities. First, it uniformly represents all image sources and outputs as *Cameras*, supporting a rich yet simple set of operations over those elements in achieving the desired abstractions. And, second, it employs a separate *Transition Planner* to translate the user's navigation commands into a sensible sequence of camera transitions and accompanying image blends. Our experiments show good support for the desired abstractions, as well as excellent scalability in the number of live video sources and *Cameras*. Support for evolution is explored through a series of changes to the application.

The paper is organized as follows. Section 3.2 describes the user experience, and Section 3.3 outlines the requirements of the system. We present a high level architectural overview of the system in Section 3.4, and then drill into the RealityFlythrough engine in Section 3.5 to reveal how the illusion of infinite cameras is achieved. Sections 3.6.1 and 3.6.2 evaluate the architecture's support of the system requirements, and Section 3.6.3 evaluates the architecture's tolerance to change and support for future enhancements. Section 3.7 concludes the paper.

3.2 User Experience

A significant part of the user experience in RealityFlythrough is dynamic and does not translate well to the written word or still-photographs. We encourage the reader to watch a short video [MG04] that presents an earlier version of RealityFlythrough, but we do our best to convey the subtlety of the experience in this section. When observing the images in Fig. 1.2, keep in mind that the transformation between the images is occurring within about one second, and the transitional frames represent only about 1/10th of the transition sequence.

The user's display is typically filled with either an image or a video stream

taken directly from a camera. When the user is "hitchhiking" on a camera in this way, the experience is similar to watching a home-video where the camera operator is walking around while filming. A still-image, then, is simply the home-video paused. When a new vantage point is desired, a short transition sequence is displayed that helps the user correlate objects in the source image stream with objects in the destination image stream. These transitions are shown in a first person view and provide the users with the sensation that they are walking from one location to another. The illusion is imperfect, but the result is sensible and natural enough that it provides the necessary contextual information without requiring much conscious thought from the users.

RealityFlythrough works by situating 2d images in 3d space. Because the position and orientation of every camera is known, a representation of the camera can be placed at the corresponding position and orientation in virtual space. The camera's image is then projected onto a virtual wall (see Fig. 1.3). When the user is looking at the image of a particular camera, the user's position and direction of view in virtual space is identical to the position and direction of the camera. As a result, the entire screen is filled with the image. Referring to Fig. 1.2, a *transition* between camera A (the left-most image) and camera B (the right-most image) is achieved by smoothly moving the user's position and view from camera A to camera B while still projecting their images in perspective onto the corresponding virtual walls. By using OpenGL's standard perspective projection matrix to render the images during the transition, the rendered view situates the images with respect to each other and the viewer's position in the environment. Overlapping portions of the images are blended using an alpha-blend. By the end of the transition, the user's position and direction of view are the same as camera B's, and camera B's image fills the screen. As shown in Fig. 1.2, additional images are displayed (if available and if desired) to help provide contextual information.

It may be easier to understand how RealityFlythrough works by envisioning the following concrete example. Imagine standing in an empty room that has a different photograph projected onto each of its walls. Each image covers an entire wall. The four photographs are of a 360 degree landscape with one photo taken every 90 degrees.

Position yourself in the center of the room looking squarely at one of the walls. As you slowly rotate to the left your gaze will shift from one wall to the other. The first image will appear to slide off to your right, and the second image will move in from the left. Distortions and object misalignment will occur at the seam between the photos, but it will be clear that a rotation to the left occurred, and the images will be similar enough that sense can be made of the transition. RealityFlythrough operates in a much more forgiving environment: the virtual walls are not necessarily at right angles, and they do not all have to be the same distance away from the viewer.

RealityFlythrough works in the wild because there is little information the system requires about each camera, and no preprocessing is required to render the transitions. The position of the camera can be obtained from whatever locationing technology is desired (we use WAAS-enabled consumer GPS's for outdoor tests), and the tilt, roll, and yaw can be determined with a tilt sensor that has a magnetic compass (we use an AOSI EZ-Compass). Since the human visual cortex is responsible for finding correlations between images the primary requirement for positional accuracy is that there be sufficient image overlap. We have found that an accuracy of 6-9 meters is adequate in outdoor settings where there is a wide field of view. Much higher accuracy would be necessary indoors—room-level accuracy, at minimum. Orientation accuracy is much more important because a camera that has less than a 40 degree field of view (typical of most web cameras) cannot be off by many degrees before images do not overlap at all. Magnetic compasses have provided good results, but may have trouble in areas of high magnetism.

3.3 Requirements

In earlier work [MG05b], we built a proof-of-concept system that revealed a number of rich requirements for harnessing ubiquitous video. Ubiquitous video is challenging because the cameras are everywhere, or at a minimum can go anywhere. They are inside, outside, carried by people, attached to cars, on city streets, and in parks.

Ubiquity moves cameras from the quiet simplicity of the laboratory to the harsh reality of the wild. The wild is dynamic—with people and objects constantly on the move, and with uncontrolled lighting conditions; it is uncalibrated—with the locations of objects and cameras imprecisely measured; and it is variable—with video stream quality, and location accuracy varying by equipment being used, and the quantity of video streams varying by location and wireless coverage. Static surveillance-style cameras may be available, but it is more likely that cameras will be carried by people. Mobile cameras that tilt and sway with their operators present their own unique challenges. Not only may the position of the camera be inaccurately measured, but sampling latency can lead to additional errors.

Our proof of concept system revealed the need for better image quality, higher frame rates, greater sensor accuracy with faster update rates, and better support for the dynamic nature of ubiquitous video.

We used a SWAT team scenario as a concrete example to help us tease out the requirements of applications that may benefit from RealityFlythrough. At a typical SWAT scene, the team commander is situated some distance from the incident site, and often must direct field operations without the aid of visuals. Commands must be issued to field officers from their point of view, straining the commander's conceptual model of the scene. Mistakes do happen [JH02]. Discussions and initial trials with the San Diego Metropolitan Strike Team (MMST) with whom we are collaborating as a part of a larger project called WIISARD (Wireless Internet Information System for Medical Response) have confirmed that video may be an effective means for providing early situational awarenes. We can expect to have 25 officers, and therefore 25 cameras, in the field.

Common knowledge about police operations combined with the previous description reveal minimum requirements for a system that could support SWAT: The system must work at novel sites with minimal configuration; the command center must be nearby and fairly mobile; cameras should be mobile and therefore wireless; the system needs to handle very incomplete camera coverage with fewer than 25 cameras in the field; and the system must work in unforgiving environments with intermittent network

connectivity.

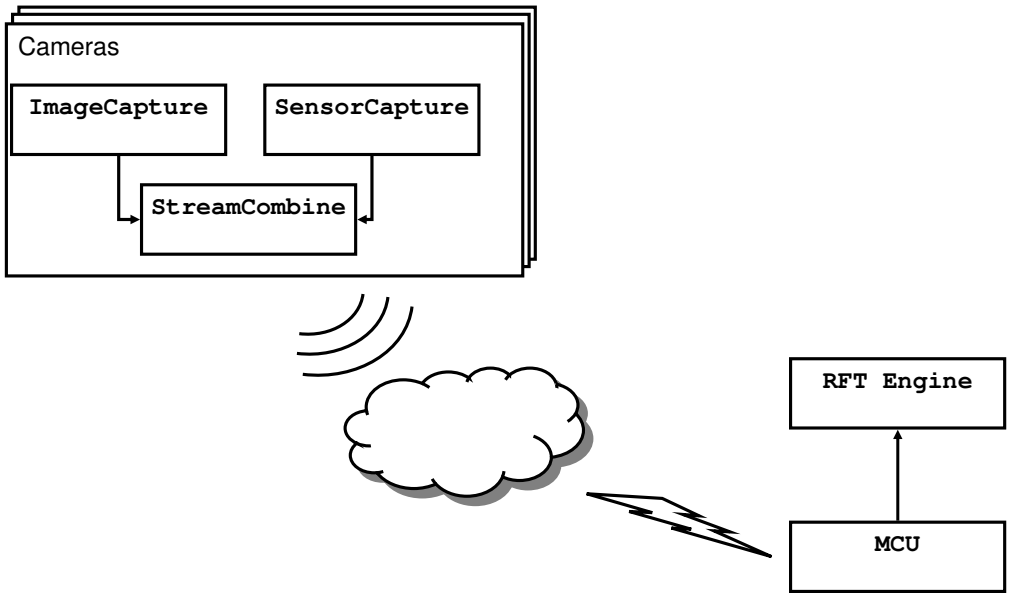


Figure 3.1: Component diagram showing system overview.

3.4 System Overview

Given the requirements just outlined, how might such a system be built? First we need some cameras and location sensors. We need to capture the image data from a camera and compress it, and we also need to capture the sensor data. We call the components that do this, *Image Capture* and *Sensor Capture*, respectively. The data then needs to be combined so that we can match the sensor data to the appropriate frame in the image data. We call the component that handles this: *Stream Combine*. The resulting stream then needs to be sent across the network to a machine that decodes the data and presents it to the user. We have a modified *MCU* (Multipoint Control Unit)

that does the decoding, and a *RealityFlythrough Engine* that combines the streams and presents the data to the user in a meaningful way. (Fig. 3.1 shows the relationships between these components.)

All of the video transmission components are based on the OpenH323 (`http://www.openh323.org`) implementation of the H323 video conferencing standard. Video can be transmitted using any H323 client without modification, but the sensor data would need to be transmitted separately and recombined on the server side. For our early prototype, though, we chose to embed the sensor data into the video stream to reduce complexity and to minimize network traffic. We mention this only because stand-alone video conferencing units that do hardware video compression are already starting to emerge, and it was a key design decision to follow standards so that we could support third party components.

RealityFlythrough is written in C++ and makes heavy use of OpenGL for 3D graphics rendering, and the boost library (`http://boost.org`) for portable thread constructs and smart pointers. A projection library (`http://remotesensing.org/proj`) is used to convert latitude/longitude coordinates to planar NAD83 coordinates, and the Spatial Index Library (`http://www.cs.ucr.edu/~marioh/spatialindex`) is used for its implementation of the R-Tree datastructure [Gut84] that stores camera locations. RealityFlythrough is designed to be portable and is confirmed to work on both Windows and Linux.

The *Engine* is roughly 16,000 lines of code (including comments), and the *MCU* is an additional 2600 lines of code written on top of OpenH323.

3.5 Engine Architecture

The *RealityFlythrough Engine* is the heart of the system. Given the available video streams and the user's intentions as input, the *engine* is responsible for deciding which images to display at any point in time, and for displaying them in the correct perspective. Fig. 3.2 shows the functional components of the engine. The standard

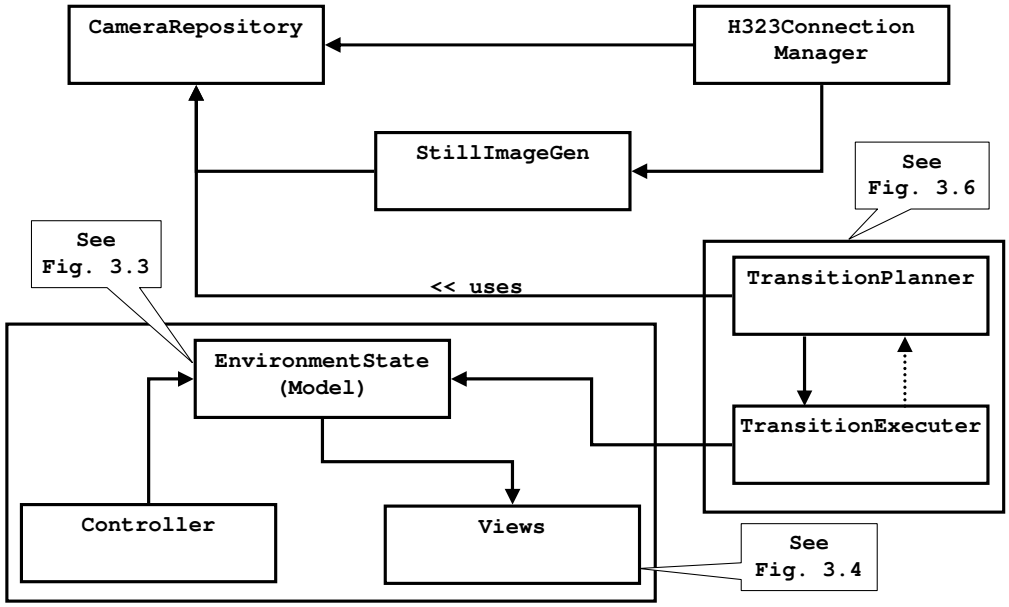


Figure 3.2: Component diagram showing an overview of the RealityFlythrough engine. Unlabeled arrows represent "calls" relationships. The dotted line is an event callback.

Model-View-Controller design pattern [GHJV95] is used to represent and display the current system state. The *Still Image Generator* is responsible for producing and managing the still-images that are generated from the live camera feeds. These still-images are used to backfill transitions, but may also be worth viewing in their own right since they may not be much older than the live feeds. The *Transition Planner/Executer* is responsible for determining the path that will be taken to the desired destination, and for choosing the images that will be displayed along that path. The *Transition Executer* part of the duo actually moves the user along the chosen path. And finally, the *Camera Repository* acts as the store for all known cameras. It maintains a spatial index of the cameras to support fast querying of cameras.

3.5.1 Model-View-Controller

The objects that comprise the Model-View-Controller support the abstraction of infinite camera coverage. The key element of our abstraction is a virtual camera (Fig. 3.3) which is simply a location, an orientation, a field of view, and a list of the "best" cameras that fill the field of view. The notion of "best" will be explored in Section 3.5.3, but for now simply think of it as the camera that most closely matches the user's wishes. A virtual camera, then, can be composed of multiple cameras, including additional virtual cameras. This recursive definition allows for arbitrary complexity in how the view is rendered, while maintaining the simplicity suggested by the abstraction: cameras with an infinite range of view exist at every conceivable location and orientation.

Model. The concept of a virtual camera is extended all the way down to the *Environment State* (Fig. 3.3) which is the actual *model* class of the Model-View-Controller. The user's current state is always using the abstraction of a *Virtual Camera* even if the user is hitchhiking on a *Physical Camera*. In that particular case the *Virtual Camera* happens to have the exact position, orientation, and field of view of a *Physical Camera*, and hence the physical camera is selected as the "best" camera representing the view. The current state of the system, then, is represented by a *Virtual Camera*, and

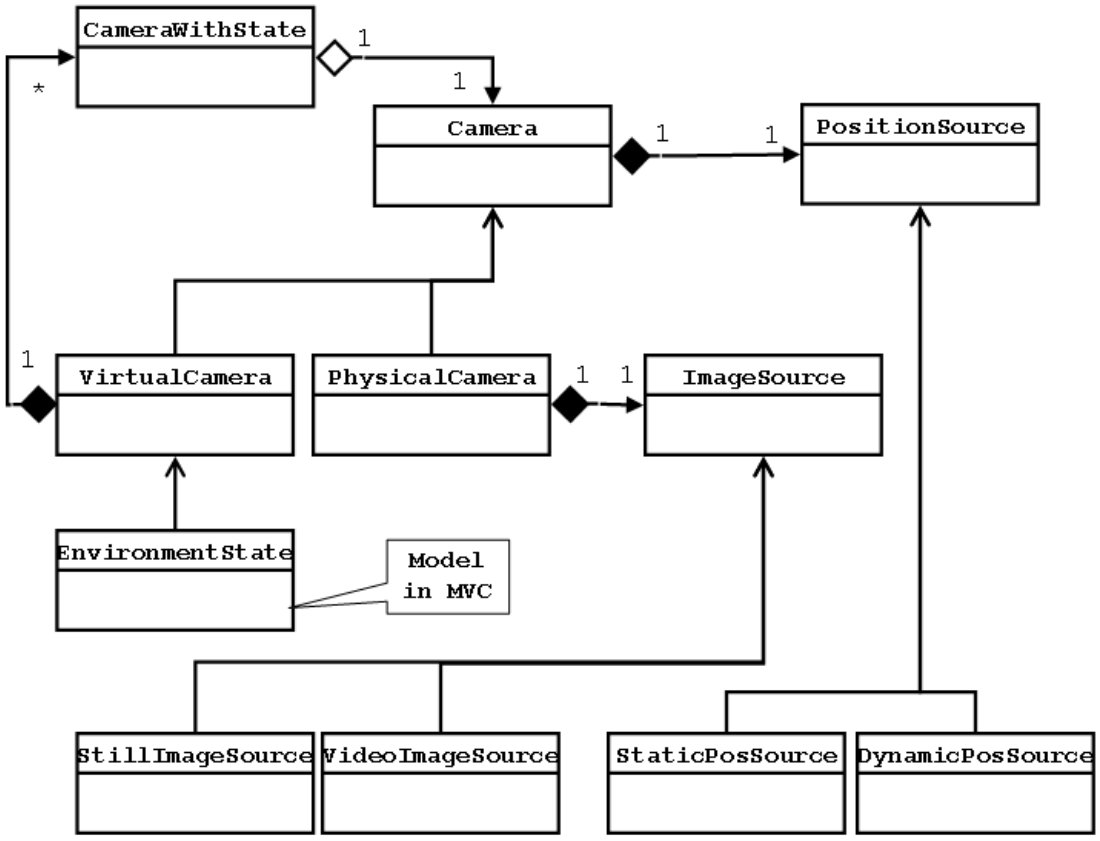


Figure 3.3: Class diagram showing the relationship of classes that are directly related to the Model in the MVC design pattern. For all class diagrams, open arrows represent inheritance, and arrows with diamonds represent containment. Open diamonds indicate that the container has only a reference to the object, while filled-in diamonds indicate ownership.

therefore by a position, an orientation, and the physical cameras that comprise the view. Changing the state is simply a matter of changing one of these three data points.

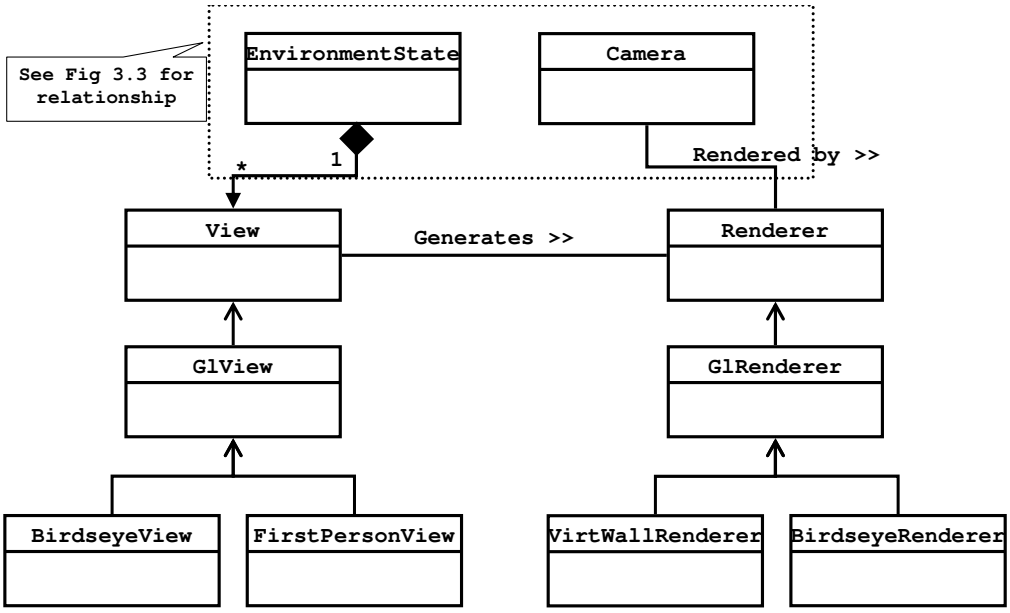


Figure 3.4: Class diagram for the classes involved in the View relationship of the MVC. The "Gl" in class names indicates that the classes are OpenGL-specific.

View. The Model-View-Controller design pattern naturally supports multiple views into the system state. There are currently two views (Fig. 3.4), but we envision more (see Section 3.6.3). The two views are the *First Person View* and the *Birdseye View*. The *First Person View* is the primary view that displays the images from a first person immersive perspective. This is the view that was described in Section 3.2. The *Birdseye View* shows a top-down perspective on the scene, with cameras rendered as arrows and the field of view of active cameras displayed as cones emanating from the arrows (Fig. 3.5).

The *Birdseye View* not only provides a wide-area map view of the scene, but

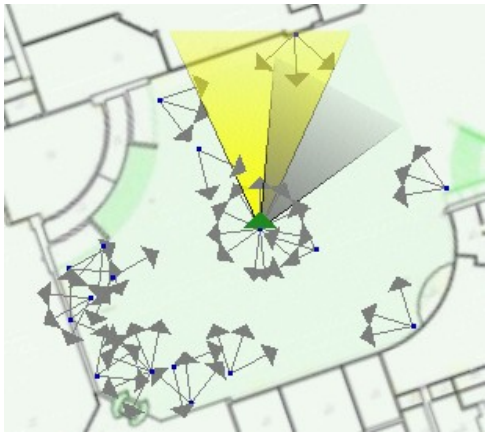


Figure 3.5: The birdseye view. The arrows represent the camera locations and directions of view.

also reveals some of the rawness of ubiquitous video that is being abstracted away by the *First Person View*. The birdseye view makes the live camera coverage (or lack thereof) obvious and it reveals the ages and density of the still-images that are used for backfill (see Section 3.5.2). There are currently three display modes available in the birdseye view: (1) show all cameras, (2) show only the cameras that have been updated within some user specifiable interval, and (3) show only the live cameras. In an ideal environment, the user could ignore the information presented in the birdseye view because a live image would be present at every vantage point. A more typical scenario, and the one we adopted in the experiment described in Section 3.6, presents the user with the birdseye view that shows only the locations of the live cameras. The assumption, then, is that the intervening space is fully populated with still-imagery. In this mode, the illusion of infinite camera coverage is still present, but the user is given some extra insight into where live camera coverage is available.

Each view instantiates one or more renderers to actually render the cameras that are involved in the current state. Since the definition of a *Virtual Camera* is recursive, there may be multiple cameras that need to be rendered. Each of these cameras has a state associated with it: the opacity (intensity) at which the camera's image should be drawn for the alpha blend. There are currently two types of renderers: *Virtual Wall Renderer* and *Birdseye Renderer*.

The *Virtual Wall Renderer* is used by the *First Person View*. It renders images using the virtual wall approximation described in Section 3.2. The images are rendered in a specific order, on the appropriate virtual walls, and with the opacity specified in their state. The animation of a transition is achieved by moving the user's view point a set distance for each frame and progressing the alpha-blend for the overlapping portions of all of the images.

The *Birdseye Renderer* simply draws either the camera arrow or the frustum cone depending on the current state of the camera.

Controller. The controller is a typical MVC controller and does not require further comment.

3.5.2 Still Image Generation

Key to the success of the infinite camera abstraction is the presence of sufficient cameras. If no imagery is available at a particular location, no amount of trickery can produce an image. To handle this problem, we take snapshots of the live video feeds and generate additional physical cameras from these. A *Physical Camera* consists of an *Image Source* and a *Position Source* (Fig. 3.3). The *Image Source* is a class responsible for connecting to an image source and caching the images. The *Position Source*, similarly, is responsible for connecting to a position source and caching the position. A camera that represents still-images, then, is simply a camera that has a static image source and a static position source. This is contrasted with live cameras that have a *Video Image Source* that continually updates the images to reflect the video feed that is being transmitted, and a *Dynamic Position Source* that is continually updated to reflect the current position and orientation of the camera.

To keep the still-imagery as fresh as possible, the images are updated whenever a camera pans over a similar location. Rather than just update the *Image Source* of an existing camera, we have chosen to destroy the existing camera and create a new one. This makes it possible to do a transitional blend between the old image and the newer image, without requiring additional programming logic. The images fit neatly into our

Camera abstraction. *In later work, described in chapter 6, we maintain a history of the still-images allowing the user to do temporal navigation through the environment. This effectively allows PVR-style (Personal Video Recorder) time-shifting on all cameras simultaneously; it is a PVR for the whole environment.*

The use of still-imagery to help achieve the abstraction of infinite camera coverage is of course imprecise. There are two ways that the limits of the abstractions are disclosed to the user:

First, the user has the option to never see older images. The user's preferences are used in the "best camera" calculation, and if no camera meets the criteria, the virtual camera will simply show a virtual floor grid.

Second, older images look different. The user can choose to have the old images displayed in a sepia tone, and can also choose whether or not to display an age-indicator-bar at the bottom of the sepia-toned or true-color images. The sepia tone makes it absolutely clear that the image is old, but it has the disadvantage that it alters the image, contradicting our aim to not mask reality. It is quite possible that this kind of image manipulation can hide information crucial to the user. An alternative is to show the age-indicator-bar on true-color images. The bar is bi-modal, giving the user high resolution age information for a short interval (we currently use 60 seconds), and lower resolution age information for a longer interval (currently 30 minutes). With a quick glance at the bottom of the screen, it is very easy for the user to get a sense of the age of an image.

3.5.3 Transition Planner/Executer

When the user changes views, the *Transition Planner* (Fig. 3.6) is responsible for determining the path through space that will be taken and the images that will be shown along this path. The *Transition Executer* is responsible for moving the user along the chosen path. There is a high degree of coupling between the *planner* and the *executer* because of the dynamic nature of ubiquitous video. Consider a typical case where the user wishes to move to a live camera. A naive approach would determine the location

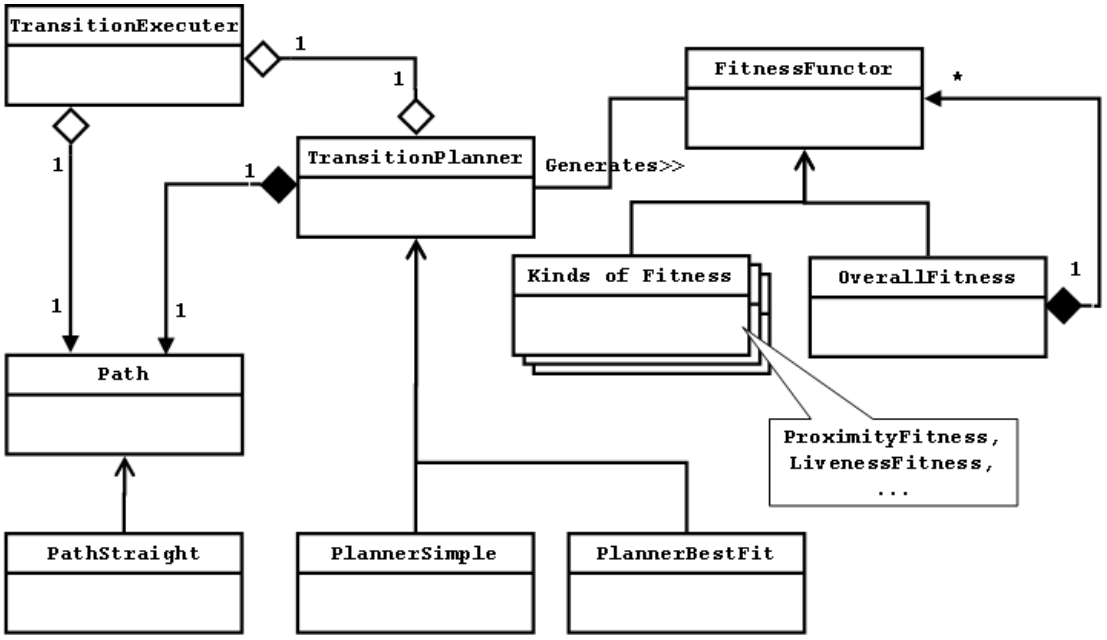


Figure 3.6: Class diagram showing the relationship of the classes involved in transition planning.

and orientation of the live camera, compute the optimal trajectory to get to the target location and orientation, determine the images to be shown along the path, and finally execute the plan that was just developed.

This approach does not work in a ubiquitous video environment for several reasons. The primary problem is that the destination camera may change its position and likely its orientation in the interval between when the plan was computed and when the execution of the plan has completed. The result will be a plan that takes the user to the wrong destination. Another problem is that the images that are selected along the path may not be the optimal ones. This is because the cameras that provide the intervening imagery may be live cameras as well, in which case their locations and orientations may have changed in the time since the plan was created. The result is that a live image that could have been shown is missed, or perhaps worse, a live image is shown that can no longer be seen from the current vantage point, so instead no image is displayed. Another possibility is that the dynamically generated still-imagery is updated after the plan is generated, but the older image is displayed instead.

To account for all of these problems the transition planning needs to be done dynamically and interleaved with the execution. There are a number of competing issues that need to be balanced when doing dynamic planning. It would seem that the ideal is to construct a plan at every time step, but some parts of the planning process are computationally expensive and need to be done sparingly. Also, the user needs to be given time to process the imagery that is being displayed, so even if a better image is available, showing it immediately may actually reduce comprehension.

The solution is to first introduce a dynamic *Path* object that takes a *Position Source* rather than a *Position* as its destination. The destination is now a moving target. At every time step, the *Path* can be queried to determine the current trajectory. With this trajectory, the *Transition Planner* can look ahead some interval and determine the best image to display. This image (camera, really) is added to the end of the camera queue. Each *Virtual Camera*—and since the *Transition Planner* acts on the *Environment State* remember that the *Environment State* is a virtual camera—maintains a fixed-length queue of cameras. When the queue is filled and a new camera is added, the camera at the front of the queue (the oldest or least relevant camera) is popped off the queue and thus removed from the *Virtual Camera*. The new camera that is added has a time-based opacity which means that the opacity gradually increases with time. We currently have the image blend to full opacity in one second.

This approach results in what appears to be a transition from one image to another, but along a dynamically changing path and with images that were used earlier still being displayed (if in view) to provide additional contextual information. The piece of the puzzle that is still missing is how the plan is constructed and adjusted dynamically. The *Transition Executer* (Fig. 3.6) is responsible for querying the *Path* at every time step and moving the user along the desired trajectory. It is also responsible for notifying the *Transition Planner* at time intervals set by the *planner*. These notification events give the *planner* the opportunity to determine which image (if any) to display next. Time is being used for signaling instead of "destination reached" because having the *Path* be dynamic means the destination may never be reached. Time is an adequate approximation of this

signal point.

To determine the images to show during a transition the *Transition Planner* applies a series of *Fitness Functors* to each camera in the neighborhood. The *Fitness Functors* are weighted based on user preference. Some of the fitness dimensions are: proximity (how close is the camera to the specified position), rotation and pitch (how well do the orientations match), screen fill (how much of the screen would be filled with the image if it were displayed), recency (how recently was the image acquired), and liveness (is the camera live or not).

To further increase the sensibility of transitions, three heuristics are used to decide which images to display: (1) The current image should stay in view for as long as possible, (2) once the *to* image can be seen from the current position, no other images should be displayed, and (3) there should be a minimum duration for sub-transitions to avoid jumpiness. The first two items are handled by always applying the *Fitness Functors* to the current camera and the ultimate target camera regardless of whether they pass the "in the neighborhood test", and then boosting the fitnesses by a configurable scalar value. This has the effect of giving extra weight to the current and target cameras, thus indirectly satisfying our heuristics. The third item is handled by adjusting the time interval used for *Transition Planner* callbacks.

3.5.4 Camera Repository

The *CameraRepository* is simply a container for all of the cameras (including the still-cameras) that are known to the system. To support efficient spatial querying of the cameras, an R-Tree [Gut84] is used to store the camera locations. The exact locations of the live cameras are not stored in the index because this would cause continuous updates to the index, and such precision is not necessary when doing "get cameras in neighborhood" queries. Instead, only location updates that are greater than a configurable threshold result in a replacement in the spatial index.

Each *physical camera* has certain fixed memory costs. To minimize the use of limited OpenGL resources, the cameras share a pool of texture maps. We have to store

the image somewhere, though, so each camera (*Image Source*, really) allocates 768KB to store a 512x512 image (the size is dictated by OpenGL's texture map size requirements) at a depth of 24bits. After a period of inactivity, the *Image Source* frees memory by storing the image to disk. Under normal loads, there is no perceptible difference in performance when an image is read from disk.

3.6 Evaluation

An architecture must be evaluated along two dimensions: does it work, and will it work in the future? In this section we first present a formative user study that captures the essence of the user experience and helps show that the abstractions presented are compelling and useful. Second, we examine performance to get insight into the scaleability of the system. Third, to evaluate how well the architecture will accommodate future changes to the application, we examine its robustness against a set of significant changes and extensions.

3.6.1 Effectiveness of the Abstraction

An earlier paper on RealityFlythrough [MG05b] suggested that the first-person perspective used in transitions generated a sense of "being there". We re-ran this experiment using our new architecture which was designed to better handle the dynamic nature of a ubiquitous environment. Unlike the first experiment where the still-images were painstakingly pre-inserted, this run made full use of the automatic still-image capture described in Section 3.5.2. This user study and the one described in the earlier paper were formative studies designed to provide evidence that RealityFlythrough research is headed in the right direction.

To determine how the system was perceived by users, we repeated the earlier experiment as closely as possible. We used the same subjects, the same equipment on the user end, and the same location for the flythrough.

There were three hand-carried camera units in the field. They consisted of a

standard logitech web camera (∼$100), a WAAS-enabled Garmin eTrex GPS (∼$125), a tilt sensor manufactured by AOSI (∼$600), and an 802.11b equipped laptop. The tilt sensor provides compass, tilt, and roll readings at ∼15hz. The video streams were transmitted using the OpenH323 video conferencing standard at CIF (352x288) resolution.

The subjects' task was to remotely explore our campus food court with the goal of getting a sense of what is happening, and to determine if there is anything to draw them to the site for lunch. The experiment was run twice because some problems with the system were encountered on the first run. We discuss this first experiment because the problems are revealing.

The first run of the new experiment was very positive from a technical standpoint. Three video streams connected successfully, and a large number of still-images were automatically generated, quickly filling the entire region with cameras. Only 61 pre-configured still-images were used in the earlier version of the experiment, but 100's were generated in this one, greatly increasing the camera density. Despite the extra overhead incurred by auto-generating the images and by planning transitions on the fly, the system performance felt about the same. In fact, the subjects made the statement that the "performance was definitely much nicer." The new H263 video codec proved to be far superior to the H261 codec used previously. The frame rate varied by scene complexity, but appeared to average about 6-8 frames per second. The frame size was the same as was used previously, but the image quality was better and the colors were much more vivid. The generated still-images were clear and of good quality. On several occasions the subjects rapidly pointed out the age of images, indicating the success of the age indicator bar.

Even with all of these improvements, though, the subjects were not left with a positive impression and had to conclude that "from a usability standpoint, it went down." Transition sequences were met with comments like "it seems like it's awkward to move through several of those stills", and "[that] transition wasn't smooth." Post-experiment analysis identified three sources for the problems: (1) Too many images were being presented to the user, not allowing time for one transition to be processed mentally

before another one was started. (2) The attempt to acquire a moving target resulted in an erratic path to the destination, causing disorientation. And, (3) no attempt was made to filter the location data by sensor accuracy. Still-images were being generated even when the GPS accuracy was very low, so transitions involved nonsensical images which detracted from scene comprehension.

Fortunately, none of these problems were difficult to handle. In Section 3.6.3 we will discuss the actual modifications made because these unplanned changes exemplify the architecture's robustness to changing requirements.

The experiment was repeated with much more positive results. Despite worse conditions at the experiment venue (we shared the space with a well attended Halloween costume contest), the subjects had much more positive comments such as, "Let's try one in the completely opposite direction. That was pretty nice.", and "It's pretty accurate where it's placing the images." "That was kind of cool. They weren't quite all in the same line, but I knew and felt like I was going in the right direction."

The costume contest placed some restrictions on where the camera operators could go, forced them to be in constant motion, and resulted in a lot of close-range video footage of people's backs (the cameras were being held at chest level). The constant motion may be typical with head-mounted cameras, and should be adressed seriously. The subjects found the constant motion to be annoying ("they're all over the map"), and the motion placed quite a strain on the new algorithm used to home in on a moving target. The subjects actually preferred the calmness of the still-images. Midway through the experiment, we asked the operators to slow down a bit, and the experience improved dramatically: "Yeah, that's what it is. So long as [the camera operators'] rotation is smooth and slow, you can catch up to it and have smooth transitions."

We have since experimented with ways to reduce the amount of motion that is experienced during transitions. The fact that our subjects preferred the calmness of the still-images is revealing. There were simply too many sources of movement in our transitions, making them difficult to comprehend and aesthetically unappealing. When we move through the real world we only have to take into account 6 dimensions of

movement—our own movement in three dimensions and the movement of the objects we are viewing in three dimensions. During a transition involving a moving camera, however, the camera is moving independently and so represents another three dimensions that have to be processed. Each additional moving camera being displayed adds three more dimensions. There is simply too much movement to process. The solution we have adopted involves pausing the live video streams whenever they come into view during a transition, and playing them back at increased speed once they have been acquired. This approach will be described in more detail in Section 3.6.3.

3.6.2 System Performance

By measuring the performance of the system we hope to provide some insight into the scalability of the architecture. Raw performance metrics mainly measure the speed of the hardware and the quality of the compiler. Seeing how the raw numbers vary under certain conditions, however, reveals important details about the architecture.

The experiments with RealityFlythrough described thus far have only been run using at most three video streams. To determine the maximum number of simultaneous streams that can be handled by the server, we ran some simulations. The capacity of the wireless network forms the real limit, but since network bandwidth will continue to increase, it is instructive to determine the capacity of the server. We should estimate the capacity of a single 802.11b access point to give us a sense of scale, however. For the image size and quality used in the user studies, the H263 codec produces data at a relatively constant 200Kbps. Empirical study of 802.11b throughput has shown that 6.205Mbps is the maximum that can be expected for applications [VS06]. This same study shows that the total throughput drops drastically as more nodes are added to the system. With more than eight nodes, total throughput decreases to roughly 2Mbps. This reduction means we cannot expect to have more than 10 streams supported by a single 802.11b access point.

We will see that the bottleneck on the server is the CPU. As more compressed video streams are added to the system, more processor time is required to decode them.

Some of the other functional elements in RealityFlythrough are affected by the quantity of all cameras (including stills), but the experimental results show that it is the decoding of live streams that places a hard limit on the number of live cameras that can be supported.

The machine used for this study was a Dell Precision 450N, with a 3.06Ghz Xeon processor, 512MB of RAM, and a 128MB nVidia QuadroFX 1000 graphics card. It was running Windows XP Professional SP2. The video streams used in the simulation were real streams that included embedded sensor data. The same stream was used for all connections, but the location data was adjusted for each one to make the camera paths unique. Because the locations were adjusted, still-image generation would mimic real circumstances. No image processing is performed by the engine, so replicating the same stream is acceptable for this study. The image streams were transmitted to the server across a 1 Gbit ethernet connection. Since the image stream was already compressed, very little CPU was required on the transmitting end. A 1 Gbit network can support more than 5000 simultaneous streams, far more than the server would be able to handle. Network bandwidth was not a concern.

To obtain a baseline for the number of streams that could be decoded by the server, we decoupled the *MCU* from the *engine*. In the resulting system, the streams were decoded but nothing was done with them. With this system, we found that each stream roughly equated to one percent of CPU utilization. 100 streams used just under 100 percent of the cpu. The addition of the 113th stream caused intermittent packet loss, with packet loss increasing dramatically as more streams were added. The loss of packets confirmed our expectation that the socket buffers would overflow under load.

Having confirmed that the addition of live cameras had a real impact on CPU utilization, we added the *RealityFlythrough engine* back to the system. We did not, however, add in the still-image generation logic. To determine the load on the system we looked at both the CPU utilization and the system frame rate as new connections were made. The system frame rate is independent of the frame rates of the individual video feeds; it is the frame rate of the transitions. It is desirable to maintain a constant

system frame rate because it is used in conjunction with the speed of travel to give the user a consistent feel for how long it takes to move a certain distance. As with regular video, it is desirable to have a higher frame rate so that motion appears smooth. To maintain a constant frame rate, the system sleeps for an interval between frames. It is important to have this idle time because other work (such as decoding video streams) needs to be done as well.

Table 3.1: Observed behavior when running the system at a system frame rate of 15fps. This table shows that the maximum number of simultaneous video streams that can be supported by our test hardware is 15. Adding more connections maxes out the CPU, making the system less responsive, and reducing the system frame rate.

# Conn	*∼CPU (%)*	*∼Fps achieved*
10	*85*	*15*
15	***95***	***14***
20	*100*	*10*

For this experiment, we set the frame rate at 15fps, a rate that delivers relatively smooth transitions and gives the CPU ample time to do other required processing. As Table 3.1 indicates, fifteen simultaneous video feeds is about the maximum the system can handle. The average frame rate dips to 14fps at this point, but the CPU utilization is not yet at 100 percent. This means that occasionally the load causes the frame rate to be a little behind, but in general it is keeping up. Jumping to 20 simultaneous connections pins the CPU at 100 percent, and causes the frame rate to drop down to 10fps. Once the CPU is at 100 percent, performance feels slower to the user. It takes longer for the system to respond to commands, and there is a noticeable pause during the transitions each time the path plan is re-computed.

To evaluate the cost of increasing the number of cameras, still-image generation was turned on when the system load was reduced to the 15 connection sweet spot. Recall that still-images are generated in a separate thread, and there is a fixed-size queue that limits the number of images that are considered. Still-images are replaced with newer ones that are of better quality, and there can only be one camera in a certain radius and orientation range. What this means is that there are a finite number of

still-images that can exist within a certain area even if there are multiple live cameras present. The only effect having multiple live cameras may have is to decrease the time it takes to arrive at maximum camera coverage, and to decrease the average age of the images. This assumes, of course, that the cameras are moving independently and all are equally likely to be at any point in the region being covered.

The live cameras were limited to a rectangular region that was 60x40 meters. A still-image camera controlled a region with a three meter radius for orientations that were within 15 degrees. If there was an existing camera that was within three meters of the new camera and it had an orientation that was within 15 degrees of the new camera's orientation, it would be deleted.

We let the system get to a steady state of about 550 still-images. The number of new images grows rapidly at first, but slows as the density increases and more of the new images just replace ones that already exist. It took roughly 5 minutes to increase from 525 stills to 550. At this steady state, we again measured the frame rate at 14fps and the CPU utilization at the same 95 percent. The system still felt responsive from a user perspective.

These results indicate that it is not the increase in cameras and the resulting load on the R-Tree that is responsible for system degradation; it is instead the increase in the number of live cameras, and the processor cycles required to decode their images. This shows that the architecture is scalable. Since the decoding of each video stream can be executed independently, the number of streams that can be handled should scale linearly with both the quantity and speed of the processors available. Depending on the requirements of the user, it is possible to reduce both the bandwidth consumed and the processor time spent decoding by throttling the frame rates of the cameras not being viewed. This would reduce the number of still-images that are generated; a tradeoff that only the user can make.

3.6.3 Robustness to Change

The investment made in an architecture is only warranted if it provides ongoing value; in particular it should be durable with respect to changing user requirements, and aid the incorporation of the changes dictated by those new requirements. Below we discuss several such changes, some performed, others as yet planned. Only one of these changes was specifically anticipated in the design of the architecture.

Planned Modification

The hitchhiking metaphor has dominated our design up to this point. Another compelling modality for RealityFlythrough is best described as the virtual camera metaphor. Instead of selecting the video stream to view, the users choose the position in space that they wish to view, and the best available image for that location and orientation is displayed. "Best" can either refer to the quality of the fit or the recency of the image.

It should come as no surprise that the virtual camera metaphor inspired much of the present design, so there is a fairly straight-forward implementation to support it. The *Virtual Camera* is already a first class citizen in the architecture. To handle a stationary virtual camera, the only piece required is a *Transition Planner* that runs periodically to determine the "best" image to display. Part of the virtual camera metaphor, though, is supporting free motion throughout the space using video game style navigation controls. The difficulty we will face implementing this mode is in minimizing the number of images that are displayed to prevent the disorienting image overload. This problem was easily managed with the hitchhiking mode because a fixed (or semi-fixed) path is being taken. The path allows the future to be predicted. The only predictive element available in the virtual camera mode is that the user will probably continue traveling in the same direction. It remains to be seen if this is an adequate model of behavior.

Another measure of a good architecture is that it is no more complicated than necessary; it does what it was designed to do and nothing more. The plan to support a virtual camera mode explains why the *Camera* is used as the primary representation

for data in the system. Once still-images, video cameras, and "views" are abstracted as cameras, they all become interchangeable allowing for the simple representation of complicated dynamic transitions between images.

Unplanned Modifications

In Section 3.6.1 we described three modifications to the system that needed to be made between the first and seconds runs of the experiment. We also examine some recent changes to the system that address the user frustrations with there being too many sources of movement during transitions. Since all of these modifications were unplanned, they speak to the robustness of the architecture.

Reduce Image Overload. The goal of the first modification was to reduce the number of images that were displayed during transitions. This change had the most dramatic impact on the usability of the system, making the difference between a successful and unsuccessful experience. The modification was limited to the *Transition Planner*, and actually only involved tweaking some configuration parameters. In Section 3.5.3 it was revealed that the current and final destination cameras are given an additional boost in their fitness. Adjusting the value of this boost does not even require a re-start of the system.

In the time since our experiments were run, we have further improved the transitions by slightly modifying our approach to finding the next camera to display. Instead of looking ahead at a fixed time-interval, we now calculate when the current image will no longer be optimal—because it has rotated off-screen, it is zoomed in too near, or zoomed out too far—and use this time interval for selecting the next image. Each image is now displayed for an optimal amount of time. We still boost the fitness of the destination camera to reduce the number of images that are displayed as the transition nears completion. These changes were all confined to the *Transition Planner*.

Moving Camera Acquisition. The second modification also involved transition planning, but in this case the change occurred in the *Path* class. The goal was to improve the users' experience as they transition to a moving target. The partial solu-

tion to this problem—implemented for the second experiment—adjusts the path that the users take so that they first move to the destination camera's original location and orientation, and then do a final transition to the new location and orientation. This makes the bulk of the transition smooth, but the system may still need to make some course corrections during the final transition. The full solution will be disclosed in the following section.

Too Much Movement. This modification has been made recently and has not been subjected to experimental evaluation. During the experiments our subjects voiced concern about the amount of movement experienced during transitions. Not only was the user virtually moving along the transition path, but the images generated by live cameras were also moving around the screen reflecting the camera movement as captured by the sensors. This problem was exacerbated by the seemingly erratic transition movement experienced during the acquisition of a moving target—helped, but not solved by the technique described in the previous paragraph.

Our current approach to this problem involves pausing the live video streams during a transition whenever they are visible on-screen. Once the destination camera has been acquired, the video stream is played back at increased speed until the users have caught up to the current time. This has two benefits: (1) it reduces the amount of motion that needs to be understood by the user during a transition, and (2) it pauses the moving target for an interval allowing for smoother final target acquisition. The video feeds are paused for only short durations (usually less than one second), so it does not take long for the user to catch up after the transition, and in early tests the pauses do not appear to be disruptive. The technique used for acquiring the moving target is still required because the target continues to move until it is actually visible on-screen.

Pausing of the video streams was handled by adding PVR-like (Personal Video Recorder) capabilities to the MCU. The incoming video streams are buffered to disk allowing for time-shifting and future replay of events. With this functionality added, the *Transition Planner* simply pauses and resumes the video feeds at the appropriate times.

Location Accuracy Filtering. The final change to the system was a little

more substantial since it required modification to both the client and server software. The goal was to filter the still-images on location accuracy. This change would have been trivial if we were already retrieving location accuracy from the sensors. As it was, the *Sensor Capture* component on the client had to be modified to grab the data, and then on the server side we had to add a location error field to our *Position* class. Now that every *Position* had an error associated with it, it was a simple matter to modify the *Still Image Generator* to do the filtering.

Future Modifications

Better High Level Abstraction. Forming continual correlations between the first-person-view and the 2d birdseye representation takes cognitive resources away from the flythrough scene and its transitions. We hope to be able to integrate most of the information that is present in the birdseye view into the main display. Techniques akin to Halos [BR03] may be of help.

This modification to the system should only affect the *First Person View*. Since we want to present the state information that is already available in the *Birdseye View*, that same information need only be re-rendered in a way that is consistent with the *First Person View*. If we want to create a wider field of view we could increase the field of view for the virtual camera that makes up the view. Another possibility is to generate additional views that are controlled by other virtual cameras. For example a window on the right of the display could be controlled by a virtual camera that has a position source offset by 45 degrees.

Sound. Sound is a great medium for providing context, and could be an inexpensive complement to video. By capturing the sound recorded by all nearby cameras, and projecting it into the appropriate speakers at the appropriate volumes to preserve spatial context, a user's sense of what is going on around the currently viewed camera should be enhanced.

Sound will be treated like video. Each *Physical Camera* will have a *Sound Source* added to it, and new views supporting sound will be created. There might be a

3D Sound View which projects neighboring sounds, and a regular *Sound View* for playing the sound associated with the dominant camera.

Scale to Multiple Viewers with Multiple Servers. Currently RealityFly-through only supports a single user. How might the system scale to support multiple users? The *MCU* component currently resides on the same machine as the *engine*. One possibility is to move the *MCU* to a separate server which can be done relatively easy since the coupling is weak. The problem with this approach, though, is that the *MCU* is decompressing the data. We would either have to re-compress the data, which takes time, or send the data uncompressed, which takes a tremendous amount of bandwidth. A better approach would be to leave the *MCU* where it is and introduce a new relay *MCU* on the new server layer. The purpose of the relay *MCU* would be to field incoming calls, notify the *MCU* of the new connections, and if the *MCU* subscribed to a stream, forward the compressed stream.

With the latter approach we could also support connecting to multiple servers. The *MCU* is already capable of handling multiple incoming connections, so the main issue would be one of discovery. How would the viewer know what server/s could be connected to? What would the topography of the network look like? We leave these questions for future work.

It is not clear where still-image generation would occur in such a model. The easiest solution is to leave it where it is: on the viewing machine. This has the additional benefit of putting control of image generation in the individual user's hands. This benefit has a drawback, though. Still images can only be generated if the user is subscribed to a particular location, and then only if there are live cameras in that location. What if a user wants to visit a location at night when it is dark? It's possible that the users want to see the scene at night, but it is equally likely that they want to see older daytime imagery. If the still-images are captured server side, this would be possible.

Since server-side still-image generation may stress the architecture as currently specified, we consider it here. The *engine* would not have to change much. We would need a *Still Image Generated* listener to receive notifications about newly gener-

ated cameras. A corresponding *Still Image Destroyed* listener may also be required. The camera that is created would have a new *Image Source* type called *Remote Image Source*. The *Position Source* would remain locally static. The *Remote Image Source* could either pre-cache the image, or request it on the fly as is currently done. Performance would dictate which route to take.

Robustness Summary

Each of the modifications presented is limited to very specific components in the architecture. This indicates that the criteria used for separating concerns and componentizing the system was sound.

3.7 Conclusion

We have presented an architecture for a system that harnesses ubiquitous video by providing the abstraction of infinite camera coverage in an environment that has few live cameras. We accomplished this abstraction by filling in the gaps in coverage with the most recent still-images that were captured during camera pans. The architecture is able to support this abstraction primarily because of the following design decisions:

(1) The *Camera* is the primary representation for data in the system, and is the base class for live video cameras, still-images, virtual cameras, and even the environment state. Because all of these constructs are treated as a camera, they can be interchanged, providing the user with the best possible view from every vantage point.

(2) The *Transition Planner* is an independent unit that dynamically plans the route to a moving target and determines the imagery to display along the way. New imagery is displayed using an alpha blend which provides the illusion of seamlessness while at the same time revealing inconsistencies. The system provides full disclosure: helping the user make sense of the imagery, but revealing inconsistencies that may be important to scene comprehension. Because the *Transition Planner* is responsible for path planning, image selection, and the blending of the imagery, it has a large impact

on the success of RealityFlythrough. Having the control of such important experience characteristics in a single component and having many of those characteristics be user controllable is key to the success of the current design.

The architectural choices made during the design of RealityFlythrough are primarily responsible for the effectiveness of the system. Complex algorithms that select the appropriate cameras to display at any given point are reduced to constructs as simple as fitness functions. The seemingly complicated rendering of multi-hop transitions to moving destinations is simplified to the rendering of a virtual camera from different perspectives along a dynamically changing path. The algorithms are simple; the architecture makes them so.

3.8 Acknowledgments

Special thanks to Robert Boyer, Jennifer Carlisle, Adriene Jenik, Charles Lucas, Michelle McCurdy, Jonathan Neddenriep, and the Active Campus team for their help with this project, and to our shepherd, Nigel Davies, whose comments on the paper were invaluable. This work was supported in part by a gift from Microsoft Research and contract N01-LM-3-3511 from the National Library of Medicine.

This chapter is, in part, a reprint of the material as it appears in A systems architecture for ubiquitous video. Neil J. McCurdy and William G. Griswold. In Mobisys 2005: Proceedings of the Third International Conference on Mobile Systems, Applications, and Services, pages 114. Usenix, 2005. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Closure: Why the Illusion Works

We have been using the word *closure* to describe the processing that is performed by the human visual system to make sense of incomplete visual information. The word *closure* is traditionally used in Gestalt psychology to describe the automatic completion of an image when only part of it is seen. For example, it is obviously important and useful for us to automatically complete the image of a leopard that is partially obstructed by other objects so that we can recognize it as a predator and not a disembodied-head creature that has never been seen before. McCloud co-opted this term and used it to explain how we are able to follow and understand the sequence of frames that make up comic books [McC93]. It is McCloud's extension of the term, *closure*, that we have used to describe the human processing that occurs in RealityFlythrough.

Even though the term has been generalized, however, the use of the term does not really explain how the RealityFlythrough illusion works. In this chapter we will attempt to weave together recent research in physiology, pyschology, and cognitive film theory to better understand the processes that are involved. The conclusions are largely speculative, but the explanation and theories that we present should provide a basis for future experimentation. This area is rich for further study, and we look forward to constructing experiments that will help elucidate the unconscious or conscious mental processes that are involved.

4.1 McCloud Closure

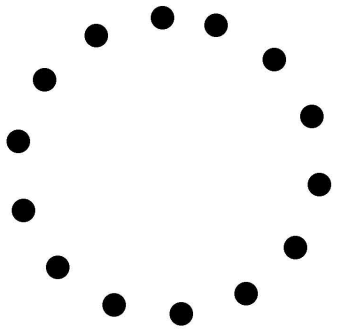


Figure 4.1: Gestalt closure. It is difficult not to see the larger circle.

In Gestalt psychology *closure* describes a perceptual process that automatically fills in gaps so that the whole is seen instead of the parts that make up the whole. The whole contains more meaning than the parts and it is nearly impossible to see the parts without seeing the whole (see fig. 4.1). McCloud extends the concept from what appears to be a low-level brain function rooted deep in the visual system, to a higher level one that works with abstract concepts. He uses the same term, closure, to explain how readers can follow a plot line from frame to frame in a comic book; how two distinct images can be transformed into a single idea. He goes so far as to state that comics *is* closure. While the closure described by Gestalt psychology appears to be involuntary, the closure of comic books is anything but involuntary. Closure, in fact, is used as a tool by artists, photographers, directors, and comic book illustrators to engage the audience. By not showing the audience everything, the audience is forced to complete the picture or story themselves, and thereby become a part of the story.

McCloud describes six different forms of closure, with all but the first one focused on comic books.

1. Continuous closure is the closure that allows people to see a sequence of images (a movie) as continuous motion. This is an automatic process that viewers have little control over. We will learn later in this chapter that *continuous closure* can be further divided into short-range and long-range apparent motion with the short-

range being indistinguishable from real motion.

2. Moment-to-moment closure is the easiest to process and is characterized by a scene that differs only very slightly between frames, indicating the passage of time.

3. Action-to-action closure shows two or more frames that involve some kind of action. For example, a frame that shows a batter standing ready to bat followed by a frame that shows the same batter hitting the ball.

4. Subject-to-subject closure requires more participation by the reader as he or she pieces together a story by looking at individual frames of subjects doing things. For example, one frame may show a runner crossing the finish line, and another, a hand holding a stopwatch. The reader, in this case, must form a connection between two distinct images by discovering the same abstract concept – winning a race – that binds them.

5. Scene-to-scene closure often forces the reader to use deductive reasoning to connect together two scenes that may be separated by a larges amounts of space and/or time. For example, McCloud shows one frame of a woman sobbing about a plane crash, and the next of a man sitting on a desert island. A great deal of knowledge about the external world, about the way that desert islands are often depicted, and about the stories in our culture of lost travelers (Gilligan's Island, in particular) is required in order to be able to piece together the connections between these two frames. The reader constructs an entire narrative across time and space.

6. Aspect-to-aspect closure requires the reader to ignore time as he or she views frames that share common themes or similar views on a common theme. For example, one frame of a Christmas tree and another of Santa Clause serve to reinforce the theme of Christmas and to tell the story about that moment in time.

McCloud's definition of closure is satisfying in its comprehensiveness. It is certainly clear that RealityFlythrough users must be using some form of closure to make

sense of the scenes they are witnessing, because, well, closure is a constant in our lives and we seem to use it to understand just about everything. So if RealityFlythrough users understand something about a scene instead of nothing they must be using closure to some degree.

The motion of the RealityFlythrough transition that evokes the sensation of walking through a scene is taking advantage of *continuous closure*. As far as the viewer is concerned, the motion in the transitions creates an involuntary sense of movement.

Moment-to-moment closure is used during the transitions between temporally adjacent images. As the experiments in section 7.3 reveal, some users focus on the temporal relationships between the images they are seeing and attempt to generate narratives that explain the sequence of shots. This is true even when the images were not in any particular temporal order. We need to exercise caution when using temporally disjointed imagery.

In the RealityFlythrough world, *Action-to-action closure* is very similar to *Moment-to-moment closure* in that there is a reliance on temporal order. If the RealityFlythrough user is paying attention to the timing of the images, he or she may be able to infer an action from, for example, a pair of images that show an empty stretcher and then a populated stretcher, and then use that information to get a better understanding about the space. *Action-to-action closure*, then, would be used when there is a greater temporal distance between images, but in order for it to work, it is crucial that the user have a good understanding about the temporal relationships between the images.

Subject-to-subject closure can happen in RealityFlythrough when the user is able to piece together understanding from a series of temporally disjointed images. Humans tend to use various spaces for very particular purposes, and even in a temporally disjointed world, at least a spatial (and sometimes even a temporal) story can be scripted. For example, a disaster scene may be partitioned into a hot zone and a cold zone with the hot zone indicating the area potentially contaminated with a hazardous substance. The hot zone may be further organized into a decontamination area and a triage area. By looking at a series of RealityFlythrough positioned images which may or may not be in

temporal order, it is possible to discover the segmentation of the scene into these areas, and then to generate a narrative based on that knowledge. An image of a woman on a stretcher logically extends to an imagined sequence of images that show the woman being carried to the triage area and then on to the decontamination zone, out of the hot zone, and finally loaded into an ambulance. All of this information can be inferred from a single image. For someone who has knowledge about what happens at a disaster scene, but no prior experience exploring this particular scene (either virtually with RealityFlythrough or on foot), the same narrative can be generated, but without the details of where these areas are and what paths and obstacles might be encountered along the way.

Scene-to-scene closure is necessary in RealityFlythrough when two scenes are not connected by a continuous sequence of images. The shortcut discussed in section 1.4.6 that cut off the right turn down the hallway created the need for *scene-to-scene* closure. The two hallways were different, and even though the motion indicated how the user got from one location to the other, there was no overlap in the images that could be used to commit closure. A user that has expert knowledge of the space, however, could use prior knowledge of the two scenes to easily commit closure. It is worth noting that expert users can probably get by without any RealityFlythrough assistance because the disjointed images that they see can be easily reconciled with their memory of the location. For example, most of us have expert knowledge of our homes and have no trouble accurately situating photographs that were taken within them.

Aspect-to-aspect closure is perhaps the most common form of closure committed in RealityFlythrough since it is responsible for connecting temporally disjointed images taken of the same scene. Since RealityFlythrough is mostly a tool for spatial (rather than temporal) exploration of a scene, a number of the transitions are going to fall into this category. In the comic book world, the aspects do not have to be even spatially related as long as they tie together a consistent theme. RealityFlythrough transitions will always be at least spatially related, and will thus share that same theme. When viewing a RealityFlythrough of a disaster scene, the user should probably expect

to see disaster response kinds of things, and anything that was not in this category would cause alarm. It would be more difficult to commit closure if there was suddenly a photo of Elmo interspersed. Indeed, users in one of our studies who were trying to make sense of jumps between seemingly spatially unrelated images (instead of transitions) seemed to take comfort in recognizing that the images were at least of the same scene. "I know we're in the same hallway. I just don't know where in that hallway."

There are also plenty of examples of basic *Gestalt closure* in RealityFlythrough. Anytime the images do not overlap and we show a black background with a floor grid in the missing areas, the user is employing *Gestalt closure* to fill in the incomplete information. This form of closure is not complete, of course, and critical information can be lost if it is located where imagery is missing, but for a tool whose primary aim is to provide increased spatial awareness, the gaps are usually very easily filled in. In figure 1.2, for example, you should have no trouble filling in the small gap that is shown in the third frame of the transition.

What is missing from McCloud's definition of *closure* and, in turn, our reliance on it as a tool for explaining how RealityFlythrough works, is that it does not explain how RealityFlythrough is any better than a collection of disjointed images (a photoalbum, if you will). Given a collection of images, the user would still be able to piece together an understanding of the scene by using all of the same closure techniques described above. It may be more difficult, yes, and it may take more time, but McCloud's closure would be employed in the same way. What is it about RealityFlythrough that makes this process of generating understanding easier and faster?

To answer this question, we will explore the world of cognitive film theory to see why it is that movies appear so real and to learn the tricks that film makers employ to enforce this illusion.

4.2 Cognitive Film Theory as an Explanatory Tool

We will use Anderson's book, *The Reality of Illusion, An Ecological Approach to Cognitive Film Theory*, as our guide into the world of Cognitive Film Theory [And96]. The goal of this investigation is to determine what qualities of RealityFlythrough transitions generate understanding at an unconscious level thus transforming the conscious closure tasks that were described above into confirmatory rather than explanatory tasks. In particular, we want to demonstrate that RealityFlythrough explorations model natural movement through space despite the imperfect stitching of images.

4.2.1 The Human Visual System

It is necessary to have a high-level understanding of how the visual system works in order to be able to understand the remainder of this section. What is of primary importance is realizing that the visual system is hierarchical with highly specialized modules responsible for detecting things like color, form, and motion. Presumably the visual system evolved in this fashion to provide animals with the best veridical description of the world that would allow them to accurately and efficiently "see" predators and prey. Efficiency, it turns out, is key because the full visual pipeline is relatively slow and the brain would not be able to detect and track fast moving objects without the shortcuts that evolved. These shortcuts are revealed and sometimes exploited by illusions, illusions that allow us to watch motion pictures, for example, as if they were real.

One such technique for overcoming the latency inherent in the visual system is the use of prediction. The visual system seems to predict where objects will be based on where they have been. In order to achieve this, the visual system follows some "rules", the same rules that are used to recognize the objects when they are observed at the predicted location. These rules are (1) an object does not simply disappear, and if it is in motion it stays in motion along a straight path, (2) an object is assumed to be rigid, and (3) an object in motion progressively covers and uncovers portions of the background [RA86]. All of these rules of motion operate at a fundamental level, well

before the scene enters consciousness, and before even the shape of the object has been classified [LH88].

The fact that these rules exist at all and the fact that they exist in the unconscious regions of the brain is critical to understanding how film makers are able to create the illusion of seamless motion and continuity of action.

4.2.2 Seamless Motion

Motion pictures are perceived as continuous motion instead of a sequence of discrete images because the motion processing that was described above, with all of its predictive abilities, appears to be the exact same motion processing that occurs when a movie plays. There is a difference between short-range apparent motion (objects that jump a few pixels per frame, for example) and long-range apparent motion (where objects travel a much greater distance). What is interesting about short-range apparent motion is that it is very similar to real motion, and the same very low level areas of the visual cortex seem to be engaged by both [Pet89, And96]. At the high frame rate (24 frames per second) that is used in motion pictures, the amount of motion that is observed between frames is in the short-range category and is thus indistinguishable from real motion. When frame rates drop, jumpyness becomes apparent most likely because the distances objects travel exceed the short-range threshold. The reason the low framerate defect manifests itself as a jump will be explained in the next section.

4.2.3 Jump Cuts

A cut is a splice of one camera shot into another without doing a cross-fade or some other transition. Cuts are extremely common in film, so common, in fact, that most viewers no longer notice them. There is one kind of cut, the jump cut, that viewers would notice, and film makers are careful to avoid them. A jump cut occurs when one of the following heuristics that film makers have stumbled upon is *not* followed: (1) the new scene has no resemblance to the previous scene, (2) the angle must be adjusted by at least 30 degrees and the image size must change, (3) the camera must stay on the same

side of a 180 degree arc so that the direction of action does not change, (4) the cut must be made at the point of greatest action, and (5) the action must overlap by approximately two frames.

These heuristics do not seem intuitive, and you may be wondering if you misread the previous sentence. A jump cut occurs when the heuristics are not followed, so that means that if the camera films a wide-angle view of a house from across a street, stops filming, crosses the street by moving directly ahead, and then resumes filming, heuristic number one will be violated and the house will appear to jump forward. Recall that rule one of vision processing states that objects persist. They do not go in and out of existence. Since the viewers have no evidence that they themselves have moved – proprioception is negative and there is no visual indication that the viewer moved – the only conclusion that the visual system can come to is that the object itself must have moved. This sudden movement of the house is jarring to the viewer and the film world describes the resultant jump as a jump cut.

4.2.4 Clean Cuts

We will now examine each of the heuristics that must be followed in order to obtain a clean cut, a cut that is nearly invisible to the viewer.

The first heuristic states that the new scene must have no resemblance to the old scene. This heuristic should be modified to state that there cannot be any similar blob-like objects in the neighboring frames. If there are similar blobs, the motion sensing part of the visual system, the magno system, will detect motion between the two shots when there should be none. If the blobs are not similar, no motion will be detected, the visual system will reset, and no jarring jumps will be seen.

The second heuristic states that the angle should be adjusted by at least 30 degrees and the image size should be different between the frames that straddle the cut. The angle change provides the visual system with enough information to make it realize that the viewer must have moved positions. The magno system that is responsible for motion detection also appears to use lines of perspective to discover depth informa-

tion [LH88]. If the second heuristic were followed in the framing of the second shot of the house in the example above, the viewer would not have sensed that the house jumped forward. The viewer's visual system would have correctly inferred that the viewer had moved forward and rotated to the right by some amount. Why would the visual system succumb to this illusion?

We primates have eyes that are in the front of our heads and we only see a very small portion of the world around us at any time. To increase our field of view, we look around. We gaze from side to side, look up and down, and move our bodies around the environment to see it from different angles. All the while, our visual system is observing everything that we see and remembering what it saw so that it can construct a model of the world around us. Our gaze will return to objects that we have seen before, but we may now be standing in a new location looking at the object from a different angle. Our visual system notes this and adds the new information to our world "map". The visual system apparently does not consider time or limits to the body's ability to move, and is quite satisfied to assume that we moved across the street and to the right in a fraction of a second. Note also that the human brain is fortunately not concerned about the lack of proprioception data to confirm the movement. If proprioception confirmation were also necessary, the illusion of motion pictures would not be possible.

The third heuristic states that the action should stay on the same side of a 180 degree arc. This heuristic is necessary to preserve the visual system's predictive power because recall that the visual system assumes that objects that are in motion continue in motion in a straight line. Objects do not suddenly reverse direction.

The fourth heuristic states that the cut should be made at the height of action. This seems non-intuitive, but recall that motion is processed before shape and color. The visual system is already sold on the cut before the rest of the brain even has a chance to see what happened.

The fifth heuristic states that approximately two frames should overlap between the cuts. This does not mean that there should be a cross fade for those two frames. Oddly enough, it means that the actual action should be repeated for those two

frames. For example, if a cut occurs right at the moment when a soccer player is heading a soccer ball, the splice should happen right when the ball bounces off of the head, and then the new zoomed in shot should show the ball bouncing off of the head again! The repetition of the action is necessary in order to make sure that the viewer does not miss any of the action. The cut, while nearly transparent to the conscious brain, is nonetheless startling to the visual system, and anytime the visual system is startled it masks the input on all stimuli except for the one that caught the attention of the visual system. This allows the brain to focus on the cause of this stimulus so that a quick determination can be made about whether to duck out of the way or turn and fight. The replay of two frames of action allows the viewer to see the action that was missed while her or she was startled. It is difficult to imagine that a visual system so startled is barely able to detect a splice into what is really a very different picture, but that is the magic of movies.

4.3 Why does RealityFlythrough Work?

What does this new insight into how the visual system works tell us about RealityFlythrough? Why is it that RealityFlythrough creates a more natural way to explore a space? What about it makes organizing a series of photos into an understanding of the space easier and faster?

The first advantage that RealityFlythrough has over the photoalbum approach to exploration is that it organizes photos spatially. Photos that neighbor one another spatially are grouped together. The experiment described in section 7.3 indicates that RealityFlythrough transitions are still more comprehendible than spatially grouped photos, however. The non-transitioned spatially grouped photos take more time to process.

Figure 4.2: A simple transition that is slightly misaligned.

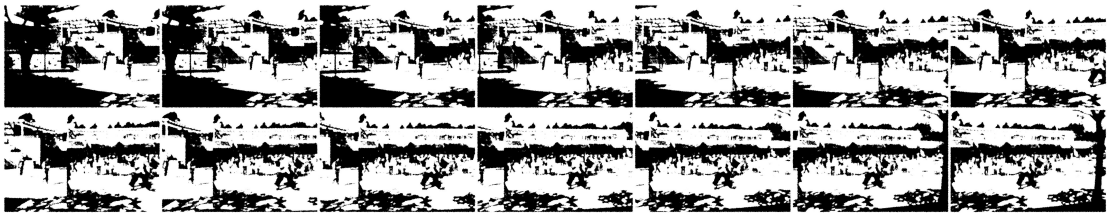


Figure 4.3: The same transition that is shown in figure 4.2, but processed through photoshop's torn edge *filter. This gives us a rough estimate of what the magno system may see when it is doing motion processing. With the detail lost, it is a little easier to imagine how the visual system may be fooled into not seeing the defects.*

The big win with RealityFlythrough is the motion that indicates the direction of travel. In this regard, RealityFlythrough is mimicking motion pictures. The motion is at a high enough frame rate that the motion is perceived as seamless, and when there is significant overlap of the appropriate kind, the scenes blend together without the user even being aware of the inaccurate stitching. What is the appropriate kind of overlap? As long as the blobs in the image are similar enough that the motion module of the visual system is sold on the transition, the user will be sold on the transition before he or she can even determine what the shapes are, let alone determine if the shapes overlap perfectly (see figure 4.3).

The automatic buy-in to a transition can be dangerous, however. The visual system can be fooled into perceiving a correspondence when there is none. Fortunately, though, the human brain is always able to use the many forms of closure to make sense of any strange inconsistencies that are detected. What RealityFlythrough transitions really appear to be doing is giving the user a head-start on understanding how a new image relates to the user's growing understanding of the scene. The user can then quickly tick through all of the various unconscious and conscious closure techniques to verify that what was observed ties in with the current understanding of the space. The transitions feed into the closure understanding and the information gleaned from closure feeds back into understanding the transitions. This feedback loop is one explanation for why users seem to get more comfortable with a RealityFlythrough space over time. It can also explain why a users' lack of expertise with either RealityFlythrough or a particular

RealityFlythrough space causes them to find the non-straightforward transitions jarring.

It would be interesting to study in the future how purposely incorrect transitions affect a user's understanding of the space. The hypothesis would be that the errant transitions would make it much more difficult to understand the space, but people who are really good at reading other closure cues would be able to eventually sort it all out. Depending on how powerful the RealityFlythrough cues actually are, it is doubtful that people would have an incorrect perception of the space. More likely they would just be confused and unable to make any sense of what they saw.

Another win for RealityFlythrough is the use of the cross-fade during transitions. We experimented with several types of fades before settling on a cross-fade of only the areas that overlap between images. The cross-fade serves the dual purpose of reinforcing the areas that do not change – because the cross-fade does not result in any visible change – while at the same time providing a visual cue that something strange is happening with the areas that do change. Recall that jump cuts should be avoided and these occur when the 30-degree heuristic is violated. We do not have any control over the camera locations in our transitions, so we probably frequently violate this heuristic. The motion of the camera makes the heuristic unnecessary, but occasionally the camera does not move when it should have (because of sensor inaccuracy) and this should cause a glaring jump. Cross-fades are one technique that motion pictures use to avoid jump cuts because they indicate to the visual system and to the user that something different is happening – the objects really are going out of existence. Our cross-fades serve the same purpose, but since the correctly overlapped areas remain solid they do not detract from the motion effects.

When dynamic objects in the scene disappear and re-appear during a transition – because they moved – the cross-fade provides the queue that something has happened and the long-range apparent motion system kicks in to complete the illusion. On occasion a dynamic object winds up in the foreground on a succession of images, and that dynamic object – a person being followed, for example – dominates the image and confuses all of the perspective and motion queues that would indicate that the camera is

moving as well. Even though the illusion breaks down, however, all of the other forms of closure are able to compensate for the somewhat ambiguous transition.

Figure 1.6 shows an example of this, and something that we failed to notice until now, despite watching this transition 100's of times, is that the sequence is not in temporal order. The sequence shows a transition between two different cameras, in fact, and the same policeman is coincidentally visible in both cameras even though the cameras filmed their respective scenes at different times. When studying figure 1.6 as a sequence of photographs, it is difficult to make sense of the policeman's motion until you realize that there are two cameras and two different times involved. The first image on the second row shows the transition from one camera to the next. In one instant the policeman is standing directly in front of the camera and then in the next he is down the hallway.

Expert users learn to ignore the dynamic content during transitions and focus instead on the stable content that better informs the spatial relationships. It would be interesting to do a future study on how the practices differ between novice and expert users. Would novice users notice the policeman anomaly?

4.4 Conclusion

The claim that RealityFlythrough employs closure to provide the compelling illusion of movement through space is accurate, but not complete. There are many different forms of closure, and all of them contribute in some way to the understanding of a RealityFlythrough space, but it is the naturalness of the RealityFlythrough experience that informs closure, a naturalness that can only be explained with a good understanding of how our visual systems function. Cognitive Film Theory provided the tools we needed.

Chapter 5

The Smart Camera

Note that this chapter is a reprint with minor changes of *A Robust Abstraction for First-Person Video Streaming: Techniques, Applications, and Experiments* [MGL06], a paper co-authored by Neil McCurdy, William Griswold, and Leslie Lenert. This chapter describes the *Smart Camera* and demonstrates how RealityFlythrough transitions can be used to augment low frame-rate first-person video. Point-matching concepts are introduced as well, but the *Composite Camera* and the integration of the *Smart Camera* into RealityFlythrough proper will be discussed in chapter 6.

The emergence of personal mobile computing and ubiquitous wireless networks enables powerful field applications of video streaming, such as vision-enabled command centers for hazardous materials response. However, experience has repeatedly demonstrated both the fragility of the wireless networks and the insatiable demand for higher resolution and more video streams. In the wild, even the best streaming video mechanisms result in low-resolution, low-frame-rate video, in part because the motion of first-person mobile video (e.g., via a head-mounted camera) decimates temporal (inter-frame) compression.

We introduce a visualization technique for displaying low-bit-rate first-person video that maintains the benefits of high resolution, while minimizing the problems typically associated with low frame rates. This technique has the unexpected benefit of eliminating the "Blaire Witch Project" effect – the nausea-inducing jumpiness typical of

first-person video. We explore the features and benefits of the technique through both a field study involving hazardous waste disposal and a lab study of side-by-side comparisons with alternate methods. The technique was praised as a possible command center tool, and some of the participants in the lab study preferred our low-bitrate encoding technique to the full-frame, high resolution video that was used as a control.

5.1 Introduction

The emergence of personal mobile computing and ubiquitous wireless networks allows for remote observation in uncontrolled settings. Remote observation is powerful in situations in which it is not possible or too dangerous for an observer to be present at the activity of interest. These include coverage of breaking news, emergency response, or grandparents joining the grandchildren on a trip to the zoo. The application investigated in this paper is video-support for a supervisor overseeing hazardous materials disposal.

Despite incredible advances in wireless networking and the mobile devices connected by it, our repeated experience is that wireless networks in uncontrolled settings are fragile, and there is seemingly unlimited demand for more video streams at higher resolution. Modern video streaming techniques heavily depend on temporal (inter-frame) compression to achieve higher frame rates, while minimizing the impact on resolution when operating at the network's capacity. Unfortunately, the panning motions common to first-person mobile video (captured from a headcam, say) virtually eliminates inter-frame compression. To stay within the available bandwidth, either the frame rate or the resolution must be reduced. In applications like hazardous materials disposal, image resolution cannot be sacrificed, making a low-frame-rate encoding the only viable option. Ironically, lower frame rates further reduce the likely overlap between frames, further reducing inter-frame compression.

The problem with low-frame-rate video is that a one-second interval between frames is long enough to disorient the viewer. This is especially true with head-mounted

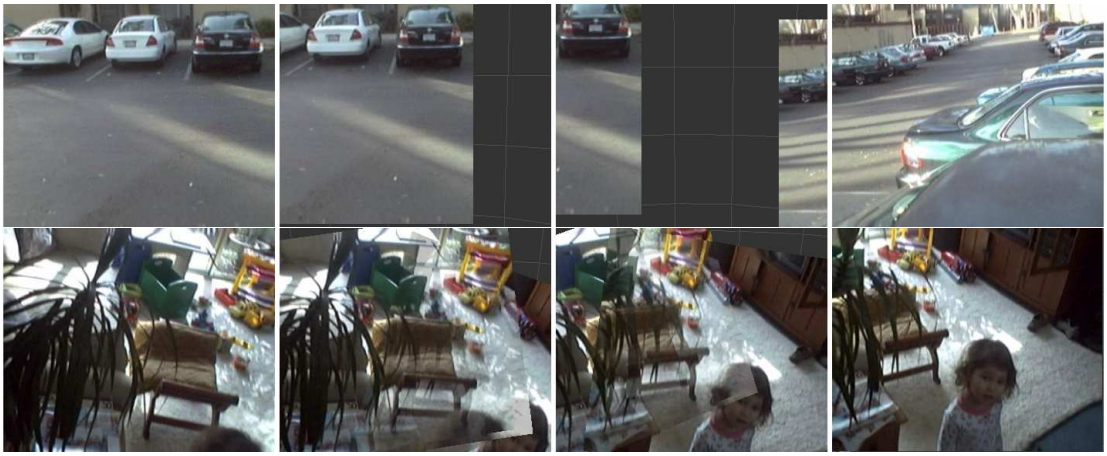


Figure 5.1: Snapshots of two transitions in progress. The top row depicts a camera pan from left to right where the frames do not overlap. The bottom is a morph from the frame on the left to the frame on the right as the camera pans down and to the right to look at the child. The live experience is one of smooth camera movement.

cameras because it may only take a fraction of a second for the view to rotate 180 degrees. With little or no overlap between successive frames, the viewer lacks the information required to understand how the frames relate to one another. Even in a relatively unchanging outdoor environment where there is a large field of view, a viewer can become disoriented looking at the camera's view of the ground when the camera operator looks down to avoid obstacles.

In this paper, we present a visualization technique that minimizes the confusion caused by low-frame-rate video, using modest hardware and processing. If the orientation of the camera is known – either by attaching tilt sensors and an electronic compass to the cameras, or by using an online vision processing algorithm on the cameras – we can generate a visualization that shows the viewer how consecutive frames relate to one another. The visualization takes the form of a dynamic *transition* similar to those described for switching between two streaming cameras located in the same environment [MG05c]. A transition (Fig. 5.1) has two components: movement (rotation) of the viewing portal from one frame to the next, and a gradual alpha-blend between the overlapping portions of the frames. If the frames do not overlap at all, the current frame rotates off of the screen, a spherical grid (as viewed from the center of the sphere)

continues to show the degree and direction of orientation, and finally the next frame rotates onto the screen. The net effect is a high-frame-rate interpolation of the camera's motion between the frames. These transitions intuitively convey the relative positions of the frames, and no users in our user study reported anything more than occasional temporary confusion when watching long sequences of these transitions. Due to the visual nature of this work, we encourage the reader to view short video clips of transitions downloadable from the web (`http://ubivideos.homeip.net`).

No knowledge of the camera's position is required, unlike the previous work involving inter-camera transitions [MG05c]. The assumption is that the amount of positional change in the interval between two frames is not significant, and the results of our user studies confirms this. Even without the explicit representation of position, however, the viewers still have a sense of movement through the environment. Not only is there the illusion of movement similar to the illusion experienced when watching any sequence of frames, but there is real movement as well. The manner in which we align the subsequent frames when there is frame overlap, and the transition between the frames, creates the sensation of movement. At times the alignment will cause the entering frame to start off smaller than it really is, and then grow in size (zoom in) until it fills the screen. This zooming creates the appropriate sensation of moving forward (or conversely, backward) through the environment.

We explore the features of this approach in part with a field study of a hazardous materials (hazmat) supervisor remotely monitoring a live video feed – transmitted over a "broadband" cellular network – of two hazmat workers disposing of hazardous chemicals. The camera was mounted on the mask of one of his team members. Such a system configuration is motivated by a response in a damaged and chaotic environment. The supervisor's impressions of our visualization technique were surprisingly favorable, and he dismissed the alternative encodings that were available. The unmodified low-frame rate video left him feeling disoriented, and the low-quality 5fps (frames-per-second) video was so choppy and disorienting that it interfered with his thinking and made him nauseated.

We explore the finer distinctions among the various approaches to low-frame-rate video with a laboratory study in which 14 subjects were asked to view video clips of three different scenes that were encoded in four different ways. A surprising result of this study is that four of the subjects actually preferred watching our 1fps transition-enhanced video over full-frame (12fps), high quality video. Nearly all of the participants preferred our visualization to the 5fps video clip that was encoded at a comparable bitrate. One further interesting result is that nearly all of the participants were unable to discern the difference between a clip that performed a simple alignment and blending between frames, and one that also performed a morph between the frames to produce more seamless transitions. This result can be explained by the brain's ability to commit *closure* with minimal cognitive load when modest amounts of visual information are missing [McC93].

The remainder of this paper is organized as follows: In section 5.2 we motivate the use of video in a disaster response setting, and describe the constraints that such an environment places on technical solutions. In section 5.3 we describe our solution, and in section 5.4 we discuss related work. Sections 5.5 and 5.6 present our field and lab studies.

5.2 Motivation

There are many situations in which high-panning low-bit-rate video can have value. Consider, for example, CNN coverage of hurricanes or remote war-torn areas where CNN resorts to satellite-phone video segments. These feeds are tolerable for the talking-head shots, but panning of the surrounding environment to show viewers what is happening results in a dissatisfying choppy, grainy image. There are also man-on-the-street news reporting scenarios where it might be desirable to look at low-bitrate video. Breaking news such as an accident, prior to the arrival of traditional television cameras, could be viewed through citizen cameras with feeds transmitted over cellular networks, or overlooked news could be streamed direct to the internet by citizen mobile phones.

Hazardous Materials Cleanup. The use of video during the early stages of a disaster response, or even during the late stages of a chemical clean-up are scenarios that can be impacted today. This paper focuses on this latter scenario, and we have used the requirements of a hazardous materials (hazmat) supervisor as the requirements for our video streaming solution. We consulted with the Hazardous Materials Business Plan Manager (hereafter referred to as Tod) at the University of California, San Diego (UCSD) to determine how live video might be used at a hazmat scene.

As a supervisor, it is Tod's job to know what is going on, to interface with the various entities on scene (such as fire fighters, witnesses, and lab managers), and to supervise the stabilization and cleanup of the environment. Live video feeds from the scene would help Tod assess the health and safety of his team, aid in identifying hazards, and allow experts outside of the hotzone to assist with operations.

Networking Challenges. The significant radio interference at a disaster scene (both natural and man-made) wreak havoc with communication. In contrast to existing hazmat video transmission systems which typically use analog signals, we have decided to use a digital signal for a number of reasons. First, in the larger-scaled deployment of our parent system, RealityFlythrough, we are piggy-backing on a state-of-the-art wireless mesh network [Ari05] that is deployed by first-responders to support the coordination of medical treatment for victims. Second, the varying conditions of the network caused by radio noise can be better managed in the digital domain. Frame rates can be throttled and image quality can be degraded in a controlled manner. Most importantly, we can guarantee eventual delivery of error-free images (with a very high latency) when conditions are so bad that only a small amount of data can trickle through the network. And third, we can use the same bandwidth managing techniques to support multiple cameras.

Radio interference in a digital mesh network results in frequent disconnects and low throughput. 802.11b has an expected bitrate of 6.4Kbps, but noise, overhead introduced by the mesh network, and the many other clients competing for bandwidth have reduced the effective bandwidth to 100Kbps for each camera in a typical 3-camera

deployment. Similar conditions are found in an alternate deployment scenario which uses a cellular network instead. In this case, immature technology is the main source of fragility.

Video Compression Challenges. The conditions that have been outlined so far present a significant challenge for video compression. The video stream produced by a head-mounted camera is typically high-panning due to the natural head movements of the wearer. High-panning video usually has very little redundancy between frames, rendering traditional codecs that rely on temporal redundancy ineffective. With low temporal redundancy in the video input, most codecs do little better than motion JPEG (MJPEG) which simply performs spatial compression on each frame in the video sequence.

In the heavily constrained networks described above, where the frame rate must be reduced to maintain image quality, the increased interval between frames further reduces temporal redundancy, minimizing the bitrate savings of the decreased frame rate. The result is a heavily decimated frame rate. As the frame rate drops, it becomes difficult to track objects, and eventually it is even difficult to orient yourself in the scene.

Traditionally, the only option at this point has been to reduce the image quality to increase the frame rate to non-disorienting levels. Our approach preserves image quality while mitigating the negative effects of low frame rates.

5.3 Our Approach

To reduce the disorienting effects of low-frame-rate video, our concept is to perform a dynamic visual interpolation between frames using meta data captured from a digital pan/tilt compass or inferred using vision techniques. In particular, we align the frames in a spatially consistent way in a 3d graphics environment, and then use rotational and translational motion to segue between the frames, producing a high-frame-rate experience that captures the effects of camera motion. Because precise frame stitching is impossible in real-time using 2D data, we use a dynamic crossover alpha-blend to help

the viewer correlate the information in the overlapping parts of the frames.

An imperfect alignment between two frames, due to, say, inaccurate sensor readings is less of an issue than might be expected. *Closure* is a property of the human visual system that describes the brain's ability to fill in gaps when given incomplete information [McC93]. It is a constant in our lives; closure, for example, conceals from us the blind spots that are present in all of our eyes. So while there is ghosting, and maybe even significant misregistration between frames, the human brain easily resolves these ambiguities.

The rest of this section describes the details of our approach.

Creating a Panoramic Effect. Our approach can be described as the creation of a dynamically changing and continually resetting spherical panorama. Each incoming frame is positioned on the panorama, and projected onto a plane that is tangential to the sphere to avoid distortion. A dynamic *transition* then moves the user's viewpoint from the current position within the panorama to the incoming frame's position (Fig. 5.1, bottom). The user's viewport has the same field of view as the source camera, so the frame fills the entire window once the transition is complete. Movement between frames looks like smooth camera panning. There may also be a translational (shifting) motion effect if the camera moves forward or backward through the scene.

A new panorama is started when consecutive frames do not overlap (Fig. 5.1, top). The frames are positioned at their relative locations on the sphere, with an appropriate gap between them. To help the user stay oriented, a wireframe of the sphere that serves as the projective surface is displayed. Horizontal and vertical rotations are thus easily recognized. The grid wireframe could be further enhanced by including markers for the equator and the cardinal directions.

The planar simplification of 3d space only works for a short interval when cameras are mobile. For this reason, at most five frames are placed in a given panorama. The oldest frame is discarded when this limit is reached. This is not a significant compromise because the source and target frames of a transition mostly fill the viewport, and any other frames in the panorama are filling in around these two.

Frame placement in the panorama is managed through a robust two-level scheme, as described in the rest of this section.

Image-based Frame Placement. When inter-frame rotations are not too large, we use an implementation of Lowe's SIFT algorithm [Low04] to find matching points between a new frame and the previous frame, and then do a best-fit alignment of the frames to fit the new frame into the panorama. The point-matching is performed on the camera units in real-time, and the list of matched points between the current frame and the previous frame are transmitted with each frame. In order to perform the matches in real-time on our camera devices, the frame is downsampled to a quarter resolution (QCIF instead of CIF) prior to analysis by SIFT. The result is good even at this lower resolution.

Each new frame is aligned to the previous frame by determining an affine correspondence between the frames. We look at the relative position, orientation, and zooming based on the two matching points in each frame that are furthest apart. After aligning the new frame, the frame is warped so that the matching points are exactly aligned. Surrounding points are warped by an amount proportional to the inverse of the distance to the neighboring control points. A transition to this new frame thus involves a morph as well as the standard rotation and alpha-blend. At the end of the transition, the new frame will be unwarped, and all of the other frames will be rotated and warped to match the control points in the new frame.

Even with point matching, the alignment is not fully precise. Our planar simplification of 3d space makes objects in the scene that are at depths different to those of the points that have been matched be less accurately aligned. Even if the depths of the matching points were recovered, the number of matching points (10-20) is very small relative to the number of objects and object depths in the scene, so any recovered geometry would be coarse. Also, since we are operating in real environments, dynamic objects that move between frames will not have any point correspondences, and thus will not be accurately aligned. Nonetheless, closure helps this technique produce very pleasing results.

Sensor-based Frame Placement. When SIFT fails to produce matching points for a new frame, the frame's placement depends on sensor data gathered from the camera rig. The camera units we use are integrated with tilt sensors and electronic compasses that record the tilt, roll, and yaw of the cameras at 15hz. This information allows us to position the frames on the sphere. However, the sensor accuracy is not good enough for generating a multi-frame panorama. Thus, the placement of such a frame initiates a new panorama with the single frame. The rotational part of the transition is still performed with the dynamic alpha-blend, using the previous frame's and new frame's relative sensor data. However, since we do not have information about the relative or absolute locations of the frames, we are unable to determine the relative translational positioning between frames. The resulting experience mitigates the confusion caused by low frame-rate video, but often lacks the aesthetics of the panorama and higher precision placement.

5.4 Related Work

We are not aware of any related work that directly addresses the conditions we have set out to handle in this chapter, but there is some work that handles subsets of these problems.

RealityFlythrough, which provides ubiquitous video support for multiple mobile cameras in an environment, uses visualization techniques similar to the one we propose in this chapter, but for inter-camera transitions [MG05c]. It requires knowledge of the positions of the cameras, as well as the orientations, limiting its use to environments where ubiquitous location sensors are available, such as outdoors.

Irani, et al. directly address the problem of encoding panning video [IAB+96]. They construct a photo mosaic of the scene, and are then able to efficiently encode new frames by using the difference between the frame and the mosaic. With this technique, it no longer matters if consecutive frames have much overlap because the assumption is that similar frames have overlapped enough in the past to construct the mosaic. Unfor-

tunately, mosaic-based compression cannot be used in our scenario because our cameras are mobile. Mosaic-based compression works well as long as the camera remains relatively static and pans back and forth over the same scene, but if the camera moves through the scene, there will be little opportunity to find matches with previous images. Essentially mosaic-based compression extends the search window for similar frames. If there are only a few similar frames, it does not matter how big the search window is, as there will rarely be a match.

There are many examples of codecs that are designed to work in wireless, low-bit-rate environments, although these codecs generally rely on the significant temporal compression that is possible in "talking-head" video. H.264 [WSBL03] (also known as MPEG4-10) represents the current state-of-the-art. When compressing first-person-video at low bit-rates, though, there is little perceptible difference between H.264 and the more common MPEG4-2 [mpe98] (commonly referred to simply as MPEG4). This is not surprising considering the low temporal redundancy.

A non-traditional approach to video compression proposed by Komogortsev, varies the quality of the video based on where the viewer is looking [KK04]. By using eye-gaze-trackers on the viewer, and predicting where the viewer will look next, the overall image quality can be low, but the perceived quality would be high. This approach would be difficult to implement in our scenario because the network latency is so high (4-5 seconds) that the gaze direction would have to predicted far in advance.

There has been substantial work on generating panoramas from still photographs [Sze94, BL03]. Real-time dynamic creation of panaoramas on a handheld camera device has been used to help with the creation of a static panorama [BTS+05]. Panoramas can also be efficiently created from movie cameras assuming the camera's position is relatively static [SPS05]. All of these techniques require some way to match points between images. We rely heavily on Lowe's SIFT algorithm [Low04], specifically the Autopano implementation of it (`http://http://autopano.kolor.com/`).

A technique to remove distortions during image morphs is described by Seitz

and Dyer [SD96]. This technique produces natural morphs, but it requires manual user intervention with each morph and is thus not applicable in our scenario. In practice, our technique rarely produces morphs that might cause disorientation, so morphing improvements would only be an aesthetic luxury.

5.5 Hazmat Field Study

We had several goals for our field study. First, we wanted to know if our visualization technique was suitable for a hazmat command center. Second, we wanted to see if our system could work in a realistic environment for an extended period of time. And third, we wanted to discover the motion model of a head-mounted camera afixed to someone doing a real job, oblivious to the presence of the camera.

5.5.1 Experimental Setup

The Scene. Every week, two members of the UCSD hazmat team perform a maintenance task that doubles as an training exercise for response to an accident. All of the hazardous waste that has been collected from labs around the university is sorted, and combined into large drums in a process that is called *bulking* of solvents. This task serves as an exercise, as well, because full hazmat gear must be worn during the procedure, giving the team members (we will call them *bulkers*) experience putting on, wearing, and performing labor-intensive tasks in gear that they will use at an incident site.

The Equipment. It was important to make the camera system as wearable and unobtrusive as possible, given our desire to discover the real motion models of the camera.

We attached a disassembled Logitech webcam ($\sim$\$100) to the front of the mask, and sewed a tilt sensor manufactured by AOSI ($\sim$\$600) into the netting of the mask that rested on the top of the head. These devices connected to a Sony Vaio U71P handtop computer ($\sim$\$2000) which was placed in a small backpack. Tod, the team

leader introduced in section 5.2, insists that the bulking experience be a replica of real-world hazmat scenarios, so we chose to transmit the video across the Verizon 1xEVDO network, which might be the only readily available network if, say, a burnt out lab were being cleaned up. The video feed was transmitted to our server, a standard VAIO laptop (FS-790P ∼$1600) connected via 802.11 to the campus network.

The 1xEVDO upstream bitrate was measured at between 60 and 79Kbps, and the campus downstream bitrate at 3.71Mbps. We fixed the frame rate of the video feed to .5fps to ensure that we would stay within the range of the 1xEVDO upstream speed.

The Task. We had Tod use the video that was being transmitted by one of his bulkers to explain to us the bulking process. This think-aloud interaction is realistic in that Tod needs to train others in how to conduct his task for times when he is on vacation or out sick. For us, this interaction served several purposes: (1) It would give Tod a reason to be viewing the video, (2) it would encourage him to verbalize his impressions of the system, and, (3) it would allow us to observe the effectiveness of the video stream as a communicative device. Did the video provide enough detail to help illustrate what he was describing, and at a fundamental level, did he understand what was going on?

As an expert, Tod's subjective opinion of the system was important to us. The requirements are domain specific, and only someone who has experience operating in a command center can know if the quality of the video is appropriate for the task.

5.5.2 Results

Our camera system was worn by one bulker for the entire exercise which lasted for roughly 64 minutes. The bulking task was very demanding for our visualization system because the close quarters of the bulking environment limit the field of view, and the heavy physical activity creates drastic camera pans from ground to horizon.

Despite these challenges, Tod reacted favorably to the visualization. He was oriented immediately: "Ok, so this is following Sam as he's moving around the room. And as you can see they have a lot of work ahead of them." The image quality was good enough for him to identify the characteristics of chemicals: "This tells me a little bit

about the viscosity. I can see the liquids, whether they're plugging up. Sometimes you get some chunks in there. And the thicker stuff – gels – looks like we had a little bit in there..."

Tod expressed an interest in flipping through the individual frames so that we could really study the pictures. We showed him how he could pause the feed and move back and forth through the images while still getting the benefit of the visualization. The visualizations helped us stay oriented as he was describing the process, and saved him from having to explain the relative positions of the images.

We then discussed how our visualization compared to the normal view of low frame-rate data which at these speeds looked more like a sequence of still photos. "Literally for me, at the moment I would just go full screen on this particular moving one (our visualization)... I'm not really even paying attention to this one (the low frame-rate stream). The individual photos clicking through. I could be disoriented with that one... It would tell me that they're moving around, but after that it's not giving me anything that I really need for decisions."

Tod concluded with his assessment of the system: "Let me tell you what I like about it. It's not overwhelming. It's appropriate. It's not a huge distraction. That's one of the things you have to be concerned about – the level of distraction.... Yeah, I think you got it."

Tod also had recommendations for improvement: he would like to have multiple cameras so that he could see the scene from multiple angles, he requested wider-angle lenses, and he wondered if he could set up fixed cameras as well.

5.5.3 Followup

During the study we were of course unable to show Tod other possible encodings of the data. Thus, we returned a few days after the experiment and presented him with a re-creation of the experiment with 5fps video encoded at bitrates comparable to the original experiment, using FFMPEG's MPEG4 codec (`http://ffmpeg.sourceforge.net`). His reaction surprised us because we assumed that the neces-

sary drop in video quality (resolution) would make the video unusable for hazmat: "I don't have a problem with the resolution on the right (the 5fps video), but it's almost flipping through so fast that you're not orienting yourself to what's going on... Yeah, I like the slower frame rate. It's not so much because of the resolution, it's the amount of time that it takes me to know what I'm looking at... [The 5fps video] is snapping too fast – it's too busy – it interferes with my thinking, literally, it's messing with my head."

Even after showing him the high quality 6.67fps feed that had been captured directly at the camera, Tod still thought our abstraction was more appropriate for a command center considering everything else that is going on. A command center needs to maintain a sense of calm [Wei95]. "This is just one piece of information that you're going to be getting. The phone is going to be ringing, people are going to be giving you status reports. The [higher frame-rate video] is just too busy."

5.6 Lab Study

Intrigued by Tod's observations during the field study that our visualization method may actually be *more* pleasurable to watch than high fidelity first-person video, we increased the scope of our planned lab study to determine if our visualization method would have broader appeal. Might it actually be an alternative to the sometimes nauseating, "Blair Witch Project" [MS99] quality of first-person video? Tod's outright rejection of the disorienting high quality, low frame-rate video feed was evidence enough that that encoding was no longer a viable candidate, allowing us remove it from consideration in our lab study, and focus on this potentially stronger result.

We were interested in uncovering people's subjective reaction to different encodings of first-person-video. Very simply, *Do you like it or not?* We wished to divorce the content and any perceived task from the judgments. It is easy to conceive of tasks that make any of the encodings succeed or fail, so a task-oriented evaluation would reveal nothing. Instead, we had to impress upon the subjects that it was the quality of the video that they were judging, and assure them that it was okay for the judgment to be

purely subjective and even instinctive. The scenarios that were viewed and the questions that were asked were designed to achieve this.

5.6.1 Experiment Setup

We recorded three 2-3 minute first-person video segments using a camera setup similar to the one described in the previous section. The first was a video of a trip through the grocery store (representing a crowded environment), the second was video of someone making breakfast for the kids (representing an indoor home environment), and the third was video of someone taking out the garbage (representing an outdoor scene). The goal was to make the camera motion and activities as natural as possible.

The three videos were then encoded in four different ways. *encFast (eF)* was sampled at 1fps and run through our visualization system. *encSlow (eS)* was similar, but sampled at .67fps. *encIdeal (eI)* was the "ideal" version, encoded at roughly 11fps (the fastest our camera system could record raw video frames) with an infinite bitrate budget. And *encChoppy (eC)* was encoded at 5fps at a comparable bitrate to the corresponding *encFast*.

The subjects were asked to watch all of the video clips in whatever order they desired, and were encouraged to do side-by-side comparisons. They had complete playback control (pause, rewind, etc.). The following questions were given to the subjects prior to the start of the experiment, and answers were solicited throughout.

What is your gut reaction? Rank the video feeds in order of preference. Describe the characteristics of each of the video clips. Why do you like it? Why don't you like it? If it was your job to watch one of these clips all day long, and there was no specific task involved, which would you choose? Why? Do any of these clips cause you physical discomfort? Which ones? Do any of the clips create confusion? If so, is it temporary or perpetual? Discounting the content, how do each of the clips make you feel? Have your preferences changed?

These questions were designed primarily to encourage the subjects to think

critically about each of the clips. Obtaining a carefully considered ranking of the clips was our main goal. However, the responses would also help shed light on the underlying reasoning.

Table 5.1: Summary of results. # *is the subject number,* S *is the sex,* A *is the age, and* G *indicates if the subject had any 1st-person-shooter game experience.* Initial Pref *is the gut reaction ranking given to each of the encodings, and* Final Pref *is the final ranking. References to our encodings appear in bold.*

#	*S*	*A*	*G*	*Initial Pref*	*Final Pref*
1	*M*	*20*	*T*	*eI, **eF**, **eS**, eC*	*eI, **eF**, **eS**, eC*
2	*F*	*60*	*F*	*eI, **eS**, **eF**, eC*	*eI, **eS**, **eF**, eC*
3	*M*	*40*	*F*	*eI, **eF**, **eS**, eC*	*__eS__, __eF__, eI, eC*
4	*F*	*40*	*F*	*__eF__, __eS__, eI, eC*	*__eF__, __eS__, eI, eC*
5	*F*	*60*	*F*	*eI, **eS**, **eF**, eC*	*eI, **eS**, **eF**, eC*
6	*M*	*30*	*T*	*eI, eC, **eF**, **eS***	*eI, **eF**, eC, **eS***
7	*F*	*30*	*T*	*eI, eC, **eF**, **eS***	*eI, **eF**, **eS**, eC*
8	*M*	*30*	*F*	*eI, **eS**, eC, **eF***	*eI, **eS**, eC, **eF***
9	*M*	*20*	*F*	*__eS__, eI, __eF__, eC*	*__eS__, eI, __eF__, eC*
10	*M*	*30*	*T*	*eI, eC, **eF**, **eS***	*eI, eC, **eF**, **eS***
11	*F*	*30*	*F*	*__eS__, __eF__, eI, eC*	*__eS__, __eF__, eI, eC*
12	*M*	*30*	*T*	*eI, **eF**, eC, **eS***	*eI, **eF**, **eS**, eC*
13	*M*	*20*	*T*	*eC, **eF**, **eS**, eI*	*eC, **eF**, **eS**, eI*
14	*M*	*60*	*F*	*eI, **eF**, **eS**, eC*	*eI, **eF**, **eS**, eC*

5.6.2 Results

The following summarizes the data found in Table 5.1. 14 subjects participated in this study, 10 male, and 4 female, ranging in age from 20 to 60. All but two of the subjects preferred at least one of our encodings to the choppy encoding, and 4 of the subjects actually preferred our encodings to the ideal encoding that was used as a control. 6 of the subjects preferred eS to eF, and in all of these cases the preference was very strong. None of these 6 subjects had first-person-shooter game experience. 4 of the subjects changed their ranking of the encodings midway through the experiment, and in all cases our encodings were ranked higher.

5.6.3 Analysis

Our encodings faired much better than expected. Not only did 4 of the subjects rank them higher than eI, there were also 4 others who were explicitly on the fence, and saw definite benefits to our encodings. Our encodings also seemed to grow on people. 4 of the subjects changed their rankings towards the end of the experiment, moving our encodings higher in preference. Everyone in the study liked our encodings, regardless of how they ranked them. The following is a sampling of the positive qualities voiced by our subjects: *calm, smooth, slow-motion, sharp, artistic, soft, not-so-dizzy*. There were of course some negative characterizations, too: *herkey-jerkey, artificial, makes me feel detached, insecure*.

The clearest pattern was the subjects' dislike of eC. We will discuss the two exceptions to this a little later. Most stopped paying attention to eC early in the experiment because the quality, to them, was obviously much poorer.

Many of the subjects had a strong personal criterion that they used for judging the videos. For some, it was clarity of the images and for others it was the lack of choppiness. There were also those who were most influenced by nausea.

The subjects in the clarity camp (subjects 2, 5, and 14) were interesting because despite the clarity of the images in eF and eS, they were bothered by the momentary blurriness during the transitions. As a result, they preferred eI even though some of them indicated that the speed of the video was too fast. The clarity camp may have responded better to transitions that do not do alpha-blends.

The slightly longer interval between frames in eS made all of the difference for some. Subject 8 actually ranked eF the lowest because it was just too "herky-jerky". Subject 9 liked eS the best, but ranked eF below eI. eF "had a jolting, motion sickness feel." Others, on the other hand, had strong negative reactions to eS, because it was too slow and boring. There appears to be a strong correlation between an individual's lack of first-person-shooter game experience and their preference for eS. None of the subjects who preferred eS had any game experience. First-person-video is not something that people get a lot of experience watching, unless they play first-person-shooter games.

With more experience, people may actually prefer the speed of eF.

This study helped explain why first-person-video can be so difficult to watch. It comes down to control and expectation. We are not bothered by our own first-person view of the world because we are controlling where we look. We can anticipate what the motion is going to feel like, and we know what to expect when the motion stops. When watching something through another person's eyes, however, that expectation is lost, so we are always playing catch-up. Subject 4 preferred our encodings over eI precisely for this reason. She said that eI was moving so fast that she could not pick up any of the details. Just as she was about to focus on the current scene to comprehend it, the view moved to something else. She liked that eF gave her the extra time to actually absorb what was going on.

Subjects 10 and 13 were the only ones who preferred eC over our encodings. Their reasons were quite different so we will consider them independently. Subject 10 simply preferred traditional video to our encodings. He could see the value in our encodings, and was not confused by them, but he felt detached watching them. One thing that bothered him in particular about them was that he did not feel they were playing at a consistent frame rate. They were, but because the "transition" time eats into the viewing time, the amount of time the image fills the screen and is completely in focus may appear to vary slightly. For example, the transitions always take a fixed .5 seconds, but if the images almost overlap exactly, and the blending between the images is barely noticeable, it will look like the image is in full view for 1 second (for eF). If, on the other hand, the *transition* rotates to an image that was off-screen, .5 seconds has already elapsed before the image is in view, and the image will only appear to be displayed for .5 seconds. We may want to consider having the images be displayed for a consistent time, and averaging the *transition* time across multiple transitions so that we keep up with the frame rate. This may create a more natural feel.

Subject 13 is an interesting outlier. Not only did he rank eC the highest, but he ranked eI the lowest! In a post-experiment interview we learned that he preferred the artistic quality of eC. It was edgy. He was bored by eI and found it a little bit nauseating.

He also liked the artistic feel of our encodings, but ultimately the "predator" feel of eC is what drew him towards that one. Clearly, there is no one solution that would appeal to everyone.

5.6.4 Secondary Study

A secondary goal of the lab study was to determine if the morphing performed during transitions was helpful. As described in section 5.3, morphing can create better alignments between the images by making all matching points overlap exactly throughout a transition. It was not clear to us that the morphing was providing much benefit, and when the vision algorithm occasionally returned incorrect matching points, the morph looked startlingly bad.

The surprising result was that none of the subjects could discern any difference between *morphed* and *non-morphed* video clips when played side-by-side. It was only when the video was slowed down by a factor of eight that some of the subjects noticed a difference, although even then they only expressed a vague preference for one over the other. Stop-motion convinced all of the subjects that the alignment between images was indeed better in the *morphed* version.

We hypothesize two explanations for this result. (1) Our brains are so good at committing closure that unless there is perfect alignment between images, varying degrees of misalignment (to a point) are perceived as being the same. There are times when closure is being performed consciously, but for the most part this is a process that happens unconsciously, and people are only vaguely aware of it happening. (2) The dynamic content is what is interesting in a scene – the very content that does not get morphed because matching points cannot be found.

5.7 Conclusion

We have presented a visualization technique for displaying low-bit-rate first-person video that maintains the benefits of high resolution, while minimizing the prob-

lems typically associated with low frame rates. The visualization is achieved by performing a dynamic visual interpolation between frames using meta data captured from a digital pan/tilt compass or inferred using vision techniques. We have demonstrated with a field study that this technique is appropriate in a command center, in contrast with traditional low-bitrate encodings which may cause disorientation and physical discomfort. Our lab study confirmed that our visualization has wider appeal and may have application in many other contexts. 12 of our 14 subjects preferred our visualization to the current state-of-the-art given comparable bitrate budgets. The surprising result is that 4 of the subjects actually preferred our visualization to the high frame-rate, high quality video that was used as a control. These results are based on subjective preferences across three different domains, and are thus untainted by task-specific evaluations that would limit the generality of our findings.

5.8 Acknowledgments

Special thanks to Tod Ferguson and the UCSD Hazmat team. This work was supported in part by contract N01-LM-3-3511 from the National Library of Medicine and a hardware gift from Microsoft Research.

This chapter is, in part, a reprint of the material as it appears in A robust abstraction for first-person video streaming: Techniques, applications, and experiments. Neil J. McCurdy, William G. Griswold, and Leslie A. Lenert. In ISM 06: Proceedings of the Eighth IEEE International Symposium on Multimedia, pages 235-244, Washington, DC, USA, 2006. IEEE Computer Society. The dissertation author was the primary investigator and author of this paper.

Chapter 6

Putting it All Together

This chapter ties together the concepts discussed in chapters 3 and 5. It explores how the *Smart Camera* and *Composite Camera* are integrated into the whole RealityFlythrough system, and it describes some of the other components of the system that are crucial to making the system work.

6.1 Smart Camera

The *Smart Camera* was at the heart of the system described in chapter 5, but getting the *Smart Camera* integrated into the rest of the RealityFlythrough system required a few modifications. The system described in chapter 5 only supported a single camera and as such did not need to worry about how transitions to and from a *Smart Camera* would be handled. In this section we will learn how the *Smart Camera* is integrated into the RealityFlythrough architecture that was described in chapter 3.

A *Smart Camera* is just like any other *Camera* in the RealityFlythrough engine in that it is defined by a location and orientation, a field of view, and the current list of cameras that should be displayed. At this point we should mention that when cameras are contained within cameras, it is not really a *Camera* that is contained, it is a *Camera With State* – a camera that has a certain intensity (opacity), and a certain mesh that is used during morphing. Cameras, themselves, do not have any state. Even the position and the image is defined by an external *Position Source* and *Image Source*.

A *Smart Camera*, then, is just a regular camera that has another camera as its image source. The *Smart Camera* registers itself as a listener on the source camera, and any updates to that camera, including position changes and image changes, are forwarded to the *Smart Camera* as a sequence of camera snapshots. The *Smart Camera* is not receiving an image stream, then. It is actually receiving a camera stream where each still frame is converted into a camera. It is then trivial for the *Smart Camera* to convert the camera stream into a sequence of transitions, if appropriate, or just a sequence of images, if not. When would it not be appropriate to use transitions? If the frame rate of the source camera is higher than two frames per second, for example, the raw video may actually be more sensible than transitions played at such a quick rate. Recall that we try to maintain a one second interval between transitions to give the user a long enough time to commit closure. Since the *Smart Camera* can move back and forth between transitioned and non-transitioned image sequences, it can act as a decorator [GHJV95] for any other camera (regardless of its frame rate) and only provide the extra sense-increasing information when the frame rate drops down because of network congestion.

A key trait of the *Smart Camera* is that it keeps up with processing the frames that are transmitted to it so that it does not fall behind. The *Smart Camera* is decorating the source camera and is not being provided as an alternative view of the data. It is imperative that the *Smart Camera* keep up with the source stream. In order to do this, we make sure that each transition takes the same amount of time as the interval between the source and target frames. Unfortunately, this necessarily adds at least one frame of latency to the *Smart Camera* view because we cannot start processing the next transition until we know where we are going and how long it should take. In situations where time is critical, it may be better to watch the raw, undecorated, stream. Even though the transitions match the speed of the original camera motion, it is still possible to get behind. The frame rate on the incoming data may be variable, and may vary even further when the user fast forwards through the stream. The *Smart Camera* maintains a fixed-length queue for the incoming frames and drops off the older frames to make sure that

the stream is never too far behind the live stream.

RealityFlythrough needs to be able to support multiple simultaneous *Smart Cameras*, and it needs to be able to support transitions between them. It is unlikely that more than one would exist for more than a second at a time since the *Smart Cameras* can be destroyed as soon as they are no longer being viewed by the user, but the system still needs to support multiple *Smart Cameras* during that one second interval. The *Smart Camera* fits elegantly into the existing RealityFlythrough architecture since it is just another camera. A *Smart Camera* can be added to the list of cameras in a *Virtual Camera*, and recall from chapter 3 that the main user interface is just another *Virtual Camera*. Since the *Smart Camera* maintains its own copy of the *Transition Planner/Executer* duo, as far as the outside world is concerned, it is just another camera that can display its image (or images) in the appropriate place on the screen when asked to do so.

What would a transition from one *Smart Camera* (SC1) to another (SC2) look like? The user would be hitchhiking on SC1 and would see a series of transitions between each of the camera's frames. Mid-transition, the user may decide to move to SC2. The transition on SC1 would continue to play out and the images would continue to blend and morph, but instead of the user's position moving to the original destination (SC1's destination), it begins to move towards SC2's current position. This change in movement is prompted by a new transition being created that likely routes the user through a series of intervening *filler* images. When the first image in SC2 would become visible, SC2 is created and transitions on that *Smart Camera* immediately begin. The user's position has not reached SC2's position, yet, though, so the user would have similar acquisition problems to the ones described in chapter 3. We use the same technique described there: we pause SC2 until we have reached it, and then resume play and quickly catch up to live time once the camera has been acquired. At this point, SC1 can be destroyed.

Chapter 3 introduced the RealityFlythrough architecture, and a central claim throughout the chapter was that the architecture was robust in the face of unforeseen modifications. The *Smart Camera* is another example of an unforeseen modification.

Everything described so far works because a *Smart Camera* has the same interface as any other *Camera*. Even though the *Smart Camera* has all of the complexity of nearly the entire RealityFlythrough engine embedded inside of it, the integration of the *Smart Camera* into the rest of the system is trivial. It is just another camera.

6.2 Composite Camera

The purpose of the *Composite Camera* is to create a panoramic image from a series of pair-wise point-matched images. A RealityFlythrough panorama does not have to be as accurately constructed as a traditional panorama, however, because just as with the rest of RealityFlythrough, there is usually only one image visible at a time. It is only during transitions that more than one image is visible. The *Composite Camera*, then, is designed only to improve the placement of images so that they line up more accurately.

6.2.1 Generating Composite Images

(a) (b) (c)

Figure 6.1: This is the first-person perspective of the example transition that is used in the image-stitching discussion in figures 6.2- 6.3. Figures (a) and (c) show the two images that are involved, and figure (c) shows what the transition between the images might look like.

Traditional panoramas, the IPIX or Quick Time VR ones that we are used to seeing on real-estate web sites, for example, are either constructed by using parabolic mirrors or by stitching together a series of images that are taken from a camera that revolves around a stationary pivot point. RealityFlythrough cameras have no such con-

straints and are free to move anywhere in the environment. Construction of an accurate panorama is impossible in most cases because all images are being forced onto the same image plane. Consider the situation illustrated in figure 6.1a. The woman is some distance behind the chair. In the image that only contains the woman (fig. 6.1c), the camera has moved beyond the table and chair. Any stitching of these two images, however, will necessarily contain a combination of objects from both images that are not visible from either camera position. The table and chair would still be visible, for example, even though the user has presumably moved past them. Panoramas work fine when the camera is rotating about a pivot point because any new content that is revealed by using the panorama could just as easily have been captured by increasing the field of view of the camera. There may be temporal disparities, but the resulting image is spatially accurate.

The only way to stitch together images from free-moving cameras while maintaining spatial accuracy is to project the images onto a perfectly accurate geometry of the scene. AVE, the system explored in chapter 2 does this, but recall that the geometry that was used only includes the exterior of buildings. People, tables, and chairs would not be included in the geometry. Unfortunately, if you head down the road of projecting images onto a geometry, you need to do it nearly perfectly or not at all because projecting content onto the wrong geometry creates gross distortions. People would be stretched out across the ground plane and then up the side of a building, for example.

Vision techniques that construct the geometry by analyzing images of the scene hold the most promise because they are being performed on the actual current view of the scene and can thus account for dynamic content. Unfortunately, to date, the vision systems have to make some strong assumptions about the environment in order to be computationally viable. Structure from motion systems [Nis03] require a nearly rigid (i.e. non-dynamic) scene, Virtualized Reality [KRVS99a] requires precise camera calibration and rigid cameras, and algorithms that can do point-matching (and thus geometry computation) on arbitrary images are too computationally intensive to work in real-time [Low04, BSW05]. Research in this field is extensive, however, and with continued increases in computational power real-time reconstruction of dynamic scenes

should soon be possible.

Since we do not have any hope of accurately reconstructing the geometry of a dynamic environment, our approach to panorama creation is to use very sparse point-match data to augment the image alignment traditionally done using sensor data. We may only find three matching points during a run of the SIFT algorithm [Low04] since we are using very low resolution images to make the match as fast as possible. With three points it is difficult to do much more than align the images, but that is usually adequate. We experimented with inferring image orientation and scale and then doing a morph from one image to the other, but the more manipulation we did of the images, the worse things looked when there were problems. The closure effects described in chapter 4 were so powerful that the improvements obtained when things did work well were barely noticeable. In fact, during one user study cited in section 5.6.4, none of the users could discern any difference between morphed and non-morphed transitions. As a result, we have since disabled both the morphing and orientation correction features. We still do image scaling, however, because we found that occasional incorrect zooming is not nearly as distracting as incorrect rotation.

6.2.2 Point-matching Inconsistencies

One might assume that when matching points are found using an algorithm such as SIFT that the position of the image dictated by the matches is more accurate than the one recorded by the sensors. This is unfortunately untrue. Point-matching, even when subjected to RANSAC [FB81] filtering which verifies that the matching points are mutually consistent across some transformation, can still be wrong. Repetitious geometric patterns such as the window panes shown on the buildings in figure 1.2 tend to confuse point-matching algorithms, especially when the images are zoomed in and there is little other context. With two unreliable positioning sources, which one do you trust?

We have developed a simple algorithm that works well in practice. We use point-matching as a refinement on the sensor data, but only if the two datasets generally

agree. Otherwise, we assume the point-matching data is in error, and rely solely on the sensor data. To check for agreement, we position the two images according to the positions reported by the sensors and confirm that the images overlap when viewed from either of the camera positions. If the images do not even overlap, either the sensors or the point-match are in error. We trust the sensors in this case because the sensors are rarely that far off. We have occasionally encountered some machinery that makes the compasses go crazy, but these instances are rare.

6.2.3 Integration with RealityFlythrough

The description in section 5.3 of the *Composite Camera* (it is not called a *Composite Camera* in that chapter, but that is what the panorama is), is accurate but incomplete. Since the system described in that chapter did not use location to position the images, there was no way to do anything other than temporal transitions to and from a *Composite Camera*. Supporting transitions to distant non-point-matched cameras is a little more complicated. The most critical issue is how the position of the *Composite Camera* relates to the real-world positions of all of the other cameras. The images (they are really just cameras, in RealityFlythrough parlance) that make up the *Composite Camera* each have real-world positions, but those real-world positions are trumped by the point-matching data. What we do is position the *Composite Camera* at the position of the most recently added image, and have all of the rest of the images orient themselves around this position. This means that the position of the *Composite Camera* moves with the addition of each new image. In an ideal world, the panorama generated by the *Composite Camera* and the sensor data would be perfect so that the logical position of the *Composite Camera* would have no effect on where the constituent images were located. In reality, though, the panorama is only an approximation of the real world and the sensor data is only an approximation of the real-world position of the cameras. As a result, the positions of all of the constituent images need to be adjusted every time a new camera becomes the origin or reference point for the panorama.

Why did we choose to make the panorama shift to the location of the newly

added image instead of incorporating the new image into the existing panorama? There are two reasons:

1. When the *Composite Camera* is part of a *Smart Camera* – in other words, when the user is viewing a consecutive sequence of video frames that have been augmented with point-matched RealityFlythrough transitions – the next image to be viewed is a temporal neighbor and as such is likely to have a similar error in its sensor readings. If that next image does not have any matching points with the existing composite, the difference in position between the new image and the previous image should be more accurate since the errors are likely to cancel out. If you have ever watched a representation of a GPS-tracked object on a map, you have probably observed that the object slowly drifts within an error circle, and rarely jumps from one position to another. It is this slow drift that we are capitalizing on.

2. The accumulation of errors that result from trying to create planar panoramas of 3d images (with cameras that have six degrees of freedom) can become quite severe. By fixing the panoramas position to the position of the most recently added image, we are effectively resetting the error. The error continues to accumulate on the oldest image in the panorama, but we combat that by limiting the number of images that can be added to the panorama. We currently have a hard limit of five, so once this limit is reached, the oldest image is dropped from the panorama with each new one added.

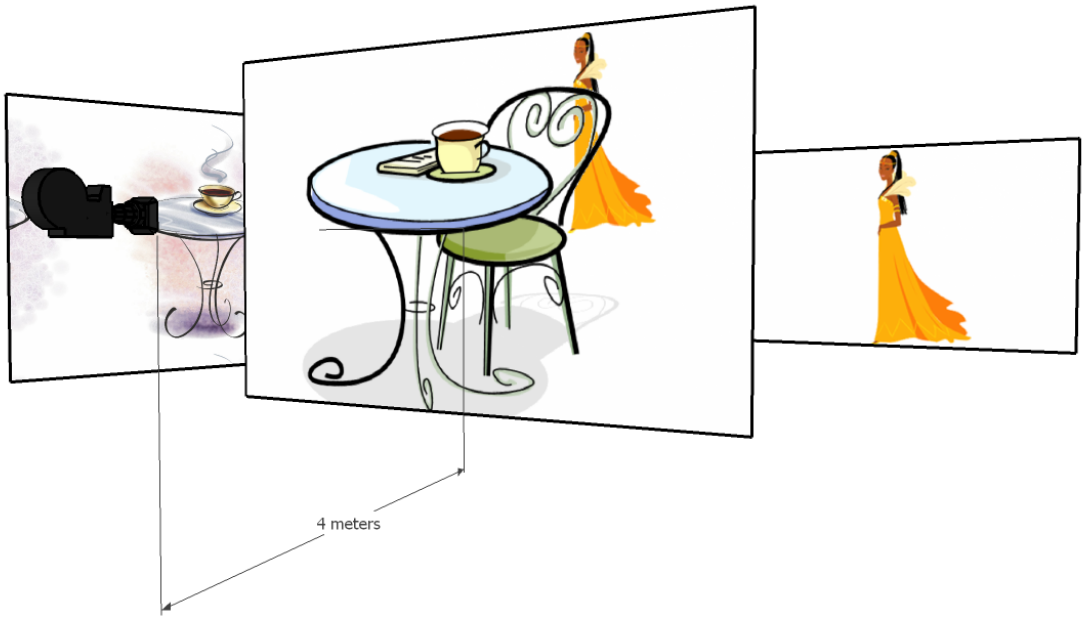


Figure 6.2: This graphic illustrates what an external observer would see when the user's view is filled with the first image in the transition. Figure 6.1a shows what the user actually sees. Notice that the user's position (represented by the camera in the image) is 4 meters from the image. If there were no matching points between figures 6.1a and 6.1c, a standard RealityFlythrough transition would simply move the user's view forward through figure 6.1a until figure 6.1c was in full view. The incorrect placement of the images due to sensor error would tax the viewer's closure abilities. Note that the image to the left of the user in this graphic is not used in the transition and only serves as a frame of reference for this graphic.

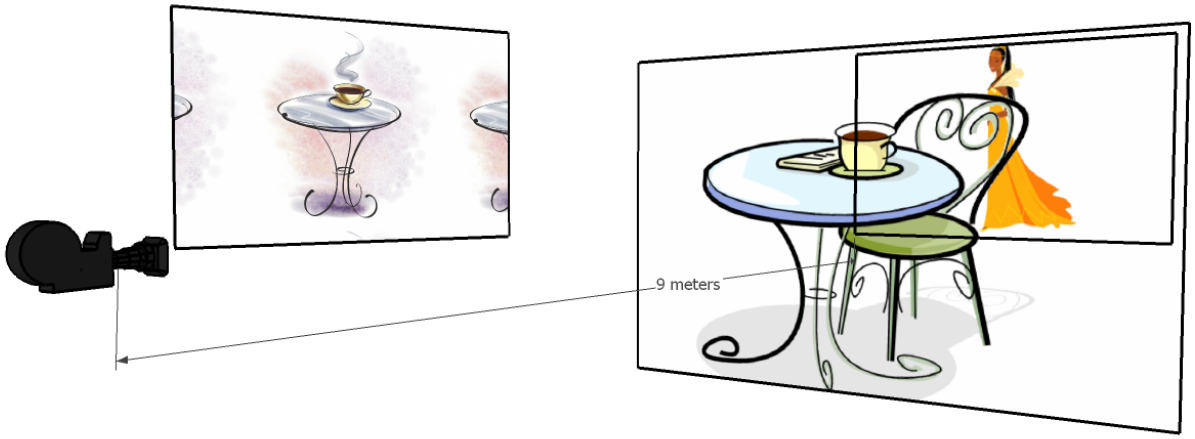


Figure 6.3: This graphic illustrates what an external observer would see after the Transition Planner has determined that there are matching points between figures 6.1a and 6.1c, but before the transition has started. As far as the users are concerned, nothing has changed and their view still looks like figure 6.1a. Notice, however, that to prepare for the image-stitching, image 6.1a has been translated and rotated to image 6.1c's image plane, and has increased in size so that the woman in both photos is the same size. To maintain a consistent view for the user, the user's position was also pulled back by a couple of meters, increasing the distance between the user and the new image plane to 9 meters.

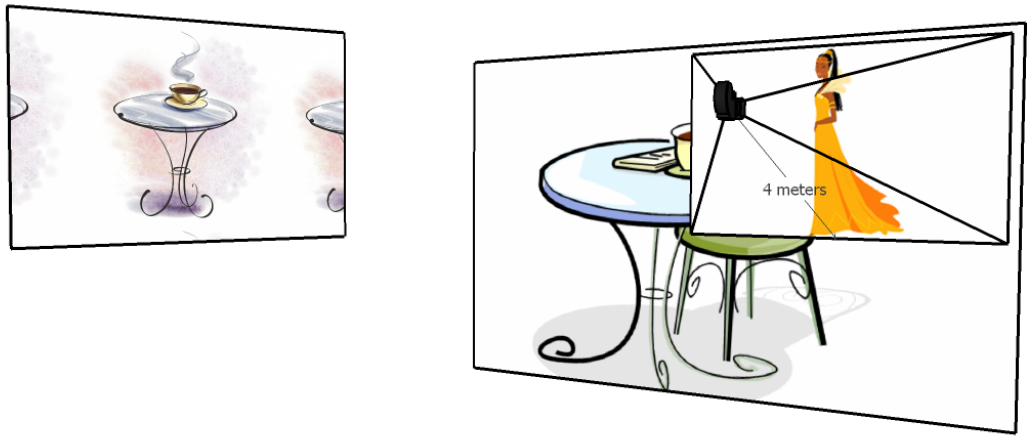


Figure 6.4: This graphic illustrates what an external observer would see at the conclusion of the transition. The user's position has moved 5 meters closer to the image plane, and the photo of the woman (as seen in figure 6.1c) is all the user sees. Note that if the user were to pan to the left to see the image of the table with the steaming coffee cup, some of the now very zoomed-in table in the foreground would be visible for a moment. In reality, the user has probably already stepped past the table in real space, but since we do not know the geometry of the environment, such artifacts are unavoidable. Fortunately they are transitory since they only appear during transitions.

There are two significant challenges with continually adjusting the *Composite Camera's* position and orientation:

1. Every time the *Composite Camera's* position and orientation is adjusted, all of the other non-composite images in the environment that may be in view must also be moved by the same amount so that the user's first-person view remains consistent despite the warp through space. In general, images that are not part of a *Composite Camera* do not remain in view for long once a transition is completed because we do not want them interfering with the comprehension of subsequent transitions. We currently have all but the current image fade out one second after a transition completes. It is still possible, though, for the user to begin another transition during that one second interval and see the lingering effects of the previous transition. The user's first-person view needs to remain consistent, so these other images need to be repositioned as well. The user should be oblivious to his or her sudden jump in location.

2. Since the *Composite Camera* also handles scaling between its constituent images, moving the composite to the position and orientation of the newly added image means that all of the other images in the composite must be rescaled (see figures 6.1-6.4). By doing this, the newly added image (which is presumably the destination of the current transition) will be displayed in its correct scale ready for the next transition which may be to a non-point-matched image.

The second point above requires more explanation because we actually have to be prepared for the next transition to happen at any time. Users can interrupt transitions whenever they want to move to a new destination. *Composite Camera* transitions cannot be implemented as a traditional morph because the user's position in space and the relative sizes of all of the images involved need to be consistent in case a transition to a non-point-matched image occurs. Instead, a RealityFlythrough morph transition is similar to a regular transition in that it moves the user's *position* through space while doing a cross-fade from one image to another. The morph is achieved by simultaneously

warping matching points on the panorama from the source position to the destination position. The key, though, is that the user's position is actually moving through space and the positions and sizes of the images involved remain constant. The motion through space is what increases and decreases the perceived sizes of the images.

In order to make this work, the destination image must be at its correct scale *before* the transition even starts. With all of the other images rescaled, and the *Composite Camera's* position and orientation already set to its new position, the user's current position in space has to be adjusted to keep the first-person view consistent. What complicates matters is that since the images in the composite have all been rescaled, the user's current position has to be adjusted by an additional offset to keep the image that was just scaled to, for example, double its size looking the same. In this example, the user's position in space would have to shift backwards by the appropriate amount so that the double-sized image would look smaller again (see fig. 6.3).

Why would the image be doubled in size in the first place? If, as figures 6.1-6.4 illustrate, the image that was being transitioned to was directly ahead of the current composite, a transition to the new image needs to create the illusion of moving forward. This illusion can be achieved by gradually increasing the size of all images involved until the destination image fills the screen. When the destination image fills the screen, that original image would be double its size. Rather than gradually increasing the size of the images, though, we instead gradually move the user's position closer to the image plane to create the perception that the size is increasing. Since we do not want the initial doubling of the image to be noticeable to the user, we need to jump the user's actual position backwards by an appropriate amount to keep the first-person view consistent despite all of the behind-the-scenes manipulation.

6.2.4 Architectural Considerations

There was a lot of software infrastructure that had to be created in order to support point-matching and *Composite Cameras*, but to continue with the theme that has been running throughout this dissertation, we will focus only on those modifications

that had to be made to the core system to accommodate this new unplanned feature.

The *Composite Camera* is just another *Camera*, and as such it integrates seamlessly into the rest of the RealityFlythrough architecture. Any of the operations that rely on the *Camera* interface can just as easily handle a *Composite Camera*. The main modification that had to be made to the core system to handle the *Composite Camera* involved adding a new rendering function to the *Virtual Wall Renderer* (see section 3.5). Since the *Composite Camera* is composed of multiple cameras each of the constituent cameras needs to be rendered at an angular offset that is governed by the *Composite Camera* and not by the cameras' sensor data.

The *Transition Planner* needed to be modified, as well. The planner needs to select for point-matched images when deciding which image to show next, and it also needs to either create a new *Composite Camera* or add the chosen image to an existing *Composite Camera* when appropriate. In order to handle the repositioning logic that was described in the previous section, the planner also needs to adjust the user's current position by the appropriate offset.

All of these modifications to the existing system amount to less than 100 lines of code in a system that has 50,000 lines of code. The integration is so clean that even the *Smart Cameras* can use *Composite Cameras* to improve their transitions. This demonstrates once again that the RealityFlythrough architecture is robust in the face of change, and that the key architectural abstractions are sound.

6.3 Hitchhiking

In the introduction of this dissertation we saw that both indoor and outdoor transitions were more effective if they took the user along common pathways – through doorways and hallways instead of through walls, for example (see section 1.4.6). The technique for accomplishing this was described as hitchhiking on historic feeds. In this section we will look at how this was accomplished and discuss some problems with the current implementation and some potential solutions.

There are two things that make transitions easier to comprehend. Better image alignment and temporal proximity. Better image alignment is helpful for obvious reasons. Temporal proximity helps because it allows the user to assign meaning to the events that are occurring and thus help with closure. Many of the users in the user study described in section 7.3 were desperately trying to make sense of the images that they were seeing by assuming temporal order. When there was no temporal order, they noticed, and wondered out loud how such an odd event could have happened.

The solution we arrived at that approximates this hitchhiking metaphor works as follows. A graph of connections between cameras is maintained. A *2* is assigned to the edges of temporal neighbors, and a *1* to the edges of cameras that have matching points. Before the start of every transition, a fitness calculation is made on the source camera and all of its neighbors. The same is done for the destination camera and all of its neighbors. Each fitness score is scaled to a range between three and some max edge weight (we use 30), and the edge weights are added to the graph. We then run Dijktra's Shortest Path algorithm on the graph and thus obtain the sequence of transitions that will move us from the source to the destination [Dij59].

The majority of the point-matched pairs will be consecutive frames not only because they are the ones that have been guaranteed to run through the point-matching algorithm but also because the similar content increases the chance of finding a match. While consecutive frames help with the goal of maintaining temporal order, they unfortunately also contribute a meandering quality to the transitions which can sometimes be punctuated with long periods of getting nowhere when the camera operator stops moving for awhile. For the remainder of this discussion, we will call these undesirable stops *knots*. Think of the path to the destination as a string. If that string has a lot of knots, progress along the string is not nearly as efficient as it could be.

Notice that we have not considered space anywhere in our shortest-path calculation except with the neighboring cameras of the source and destination. We are unable to avoid knots without looking at the sequence of cameras and analyzing the flow of neighboring positions. Such an analysis, while seemingly trivial, cannot be done if you

do not trust the sensor data. By removing a knot, we may in fact remove a critical sense-preserving sequence that takes the user around a corner in a hallway, for example.

We still have a knot problem, but the inclusion of point-match data and the spatial neighbors of the source and destination cameras create intersection points, or worm holes, through which the user can slip past a knot by either hitchhiking on another camera for awhile or by doing a temporal jump across the knot if one of the fitness functor weights is shorter than the travel through the knot.

One other problem is caused by the preservation of temporal order, or should we say the *almost* preservation of temporal order? We have observed that users use temporal cues to help them commit closure. The problem we have now, though, is that temporal order is not preserved when we go through one of the worm holes. So what we have created is a system that encourages users to rely on temporal cues, but then at key times during the sequence we ask them to ignore temporal inconsistencies. If you look carefully at figure 1.6, you will notice that there is a problem with time in that sequence. Chapter 4 has a discussion on this problem as well. The other issue with time is that the transitions are allowed to move backwards in time. We could have a sequence, then, where we walk down one length of a corridor hitchhiking on one camera, and then switch to a second camera and walk backwards down another corridor. While spatially sensible, these temporal oddities could confuse users.

At a minimum we need some way to alert the user about temporal shifts. We already have an age-indicator bar at the bottom of our images which give some clue to the age of the images, but we probably need a more obvious indicator that invovles an audible or visual alert when something strange happens with time. To limit the backwards walking, we may want to assign different weights to the graph based on temporal order. Make the weight three instead of one for backwards travel, for example.

It should be noted *Hitchhiking* creates a sequence of transitions. Each one of those transitions still goes through the *Transition Planner* and is possibly augmented with the other properties of transitions. For example, one of the transitions in the sequence may have the user's view rotate by 180 degrees. The standard RealityFlythrough

transition can still fill that transition with *filler* images if any are available. *Hitchhiking* does not replace the transition mechanism that was described in chapter 3. Both techniques mutually augment one another.

6.4 Walking Metaphor

When not *Hitchhiking*, transitions travel "as the crow flies". We initially found that the cleanest transition from point A to point B involved a smooth rotation and translation with both ending at the same time. If we do not do this, the transition has a jerky feel if the rotation completes before the translation. Even though smoothing out the transitions makes them aesthetically more pleasing, this approach creates two problems:

1. The combination of rotational and translational motion is difficult to comprehend when traveling large distances that are not covered by filler images. The floor grid does not provide enough clues to the distance traveled and the degrees rotated.

2. The smooth path is not consistent with the kind of path a camera operator would take, and there is therefore less opportunity to find and display filler images. People, and therefore camera operators, do not tend to walk from one end of a courtyard to the other while slowly rotating their gaze 180 degrees. They instead rotate to the direction of travel, walk forward to the destination, and then rotate to the new direction.

If we made RealityFlythrough transitions mimic natural walking, we could probably make the transitions more sensible and also find more *filler* imagery to display. People, and camera operators, naturally look around quite a bit. They probably do not do 360 degree rotations that often, but they definitely move their heads back and forth to increase the field of view. By rotating first to the direction of travel, it is likely that we would find some nearby images that showed that rotation. The forward walk would then be sensible because the image that was being viewed would increase in size. We may then find a destination image that is in a similar location to the one we wish to

move to, but at a similar orientation to the direction of travel. The final rotation would be comprehendible because rotations are the easiest transitions to understand.

Our first effort to model human walking failed, however. We wound up with jerky transitions like the ones we were trying to avoid in the first place. The problem was that we were using the walking metaphor even for short distances that were traveled. That is not how people walk either. Sometimes people *do* walk backwards. Sometimes they *do* walk sideways.

We use the following criteria to determine what kind of walk to perform:

- If there is a small rotation delta and a short distance to the destination, just do a normal transition.
- Otherwise, if the distance is short, move to the destination camera's location, and then rotate to the new orientation.
- Otherwise, rotate either to a forward-walk or to a backward-walk depending on which one is closest to the current view, walk to the destination, and then rotate to the new orientation.

We have had good success with this approach when using a rotation delta of 60 degrees.

6.5 Temporal Controls

Implementing a temporal control system in RealityFlythrough was quite complex because the system needed to support the live collection of data while allowing historical exploration, and also support a replay of live data while allowing historical exploration. The historical exploration mimics the conditions as they were at the time. The live cameras move through the environment just as they did at that time, and all of the snapshots that are available for viewing were taken in the past. We would not want future images to be visible when we are exploring a historical scene.

To accomplish this, we modified the system to maintain two times: a live time which is the actual look-at-your-watch time, and the current time which is the period of time that the user is actually exploring. The live time could be historic, too, however if the user is replaying a live event. We will ignore this confusing detail for the purpose of this discussion, however.

When the user is viewing live data, the live time and current time match. The user can choose to move back and forth through current time, can control the speed of the current time, can make time stop, and can make current time travel backwards. Everything in the system, except for the processing of incoming data, is controlled by the current time. All of the camera locations that are displayed on the map are consistent with what would have been visible at the time being explored, and all live video feeds play back from that time. If the current time moves forward at twice its normal rate, all video feeds in the system will play back at this faster rate. For any image the user views in the environment, the user can choose to play the source camera's video stream from that point forward. The user can also single-step through a sequence of images and manipulate the current time in that way.

What we have created is a DVR (digital video recorder) for an entire RealityFlythrough event, and the transformation to the system is as powerful as what DVRs did to television. Users of our system rely heavily on the temporal controls (see the study described in section 7.4).

6.6 Conclusion

This chapter concludes the discussion of the major features of RealityFlythrough. At this point we have a system that can listen to multiple video feeds, archive images from these feeds, position the feeds and images on a map of the environment, generate simple transitions between images that reveal spatial context, fill gaps with the appropriate imagery from other cameras (live, still, or archived frames), handle transitions to and from moving cameras, augment low-frame-rate video with transitions,

point-match images and generate composite views, intelligently choose paths that mimic walking that respect walls and other boundaries, and support historical exploration of a scene with full temporal controls that allow navigation both through time and space. This is the system that our user studies explore in chapter 7.

Chapter 7

User Studies

This chapter presents four user studies. The first two user studies were done on earlier versions of the system, and the last two build on these results by investigating the complete RealityFlythrough system that is described in this dissertation.

7.1 How Transitions Affect User Behavior

Of the many system components we depend on for the sense-making properties of RealityFlythrough—GPS accuracy, compass accuracy, good image quality, and our approach to transitions—transitions are the one upon which we can exercise considerable control. Consequently, we report on early results from an experiment in assessing how our transitions affect sense-making. In particular we investigate how they are similar to (or different than) a system with no transitions at one end of the spectrum, and a perfect tele-reality system at the other. In this study we only address the first half of this spectrum, and assume that users would have no trouble comprehending the perfect system. To focus on the qualities of transitions and dramatically simplify their assessment we have explored this question through the use of stationary still images in a space, rather than confuse things with live video.

7.1.1 Experiment

In order to learn more about the effectiveness of transitions, we created another version of RealityFlythrough that was identical to the original except that no transitions were performed when a user switched between cameras. An initial pilot study demonstrated the difficulty of obtaining conclusive quantitative results, due to the large number of experimental variables that must be controlled. There is a high variability in the experience and abilities of the users that would make statistical comparison difficult. The experiment that resulted was designed to help us answer the following questions: (1) How is the user's behavior affected by the transitions? (2) Do transitions help the user more quickly grasp the spatial relationship between images? and, (3) Do users automatically understand transitions, or is this a skill that needs to be learned?

By studying the results of the pilot study, we chose the following partial operationalization of user behaviors for our questions: For question one, do users who do not have transitions flip back and forth between images more often (presumably trying to figure out how the two images relate)? For question two, do users who do not have transitions linger longer in certain parts of the space, trying to make sense of how the images relate to each other? For question three, do users who have transitions show or voice confusion during certain transitions?

The experiment we constructed was designed to give the subjects a very concrete task to provide us with results that could be compared across all subjects. Each participant was randomly given one of the two versions of the system and was given two minutes to remotely explore a portion of the ground floor of a 1500 square foot house. 31 images were made available that gave nearly complete coverage of three rooms. After exploring the space, the subject drew a floor plan from memory and tried to position as many objects as he/she could recall on the plan. The subject was not allowed to consult the images while doing the sketch, but was given a list of objects that may have been present to help with recall. During the exploration, the participants were allowed to use the birdseye view to glean information about the relative positions of the cameras, but the birdseye view did not contain a map of the house.

7.1.2 Results

Eleven subjects participated in the experiment. Six saw transitions between images, and five did not. We will identify the former subjects as the transitions group and the latter as the no-transitions group. Analysis of the resulting floor plan sketches is subjective and inconclusive. More experiments need to be done to control for the experience subjects bring to the task. First-person shooter game experience, innate spatial ability, comfort with spatial abstractions, and comfort with computers all played a role.

All participants were given the chance to use both versions of the experiment at the conclusion of the study, and there was unanimous agreement that transitions are better than no transitions. There were two exceptions to this sentiment during the pilot study, one of which may have been prompted by the way the pilot study was set up. The other case cannot be attributed to flaws in the experiment, and appears to be a genuine preference for no transitions. This subject was exhibiting all of the signs that indicate he was doing transitions mentally in his head (repeatedly flipping back and forth between two images), so it is interesting that he preferred not to have them. He said that he did not like the time required for a transition to take place, suggesting a desire for speed or efficiency.

To answer questions one and two, we present the following results: Two of the five no-transitions subjects spent about half their allotted two minutes stuck in the hallway which was covered by less than one quarter of all the images. None of the transitions subjects exhibited this behavior. A third no-transitions subject who is a hardcore gamer spent a little extra time in the hallway and did a fair amount of flipping back and forth between those images. A fourth no-transitions subject did not linger in the hallway, but was slow and methodical and only got to 3/4 of the images before time expired. This contrasts with the transitions subjects, all of whom covered the space completely and saw all images. For the subjects who did linger in the hallway, no extra detail about the hallway was revealed in their sketches of the floor plan. We should mention that the fifth no-transitions subject had what was clearly the worst floor plan sketch and apparently had no concept of the space being explored, but it's hard to tell why, so we shall

ignore the practices used by that subject.

What the above results indicate is that subjects who did not have transitions had more difficulty making sense of the images they saw and had to move more slowly through the space or do more flipping back and forth between images to compensate. Subjects who had transitions may or may not have had more comprehension of the space, but it is clear that they thought they understood it because they did not linger.

We now address question three: There were several instances where transitions subjects showed surprise or confusion during their explorations, even though they had transitions to help them. These cases fall into two categories. The first category involves walking through walls, and the second, poorly constructed 180 degree turn transitions that turned towards a wall, rather than away from it. The ability to walk through walls is a useful feature we want to include [HS92], but it was clear from these experiments that a feedback mechanism needs to be employed to alert the user of this odd phenomenon.

We received a comment from one subject that speaks to the naturalness of the transitions. He said that the rotations were more natural than the backward and forward translations, and that the latter took some getting used to. This is consistent with our experience. As expert users now, we are quite adept at internalizing the myriad sense-making cues, and while the transitions cannot be described as natural, they do seem to convey the information required for sense-making. Browser style *Back* and *Forward* controls would be useful to help the user see a transition multiple times if there is confusion. Repetition is a good sense-making device.

7.2 The Effectiveness of Simple Transitions

Note that this section is a reprint with minor changes of *Harnessing Mobile Ubiquitous Video* [MCG05], a paper co-authored by Neil McCurdy, Jennifer Carlisle, and William Griswold.

(a) (b) (c)

Figure 7.1: A sample transition used in the user study. Image (b) shows the transition in progress as image (a) moves off the screen to the right and image (c) moves in from the left. This transition represents rotating to the left while moving forward.

To quantify how well our transitions convey additional information about the spatial relationships between cameras, we constructed an experiment that compared simple two-camera transitions with a no-transition alternative. We will call these two scenarios *transition* and *no-transition* respectively. We assumed that the ideal would be perfect, seamless transitions that could convey spatial relationships with 100% accuracy. Our target was 100% accuracy.

For the *transition* tests, a short video was played that showed a transition between two still photographs. The subject could watch the transition multiple times and could control the playback speed. While watching the video the subjects had to choose the best of four possible birdseye depictions of the scene that showed the relative positions of the cameras (Fig. 7.2). The *no-transition* tests were similar, only instead of a video the subjects viewed two photographs while making the selection. The transition represented in Fig 7.1 is an example of a transition that might have been shown in a *transition* test, and Figs. 7.1a and 7.1c are examples of still photos that might have been used in a *no-transition* test.

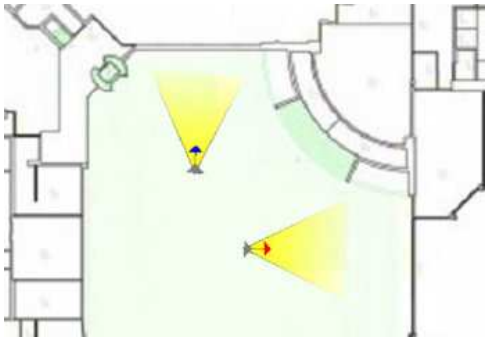


Figure 7.2: An example answer sheet for the user study. A birdseye view that shows the position and direction of two cameras. This is an example of one of the multiple choice answers used in the experiment. For each question the subjects were presented with four images like this and had to decide which one best represents the position and orientation of the two cameras. Note that this is an incorrect answer for the transition depicted in figure 7.1.

We had 30 subjects participate in the study. The majority were university students, but their experience with computers varied. Each subject was tested on both *transition* and *no-transition* questions. The scenes depicted in the photographs fell into two categories: *familiar* and *unfamilar*. The *familiar* location was a campus foodcourt that all participants were very familiar with. The *unfamilar* location was a disaster scene that no one was familar with and was difficult to interpret even when familiar with it. Twenty questions were asked of each participant—five in each category. We attempted to make the questions increase in difficulty based on our experience with which motions are difficult to visualize. Rotations were considered simple. Rotations combined with motion were considered more difficult. Questions were randomly interleaved from the four categories, but each participant was asked the questions in the same order.

We hypothesized that the *transition* responses would be quicker and more accurate. Given the difficulty we had with determining the locations of the cameras at the *unfamiliar* location, we also hypothesized that the *no-transition* answers would do no better than random guessing, and the *transition* answers would do much better.

Table 7.1: Mean of correct responses for all 30 participants.

	No-Transitions	**Transitions**
Unfamiliar Location	1.63/5 (σ = 1.10)	2.60/5 (σ = 1.19)
Familiar Location	2.73/5 (σ = 1.01)	4.10/5 (σ = 0.87)

7.2.1 Results

As Table 7.1 shows, the mean scores for the *transition* questions exceeded those of the *no-transitions* questions. Furthermore, of the 30 subjects 26/30, 86.67% achieved a greater or equal score on the *transitions* questions. This indicates that the transitions provide the user with additional information that is beneficial in determining the spatial relationship between cameras.

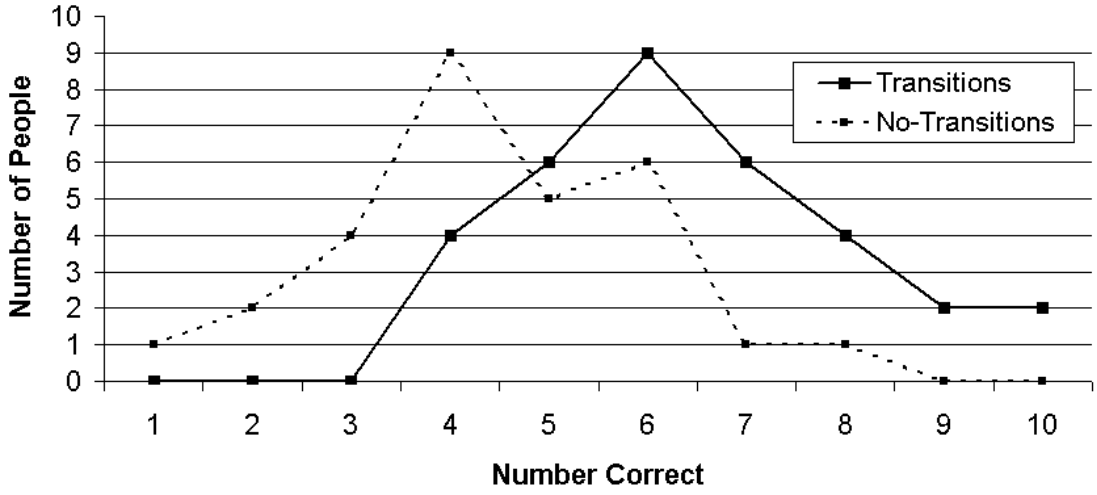


Figure 7.3: Number of people who answered ***x*** *number correct. Compares* transitions *to* no-transitions.

Fig. 7.3 demonstrates that subjects attained a higher level of success answering transition questions. Scores in the 50th percentile and higher demonstrated a greater rate of success for the transition questions.

When drilling into the data we determined that the success rate on the *transition* questions increased as the experiment progressed. This suggests that as the users become more familiar with transitions, the transitions become easier to interpret. Most notably, the second to last and the last *transition* questions at the *familiar* location were

answered correctly 93.33% and 100% of the time respectively. We also looked at the relative increase in speed between the *transition* and *no-transition* questions and noted general patterns that indicated that the *transition* questions were answered more quickly as transition interpretation was learned. In fact, by the end of the experiment all *transition* questions were answered faster than the *no-transition* questions. These results are supported by a comment from one of our subjects in a post-experiment interview: "[transitions are] different than anything I had really seen. At first it seemed very strange and took me by surprise. By the middle or end of the test I had really gotten the hang of it and the transition questions seemed much easier."

Our hypothesis that *no-transition* questions in the *unfamiliar* location would be answered randomly was supported by the data and by user comments. Random guessing would produce an average score of 1.25 out of 5; the 1.67/5 average score obtained in the study is not much better than random. The average *transition* score of 2.73/5 is better, but not quite as good as we had hoped. The subjects lack of experience with transitions may explain this. The transitions were much more difficult at this location because the images and the subject's knowledge of the space provided little additional help. Complete trust had to be placed in the transition, and some subjects were not ready to extend that trust. Referring to Fig. 7.3, notice that two subjects scored perfectly on the transition questions, and two scored in the 90th percentile. All four of these individuals reported having a great deal of experience playing 1st person shooter games, suggesting that cognition of image-to-image transitions is a skill that is honed through exposure. The subjects who were not able to trust the transitions reported no such experience.

This section is, in part, a reprint of the material as it appears in Harnessing mobile ubiquitous video. Neil J. McCurdy, Jennifer N. Carlisle, and William G. Griswold. In CHI 05: CHI 05 extended abstracts on Human factors in computing systems, pages 1645-1648, New York, NY, USA, 2005. ACM Press. The dissertation author was the primary investigator and author of this paper.

7.3 The Effectiveness of Complex Transitions

The previous section explored the effectiveness of simple two-camera transitions, some of which actually turn out to be quite confusing because of the lack of filler images to provide additional context. In this section, we will explore more complex transitions that navigate through narrow hallways and stitch together multiple images. The transitions use all of the techniques described in chapter 6, and thus show off the best we have been able to achieve to date. Frustrated with our earlier inabilities to demonstrate that all users find RealityFlythrough transitions sensible, we set out to prove once and for all that the transitions could be understood by everyone. This seemed to be such an obvious result and it was frustrating to not be able to demonstrate it through user studies.

The experimental setup was designed to overcome the problems that have plagued our experiments in the past. There is so much user variability in spatial ability that even the seemingly simple task described in the previous section was difficult for some. In that task we had users watch a transition and then select the best of four bird-seye representations of the beginning and ending positions (see fig. 7.2). With the user study described in section 7.1 we ran into similar problems – problems that prompted us to create the multiple-choice study that we just described. In that study, we had users explore one room inside of a home and then draw a floor plan from scratch identifying the locations of key elements such as the dining table. This turned out to be a very difficult task and we probably would have realized this ourselves had we not been so intimately familiar with the rooms.

The key insight from both of these user studies is that converting from a 3d immersive view to a top-down view does not come naturally to many people. Before the advent of maps, and the now ubiquitous satellite imagery of the world, humans had little opportunity to view the world from above. One of the users in the floor-plan study was unable to even begin the experiment because she could not understand what a top-down view would even look like. We had her trying to draw the room that she was in, and no matter how we tried to explain it, she would inevitably draw the room

from her perspective. She was a 60 year old woman from another country and had obviously never used (or at least never understood) a map. There is much debate among psychologists about the representation of spatial data in the brain. J. J. Gibson argues that we do not possess an internal map of the world, nor do we need one. Instead he says that we apprehend the structure of the world by navigating through it. Occasionally we may look down on our environment from an elevated location to get a better perspective, or we may even consult a map, but, he argues, this does not mean that we have a map in our heads [Gib].

To avoid these problems, we had our users describe what they saw using whatever means they felt comfortable with. They could use words, gestures, or even resort to drawings if desired. All we wanted to know was that they knew what they saw, regardless of their ability or inability to put what they saw into culturally understood geographic terms or symbols. For the most part this mechanism for eliciting understanding was successful, although there was one case where we are still uncertain if the user really meant want she said. She seemed to have trouble with her left and right and corrected herself one time, but on several other occasions did not. On a rather simple transition where the screen obviously rotated to the right indicating motion to the left, she claimed that she had rotated to the right. Other users were definitely helped by using gestures. One user could not find the words to describe what she saw, but was able to point to where she had been looking before the transition ended.

That explains how we planned to accurately capture the data, but what data were we after? From the discussion on closure in chapter 4 we know that closure is a very powerful mental device that will allow people to make sense of photographs regardless of what additional tools we give them. What we wanted to show was that the RealityFlythrough transitions, independent of all of the other tools that make up the complete system, are adding value. We know that users can glean a lot of information about spatial orientation just by looking at two photographs. Once the locations of the photos are positioned on a map, it is almost certain that users can figure out how they relate to one another. Add the hundreds of other spatially positioned images, include

a temporal navigation mechanism, and people can most certainly explore a scene and understand it. We hypothesized, however, that the RealityFlythrough transitions were making it *easier* to explore the space. The transitions were helping people commit closure.

Unfortunately, easiness is not that easy to measure. Perhaps something can be inferred about the time of task completion, but how do you measure that across individuals? Within individuals, the scene would have to be changed between time trials so that learning effects could be discounted. How do you guarantee that the various scenes have the same complexity? How do you ensure that the user's attention is the same across all tasks? How do you account for the learning effects that are independent of the scene? Users will learn how to explore better and learn how to commit closure better once they have done it once. Watching users in previous experiments, it was clear that video game experience drastically improved peoples' abilities to navigate and explore an environment. They know what to look for. They know what will help them remember key information about a scene. They know what that key information is. Even with the study that is described in this section, we still could not avoid learning effects. In transitions that were watched later in the experiment, the user would obviously study the first image for a time looking for cues that they might use later. They would learn to avoid temporal traps and fixate on objects that likely would not change.

We chose to demonstrate the effectiveness of transitions by looking at transitions in isolation. If ungrounded transitions, transitions that have no context, can be understood on their own, then it is almost certain that the spatial information being conveyed by the transition is contributing positively to the exploration experience. We decided to dive even further into the transition to determine if the motion component of the transition was providing value above the sequencing of images that is a central part of complex transitions. Sequential images look a lot like low frame-rate video, and we know that low frame-rate video, while confusing during camera panning, is quite sensible in other situations. Perhaps the selection of filler images during transitions and the hitchhiking metaphor described in section 6.3 was enough to provide understanding

and the motion component of transitions was contributing little.

We entered this user study with trepidation because we were undoing the ecology of RealityFlythrough and trying to understand the whole by looking at its parts – a folly that Gibson cautioned against leading to the whole movement of Gibsonian Psychology [Gib]. We were removing the very closure properties that we have argued contribute to the user's understanding of transitions. Transitions are not meant to be viewed in isolation. Still, we were left with few options short of evaluating RealityFlythrough solely on qualitative terms, and since we were doing that as well (see section 7.4), we had little to lose.

7.3.1 Experimental Setup

Figure 7.4: A Jump *type of transition that only shows the source and destination image.*

Figure 7.5: A Sequence *type of transition that shows a sequence of images that lead to the target but does not use the motion component of a RealityFlythrough transition.*

We created 27 videos that were each no longer than four seconds. The videos

Figure 7.6: A Transition *type of transition that shows a full RealityFlythrough transition. Notice that in this transition the rotation to the right down the hallway is a little more evident. In this case, astute users were able to tell that there was a motion to the right even when viewing a* Sequence *type of transition because the wall clearly deadends. In other* Hallway *scenes, however, the users were uncertain whether they turned left or right at intersections when viewing the* Sequence *transition.*

depicted movement from a source image to a destination image and, as just described, the user's job was to explain to us how the the source image relates spatially to the destination image. In other words, how do you get from point A to point B? The user had two opportunities to view each video, but was required to provide an answer after each viewing. Along with their answer, the users also stated their confidence on a 5 point likert scale.

We required 27 videos because there were two dimensions – each of which contained three categories – that we wished to explore and each of those required three versions so that we could maximize the information gleaned from each user without succumbing to learning effects. The first dimension was the type of transition. On this dimension, we had:

1. *Jump*. Jump is basically no transition. The video jumps from the first image to the last image without showing any motion in between. In our videos, each image was displayed for 2 seconds, and the user was able to study the source image before the transition started and the destination image after it ended (see fig. 7.4).

2. *Sequence*. These videos showed a sequence of images that took the user from the source to the destination, but the only transitional effect shown between each image was a cross-fade. (see fig. 7.5).

3. *Transition*. These videos showed the sequence along with a full RealityFlythrough transition between each image (see fig. 7.6).

The second dimension was the scene type. On this dimension, we had:

1. *Simple*. The simple scene is large outdoor open space with transitions that only involve rotations about a central point. Transitions are particularly good at handling rotations so we hypothesized that this scene would be relatively easy for the *Transition* type of transitions. The simple scenes were filmed at the Price Center, a food court on the UCSD campus.

2. *Hallway*. The hallway scenes involved a walk down an L-shaped corridor. We hypothesized that the *Sequence* type of transitions would be quite sensible in hallways because there was little side to side movement that had to be accounted for. The transitions shown in figures 7.4-7.6 illustrate a hallway scene. The hallway scenes were filmed inside a building at UCSD that was the scene of a county-wide disaster drill. The halls were full of victims, policemen, and medical personnel wearing hazmat suits.

3. *Complex*. The complex scenes involved 180 degree rotations along with some walking. The word "complex" was chosen to convey complexity in the *Transition* type of transitions. In hindsight they were not quite as complex for the other types of transitions because they were confined to a single room. The complex scenes were also filmed during the disaster drill at UCSD but in a different area of the building.

We pseudo-randomly selected which of the three versions of a scene the user would see; pseudo so that we could ensure that all of the scenes were viewed an equal number of times. We gave the users no training on RealityFlythrough transitions other than to tell them that the videos they were seeing were designed to *help* them and not *trick* them. One of our early users thought he was being tricked. We did not want to give users any training because part of the appeal of transitions are their naturalness. The

first-person immersive view mimics the view that humans have evolved to comprehend. This decision unfortunately negated the brief learning curve that some users need to comprehend transitions.

7.3.2 Experimental Results and Analysis

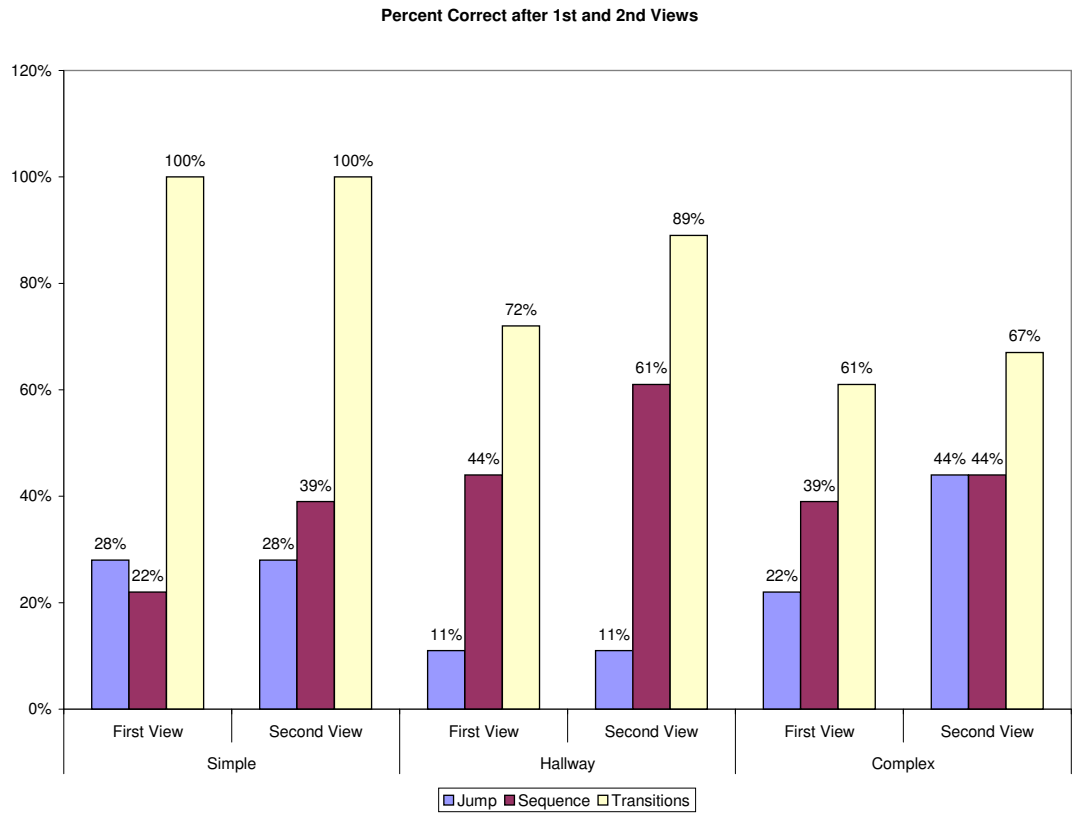


Figure 7.7: Experiment results that show the percentage of correct responses across both the transition type and scene type dimensions. Results are shown for both the first and second viewings of the same transitions.

We had 18 users complete the experiment. There was an equal split of males and females and a nice distribution of age ranges from 15 to 65 with a slightly heavier bias towards the young. Answers were either correct or incorrect. No partial credit was given. The answers had to include both rotational and translational information in order to get credit. For example, a correct answer for the transition shown in figure 7.6 sounded like this, “I walked down the hall, turned right, and then continued down that

hall until I came to a wall."

The chart in figure 7.7 summarizes the results. The results are self-explanatory so we will use this section to drill into some of the questions that are prompted by this data that cannot be explained with this chart alone.

Why not 100% Comprehension?

The most glaring question is why the *Transition* comprehension drops rather dramatically from 100% in the *Simple* scenes to much lower in the *Hallway* and *Complex* scenes.

We consider the *Complex* scene, first. Four of the users who had trouble with the complex scene struggled with the same transition. This was a particularly difficult transition that most users could not figure out. It involved a walk down a hall followed by a 180 degree turn. The walk down the hall was odd in that the entire length of the hall was traveled in a single transition with no filler images. Worse, the images did not line up quite right so the transition was not natural. To confuse matters more, only one or two filler images were found for the 180 degree rotation so a large part of the rotation had to be understood by viewing the rotation of the spherical grid. One of the users did not understand what the grid was telling him, but immediately understood the transition once it was explained to him after the experiment. The initial zoom down the hallway would have been understood by experienced users as well. This reveals a flaw in the comprehendability of transitions because they obviously are not completely natural, but if a lack of filler images makes such transitions necessary, the good news is that they can be learned.

Users who viewed other transition types had trouble with this particular transition, too. Only half of the *Jump* and *Sequence* viewers were able to figure out this transition, and they actually had an easier time of it because they did not have the novel transition effects creating additional confusion. Some of the users were able to notice similar content in the source and destination images and were able to use this information to correctly interpret the spatial orientation of the images. Recall that the complex-

ity ranking of this category was issued because of tricky *Transition* transitions. For the users trying to make sense of the *Jump* transitions, there was actually more information present in this scene to aid closure than there was in the *Hallway* scenes (and that is one reason for the increase in *Jump* comprehension between the *Hallway* and *Complex* scenes).

We should mention that there were two *Transition* users who did get this transition correct, and both were after the first viewing. One of these users stands out because he had a confidence of 5 and knew exactly what happened despite the weird transitions. He is a young man that plays a lot of video games.

The other two misses on the complex transition occurred on another complicated transition that was almost the inverse of the one just described. In this case, though, the rotation did have filler images. The rotation was to a washed out scene, however, so two users were confused by this and thought that they had continued down the hallway that is visible halfway through the rotation. All of the *Sequence* transition viewers made this same mistake since they were not aware of any rotation so the fact that four of the six *Transition* viewers were not confused is actually good news and validates the effectiveness of the transitions.

For the *Hallway* transitions, two users had trouble comprehending the transitions after the second viewing. One of those users was the woman I mentioned earlier who seemed to have trouble with her left and right. We can think of no other explanation for her misunderstanding of that particular transition. The other user had trouble with the transition that is depicted in figure 7.6. She missed the turn to the right. It is possible that in her mind her description of walking down the hall meant that she had turned right since to her maybe it was obvious that she had followed the corridor in the only direction that was possible. There was one other user who made the same mistake on the first viewing of that particular transition, however, so it is possible that the rotation to the right was just missed.

The speed of the transitions is another possible explanation for the drop in comprehension. The playback rate of the transitions was a little too fast for novice users.

This was not an issue in the simpler scenes, but may have contributed to confusion in the more complicated scenes. We did not assign partial credit for answers; an incorrect response does not mean the user did not understand anything about what they saw. They could follow most of the transition but would report getting lost at a key area. The same would happen with the *Sequence* transitions, too. Contrast this with the many (44%) "I don't have a clue" answers that we heard on *Jump* transitions. Not all incorrects are the same.

Low Jump and Sequence Scores on the Simple Scene

The next question is why the *Jump* and *Sequence* scores are so low on the *Simple* scenes. Even though this scene is labeled simple, the lack of image overlap between the source and destination images makes it difficult to commit closure. The *Sequence* transitions just confused matters more by showing a lot of images that seem to not have much relationship to one another. The lack of rotational cues hurt comprehension. The only doubt that any users had about the *Transition* transitions was the degree of rotation. They spent time trying to decide whether it was a 160 or 175 degree rotation.

The industriousness of the users surprised us and their success (and even their failures) really illustrated how powerful closure can be. One user used a temporal cue that we had never noticed even after viewing these photos nearly daily for the last four years. A woman in red that was visible in the first frame had walked into the final frame by the time we had taken that photo. The user assumed we were tracking this woman, and correctly inferred that we had rotated 90 degrees to the right. Another user tried to use shadows to determine orientation. He was unfortunately unsuccessful, but was quite confident in his answers.

Better Sequence Scores in Hallways

Users were able to get 61% of the *Sequence* transitions correct after the second viewing. The intersections were the source of most of the problems for these users. They were uncertain whether they had turned left or right, and often times found some-

thing in the image (sometimes incorrectly) that convinced them that they had turned one way. They occasionally tried to use information from previous scenes even though the transitions were supposed to be of isolated scenes. Once again, the power of closure surprised us. One user nearly made a phenomenal inference on a *Jump* transition based on her knowledge of where the light sources were from previous questions in the experiment. When we chose this particular transition sequence we thought there was nothing that could possibly tell the users how these two images were related unless they saw filler images, but she almost got it.

As we hypothesized, the *Sequence* transitions were pretty effective at conveying forward and backward motion. In a hallway, that is about the only motion that is possible. The transitions were not necessarily aesthetically pleasing, though, because even though the camera operator only had one direction to move in, he or she could still look around in the hallway. The transitions were mostly sensible only because the user knew nothing too crazy could happen in such a confined space.

The *Jump* scores dropped in the *Hallway* scenes because of the nature of the L-shaped transitions. There was only one version of the *Hallway* scene in which users had any hope of inferring spatial orientation, and two users *did* manage to do it with that scene. There was a third user who got it right on the first try but then changed his answer after viewing the two images a second time.

Jump Improvement on Complex Scenes

Users who viewed the *Jump* transitions did substantially better on the *Complex* scenes. There are two possible reasons for this improvement. The first has to do with the environment. The source and destination images were all in the same room, so the very observant user could detect correspondences between the two. Many users were able to do this. Why? That leads to the other reason for the improvement.

There was quite a substantial learning effect during this experiment. By the time the users reached the final video sequences, nearly all of them were paying much more attention to the images. They did not know which type of transition was going to

come next, so they all prepared for a *Jump*. They would really study the source image and try to remember everything about it, and then they would spend a long time studying the destination image before answering. It was almost possible to see them computing RealityFlythrough transitions in their heads.

Confidence

So far we have made no mention of the users' reported confidence. Confidence is quite subjective and we encountered users who answered questions correctly with apparent confidence but never scored themselves above a 3. Note that confidence scores were on a 5 point likert scale with 1 representing a complete guess and 5 representing complete confidence. Others reported absolute certainty about something for which there was no evidence supporting the claim. As an example of an incorrect assessment of confidence, consider the user who while watching a *Jump* transition of a *Simple* scene recorded a confidence of 3 on her second viewing of the video despite saying, "Wait, now I think I turned to my left. I'll just stick with my first answer."

What is more shocking, however, is false confidence. This first manifested itself in our 5th subject and we assumed it was an anomaly until we saw examples of this in our later subjects as well. This user would construct an entire narrative that described how two images were related to one another. She would convince herself that that was exactly what happened and report a confidence of 5. Even after watching the video clip a second time, she would keep her answer at a 5. "Yep, that's what is happening." There was absolutely nothing in some of these image pairs that would give any clue how they were related, but she would find something, and convince herself of it. One would think that this is a person who has false confidence, but that does not explain why she would score a *Transition* on a *Simple* scene a 3, even though the *Transition* in this case made it absolutely clear how the images were related.

Curious about this phenomenon, we analyzed the data looking for false confidence. We flagged all incorrect answers that were given a confidence of 4 or higher after the second viewing (see table 7.2). Less than 10% of the results fell into this category,

Table 7.2: User confidence in incorrect results by transition type after second viewing of transitions. Also shows the total number of incorrect answers by transition type and the number that are false confident. False confidence is determined by an incorrect answer that is assigned a confidence of 4 or 5.

Type	*Avg Confidence*	*# Incorrect*	*# False Confident*
Jump	*2.10*	*39*	*6*
Seq	*2.61*	*28*	*8*
Tran	*2.50*	*8*	*2*

but it is indicative that only two of the 16 belonged to *Transition* transitions. Six were associated with *Jump* transitions and the remaining eight with *Sequence* transitions.

The false confidence with *Sequence* transitions has to do with changes in direction at intersections. The users see something that convinces them that the camera moved left and they stick with this story even if it is not true. False confidence is obviously quite dangerous because it can lead to misunderstandings of the environment. It is likely that such misconceptions would be corrected when the full sytem is running and the user has access to additional context, but they may not be. If RealityFlythrough transitions make the user less confident about their incorrect assessments, this is a very good thing.

Table 7.3: User confidence in correct results by transition type after second viewing of transitions. Also shows the total number of correct answers by transition type and the number that are false pessimistic. False pessimism is determined by a correct answer that is assigned a confidence of 1 or 2.

Type	*Avg Confidence*	*# Correct*	*# False Pessimistic*
Jump	*3.27*	*15*	*3*
Seq	*3.77*	*26*	*4*
Tran	*3.98*	*46*	*3*

Transition viewers also had a higher confidence in their correct answers. 3.98 for *Transition*, 3.77 for *Sequence* and 3.27 for *Jump* (see table 7.3). The high confidence on correct responses and the low incidence of false confidence provide further evidence that the RealityFlythrough transitions are providing the users with a good understanding of the space.

7.3.3 Conclusion

The results from this experiment clearly indicate that RealityFlythrough transitions viewed in isolation provide users with a good understanding of the spatial arrangement of all images involved. We were still not able to demonstrate 100% comprehension, but we now realize that with no experience and seeing transitions in isolation, some users do not completely understand the more complicated transitions. We were able to show that 100% of our users comprehended the simple rotational transitions, however.

The study described in the next section will investigate how users respond to transitions when they are viewed in context and can use inter-transition closure to make sense of the scene as a whole.

7.4 Using the Complete System

The user study described in the previous section provided evidence that RealityFlythrough transitions convey additional spatial information that may assist comprehension of complex scenes. As discussed in that section, it is unlikely, however, that we can construct a user study that can quantify the value added by these transitions. User ability is too variable and if a single user performs multiple experiments, it is very difficult to control for learning effects.

The qualitative data that can be recovered from a carefully constructed end-to-end user study can be extremely rich, however, and often leads to insights in design and usage that would never have otherwise been discovered. In the study explored in this section we had pairs of users navigate through a complete RealityFlythrough environment.

Getting the system to a state where we felt comfortable letting novice users explore on their own was no small feat. In such a test, we would lose control over which transitions were displayed, and some of the ugliness of the system that can be hidden during staged studies would be exposed. The users could move anywhere through the

environment, and they would probably be using the system in unanticipated ways. We needed to be confident that any transition that they happened upon would be sensible and contribute to their understanding of the environment. This meant, though, that RealityFlythrough had to be robust in the face of the challenges of the real world. A world in which sensors are inaccurate; large sources of magnetism effect compasses; lighting conditions vary from day to day and minute to minute; dynamic objects and people confuse both users and point-matching algorithms; buildings that are divided into small rooms and hallways obscure views that may better convey how a space is laid out; and network conditions limit the quality and quantity of data that can be transmitted.

RealityFlythrough would have to contend with these conditions, and yet still be able to provide users with enough information to make sense of the scene that they were exploring. The solutions that we have outlined in this dissertation to the myriad problems the real world presents would help, but despite these efforts, the occasional ugly transition sneaks in, a transition like the complex one in the previous section that gave our users so much trouble. We hoped that our users in this full-system study would be able to use their closure abilities to comprehend even the ugly transition, and that an ugly transition would not cause them to lose faith in what mostly were sense-contributing transitions.

This study, where users are free to explore an environment using the entire RealityFlythrough system, really represents the culmination of all of the work outlined in this dissertation. The user experience would not be possible if any one of these components of RealityFlythrough were not implemented. The positioning of cameras on a map shows users where cameras are located; the archived imagery provides views of the scene from different angles that can be explored out of temporal order; the basic transitions contribute to spatial understanding when sensor data is all that is available; the path selection, the choice of filler images, and the handling of dynamic environments help with transitions that cover long distances or angles; the *Composite Camera* and point-matching improve the aesthetics and also contribute to understanding; the *Smart Camera* allows low frame-rate video to be watched without disorientation; the *Hitchhik-*

ing transition sequences improve navigation through hallways; and the temporal controls allow users to revisit previous views of the scene that need to be watched or re-watched. Without any one of these components, the system as a whole is severely degraded. The users in this study would be using a complete system.

7.4.1 Experimental Setup

The users were presented with the following scenario:

> A report has been received that a building at UCSD has been attacked by terrorists. The police have secured the scene and have nullified the threat. There are a number of victims in the building, however, who appear to be exhibiting signs of chemical exposure. A hazmat team has entered the premises and is in the process of treating and removing the victims. Two members in this entry team are equipped with head-mounted cameras and are transmitting images from the scene to your command and control center. Your job is to learn as much about the scene as you can so that you can make decisions now (and in the future) about where to send the doctors who will soon be entering the scene. Your primary concern is the safety of your personnel. Figure out how to get them in and out as safely and efficiently as possible. You want to care for and transport all of the patients with a minimum of risk to your staff.
>
> You will be viewing the live camera feeds of the two personnel. The frame rate is very poor (1 frame per second), but is the best we can do because of the network conditions at the disaster scene. You will notice that the camera positions are drawn on a map. The positions and orientations are mostly accurate, but occasionally the compasses get confused by the magnetism in the building. Trust your instincts.
>
> In the 5 minutes that you have to explore the scene, please answer as many of the following questions as you can. Try to be thorough, but remember to explore as much of the scene as possible. You may mark your answers on the map that is provided.
>
> 1. Who are the victims (men, women, children)? Please get an approximate count of each.
> 2. Where are the victims? Get an approximate victim count by room.
> 3. How serious do the injuries appear to be? For each injury, try to gauge the severity of the injury.
> 4. Are there any deaths?
> 5. How many hazmat personnel are on the ground?

6. Are there any patients who are being neglected?
7. Is there any structural damage?
8. Are there any dangerous areas to avoid?
9. Are there any violent suspects? What weapons do they have? What are they wearing? How many suspects are there? Is there any active violence? Is the scene stable or are there still problems?
10. Is there anything outside to be aware of?
11. Are there any suspicious people entering or leaving the building?
12. Is there any damage to utilities? Water-main break? Electricity outage?
13. What is the status of potential escape routes? Can our personnel evacuate quickly if necessary?
14. Are there any areas that need to be explored? Where would you suggest that the camera operators go next?

The users would experience a live copy of a RealityFlythrough recording that had been staged earlier. The copy takes advantage of some of the temporal controls in RealityFlythrough that allow an event to be re-viewed and re-explored as if it were live. It is a good tool for training and for doing post-mortem analysis of events that involve disaster response. The event was a staged disaster drill that involved SWAT, hazmat, and medical response teams from all over San Diego county. Bombs exploded, SWAT teams dropped from helicopters, hazmat teams wore full hazmat gear, and victims doused themselves in fake blood. It was a big deal, and mostly realistic except for the civilians who kept order and the media who filmed the event.

During this drill we embedded two camera operators with the medical response team that would be treating and evacuating victims (see fig. 7.8). During the actual drill the video feeds captured by the camera operators were streamed live to the command center where personnel experienced a RealityFlythrough very similar to the one that the users in our user study would see.

The main improvement to the system that our users saw came from the scrubbing of the indoor data to correct camera locations. RealityFlythrough obviously requires information about the location of cameras, and when this information is not available the system gracefully degrades to the one described in chapter 5. We wanted our

(a) (b)

Figure 7.8: The camera unit worn by camera operators. (a) shows all of the gear: a camera, gps device, tilt sensor, and palm-top computer. (b) shows the camera attached to the mask.

users to experience the full RealityFlythrough system, so we artificially inserted the locations of the cameras for the indoor scenes. RealityFlythrough has a location correction algorithm that interpolates locations based on a few corrections, so only about 25 corrections were required for each camera. It is not unreasonable to have these corrections done live when necessary, but we eagerly await an indoor locationing solution that is robust enough to work in disaster settings.

We had each pair of users use one of the three systems that was studied in section 7.3. The *Transition* version was the full RealityFlythrough system. The *Sequence* version stripped the motion component from the RealityFlythrough transitions but still showed the intervening images as a sequence of stills. The *Jump* version gave the user no context and simply jumped between chosen locations.

One member of each pair, the *operator*, was trained on the computer in an environment similar to the one that they would ultimately be exploring. The other member, the *recorder*, was responsible for recording the findings on an architectural floor plan of the building. The questions that the users were asked to answer were intentionally varied and long. Those questions along with the five minute time limit put the users

under a similar kind of stress to what real command center personnel likely experience.

The primary reason we used pairs in this study was to encourage the users to *naturally* verbalize while completing the task. This approach to experimental observation, known as *constructive interaction* [Miy86, Wil95], allows us to get access to the operators' thoughts through their substantive interactions without prompting on our part, which could disturb the activity or otherwise bias the study. A secondary consequence is that the two operators together might be more effective than a single operator, because the co-pilot may see something the pilot has overlooked (as one example). However, we note both that paired work practices are common (and hence our set-up is realistic in this sense), and our questions are qualitative in nature and not tied to single-operator use.

Seven pairs participated in this study and most of them used the *Transition* version of the system. We heavily weight the *Transition* version because that was the one we wished to learn most about. We label the pairs *Pair 1-7* so that we can easily cross reference them.

7.4.2 Experiences of the *Jump* Group

Pair 1

One pair of users experienced the *Jump* version of RealityFlythrough. *Pair 1* ended up with a pretty good understanding of the environment, despite the lack of transitions. The recorder on the team appeared to be quite lost most of the time, however, and it is easy to see why. The operator flipped around between images so rapidly that even we had trouble following him. Sometimes he would be exploring a hallway at one end of the building and then suddenly jump outside and look around there for awhile. Nearly all of his knowledge was internal and he did not communicate much with his partner – perhaps because he was struggling for understanding himself.

The recorder would look down at his floorplan to write something down, and when he looked up again, the operator would be viewing a completely different scene. The recorder would have to try to catch-up but before he could, the operator had moved

on. The recorder eventually resigned himself to the role of a secretary and simply asked the operator for answers to the questions.

The operators eyes darted back and forth constantly between the immersive view and the birdseye view (see fig. 1.1). It was almost possible to see him building his internal map of the environment as he committed closure on every image pair. He spent the bulk of his time exploring the space on his own. This tactic may have hurt him because the live feeds would have shown him sequences of related images that would have shown him the path that the camera operators had taken. The video would have been one frame per second, but even without the *Smart Camera* the video is quite comprehendible.

Other users of other versions of the system relied quite heavily on the live feeds. We suspect that the operator got so heavily involved in the game of trying to figure out how these disembodied images related to one another that he forgot about or did not have time to revisit the live feeds. It is also possible that he has a more active personality and preferred active exploration over passive exploration. His partner would certainly have benefited from a more linear story, however.

The operator ended up having a pretty good knowledge of the space; he could describe where the major areas of the environment were. He saw less than 80 of the photos, however, out of a total of roughly 700. We counted 79 clicks on images, but many of those were repeats seen during back-and-forth clicking presumably to confirm understanding. While the user's understanding of the larger space was sound, he must have lacked detailed knowledge. There was not much continuity in his exploration, so he probably had islands of understanding.

Pair 1 was also given the opportunity to experience the other two versions of the system. The operator loved the sequencing that was present in those versions. He described the jumps that he saw as "looking at all of the different angles at once... The transitions show us how the images are related and the sequence show us how we get from one area to another."

7.4.3 Experiences of the *Sequence* Group

Two pairs of users experienced the *Sequence* version of RealityFlythrough.

Pair 2

Pair 2 did a really nice job of exploring. They relied quite heavily on the sequencing of photos used on the live feed, and also used the temporal controls to go back and revisit areas when the recorder missed something. They would then move back to the live camera and see the sequence of images that took them there. They never seemed to be lost. There was one sequence that they saw that is similar to a sequence that was included in the experiment described in section 7.3. We know from that experiment that none of the *Sequence* users understood that particular transition. A quick look at the map (not present in the previous study) seemed to confirm to the operator what had happened because he never showed signs of confusion. It is also possible that he simply waited a little longer until the subsequent images corrected his initial misperception. We humans are quite tolerant to confusion and rarely throw up our arms in desperation when we do not understand something. We remember the source of confusion and try to let future events resolve things. We do this all the time when watching complicated plots in movies, for example.

The recorder was once again a little behind. He showed some signs of confusion during the exercise, and then in later questioning he only gave himself a three on a ten point likert scale for his understanding of the environment. “I was pretty confused,” he said. He was trying to watch the video and orient himself on the map at the same time.

Both users were quite excited when they saw the RealityFlythrough transitions. “That other one (the *Sequence* version) was like YouTube trying to load.” “That’s cool. Look at that. It’s showing us how the images line up. The other one wasn’t doing that.” How close was the *Transition* version to a video of the scene? “A lot closer than the last one.” On all of the questions they scored the *Transition* version two or three points higher than the *Sequence* version.

Pair 3

This stands in stark contrast to the other *Sequence* users, *Pair 3*, who did not like the *Transition* version at all. "It felt like it took you out of things for a minute. I didn't like that. I was more confused." What he disliked was the floor grid that appears when there are no filler images. He makes a good point that the floor grid stands out, removing the user from the immersive feel of the scene. This actually indicates success, to some degree, because the rest of the experience feels so natural that the artificial insertion of a floor grid feels like a visual slap. "No, this is not real," it seems to be saying. We are not sure how to overcome the problem of missing imagery. The borders of the nearby images could be stretched to fill in the gaps, but if this were somehow perceived to be less artificial, it could trick the user into seeing something that was not there. If you recall from the discussion on presence in chapter 2, it is sometimes better to not achieve full presence. In disaster response settings, we think it is better, actually, for the user to realize that something is amiss. It may be a good thing that the user is removed from the immersive experience because there is, in fact, something missing and the user better be aware of that when making decisions.

The other user, the operator, disagreed, about being more confused by the *Transition* version. "Actually, I was less confused than last time." During the original *Sequence* version, the operator seemed to have a little trouble understanding where he was. "I can't quite tell which way it's going?" And then after the experience: "I personally got confused because I was switching camera angles and people so it's kind of hard to tell exactly where you are." The operator spent most of his time viewing the live feed, so the "switching of camera angles" statement referred to the camera operator's angle changes as he walked through the scene.

7.4.4 Experiences of the *Transition* Group

We had four pairs experience the *Transition* version of RealityFlythrough.

Pair 4

Pair 4 only managed to explore one third of the entire space. It is easy to dismiss them as an ineffective team who did not understand the system or the task. This was our first impression, anyway. Closer examination of the video, however, reveals something quite interesting. Both the operator and the recorder were very detail oriented and spent the early moments exploring the entryway in great detail looking for answers to many of the questions. They were not just looking for victims, they were examining the wounds on the victims and trying to decide if the wounds were fatal.

This is not what prevented them from exploring the rest of the scene, though. The operator explored by single-stepping through the images, and for the first two and a half minutes he never looked at the map and was oblivious to the extra information the map provided – key information that there was much more of the scene to be explored. How do we know he did not look at the map? We could see his eyes, but more importantly he pointed with his mouse and we could monitor his eye gaze by watching the mouse. It never left the first-person-view.

Halfway into the experiment he finally noticed the map and clicked on a camera that was outside. Finding nothing of interest outside, he moved back inside and once again focused his gaze on the immersive view.

What at first blush appeared to be a negative result for the *Transition* version is actually quite possibly the most positive result we could have obtained. The RealityFly-through transitions are so effective at conveying spatial information that the map never needed to be consulted. Contrast this with the operator of *Pair 1* whose eyes darted back and forth dramatically between the two views. The *Pair 4* operator did not need the map.

This reveals a serious flaw in the navigation interface, however. If users do not need the map to stay oriented, they should not have to use the map in order to move spatially through the environment. Many of the users figured out how to turn temporal control into spatial control, but this is obviously not a good solution. As chapter 8 will explore, we need a navigation interface that integrates nicely with the immersive view.

Pair 5

Pair 5 also relied heavily on single-stepping, but quickly learned to look at the map periodically to see what additional information could be gleaned. While exploring the entryway, the operator glanced over at the map and noticed that the green camera was moving quickly through the building. He clicked on it and while watching the ensuing sequence noted, "We're getting to see everything that that camera has been seeing which is pretty cool." From that point on he knew to explore both camera paths and to occasionally pay attention to the map. For each question that they were tasked to answer, he would say, "let's go back to the beginning", and hold down the single-step key that took him backwards until he was back at the beginning and then he would quickly single-step through every image, noting key findings for his partner. He would then do the same thing on the other camera.

The operator also made use of the spatial exploring capabilities of the system. At one point he and his partner were counting victims, and uncertain about what he was seeing, he said, "Maybe 5. Let's go down and see." He then clicked on a camera far down the hallway knowing that the transitions would do the work of taking him through the hallway. Contrast this with the experience of the *Pair 1* operator who would have had to click on each camera in succession to achieve the same effect. With each view, he would have to divert his eyes from the immersive view so that he could orient himself and select the next camera.

The *Pair 5* operator's repeated viewings of the same scene seemed to enhance his memory of the environment, too. At one point when viewing an area that he was familiar with, but through a different camera's lens, he correctly noted: "Seems different. Is this a different time?"

After the experiment, I asked the operator if he thought the transitions were providing him with any extra information because it seemed that with his reliance on the temporal controls he could have done just as well without them.

> You mean the little Hollywood style ones that do this? Maybe they're hardly necessary but they are nice. But the ones that are really useful, I thought,

> was like that. (Pointing to a transition that had a gap and no image overlap). If you see these two images you don't know, if it's just flipped, that that's the same guy because they all look the same. So even if it doesn't match perfectly you'd say that's probably the same fellow. Those ones are really nice. I can't look at both of these screens (the immersive view and map view) at the same time. Whenever Anne (his partner) asked where are we and what are we looking at, you go to this one (the map). I mostly looked at this one (the immersive view), and it helped. I knew roughly where I was. I didn't know necessarily exactly which direction I was pointing, but I knew where I was from one frame to the next. I knew which way to point, but I didn't know in a global view which way I'm pointing at any one time.

Pair 6

Pair 6 struggled the most of all of the *Transition* version viewers. The primary problem was the operator's misunderstanding of the user interface. The somewhat klunky interface for selecting a camera requires the user to select the desired camera first by hovering over it and then clicking on it. This operator was simply clicking on areas of the map and expecting to be taken there. The click instead would take him to whatever random camera his mouse had selected while in transit to the click position. The operator was definitely not controlling the system. It seemed to be controlling him. It is actually quite surprising that the random path that he took through the environment and the stress of not understanding why the system was misbehaving did not have a more adverse effect on the experience. By watching the transitions unfold (as a viewer of television would do) he was able to still learn quite a bit of information about the environment. He still understood the space and *Pair 6* found answers to the basic questions such as how many victims there were.

Pair 7

The final pair of users in the *Transition* group were by far the strongest of all of the users if measured on task performance. They answered all of the questions completely, and had plenty of time left over to explore and double-check their answers. Both the operator and the recorder were fully engaged in the activity and naturally filled

their respective roles. The recorder, in this case, was actively looking at the screen and discovering information on his own. In most of the other groups, the recorder took a more passive role and received information rather than seeking it out.

The operator explored all of the different modes of navigation. He single-stepped backwards when the recorder missed something and clicked on the live views to catch up to real-time again. When asked to move ahead, the operator used a similar technique to the one observed in *Pair 5*, clicking on a camera at the end of a long corridor so that he could automatically see the transition sequence that took him there. The operator had a slightly better grasp of the navigational components of the system than the recorder. The recorder asked, "Let's work from over here. Is he real-time over here?" "No," the operator responded. After a pause the recorder got his bearings, "This is where the real-time is over here, right?" "Yeah," the operator responded.

This pair had plenty of time to review questions they had already covered. They had completed counting victims, but then realized they had not noted the sex of the victims. The recorder asked, "Do you want to scoot on forward so we can see what we have here and start observing? Let's start counting. You count the guards and I'll count the people on the floor." Contrast this level of engagement with that of the *Pair 1* recorder who was unable to be an active participant in his task. That recorder had the same number of different things to keep track of (his map, the RealityFlythrough map, and the immersive view), but he also had to try to make sense of his partner's somewhat erratic jumps through space. The jumps were intentional and calculated, but to anyone other than the person doing the exploring, they appeared to be erratic.

Later, *Pair 7* watched the camera follow a stretcher out of the building. Nervous about what they may be missing, the operator glanced at the map and observed, "That blue guy is still in the hallway. We already have a pretty good idea of what is over there." They spot something suspicious outside. "What's this?" They examine it closely over a few frames: "Can't see. Bomb? Is that a threat?"

We mention these mundane details because they demonstrate that this pair of users was not struggling at all in their use of the system. They had a very clear under-

standing of everything they saw and had plenty of time to doublecheck their answers, to do some fun exploring, and to even think about ways that the system could be improved. They suggested that we provide an interface for the camera operators to key in what they were seeing, and further suggested that we mount the cameras on the guns. These users were clearly not stressed.

We will close with this pair's debriefing which demonstrates quite an extensive knowledge of the environment after such a short period of exploration:

> *Recorder:* Victims. They're all in the hallways. The guards didn't go into any of the rooms. They just went in hallways. And they went all the way into this area of the building and they followed this path. And they also went on the outside which is not on the map. Yeah in the parking lot which is somewhere over here.
>
> *Operator:* Yeah, I mean the camera man didn't really go anywhere except for the hallways. Yeah. So there weren't really any threats that we could see. There were several guards at different locations throughout the building. For instance right here and there.
>
> *Recorder:* At each doorway, there were two guards. Every hall and corridor.
>
> *Operator:* And at major crossroads. However we didn't really go past the guards. There were many, many different people.
>
> *Recorder:* Yeah there were many people dead. Many people severely injured. And some who were not. There were a couple of people who were moving around. There were many people severely injured, and there were a couple of people who we assumed are dead because no one was going to them anymore. Because usually when they're dead no one tries to mess with them. I'm guessing there are about a dozen or so who are dead. The number of hazmat people in white – there's about a dozen of them. Maybe two dozen, about 24. Neglected patients? I know I saw five neglected patients.
>
> *Operator:* The people who we thought were dead could have also been neglected patients. It was very difficult to tell. Also the gender of the victims we felt was difficult to tell. Although in some cases I could tell whether they were male or female.
>
> *Recorder:* I didn't see any dangerous area. Violent suspects? I didn't see anything of the sort. The guards were standing their ground. They were pretty calm. Also if there was anything violent, there wouldn't be any camera crews or people helping people, yet. The guards would stop that. Damage to water and electricity? There was nothing. Escape routes? There were two. Actually three. There was one here from this door that people could have gone out. Also on the right, but I don't know what's on this part of the

building. The guy didn't see it. And then down here there's another escape route down here. I think this is an emergency exit. I know because I saw sunlight from there. Also the front door of course. Weapons? There was tactical gear and hazmat, and a couple of people had rifles. The rest didn't have anything.

Operator: It was mainly just the guards. The majority of the casualties were in these first two halls. This one was somewhat less populated from what we could tell.

Recorder: Also, an unexplored area was here on the right. It splits off into two directions. He went left, not right. So we don't know what's right. And also there were these four rooms that you could go into to "neutralize a threat".

Operator: Another thing. If you were to make a choice of what to send into this disaster area, I only saw maybe two or three stretchers throughout this entire thing. There were so many people on the floor. There were a lot of personnel in there, but they really need to get the people out. There weren't adequate stretchers. Lot's of neglected patients on the floor. Yeah, that was kind of a major concern. No threats that we saw.

Recorder: No bombs or anything like that.

Operator: Except for that thing by the flag, but I wasn't really sure what that was.

Chapter 8

Conclusion

This concludes a long journey through the internal workings of a rather complex system that at its core is based on a very simple idea. That idea is that if you place two appropriate photographs next to each other, you position yourself so that you can see nothing but the first photograph, and then you move your head and body so that you can see nothing but the second photograph you may experience a sense of motion from one to the other . If you repeat this process and appropriately situate a lot of photos taken from many different positions in an environment perhaps you could feel like you were exploring through the environment as you moved from photo to photo. Replace the photographs with live video feeds, and you have live remote-exploration.

That is the simple idea. This disseration was about how to make this work in the real world, a real world that transforms simple ideas into complicated ones as we try to manage the complexities of the real world. But at its core, RealityFlythrough remains a simple system. It is a collection of *Cameras*, and a mechanism for planning and executing transitions between them. All of the components of RealityFlythrough that we discussed can be described in these terms, and it is through that simplicity that we have created a complex system that to the user looks simple.

We demonstrated that novice users can explore a highly chaotic environment by using less than 700 grainy, unfocused, low quality, 352x288 web cam snapshots whose locations and orientations are only approximately inferred from sensors. After

spending only five minutes in such an environment they can answer detailed questions about what they saw and they can describe with incredible detail what they experienced. It was almost as if they were there, but in fact their experience really went *beyond being there* [HS92].

This simple idea that became RealityFlythrough has created a wealth of research opportunities, but our investigations so far have only just scratched the surface. Indeed, it was difficult to stop working to present what we have so far because there is so much more that we want to do. We conclude this dissertation with a study of future possibilities:

8.1 User Interface

8.1.1 Spatial Navigation

Improvements to the user interface for spatial navigation would probably be the most satisfying enhancement to RealityFlythrough. The current method for moving through the environment is unsatisfactory because it forces the user to draw his or her attention away from the immersive view to look at the birdseye view. As chapter 7 described, one of the users in our final user study almost never took his gaze away from the immersive view and as a result failed to realize that a whole section of the building was unexplored. Not only do we need to provide a user interface for spatial exploration that is compatible with the immersive view, but we also need to make sure that it clearly indicates to the user what is available to be explored. It needs to show the users where they can go.

The interface used for selecting cameras in Photosynth would work well in RealityFlythrough [SSS06]. What it does is display the rectangular outline of an overlapping photo anytime the user moves the mouse over the overlapping region of the display. Clicking on the rectangle initiates a transition to that photo. We could do the same although this technique really only works for images that overlap. It is possible that by exaggerating the field of view (thus making more images overlap with the current

image) we could increase the number of images that could be moved to without causing much confusion to the user. The artificially overlapping rectangles would only be displayed on the borders of the image, and the transitions to the associated images should make the actual relationships between the images more clear. To prevent the increase in field-of-view from casting too wide a net and cluttering the display with less desirable candidates for where to move to next, we could only increase the field-of-view when no overlapping images are found.

There is still an issue with how to move backwards and how to strafe left and right. Photosynth draws arrows on the screen to support that kind of movement. We could do the same or perhaps do something with 3d halos [BR03].

8.1.2 Virtual Camera Interface

As was heavily discussed in section 3.6.3, a planned modification all along for RealityFlythrough was the creation of a Virtual Camera interface. This still needs to be implemented. The idea is to have the user choose a location to view rather than a camera at a location. Regardless of what the camera does, the view would remain on the chosen location. Each time a physical camera pans across the desired view, the view would be updated with the new image. That new image will remain in view until another update is available from that camera or any camera in the system. The Virtual Camera interface would be useful for watching a doorway to see who is entering or leaving a building, for example.

8.2 Going Beyond Being There

8.2.1 Augmented Reality

There are a number of possiblities for augmenting the immersive view with meta-data. Hyperlinks to relevant web pages can overlay buildings or other points of interest. Additionally, the camera operator can tag locations with audio clips or other meta-data to provide information that may not be visible in a photo. For example, a first-

responder might flag structural damage or severe injuries. The absence of information would be important to report as well. "I explored this area of the building, but found nothing worth noting. Do not bother to go here."

Explicit camera-operator picture taking (possibly with higher resolutions if needed) would be helpful, too. It is still important to transmit continuous automatic video (or still-image sequences), but it may be useful to have the camera operator take explicit photos of important locations to ensure that high-quality images of that location exist. Snapshots taken from a moving camera are often blurry, even those from a high-quality camera.

The communication between the viewer and the camera operator should be two-way, as well. Human camera operators are autonomous, but humans are also very adept at following directions. A simple, "Go back to that intersection but turn right" can be understood and instantly processed by the camera operator. We should be able to take full advantage of having an intelligent agent on the ground.

There is no reason to limit the augmenting of the data to the camera operators, either. People who have explored the scene virtually have gained insight about certain conditions. They should also be able to annotate points of interest. Perhaps one pass would be made by a structural engineer who is looking for potential collapse points in a building and another by a physician looking for injuries. These expert findings would augment the display and would take the experience well beyond being there.

8.2.2 Enhancing Temporal Controls

We have only just barely touched what is possible with the spatio-temporal controls in RealityFlythrough. The timeline views and scrubbing controls that are prevalent in video editing software would make the data much easier to navigate. In addition, combining temporal controls with the Virtual Camera Interface discussed earlier would allow a particular view of a space to be explored through time. It would be like time-elapsed photography, but potentially with more than one camera providing the views.

8.3 Improving Path Plans

Path planning with inaccurately sensed data is very difficult and we have only *just* achieved path plans that are consistently sensible. A lot more work can be done to improve the plans to make them even more sensible and aesthetically pleasing. With our current implementation that uses the walking metaphor (see section 6.4), we still occasionally encounter knots that make the transition stall for long periods of time. In addition, a simple transition to the left will occasionally start with a transition to the right (since that is the motion the camera operator took). The transition is sensible, but certainly not very appealing. Much work needs to be done to further limit these annoyances while still honoring the commitment to real-time path-planning.

8.4 Increasing Sensory Breadth

RealityFlythrough currently only mediates the sense of vision. As we learned in chapter 2, one way to increase the feeling of presence in a virtual environment is to involve more senses. Had RealityFlythrough included sound in the user study described in section 7.4 some of the tasks such as determining if patients were alive or dead would have been much easier. There is an eerie calm when exploring the scene using RealityFlythrough, but in reality there was a cacophony of screaming and moaning that revealed that patients were in pain and very much alive. Having experienced both the reality of the scene and the RealityFlythrough version, we can say that the lack of sound detracts from the experience. At the same time, however, the lack of sound focuses attention on the other senses and may create the necessary distance from the scene that enables rational decision making.

When given a choice, mediating a sense is always better than not mediating it because there is nothing to prevent the viewer from dialing down the sensory input. Mediating sound opens up a whole other venue for research because in addition to determing how the sounds from multiple microphones can be mixed to produce sound at a novel location, the medium is ripe for *Beyond Being There* extensions. We can listen

through walls, hear all sounds at once, do spatio-temporal exploration through sound, and have certain sounds such as gunshots or explosions alert the viewers and possibly even triangulate their origin [Sho97].

8.5 Understanding Closure

Our discussion on closure in chapter 4 hypothesizes why the imperfect illusions of RealityFlythrogh transitions are effective. We drew on existing research that studies the human brain's ability to understand and comprehend motion pictures, but there is still much research to be done on the cognitive processes that are involved during the comprehension of transitions. These studies may provide additional insights into how the brain processes information.

8.6 From Research to Product

RealityFlythrough was designed as a research system to identify and accomodate the challenges of creating a live, real-time telepresence experience in real-world situations. We focused on the elements of the real-world that we believed would provide the largest challenges and as a result have ignored some problems that we perceived to be easier or less interesting. Now that the system is maturing, it is time to engineer solutions to these remaining problems.

8.6.1 Multi-viewer Support

RealityFlythrough is currently a single-viewer system with the server playing the role of both server and viewing station. The ideal system would be web-based and allow many producers and consumers of video data. The producers would be the camera operators, and the consumers would be the RealityFlythrough users who explore the remote scenes.

8.6.2 Multi-story and Altitude Support

RealityFlythrough currently assumes that all cameras are five feet off the ground. We have not looked into the accuracy of GPS-based altitude sensors, but even if they are accurate enough for our purposes, raw altitude is not necessarily the appropriate measure for height. In many situations it is most helpful to think of the ground as a plane and it is the height from the ground that is actually important. In buildings it is the height from the various floors of the building that matter. We may find that the best way to handle altitude is to assume the camera *is* five feet above the ground unless the camera operator (or an astute viewer) indicates differently. The floor that the camera operator is on in a building could similarly be manually entered.

8.6.3 Better Hardware

The choice of hardware used in this experimental system was governed by two constraints that are intertwined: cost and what was available. Commodity hardware is cheaper, but is often designed for specific purposes. Part of our effort to demonstrate that RealityFlythrough can work in the real world was to also show that it could be done using commodity hardware. We do not need multi-million dollar equipment to experience flythroughs of live real-world environments.

That being said, the experience would certainly be greatly enhanced with better hardware. The image quality is a function of both the video cameras and bandwidth. The sharper and more detailed the source image, the more bits that are required to encode it – regardless of the form of compression being used. The bandwidth on wireless networks is relatively fixed at the moment, so the resolution of the images cannot go much past 352x288. A better camera than the simple webcams that we used could at least produce focused and light-balanced images. The higher fidelity – the increase in sensory depth – would help quite a bit. As discussed earlier, we could also have the camera operator take explict snapshots of important locations in the scene that could be transmitted at higher resolutions.

It would also be interesting to integrate with existing cameras that are part

of the infrastructure of a building or city. Bandwidth would be less of an issue in these cases as long as the infrastructure is not disturbed by a power outage or more serious disaster. Arrays of high-resolution omnidirectional cameras could provide more seamless walkthroughs of an area [INCT03].

Using better hardware to improve the accuracy of the sensors would improve the aesthetics of the experience and would make the path-planning problem much simpler. GPS devices currently have a typical error of 10 meters. More localized positioning systems can do far better, but currently require a large infrastructure installation effort [HHS+99]. In the near future we may have ultra-wideband solutions that meet our needs and deliver the promised 10 centimeter accuracy [AGCL01].

In the meantime, however, the orientation accuracy can be much improved by using better hardware. We currently rely soley on a magnetic compass to provide the yaw angle and compasses take time to stabilize during rapid movement. More expensive tilt sensors use gyros to compute the angular rotation and periodically recalibrate to the ground truth with compass data when the compass has stabilized. Many of the aesthetic problems with RealityFlythrough transitions are a result of errant tilt-sensor data – improvements in sensing would help greatly.

8.7 Final Thoughts

The above is only a sampling of the future work that is possible with RealityFlythrough. The possibilities are endless and we encourage the reader to join us in further study of live, real-time remote exploration of the real world.

Bibliography

[AGCL01] J.C. Adams, W. Gregorwich, L. Capots, and D. Liccardo. Ultra-wideband for navigation and communications. In *Aerospace Conference, 2001, IEEE Proceedings*, volume 2, pages 785–792, 2001.

[And96] J.D. Anderson. *The reality of illusion: an ecological approach to cognitive film theory.* Southern Illinois University Press, Carbondale, 1996.

[Ari05] M Arisoylu. 802.11 wireless infrastructure to enhance medical response to disasters. In *Proc. AMIA Fall Symp*, 2005.

[BL03] M. Brown and D. G. Lowe. Recognising panoramas. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 1218, Washington, DC, USA, 2003. IEEE Computer Society.

[BR03] Patrick Baudisch and Ruth Rosenholtz. Halo: a technique for visualizing off-screen objects. In *Proceedings of the conference on Human factors in computing systems*, pages 481–488. ACM Press, 2003.

[Bri98] D. Brin. *The Transparent Society*. Perseus Books, 1998.

[BSW05] Matthew Brown, Richard Szeliski, and Simon Winder. Multi-image matching using multi-scale oriented patches. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 510–517, Washington, DC, USA, 2005. IEEE Computer Society.

[BTS+05] Patrick Baudisch, Desney Tan, Drew Steedly, Eric Rudolph, Matt Uyttendaele, Chris Pal, and Richard Szeliski. Panoramic viewfinder: providing a real-time preview to help users avoid flaws in panoramic pictures. In *OZCHI '05: Proceedings of the 19th conference of CHISIG of Australia on CHI*, pages 1–10, Narrabundah, Australia, Australia, 2005. CHISIG of Australia.

[Che95] Shenchang Eric Chen. QuickTime VR — an image-based approach to virtual environment navigation. *Computer Graphics*, 29(Annual Conference Series):29–38, 1995.

[Cla] Peter Clay. Surrogate travel via optical videodisc. Thesis Urb.Stud 1978 B.S., Massachusetts Institute of Technology.

[CNSD93] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 135–142. ACM Press, 1993.

[Dij59] Edsger. W. Dijkstra. A note on two problems in connexion with graphs. In *Numerische Mathematik*, volume 1, pages 269–271. Mathematisch Centrum, Amsterdam, The Netherlands, 1959.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[FPB99] Jr. Frederick P. Brooks. What's real about virtual reality? *IEEE Computer Graphics and Applications*, November/December:16–27, 1999.

[GHJV95] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[Gib] James J. Gibson. *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates.

[GSO95] William W. Gaver, Gerda Smets, and Kees Overbeeke. A virtual window on media space. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 257–264. ACM Press/Addison-Wesley Publishing Co., 1995.

[Gut84] Antonin Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57. ACM Press, 1984.

[Hee03] Carrie Heeter. Reflections on real presence by a virtual person. *Presence: Teleoper. Virtual Environ.*, 12(4):335–345, 2003.

[HHS+99] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 59–68, New York, NY, USA, 1999. ACM Press.

[HHT01] B. Hall, K. Huang, and M. Trivedi. A televiewing system for multiple simultaneous customized perspectives and resolutions. *Proceedings of the Intelligent Transportation Systems Council*, 2001.

[HS92] Jim Hollan and Scott Stornetta. Beyond being there. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 119–125. ACM Press, 1992.

[IAB+96] M. Irani, P. Anandan, J. Bergen, R. Kumar, and S. Hsu. Efficient representations of video sequences and their applications. *Signal Processing : Image Communication*, pages 327–351, 1996.

[INCT03] H. Ishiguro, K. C. Ng, R. Capella, and M. M. Trivedi. Omnidirectional image-based modeling: Three approaches to approximated plenoptic representations. *Machine Vision and Applications*, pages 94–102, 2003.

[JH02] Hank Jones and Pamela Hinds. Extreme work teams: using swat teams as a model for coordinating distributed robots. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 372–381. ACM Press, 2002.

[Joh04] Steven Johnson. *Mind Wide Open: your brain and the neurosciences of everyday life*. SCRIBNER, New York, NY, 2004.

[Kan97] S. B Kang. A survey of image-based rendering techniques. In *Technical Report 97/4, Digital Equipment Corporation.*, 1997.

[KIN+95] Hideaki Kuzuoka, Gen Ishimo, Yushi Nishimura, Ryotaro Suzuki, and Kimio Kondo. Can the gesturecam be a surrogate? In *ECSCW*, pages 179–194, 1995.

[KK04] Oleg Komogortsev and Javed I. Khan. Predictive perceptual compression for real time video communication. In *ACM Multimedia*, pages 220–227, 2004.

[KRVS99a] T. Kanade, P. Rander, S. Vedula, and H. Saito. Virtualized reality: digitizing a 3d time varying event as is and in real time. In *Mixed Reality, Merging Real and Virtual Worlds*. SpringerVerlag, 1999.

[KRVS99b] T. Kanade, P. Rander, S. Vedula, and H. Saito. Virtualized reality: digitizing a 3d time varying event as is and in real time. In *Mixed Reality, Merging Real and Virtual Worlds*. SpringerVerlag, 1999.

[LD97] M. Lombard and T. Ditton. At the heart of it all: The concept of presence. *Journal of Mass Communication*, 3(2), 1997. `http://www.ascusc.org/jcmc/vol3/issue2/lombard.html`.

[LH88] M Livingstone and D Hubel. Segregation of form, color, movement, and depth: anatomy, physiology, and perception. *Science*, 240(4853):740–749, 1988.

[LJDB99] Jason Leigh, Andrew E. Johnson, Thomas A. DeFanti, and Maxine D. Brown. A review of tele-immersive applications in the CAVE research network. In *VR*, pages 180–, 1999.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[MC02] McKinney and Company. *Post 9-11 Report of the Fire Department of New York*. August 2002.

[McC93] S. McCloud. *Understanding comics: The invisble art.* Harper Collins Publishers, New York, 1993.

[MCG05] Neil J. McCurdy, Jennifer N. Carlisle, and William G. Griswold. Harnessing mobile ubiquitous video. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1645–1648, New York, NY, USA, 2005. ACM Press.

[MG04] Neil J. McCurdy and William G. Griswold. Tele-reality in the wild. UBICOMP'04 Adjunct Proceedings, 2004. `http://activecampus2.ucsd.edu/~nemccurd/tele_reality_wild_video.wmv`.

[MG05a] Neil J. McCurdy and William G. Griswold. An abstraction for ubiquitous video. UBICOMP'05 Adjunct Proceedings, 2005. `http://activecampus2.ucsd.edu/~nemccurd/ubicomp05_video.wmv`.

[MG05b] Neil J. McCurdy and William G. Griswold. Harnessing mobile ubiquitous video. Technical Report CS2005-0814, University of California, San Diego, February 2005.

[MG05c] Neil J. McCurdy and William G. Griswold. A systems architecture for ubiquitous video. In *Mobisys 2005: Proceedings of the Third International Conference on Mobile Systems, Applications, and Services*, pages 1–14. Usenix, 2005.

[MGL06] Neil J. McCurdy, William G. Griswold, and Leslie A. Lenert. A robust abstraction for first-person video streaming: Techniques, applications, and experiments. In *ISM '06: Proceedings of the Eighth IEEE International Symposium on Multimedia*, pages 235–244, Washington, DC, USA, 2006. IEEE Computer Society.

[Miy86] N. Miyake. Constructive interaction and the iterative process of understanding. *Cognitive Science*, 10(2):151–177, 1986.

[mpe98] International Organisation for Standardisation: ISO/IEC JTC1/SC29/WG11MPEG 98/N2457. 1998.

[MS99] Daniel Myrick and Eduardo Sanchez. Motion picture: Blair witch project. 1999.

[MTUK] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. In *SPIE Proceedings volume 2351:Telemanipulator and Telepresence Technologies*.

[Nis03] David Nistér. Preemptive ransac for live structure and motion estimation. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 199, Washington, DC, USA, 2003. IEEE Computer Society.

[NYH+03a] Ulrich Neumann, Suya You, Jinhui Hu, Bolan Jiang, and JongWeon Lee. Augmented virtual environments (ave): Dynamic fusion of imagery and 3d models. In *VR '03: Proceedings of the IEEE Virtual Reality 2003*, page 61. IEEE Computer Society, 2003.

[NYH+03b] Ulrich Neumann, Suya You, Jinhui Hu, Bolan Jiang, and JongWeon Lee. Augmented virtual environments (ave): Dynamic fusion of imagery and 3d models. In *VR '03: Proceedings of the IEEE Virtual Reality 2003*, page 61. IEEE Computer Society, 2003.

[PC97] E. PAULOS and J. CANNY. Ubiquitous tele-embodiment: Applications and implications. In *International Journal of Human-Computer Studies/Knowledge Acquisition*, 1997.

[Pet89] J. Timothy Petersik. The two-processs distinction in apparent motion. *Psychological Bulletin*, 106:118, 1989.

[PPW97] Randy Pausch, Dennis Proffitt, and George Williams. Quantifying immersion in virtual reality. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 13–18. ACM Press/Addison-Wesley Publishing Co., 1997.

[RA86] Vilayanur S. Ramachandran and Stuart M. Anstis. The perception of apparent motion. *Scientific American*, 254(6):80–??, June 1986.

[Ros] Becky Rosner. Awareness is key to personal safety. In *The Current Online (September 7, 2004)*. Retrieved January 18, 2005, from http://www.thecurrentonline.com.

[RS01] G. Reitmayr and D. Schmalstieg. Mobile collaborative augmented reality. In *Proceedings of the 2nd International Symposium on Augmented Reality (ISAR 2001)*, New York, USA, 2001.

[SD96] S. M. Seitz and C. R. Dyer. View morphing. In *Proc. SIGGRAPH 96*, pages 21–30, 1996.

[SH01] Jussi Suomela and Aarne Halme. Tele-existence techniques of heavy work vehicles. *Auton. Robots*, 11(1):29–38, 2001.

[Sho97] R. Showen. Operational gunshot location system. In A. T. Depersia, S. Yeager, and S. Ortiz, editors, *Proc. SPIE Vol. 2935, p. 130-139, Surveillance and Assessment Technologies for Law Enforcement, A. Trent DePersia; Suzan Yeager; Steve Ortiz; Eds.*, volume 2935 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 130–139, February 1997.

[SPS05] Drew Steedly, Chris Pal, and Richard Szeliski. Efficiently registering video into panoramic mosaics. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 2*, pages 1300–1307, Washington, DC, USA, 2005. IEEE Computer Society.

[SSS06] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 835–846, New York, NY, USA, 2006. ACM Press.

[Ste95] Jonathan Steuer. Defining virtual reality: Dimensions determining telepresence. *Communication in the Age of Virtual Reality*, pages 33–56, 1995.

[Sze94] R. Szeliski. Image mosaicing for tele-reality applications. In *WACV94*, pages 44–53, 1994.

[Tac98a] Susumu Tachi. Real-time remote robotics - toward networked telexistence. In *IEEE Computer Graphics and Applications*, pages 6–9, 1998.

[Tac98b] Susumu Tachi. Real-time remote robotics - toward networked telexistence. In *IEEE Computer Graphics and Applications*, pages 6–9, 1998.

[TP03] Bruce H. Thomas and Wayne Piekarski. Outdoor virtual reality. In *ISICT '03: Proceedings of the 1st international symposium on Information and communication technologies*, pages 226–231. Trinity College Dublin, 2003.

[VS06] Arunchandar Vasan and A. Udaya Shankar. An empirical characterization of instantaneous throughput in 802.11b wlans. `http://www.cs.umd.edu/~shankar/Papers/802-11b-profile-1.pdf`, 2006.

[Wei95] Mark Weiser. The computer for the 21st century. *Human-computer interaction: toward the year 2000*, pages 933–940, 1995.

[Wil95] D. Wildman. Getting the most from paired-user testing. *ACM Interactions*, 2(3):21–27, 1995.

[WSBL03] Thomas Wiegand, Gary J. Sullivan, Gisle Bjntegaard, and Ajay Luthra. Overview of the h.264/avc video coding standard. *IEEE Trans. Circuits Syst. Video Techn.*, 13(7):560–576, 2003.