UNIVERSITY OF CALIFORNIA
SANTA BARBARA

Interactive Remote Collaboration Using Augmented Reality

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Steffen Gauglitz

September 2014

The dissertation of
Steffen Gauglitz is approved:

Pradeep Sen

Susan Fussell

Gerhard Reitmayr

Tobias Höllerer, Committee Co-Chair

Matthew Turk, Committee Co-Chair

August 2014

Interactive Remote Collaboration Using Augmented Reality

Acknowledgements

I thank my advisors Tobias Höllerer and Matthew Turk for advising, supervising, guiding and supporting me throughout graduate school. You have taught me about the art and science of human-computer interaction and computer vision, about research, academia, writing grants and papers. I feel that it has always been a comfortable environment and respectful relationship, which I believe cannot be overrated.

I thank Sue Fussell, Gerhard Reitmayr, and Pradeep Sen for serving on my dissertation committee: You have provided expertise and feedback from other angles; and I believe that this has made this work stronger and more well-rounded.

I thank Qualcomm, and in particular Serafin Diaz, Daniel Wagner and Alessandro Mulloni, for making an important piece of software available to us for this work, and for supporting us to put it to use most effectively. This has allowed us to build upon a very reliable solution and concentrate on other research aspects; I hope that the collaboration was to mutual benefit.

I thank my collaborators and co-authors Lukas Gruber, Stefanie Zollmann, Victor Fragoso, Luca Foschini, Cha Lee, Jon Ventura, Chris Sweeney, and in particular Ben Nuernberger; you have helped me to implement and realize ideas and write great papers about them. I hope that the collaborations and friendships may endure. I also thank all other members, and visitors, of the Four Eyes Lab (and my advisors for attracting them); all of you let me bounce ideas off, have given me input on my work, or helped shape it in other ways.

I further thank Janet Kayfetz, Andy Beall, Yuqi Chen and Daniel Sheinson, who have provided expertise and advised me professionally and/or scientifically in different ways.

Last, but certainly not least, I thank my parents for supporting me to go on an adventure 9000 kilometers away, and my wonderful wife, Julia, for supporting me throughout this journey, proofreading, bringing food to lab, lending her voice to our videos, celebrating successes and enduring stressful times, and spending lots of time with our daughter on top of pursuing her own career.

Curriculum Vitæ

Steffen Gauglitz

September 2014

Education

2014	Doctor of Philosophy, Computer Science, UC Santa Barbara
2008	Diplom-Ingenieur, Computer Engineering, RWTH Aachen University

Work Experience

2012	Intern, Google, Inc.
2010	Intern, Qualcomm, Inc. Corporate Research & Development
2008 – 2014	Graduate student research assistant, Department of Computer Science, UC Santa Barbara
2007 – 2008	Intern, WorldViz, LLC
2005 – 2006	Student research assistant, Institute of Man-Machine Interaction, RWTH Aachen University (part time)
2003 – 2005	Programmer, linear GmbH (part time)

Awards & Fellowships

2014	Outstanding Dissertation Honorable Mention, Department of Computer Science, UC Santa Barbara
2013	Second-best Paper Award, UC Santa Barbara Computer Science Graduate Student Workshop on Computing
2012	Best Paper Award, IEEE International Symposium on Mixed and Augmented Reality (ISMAR)
2012	Best Paper Honorable Mention, ACM SIGCHI's International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI)
2012	Outstanding Graduate Student Award, Department of Computer Science, UC Santa Barbara
2010	Outstanding Teaching Assistant Award, Department of Computer Science, UC Santa Barbara
2010	Springorum Coin, RWTH Aachen University, "in recognition of outstanding academic achievement"
2008 – 2013	Chancellor's Fellowship, UC Santa Barbara

2006 – 2007 Fellow of the Deutsche Akademische Austausch Dienst (German Academic Exchange Service)

2003 – 2008 Fellow of the Studienstiftung des deutschen Volkes (German National Academic Foundation)

Publications

In Journals:

S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Höllerer: Model estimation and selection towards unconstrained real-time tracking and mapping. In *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol.20, no.6, pp.825-838, 2014

S. Gauglitz, T. Höllerer, and M. Turk: Evaluation of interest point detectors and feature descriptors for visual tracking. In *International Journal of Computer Vision (IJCV)*, vol.94, no.3, pp.335-360, 2011

In Conference Proceedings:

S. Gauglitz, B. Nuernberger, M. Turk, and T. Höllerer: In Touch with the Remote World: Remote Collaboration with Augmented Reality Drawings and Virtual Navigation. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST) 2014*, Edinburgh, UK, *to appear*

S. Gauglitz, B. Nuernberger, M. Turk, and T. Höllerer: World-stabilized Annotations and Virtual Scene Navigation for Remote Collaboration. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST) 2014*, Honolulu, USA, *to appear*

S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Höllerer: Live tracking and mapping from both general and rotation-only camera motion. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR) 2012*, Atlanta, USA

S. Gauglitz, C. Lee, M. Turk, and T. Höllerer: Integrating the physical environment into mobile remote collaboration. In *Proceedings of the ACM SIGCHI International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI) 2012*, San Francisco, USA

S. Gauglitz, M. Turk, and T. Höllerer: Improving keypoint orientation assignment. In *Proceedings of the British Machine Vision Conference (BMVC) 2011*, Dundee, UK

S. Gauglitz, L. Foschini, M. Turk, and T. Höllerer: Efficiently selecting spatially distributed keypoints for visual tracking. In *Proceedings of the IEEE International Conference on Image Processing (ICIP) 2011*, Brussels, Belgium

V. Fragoso, S. Gauglitz, S. Zamora, J. Kleban, and M. Turk: TranslatAR: a mobile augmented reality translator. In *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV) 2011*, Kona, USA

L. Gruber, S. Gauglitz, J. Ventura, S. Zollmann, M. Huber, M. Schlegel, Gudrun Klinker, Dieter Schmalstieg, and Tobias Höllerer: The City of Sights: design, construction, and measurement of an augmented reality stage set. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR) 2010*, Seoul, Korea

B. Sturm, J. Shynk, and S. Gauglitz: Agglomerative clustering in sparse atomic decompositions of audio signals. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2008*, Las Vegas, USA

Extended Abstracts/Posters:

B. Nuernberger, S. Gauglitz, T. Höllerer, and M. Turk: Investigating viewpoint visualizations for Click & Go navigation. In *Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI) 2014*, Minneapolis, USA

C. Lee, S. Gauglitz, T. Höllerer, and D. A. Bowman: Examining the equivalence of simulated and real AR on a visual following and identification task. In *Proceedings of IEEE Virtual Reality (VR) 2012*, Orange County, USA

S. Gauglitz, T. Höllerer, P. Krahwinkler, and J. Roßmann: A setup for evaluating detectors and descriptors for visual tracking. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR) 2009*, Orlando, USA

Abstract

Interactive Remote Collaboration Using Augmented Reality

Steffen Gauglitz

With the widespread deployment of fast data connections and availability of a variety of sensors for different modalities, the potential of remote collaboration has greatly increased. While the now ubiquitous video conferencing applications take advantage of some of these capabilities, the use of video between remote users is limited to passively watching disjoint video feeds and provides no means for interaction with the remote environment. However, collaboration often involves sharing, exploring, referencing, or even manipulating the physical world, and thus tools should provide support for these interactions.

We suggest that augmented reality is an intuitive and user-friendly paradigm to communicate information about the physical environment, and that integration of computer vision and augmented reality facilitates more immersive and more direct interaction with the remote environment than what is possible with today's tools.

In this dissertation, we present contributions to realizing this vision on several levels. First, we describe a conceptual framework for unobtrusive mobile video-mediated communication in which the remote user can explore the live scene independent of the local user's current camera movement, and can communicate information by creating spatial annotations that are immediately visible to the local user in augmented reality. Second, we describe the design and implementation of several, increasingly more flexible and immersive user interfaces and system prototypes that implement this concept. Our systems do not require any preparation or instrumentation of the environment; instead, the physical scene is tracked and modeled incrementally using monocular computer vision. The emerging model then supports anchoring of annotations, virtual navigation, and synthesis of novel views of the scene. Third, we describe the design, execution and analysis of three user studies comparing our prototype implementations with more conventional interfaces and/or evaluating specific design elements. Study participants overwhelmingly preferred our technology, and their task performance was significantly better compared with a video-only interface, though no task performance difference was observed compared with a "static marker" interface. Last, we address a particular technical limitation of current monocular tracking and mapping systems which was found to be impeding and present a conceptual solution; namely, we describe a concept and proof-of-concept implementation for automatic model selection which allows tracking and modeling to cope with both parallax-inducing and rotation-only camera movements.

We suggest that our results demonstrate the maturity and usability of our systems, and, more importantly, the potential of our approach to improve video-mediated communication and broaden its applicability.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the widespread deployment of fast data connections and availability of a variety of sensors for different modalities, the potential of remote collaboration has greatly increased. While the now ubiquitous video conferencing applications take advantage of some of these capabilities, they lack the ability to interact with the remote physical environment; the use of video between remote and local users is limited largely to passively watching disjoint video feeds. This is aptly summarized by Gelb et al. (2011): "Current systems, however, do a poor job of integrating video streams [...]. Real and virtual content are unnaturally separated, leading to problems with nonverbal communication and the overall conference experience." Thus, teleconference-like applications have been largely successful when the matter at hand can be discussed verbally or with the help of purely digital data (such as presentations slides), but they hit severe limitations when real-world objects or environments are involved.

As effective collaboration often involves sharing, exploring, referencing, or even manipulating the physical environment, tools for remote collaboration should provide support for these interactions. For example, gesturing and pointing are very natural and effective parts of human communication, without which communication can be ineffective and frustrating ("If I could just point to it, its right there," "if only I could show you how to do it" (Fussell et al., 2004)). To incorporate these means of communication, researchers have explored various ways to support remote pointing. However, two of the most common limitations in existing work are that the remote user's view onto the scene is constrained to the typically small field of view of the local user's camera, and that any support for spatially referencing the scene is contingent upon a stationary camera, as otherwise the pointers lose their referents.

Advances in computer vision facilitate applications which are to some degree able to understand where mobile cameras are pointed and what is seen. These new capabilities should be exploited to enable the remote user to interact with what he/she sees, instead of forcing him/her to passively watch whatever is in view of the local user's camera. Further, we suggest that mobile augmented reality (AR) provides a natural and user-friendly paradigm to communicate spatial information about the scene and to browse an environment remotely. This concept leads to the following thesis statement:

> Integrating computer vision and augmented reality can significantly improve video-mediated communication and broaden its applicability. In particular, integrating those technologies enables telecommunication participants to effectively reference and communicate information about the remote physical environment. Significant benefits can be achieved without instrumentation of

the environment and with off-the-shelf hardware with relatively low sensory fidelity, making this approach suitable for widespread adoption.

In this dissertation, we support this thesis with contributions on multiple levels:

- We describe a system framework for unobtrusive mobile remote collaboration that integrates the physical environment. Our concept does not require specialized equipment or preparation of the environment and is compatible with a wide range of hardware configurations, including systems that are already ubiquitous (e.g., smartphones) as well as more advanced immersive systems (Chapter 3).
- We designed three prototypes that implement the aforementioned framework and feature several novel interface elements for unobtrusive mobile telecollaboration in unprepared environments. Our prototypes use model-free, markerless, expanding visual tracking and modeling to enable a remote user to provide visual/spatial feedback by means of world-stabilized annotations that are displayed to a local user in AR. Further, our system enables *decoupling* of the local user's view from the remote user's view while maintaining live updates. This gives the remote user control over his/her viewpoint and allows him/her to study and annotate (and thus direct attention to) parts of the scene that are not currently in the field of view of the local user. The prototypes' interfaces include several other novel features, such as the interpretation of 2D drawings as world-stabilized annotations in 3D and virtual navigation designed specifically to explore a partially modeled remote scene (Chapter 4).

- We conducted user studies to validate the usability of our prototypes and evaluate their benefits over more conventional interfaces, and discuss both quantitative results and qualitative observations in detail. To our knowledge, our studies are among the first formal user studies overall to rely solely on markerless, model-free visual tracking (Chapter 5).
- Lastly, we address one particular limitation of current visual environment modeling, as an important enabling technology of our framework, and describe a conceptual solution and proof-of-concept implementation. Namely, we describe a novel approach to live vision-based environment modeling which supports both general (parallax-inducing) and degenerate (rotation-only) camera motions in environments of arbitrary geometric complexity (Chapter 6).

We will discuss related work in Chapter 2, then present these contributions in the above order. We conclude in Chapter 7.

Authorship. I am the primary author of all aspects of the work constituting this dissertation. However, in addition to the guidance of my advisors Tobias Höllerer and Matthew Turk on all parts of this work — and several people to whom we are indebted for providing additional advice, as expressed in the acknowledgements — several of my colleagues have contributed to individual parts of this work, as attributed via their co-authorship on the respective peer-reviewed publications.

Specifically, in the order of appearance of the respective contents in this dissertation: Ben Nuernberger has contributed to the design, implementation, and description of prototypes 2 and

3 (Sections 4.2 and 4.3) as well as the administration, analysis, and description of user study 2 (Section 5.2); Cha Lee has contributed to the design, administration, analysis, and description of user study 1 (Section 5.1); and Chris Sweeney and Jon Ventura have contributed to the formulation, implementation, testing, and description of the tracking and modeling approach described in Chapter 6.

Chapter 2

Related Work

2.1 Video-Mediated Communication

Research on video-mediated communication is multifaceted. The body of research includes investigations on various video configurations (Kraut et al., 1996; Gergle et al., 2004; Fussell et al., 2000, 2003a) and systems aiming at increasing the level of immersion of teleconferences by transferring live three-dimensional or perspectively corrected imagery of the participants (Raskar et al., 1998; Regenbrecht et al., 2004; Sadagic et al., 2001; Kurillo et al., 2008; Maimone and Fuchs, 2011; Maimone et al., 2013). Collaboration is possible on purely virtual data (as demonstrated in Sadagic et al., 2001; Regenbrecht et al., 2004; Kurillo et al., 2008), but spatial references to the physical world are not supported.

There are several AR frameworks that focus on collaborative work and more mobile infrastructure (Billinghurst et al., 1998a,b; Butz et al., 1999; Reitmayr and Schmalstieg, 2001). However, most of these also facilitate collaboration on virtual data only.[1]

2.2 Spatial References to the Remote Physical Environment

Support for spatial references to the remote physical scene has been an active research topic. Notable early works include "VideoDraw" (Tang and Minneman, 1991) and the "DoubleDigitalDesk" (Wellner and Freeman, 1993).

While many of the above systems focus on symmetric setups (i.e., both participants have the same equipment and share their own environment to the same degree), further work has been done on asymmetric local worker/remote expert scenarios (Alem et al., 2011; Bauer et al., 1999; Fussell et al., 2004; Kurata et al., 2004; Chastine et al., 2008; Ou et al., 2003; Kirk, 2006). These systems typically focus on tasks involving objects in the local worker's physical environment ("collaborative physical tasks" (Kirk, 2006)).

Various ways to support the visual/spatial referencing have been studied, for example remote pointers or markers (Bauer et al., 1999; Fussell et al., 2004; Chastine et al., 2008; Kim et al., 2013), laser pointers (Kurata et al., 2004), drawing onto a live video stream (Ou et al., 2003; Kirk and Fraser, 2006; Fussell et al., 2004; Gurevich et al., 2012; Kim et al., 2013),

[1]Cf. the following quote by Butz et al. (1999): "We present [...] a prototype experimental user interface to a collaborative augmented environment. Users share a 3D virtual space and manipulate virtual objects that represent information to be discussed."

or directly transferring videos of hand gestures (Kirk and Fraser, 2006; Li et al., 2007; Alem et al., 2011; Kirk, 2006; Huang and Alem, 2013; Oda et al., 2013). These annotations are then displayed to the collaborator on a separate screen in a third-person perspective (Fussell et al., 2004; Huang et al., 2013), on a head-worn display (Bauer et al., 1999; Huang and Alem, 2013; Oda et al., 2013), or via projectors (Gurevich et al., 2012).

The studies by Fussell et al. (2004), Bauer et al. (1999), Chastine et al. (2008), and Kurata et al. (2004) are especially pertinent to our work in several regards. In Fussell et al. (2004), the local user had to assemble a toy robot with guidance from the remote user. The remote user saw the local user's workspace by means of a static camera looking over the local user's shoulder, while the local user saw the same view on a separate monitor in front of him/her. The remote user controlled a cursor which was visible on both screens.

In Bauer et al. (1999), the local user was wearing a video-see-through head-worn display (HWD) and had to solve a puzzle-like task. The remote expert saw the video and controlled a pointer visible to both of them. To be able to accurately reference objects despite movement of the local camera, the remote expert could freeze the video. However, they did not employ any kind of tracking and thus had to freeze the local worker's view simultaneously to be able to display the pointer.

In Chastine et al. (2008), the local user was asked to build a structure of wooden blocks, for which the remote user sees a virtual model in AR. Fiducial marker-based tracking was used to establish a shared coordinate system for both the virtual model (on the remote user's side) and the physical model (to be built on the local user's side), and the remote user could place pointers by placing additional fiducial markers around the virtual object. Note that in their setup, the virtual model serves not only as "expert knowledge," but additionally as surrogate for the physical model when placing the markers around it.

Kurata et al. (2004) developed a shoulder-worn device with a camera and laser pointer mounted on a pan-tilt unit which is controlled by a remote user. The remote user thus has some control over the viewpoint; in addition, their system can build pseudo-panoramic maps (while the local user is holding still).

However, in order to support spatially referencing objects in the physical world, all of these systems either assume a static camera (at least for the duration of the relevant interaction), since otherwise virtual annotations lose their referents (Fussell et al., 2003b, 2004; Bauer et al., 1999; Kurata et al., 2004; Kirk and Fraser, 2006; Kirk, 2006; Li et al., 2007; Alem et al., 2011; Gurevich et al., 2012; Huang and Alem, 2013; Kim et al., 2013), or require extensive equipment and prepared environments to track and thus maintain the annotations' locations (Chastine et al., 2008; Oda et al., 2013). Furthermore, in all of these systems, the remote user's view into the local environment is either restricted to a static camera (Fussell et al., 2004; Kirk and Fraser, 2006) or tightly coupled to the local user's head or body movement (Bauer et al., 1999; Alem et al., 2011; Kurata et al., 2004; Chastine et al., 2008; Huang and Alem, 2013; Kim et al., 2013), thus forcing the remote user to constantly re-orient and ask the local user to hold still (or, in the case of Bauer et al. (1999), enforcing this by freezing both users' views) while pointing at an object.

2.3 World-stabilized Spatial References

To overcome the restrictions mentioned above, we leverage computer vision-based tracking and modeling and the paradigm of AR in order to (a) correctly anchor virtual annotations to their real-world referents despite a moving camera, and (b) *decouple* the views of local and remote user, thus enabling the remote user to explore the environment virtually.

Using world-stabilized annotations for the purpose of remote collaboration has been envisioned before. Davison et al. (2003) describe this goal, and present a Simultaneous Localization and Mapping system (cf. Section 2.4) for a shoulder-worn robot with camera towards its realization. Similarly, Reitmayr et al. (2007) discuss tracking of landmarks of varying complexity in order to anchor annotations for the same purpose. Ladikos et al. (2008) present a tracking system to maintain the location of annotations on planar objects. Also designed for planar environments, the system by Lee and Höllerer (2006) supports not only world-stabilized drawings, but also viewpoint stabilization for the remote user, which is closely related to our notion of camera control.

AR-based collaboration between a local ("outdoor") user and a remote ("indoor") user was also discussed by Höllerer et al. (1999) and Stafford et al. (2006), albeit using sensor-based localization and maps and/or 3D models of the local user's environment rather than real-time capture and reconstruction.

Sodhi et al. (2013) present a system which is very closely related to our work. They use a customized setup with a mobile device and two active depth sensors mounted onto a monopod which gives the ability to reconstruct the environment (on the local user's side) and to reconstruct and transfer hand gestures (from the remote user's side) in 3D. In contrast, our system needs only an off-the-shelf tablet, but uses simpler annotations. Another difference worth noting is the method of scene navigation by the remote user: Sodhi et al. equip the remote user with a mobile device as well and use the device's physical movement to navigate the remote (virtual) environment, while we use virtual navigation. We will contrast the two approaches with respect to both navigation and input modality in more detail in Sections 4.2.5 and 4.3.1, respectively. They conducted a usability study reporting results with respect to ease of use, etc., but no comparative or performance-based evaluation was reported.

The system by Adcock et al. (2013) also reconstructs the environment via active depth sensors and allows the remote user to control the viewpoint (via a touch interface) and draw annotations; however, these annotations are displayed using a statically mounted projector.

Further, Jo and Hwang (2013) presented an interactive, fully mobile system which allows for world-stabilized drawings, albeit only for panoramas (i.e., rotation movements). Two particular interesting aspects of this work are the physical navigation of the panorama by the remote user and switching between front and back cameras (upon permission by the local user).

One alternative to a wearable setup is to use a camera and projective device mounted to a robot which is controlled by the remote user (Kuzuoka et al., 2000; Gurevich et al., 2012). However, this requires specialized hardware which needs to be carried around and put in place, and the range and speed of operation are limited by the robot.

2.4 Vision-based Tracking and Mapping

A key component of our approach is the integration of vision-based tracking and mapping of the environment to support navigating the environment and anchoring of annotations to it. Our work is thus related to work in vision-based tracking and mapping, which are important enabling technologies for the area of AR in general.

Monocular vision-based Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is the problem of determining the pose of the observer relative to an unknown environment while concurrently creating a model of the environment (which may have arbitrary geometric complexity) as the observer moves around. In the case of monocular vision-based SLAM (Davison et al., 2007), the only sensor used to accomplish this task is a single camera.

Filter-based SLAM systems (Civera et al., 2008a; Davison et al., 2007; Eade and Drummond, 2008) maintain estimates of both camera and feature positions (i.e., the map) in a large state vector which is updated using Kalman filters in each frame. In contrast, keyframe-based systems (Klein and Murray, 2007, 2008; Mouragnon et al., 2006) track features in each frame, but use only selected frames to update their map, typically using bundle adjustment for the latter. While all aforementioned systems are based on sparse features, Newcombe et al. (2011a) presented a keyframe-based system that uses dense mapping. They report extremely robust results, but (in contrast to the above systems) require a powerful GPU for real-time operation.

In both types of systems, the map is designed to store structure-from-motion (SfM) data, that is, feature positions in 3D that have been triangulated using observations from multiple viewpoints. Thus, they require parallax-inducing camera motion in order to bootstrap their map (Davison et al., 2007; Klein and Murray, 2007; Newcombe et al., 2011a), as otherwise, the features cannot be triangulated and integrated into the map. In some systems (Klein and Murray, 2007; Newcombe et al., 2011a), the initialization is performed as a dedicated separate step, and tracking quality crucially depends on the quality of this initialization. Rotation-only motions are supported only if they are constrained to the already observed part of the scene.

For filter-based systems, an alternative, six-dimensional parametrization of the feature locations (Civera et al., 2008a) can provide a remedy: Here, rotation-only motions are supported by admitting features with a depth prior that represents extreme uncertainty and filtering the features through multiple motion models (Civera et al., 2008b) to constrain their uncertainty. However, this support comes at a high computational cost: The already high cost of filtering of each feature point is further increased by doubling the dimensionality of the feature state vector as well as computing the results for multiple motion models (note that Civera et al. (2008b) use seven models in each frame).

We will return to this particular problem in Chapter 6.

Panorama tracking and mapping

Like a SLAM system, a panorama tracking and mapping system aims at modeling the environment while determining the pose of the camera,[2] but in this case, the camera is assumed to rotate around its optical center, so that only its orientation has to be determined. An early real-time system is Envisor (DiVerdi et al., 2009). Wagner et al. (2010) describe a system that operates robustly and in real time on a mobile device.

The tracking system that we developed for our first prototype system (Section 4.1) is technically equivalent to a panorama tracking system.

2.5 Virtual Navigation

Virtual navigation is a large research area in itself, including a large body of work concerning 3D navigation from 2D inputs (Hanson and Wernert, 1997; Zeleznik and Forsberg, 1999; Tan et al., 2001; Hachet et al., 2008; Christie and Olivier, 2009; Jankowski and Hachet, 2013; Bowman et al., 2005; Marchal et al., 2013). Most of these works assume that the scene to be navigated is known completely (that is, it can be rendered from any viewpoint, such as an architectural model, the virtual environment in a game, etc.).

[2] Depending on the exact interpretation of the word "localization," one may argue that a panorama tracking and mapping system *is* a SLAM system, as advocated by Lovegrove and Davison (2010). This interpretation would render the term "tracking and mapping," which we chose to refer to the general class following Klein and Murray (2007); Pirchheim and Reitmayr (2011); Wagner et al. (2010), obsolete.

In our context, the scene is only partially reconstructed from image data, with highest rendering fidelity from a constrained set of previously visited viewpoints, and can be navigated only based on this reconstruction. In this respect, our work bears similarity in particular with the work by Snavely et al.'s "Photo tourism" (Snavely et al., 2006) and similar works. However, while Photo tourism deals with a (potentially large) set of photos in a batch manner and offers offline navigation, we process a live video in real time and offer live viewing of the evolving model.

Chapter 3

A Concept for Mobile Unobtrusive Remote Collaboration

Figure 3.1 illustrates the proposed concept which enables the remote user to explore a physical environment by means of live imagery from a camera that the local user holds or wears. The remote user is able to interact with the model fused from these images by creating virtual annotations in it or transferring live imagery (e.g., of gestures) back.

At the core of the concept is a tracking and environment modeling core, which enables the system to (1) synthesize novel views of the environment and thus decouple the remote user's viewpoint from that of the local user, giving the remote user some control over his/her viewpoint, and (2) correctly register virtual annotations to their real world referents.

This concept has first been presented in our paper in the proceedings of the ACM International Conference on Human-Computer Interaction with Mobile Devices and Services 2012 (Gauglitz et al., 2012a).

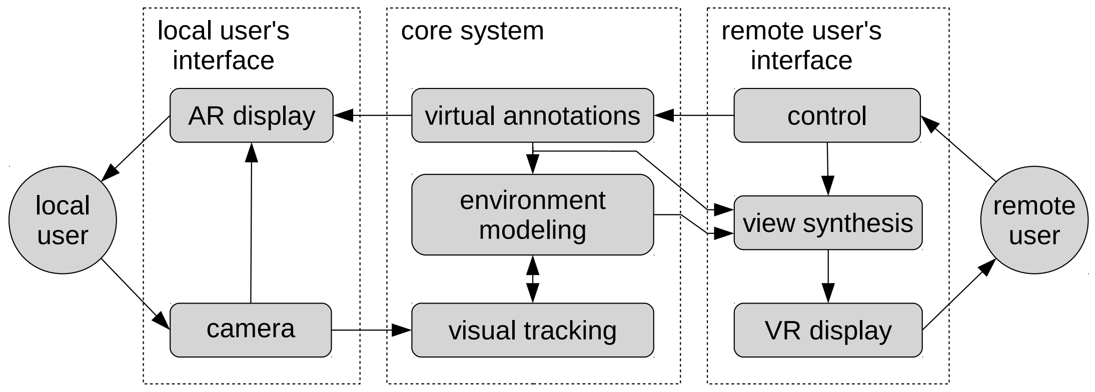


Figure 3.1: Overview of the proposed concept for unobtrusive, mobile remote collaboration that integrates the physical environment. This figure depicts the situation that the user on the left is situated in the physical task environment and thus assumes the role of the local user, while the user on the right assumes the role of the remote user.

3.1 Local User's Interface

The local user is assumed to hold or wear a device that integrates a camera and a display system (e.g., hand-held tablet or head-worn display with camera), which is used to both sense the environment and display visual/spatial feedback from the remote user correctly registered to the real world. In the case of a video see-through hand-held device, it acts as a *magic lens*, that is, it shows the live camera feed plus virtual annotations. Since a collaboration system has to aid the user in his/her actual task rather than distract from it, an interface which is simple and requires a minimal mental load to operate is essential. It should facilitate an active user who may be looking at and working in multiple areas.

3.2 Remote User's Interface

The remote user is presented with a view into the local user's environment, rendered from images obtained by the local user's camera. The remote user can place annotations that will be displayed to both users, correctly registered to their real-world referents from their respective point of views. Annotations may include point-based markers, more complex three-dimensional annotations, drawings, or live imagery (e.g., of hand gestures).

In the simplest case, the remote user's viewpoint may be restricted to being identical to the local user's current camera view. In this case, no further image synthesis is needed. Ideally, however, the remote user should be able to decouple his/her viewpoint and control it independently, as far as supported by the available imagery of the environment.

If the system allows for decoupled views, it is important that only the *viewpoint* is decoupled; the video is still synthesized and updated from live images in order to enable consistent communication. (For example, in the case of our first prototype system, the effect of this can be observed in Figure 4.2(b): although the viewpoints are different, the remote user can observe how the local user is pointing to a control element on the panel in front of him.)

3.3 Visual Tracking & Environment Modeling

We assume that the environment may be completely unknown prior to the start of the system, that is, we do not require any model information. While using model information bears the potential to make the system more robust, any kind of model information has to be collected in some way prior to the task, which either severely limits the generality of the system or puts a burden on the user.

Instead, we assume that the system starts with no prior knowledge about the scene and builds up an internal representation on the fly, which automatically expands to include new areas as the camera moves. This does not preclude the use of additional data that may be available from online sources — cf. the “Anywhere Augmentation” paradigm (Höllerer et al., 2007) — to provide additional information should a specific location or object have been identified, but it means that our system will not require such data (in particular, no model of the environment), in order to operate. Our concept is compatible with environment modeling systems of different levels of flexibility and generality, including panorama mapping and SLAM. If admissible by the application, other sensors such as active depth cameras could also be used (Newcombe et al., 2011b).

3.4 Hardware

We note that only very few requirements for the hardware that this concept could run on were specified thus far. Indeed, this concept is compatible with a wide range of hardware systems, including systems that are already ubiquitous (e.g., smartphones), as well as more immersive systems which may serve specialized applications or become more wide-spread in the future. This flexibility is important as the decision of which type of hardware to use may depend on external factors such as cost, market penetration and task- or environment-specific considerations.

In particular, our concept is compatible with various types of displays for the local user, including smartphones or tablets, head-worn displays (used in the context of remote collaboration for example by Bauer et al. (1999); Huang and Alem (2013); Oda et al. (2013)), as well as projective displays (used in the context of remote collaboration for example by Gurevich et al. (2012)). To demonstrate that our concept does not hinge on benefits of more immersive technology and is suitable for wide-spread adoption, we used hand-held tablets in this work.

Likewise, the remote user's interface could consist of a standard PC interface with screen, mouse, keyboard; a touchscreen; a head-worn (and -tracked) virtual reality (VR) display; or a multitude of other VR displays. With the system prototypes described in Chapter 4, we explore the first two options.

Chapter 4

Design of Interfaces & Implementation of Prototype Systems

In this chapter, we describe the design and implementation of three prototype systems that implement the concept presented in Chapter 3. From Prototype 1 to 3, they describe an evolution from an initial proof-of-concept system to a flexible, feature-rich mobile system that allows untrained users to interact with the remote environment and communicate spatial information. Important characteristics of and differences between the three prototypes are summarized in Table 4.1.

Prototype 1 (Section 4.1) is an initial proof-of-concept implementation. It is not actually networked, but uses two interfaces connected to the same physical computer. It is a complete, but in several ways the simplest or least immersive implementation of our concept. Most notably, the remote user's camera control is limited to a "freeze" (and "un-freeze") feature. Technically, it is limited in generality in one particular aspect: its tracking and modeling capabilities are restricted to homographic warps, that is, it is compatible with planar scenes or rotation-only

	Prototype 1 Section 4.1	**Prototype 2** Section 4.2	**Prototype 3** Section 4.3
System architecture	both users' interf. tethered to same physical computer	independent entities communicating via wireless network	
Local user's hardware interface	hand-held (but tethered) display & camera	off-the-shelf Android-based tablet or smartphone	
Remote user's hardware interface	screen, keyboard, mouse		touchscreen
Environment modeling	restricted to planar scenes	arbitrary geometric complexity	
Virtual navigation/ camera control	live & “freeze”	flexible navigation	flexible gesture-based navigation
Annotations	single-point markers		single-point markers & freehand drawings

Table 4.1: Characteristics of the three prototype implementations.

movements, but not with general camera motion in general environments. With the user study described in Section 5.1, we will show that even with this low-immersion implementation, our concept provides significant benefits in a remote collaborative task.

Prototype 2 (Section 4.2) is a more general and feature-rich implementation. In particular, (1) it consists of two stand-alone entities communicating via wireless network, (2) it operates in environments of arbitrary geometric complexity, and (3) it enables the remote user to explore the remote scene via a set of virtual navigation controls. This prototype is evaluated via a second user study which will be described and discussed in Section 5.2.

Prototype 3 (Section 4.3) builds upon Prototype 2, sharing much of the underlying system infrastructure and architecture, but provides a more intuitive and immersive user experience for the remote user, and improves individual interface aspects by building upon user feedback from the second study. In particular, we incorporated a touchscreen interface to allow more direct interaction, added support for free-hand drawings as more flexible and expressive annotations, and added gesture-based virtual navigation. Adding these new elements involved new, unique design challenges, which are evaluated via a qualitative user study to be described in Section 5.4.

4.1 Prototype 1

The prototype presented in this section has first been described in our paper in the proceedings of the ACM International Conference on Human-Computer Interaction with Mobile Devices and Services 2012 (Gauglitz et al., 2012a).

4.1.1 Local user's interface

As hardware interface for the local user, we decided to use a hand-held tablet screen, since we wanted to show that benefits can be achieved using hardware setups that are already in widespread deployment. This prototype being the initial proof-of-concept system, for ease of implementation and flexibility with respect to the hardware components to be used, we use a USB-driven screen with a camera mounted on the back (Figure 4.1) instead of a stand-alone

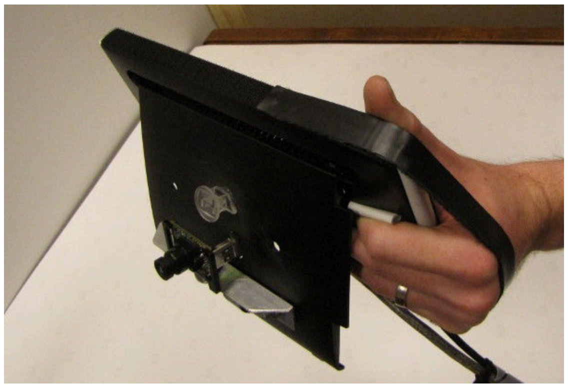
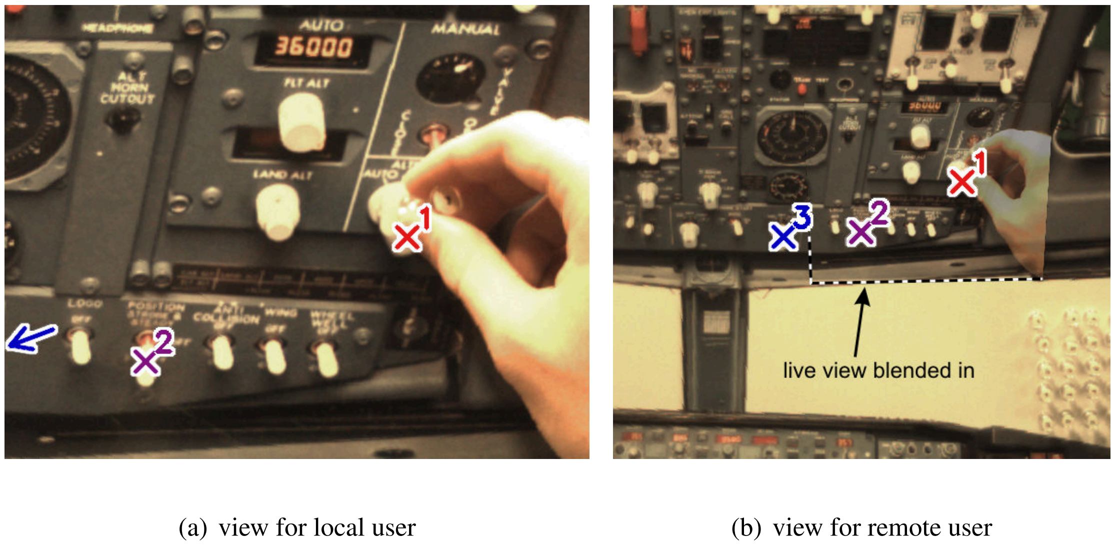


Figure 4.1: Hand-held device for the local user: 10” USB tablet screen with camera mounted on back.



(a) view for local user

(b) view for remote user

Figure 4.2: View for local and remote user. The remote user’s viewpoint (b) is frozen, but the live video frame is — correctly registered with the frozen frame — blended in, such that the remote user can still observe the local user’s actions. The remote user has set three markers in his view. Two are inside the local user’s current field of view (a), the third lies outside his view on the left, as indicated by the (accordingly colored) arrow on the left. Note that the view is correctly tracked (as apparent from the correctly registered frames and markers) despite significant occlusion from the local user’s hand.

tablet computer, and all computations are executed on the connected PC. However, we do not make use of a GPU and the computation-intensive part (i.e., the tracking with the template matching core, cf. Section 4.1.3) is very similar to the tracking system by Wagner et al. (2009), which operates in real time on a 2009 smart phone. Thus, our system could be implemented on a mobile device (such as a light-weight tablet or smartphone) given appropriate optimizations.

When tracking is lost and needs to be recovered, the local user sees a large red ‘X’ across the screen to indicate that tracking is lost and hence virtual annotations cannot be drawn. By pointing back to a previously seen location, the user can help the system to recover tracking (cf. Section 4.1.3). Even if tracking is lost, the live video feed is not interrupted and the local user may continue to work.

As virtual annotations, our prototype supports point-based markers which the remote user controls. They are displayed to both users in their respective views as an ‘X’ anchored to the real world, with a number attached to it and additionally color coded to disambiguate between multiple markers. When a marker is set outside the current view or moves outside the current view, a correspondingly colored arrow appears on the border of the screen pointing towards the marker’s location (see Figure 4.2(a)). (Cf. (Baudisch and Rosenholtz, 2003; Gustafson et al., 2008) for other visualizations of off-screen objects.)

4.1.2 Remote user's interface

The remote user is presented with a view of the local user's environment and can place markers by clicking into this view (Figure 4.2(b)).

We provided one particular feature that allows control over the remote user's viewpoint: the remote user can "freeze" (and un-freeze) his/her viewpoint at any time. Despite its simplicity, this feature allows decoupling of the remote view from local movement, and thus enables for example precise clicking on objects despite the movement of the camera on the local user's side. This feature resembles the "Frame & Freeze" technique by Güven et al. (2006), which was developed for a mobile AR user to interact with his own view.

The local user's screen is *not* affected by the freezing and remains "live" at all times. Using visual tracking, the remote user's view is still registered to the local user's live view. Therefore, when markers are set, they immediately appear at the correct position with respect to the world on both the local and remote views. The remote user can return to the live view at any time by right-clicking again, and the view transitions back to the live view, animated by interpolating a few frames between the two viewpoints.

Note that rather than freezing the remote user's *video frame* — this would have the crucial disadvantage that the remote user would not receive any visual updates and could not observe the local user's action — we freeze the *viewpoint* of the remote user and display a transparent image of the live stream on top of the remote user's frozen view, correctly registered with the frozen viewpoint. Thus, when the local user points to something within his camera's field of view, the remote user sees a half-transparent hand correctly indicating the object of interest. Blending the two frames, rather than displaying only the warped live frame, has the advantage that the initial frozen frame remains visible and stable even if the warped view becomes jittery or blurry (due to jittery tracking, motion blur, or extreme warping angles), and it reduces artifacts along the border of the live frame. The blended view with the local user's half-transparent hand (Figure 4.2(b)) bears noteworthy resemblance to the blended video feeds in several related systems (Tang and Minneman, 1991; Wellner and Freeman, 1993; Kirk and Fraser, 2006), but these systems require a static setup with known camera pose.

4.1.3 Visual tracking & environment modeling

As visual tracking system we used a multi-level, active search patch tracker with normalized cross-correlation (NCC)-based template matching and keyframe-based recovery, inspired by the systems of Wagner et al. (2009) and Klein and Murray (2008). In preliminary investigations, this algorithm was found to perform favorably in terms of speed/robustness trade-off compared to several image alignment- and other feature-based algorithms.

As postulated by our concept, we do not use any model information; that is, the environment is completely unknown prior to the start of the system. Instead, our system builds up an internal representation on the fly which automatically expands to include new areas as the camera moves.

The tracking system could enable further features that may improve the collaboration. Most notably, it could effectively increase the field of view of the remote user by displaying the entire map collected by a panning camera. However, we decided to make only minimal use of the tracking by only world-referencing annotations and the remote user's viewpoint, to allow evaluation of these features in particular.

Details of the Tracking Algorithm. From the first frame, the tracker creates an image pyramid by half-sampling the image twice. On each level, keypoints are detected with the FAST corner detector (Rosten and Drummond, 2006), and a subset that is spatially well-distributed across the image is selected with the algorithm described in Gauglitz et al. (2011). Those keypoints constitute the features that are tracked and based on which the camera viewpoint (modeled as homography) is estimated for each frame.

Each incoming frame is projected back to the initial viewpoint, taking the previous frame's estimated homography as prior estimate (such that the patch tracker has to be robust to the distortion between two subsequent frames only). Then, each feature's new position is established by NCC-based template matching of an 11×11 pixel image patch. This is done for the top-most (smallest) image level first, then the homography is re-estimated from the putative feature correspondences using RANSAC (Fischler and Bolles, 1981) and used to project the features into the next level. This ensures robustness to relatively large inter-frame movements despite a small template matching area. We used a radius of 5 pixels on the top-most level, resulting in tolerable movements of 20 pixels between two subsequent frames.

As new areas come into view, features are detected in those areas as well and added to the internal map by storing their position and NCC template with respect to the initial frame's coordinate system.

Tracking Loss & Recovery. We implemented a recovery algorithm similar to that of Klein and Murray (2008). In certain intervals and if tracking quality is deemed good (as determined by the fraction of inliers found by RANSAC), the system creates a keyframe by downsampling the current frame to 80×60 pixels, blurring it with a Gaussian kernel ($\sigma = 1.5$), and storing it together with the current pose information.

The system declares tracking to be lost if RANSAC fails to find a certain fraction of visible features (25%) that agree on one pose estimate. If tracking is lost, each new frame is also downsampled and blurred, and the NCC score between this frame and all stored keyframes is computed. The keyframe with the highest score is then aligned to the downsampled current frame. During this image alignment step, the homography is restricted to be affine, which increases both the speed and the convergence rate. The refined pose of the stored keyframe is then fed back into the tracking algorithm. If RANSAC is able to again find a sufficiently large fraction of inliers among the feature correspondences, tracking is assumed to be restored successfully; otherwise, the recovery algorithm is run again with the next frame.

When It Is Not Necessary to Recover Tracking. Unless tracking recovery succeeds quasi-instantaneously and thus fully automatically, the user has to help out by pointing towards a previously seen location. This is a distraction from his actual task, and thus should only be done if necessary. Therefore, it is important to understand when recovery is dispensable and design the system appropriately.

In our system, tracking is needed in two cases: when annotations (markers) are present, and when the remote user has frozen his/her viewpoint. If neither is the case, tracking is not currently needed and hence the user should not be bothered with requests to recover it. In this case, recovery is attempted only for a maximum of 10 frames. If unsuccessful, tracking is simply reset and re-initialized with the current frame. If, however, tracking is needed to maintain currently active markers or the registration with the remote user's frozen viewpoint, recovery is attempted for much longer, asking the user to help with recovery by displaying the red 'X' if needed. If recovery is not successful after 240 frames, it is assumed that the user does not want or is unable to recover tracking, and the tracking is reset.

Real-time Performance. Overall, this system is fast enough to operate in real time — the framerate is limited by the camera (30 Hz) and not by the tracking algorithms — and is robust enough to be used by our study participants without any specific instructions, while coping with (or successfully recovering from) free camera movement, motion blur, specular reflections, and significant occlusion (e.g., due to the user's hand, cf. Figures 4.2(b) and 5.2(b)).

4.2 Prototype 2

While Prototype 1 is a complete proof-of-concept implementation of our concept — and was rated very favorably by users, as we will describe in Section 5.1 in detail — it suffers from one particular technical limitation (namely, it can operate only in planar environments) and provides only a very simplistic camera control for the remote user, which however is one of the key components of our concept. We addressed both limitations with the implementation of the prototype described in this section. As a further advance denoting a shift from a proof-of-concept to a fully operational system, this prototype consists of two stand-alone entities communicating via wireless network; the local user's device is an off-the-shelf Android-based smartphone or tablet.

This prototype will be published in the proceedings of the ACM Symposium on User Interface Software and Technology 2014 (Gauglitz et al., 2014a).

Figure 4.3 shows an overview of the system architecture of both the local user's and the remote user's system. Since device hardware (camera and display), network communication, real-time processing, and background tasks are involved, both systems employ a host of components and threads.

4.2.1 Local user's system

The local user's interface, running on a lightweight tablet, is intentionally simple. As in Prototype 1 Section 4.1.1, from the user's perspective, it behaves exactly like a live video of the

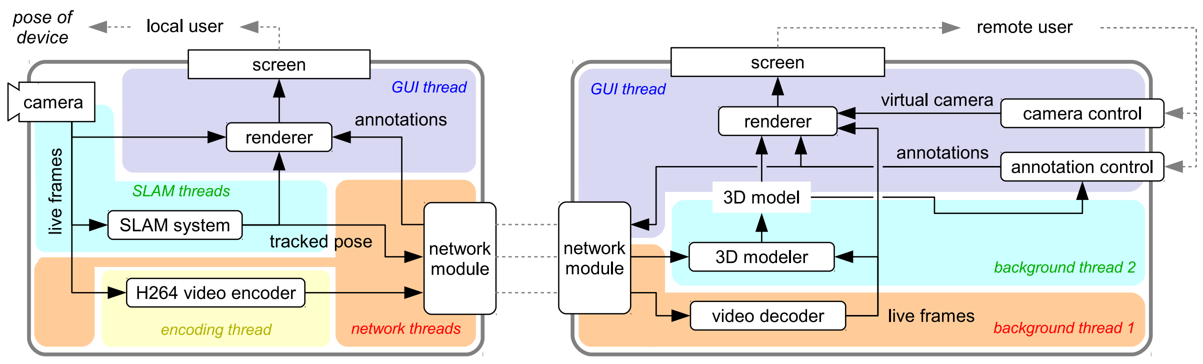


Figure 4.3: System architecture of Prototype 2 including the local user's system (left; running on an Android-based lightweight tablet or smartphone) and the remote user's system (right; running on a commodity PC with Ubuntu). The main components are described in the text in detail.

user's own view plus AR annotations, i.e., a classic magic lens. The only control the user exerts during its operation is by manipulating the position and orientation of the device.

Under the hood, the system runs a SLAM system and sends the tracked camera pose along with the encoded live video stream to the remote system. The local user's system receives information about annotations from the remote system and uses this information together with the live video to render the augmented view.

As in Prototype 1, tracking can get lost, which is communicated to the user via a red border around the screen. The SLAM system continuously tries to relocalize until successful; the user can help this effort by going back to a pose in which tracking is known to work. The live video transmission continues despite tracking loss.

The system is implemented as an Android app, running on several state-of-the-art Android devices. For the user study, we used a Google Nexus 7 2013, a 7" tablet powered by a Qualcomm Snapdragon S4 Pro with 1500 MHz Krait quad core CPU. The core SLAM implementation, including access to the live camera frames and support for rendering them, was provided to us for this work by Qualcomm, Inc. Communication with the SLAM system, handling of the raw image data, and rendering are implemented in C/C++, while higher-level app structure, user interface, and network communication are implemented in Java, with data exchange between the two layers via JNI. The live frames are encoded as a H.264 video stream. A minimal HTTP server streams the data (encoded video, tracked camera pose, and other meta-data) upon request from a remote connection, and manages remote requests for insertion/deletion of annotations (encoded as HTTP requests).

The system operates at 30 frames per second. We measured system latencies using a camera with 1000 fps which observed a change in the physical world (a falling object passing a certain height) as well as its image on the respective screen. The latency between physical effect and the local user's tablet display — including image formation on the sensor, retrieval, processing by the SLAM system, rendering of the image, and display on the screen — was measured as 205 ± 22.6 ms (average and standard deviation over a series of measurements).

4.2.2 Remote user's system: Overview & Architecture

The remote user's interface, running on a commodity PC (without high-end GPUs or such), starts off as a live video stream, but is augmented by two controls, the *camera control* and the *annotation control*.

The system consists of five main modules — network module, 3D modeler, camera control, annotation control, and renderer — and the framework to hold them together. Due to this modular architecture, different implementations for each module can be readily swapped in and out upon the start of the program via command line arguments. This flexibility enabled several use cases:

1. For comparing our interface against two baseline interfaces in our user study (Section 5.2), we simply replaced the respective components with simpler ones;
2. For development and testing, we also implemented modules that load virtual 3D models (instead of modeling from live video) and allow for other types of camera control;
3. Prototype 3 utilizes the same architecture and was realized by implementing appropriate new modules (Section 4.3);
4. Due to this flexibility, the overall system was used not only to implement and evaluate the prototypes presented here, but was also used as platform for other studies, published in Nuernberger et al. (2014), targeting the evaluation of other interface elements.

Here, we describe the module implementations that constitute our Prototype 2.

Using the same setup as above (Section 4.2.1), the latency between physical effect and the remote user's display — including image formation on the local user's sensor, retrieval, processing by the SLAM system, video encoding, transmission via wireless network, video decoding, rendering, and display — was measured as 251 ± 22.2 ms.

4.2.3 Remote user's system: Network module

The network module receives the data stream from the local user's device, sends the incoming video data on to the decoder, and finally notifies the main module when a new frame (decoded image data + meta-data) is available.

4.2.4 Remote user's system: 3D modeler

From the live video stream and associated camera poses, we construct a 3D surface model on the fly as follows. We select keyframes based on a set of heuristics (tracking quality, low device movement, minimum time & translational distance between keyframes), then detect and describe features in the new frame using SIFT (Lowe, 2004). We then choose the four closest existing keyframes, match against their features (one frame at a time) via an approximate nearest neighbor algorithm (Muja and Lowe, 2009) and collect matches that satisfy the epipolar constraint (which is known due to the received camera poses) within some tolerance as tentative 3D points. If a feature has previously been matched to features from other frames, we check for mutual epipolar consistency of all observations and merge them into a single 3D point if possible; otherwise, the two 3D points remain as competing hypotheses.

Next, all tentative 3D points are sorted by the number of supporting observations and are accepted one by one unless one of their observations has been previously "claimed" by an already accepted 3D point (which, by construction, had more support). We require at least four observations for a point to be accepted, and we further remove a fraction of points with the largest 3D distances to their two nearest neighbors. The algorithm is thus robust to even large fractions of mismatches from the stereo matching stage.

To obtain a surface model from the 3D point cloud, we implemented the algorithm by Hoppe et al. (2013): First, a Delaunay tetrahedralization of the point cloud is created. Each tetrahedron is then labeled as "free" or "occupied," and the interface between free and occupied tetrahedra is extracted as the scene's surface. The labeling of the tetrahedra works as follows: A graph structure is created in which each tetrahedron is represented by a node, and nodes of neighboring tetrahedra are linked by edges. Each node is further linked to a "free" (sink) node and an "occupied" (source) node. The weights of all edges depend on the observations that formed each vertex; for example, an observation ray that cuts through a cell indicates that this cell is free, while a ray ending in front of a cell indicates that the cell is occupied. Finally, the labels for all tetrahedra are determined by solving a dynamic graph cut problem. For details on how the edge weights are computed we refer the reader to Hoppe et al. (2013).

We refined Hoppe et al. (2013)'s algorithm by taking the orientation of observation rays to cell interfaces into account, which reduces the number of "weak" links and thus the risk that the graph cut finds a minimum that does not correspond to a true surface.

As Hoppe et al. describe, both updating the graph costs and solving the graph cut can be implemented in an incremental manner, with cost almost independent of the overall model size. As these steps were found to take up a negligible amount of time in our application (cf. Table 4.2), for simplicity we did not even implement the incremental algorithm. Nonetheless, the entire processing of a new keyframe and updating of the 3D surface model is completed

# of keyframes in model	1–10	11–25	> 25
Keypoint detection	32 ± 2	32 ± 3	34 ± 2
Keypoint description	904 ± 124	868 ± 144	795 ± 167
Stereo matching	121 ± 59	166 ± 38	153 ± 45
Merging & filtering of vertices	$<1 \pm 1$	5 ± 1	9 ± 2
Updating tetrahedralization	$<1 \pm 1$	2 ± 2	3 ± 3
Calculating graph costs	10 ± 6	51 ± 14	128 ± 28
Solving graph cut	$<1 \pm <1$	$1 \pm <1$	1 ± 1
Extracting & smoothing surface	4 ± 8	9 ± 21	21 ± 9
Total time	1082 ± 140	1159 ± 175	1201 ± 208

Table 4.2: Average timings of the 3D modeler to process and integrate one new keyframe into the model, running on a single core of a 3 GHz Intel i7 CPU with 4 GB RAM. All times are given in milliseconds, with associated standard deviations. Incorporating all logs from the user study to be discussed below, the data in the three columns are based on 300, 429, and 481 keyframe integrations, respectively.

within 1-1.5 seconds (cf. Table 4.2), which is easily fast enough for our purposes, as it is smaller than the interval at which keyframes are added on average. Currently, the vast majority of the time is taken up by the keypoint description, which is independent of the model size. If necessary, the computation could be significantly sped up by using a more efficient descriptor implementation or algorithm and/or implementing the incremental graph update. Thus, the overall modeling algorithm is highly scalable to environments much larger than demonstrated here.

4.2.5 Remote user's system: Camera control (virtual navigation)

The remote user's ability to navigate the remote world via a virtual camera, independent of the local user's current location, is one of the key contributions of our work.

As mentioned earlier, Sodhi et al. (2013) provide a similar feature, but use physical device movement for navigation. While this is arguably very intuitive, using physical navigation also has two disadvantages; one being generally true for physical navigation and the other one being specific to the application of live collaboration.

First, the remote user needs to be able to physically move and track his/her movements in a space corresponding in size to the remote environment of interest, and "supernatural" movements or viewpoints (e.g., quickly covering large distances or adopting a bird's-eye view) are impossible (cf. Bowman et al. (2005) for a more detailed discussion of physical vs. virtual travel, and Sukan et al. (2012) for a comparison on a particular task in the context of AR).

Second, it does not allow coupling of the remote user's view to the local user's view (and thus have the local user control the viewpoint) without breaking the frame of reference in which the remote user navigates. Lanir et al. (2013) presented a study on the issue of control of viewpoint with mixed results, suggesting that it may be dependent on the particular task.

We thus decided to use virtual navigation, and we deem it important that our navigation gives the remote user the option of coupling his/her view to that of the local user.

Within virtual navigation, we decided to use a keyframe-based navigation as explained in the following for several reasons: first, mapping unconstrained 3D navigation to 2D controls requires relatively complex interfaces (Jankowski and Hachet, 2013); second, our model is inherently constrained by what has been observed by the local user's camera, and a keyframe-based approach offers a natural way to constrain the navigation accordingly; third, a keyframe-based approach allows for image-based rendering with high levels of fidelity. (We note that these keyframes do not have anything to do with the ones used by the modeler, they serve a different purpose and are maintained separately.)

The resulting interactions were designed to be as intuitive as possible and resemble controls familiar from other exploration tools such as Google Street View and Photo tourism (Snavely et al., 2006). Viewpoint transitions are implemented by smoothly interpolating between the camera poses and are rendered as visually seamlessly as possible (see below).

Freeze & back to live

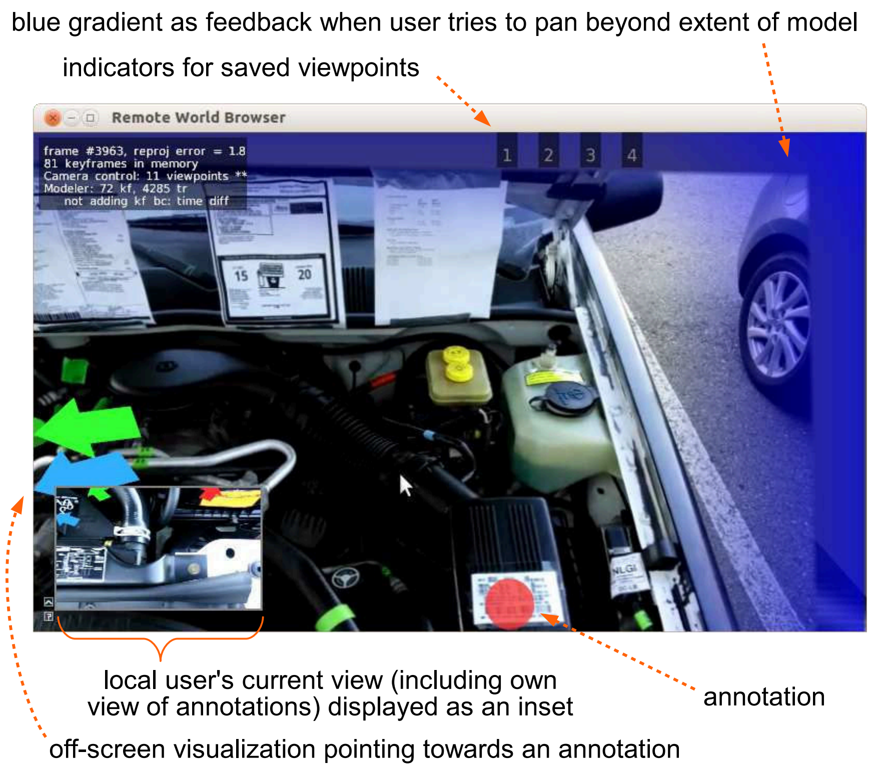
The application starts with the remote view coupled to the local user's live view. With a single right-click, the remote user "freezes" his/her camera at the current pose, for example in order to precisely set annotations, or as a starting point for further navigation. Whenever the remote user's view is not coupled to the local user's view, the latter is displayed to the remote user as an inset (cf. Figure 4.4). A click onto this inset or pressing 0 immediately transitions back to the live view. (This feature is the only camera control feature provided in Prototype 1 (Section 4.1).)

Panning & zooming

By moving the mouse while its right button is pressed, the user can pan the view in a panorama-like fashion (rotate around the current camera position). We maintain the original up vector (typically: gravity, as reported by the local user's device) by keeping track of the original camera position, accumulating increments for yaw and pitch separately and applying *one* final rotation for each yaw and pitch (rather than a growing sequence of rotations, which would skew the up vector).

To prevent the user from getting lost in unmapped areas, we constrain the panning to the angular extent of the modeled environment. To enforce this constraint, we allow a certain amount of "overshoot" beyond the allowed extent; here, further mouse movement away from the modeled environment causes an exponentially declining increase in rotation and we show visual feedback in the form of an increasingly intense blue gradient along the respective screen border (Figure 4.4). Once the mouse button is released, the panning quickly snaps back to the allowed range. Thus, the movement appears to be constrained by a (nonlinear) spring rather a hard wall.

The user can also zoom into and out of the view with the scroll wheel. Zooming is implemented as a change of the virtual camera's field of view (rather than dollying) to avoid having to deal with corrections for parallax or occlusions from objects behind the original camera position.


blue gradient as feedback when user tries to pan beyond extent of model
indicators for saved viewpoints
Remote World Browser
1
2
3
4
local user's current view (including own view of annotations) displayed as an inset
annotation
off-screen visualization pointing towards an annotation

```
frame #3963, reproj error = 1.8
81 keyframes in memory
Camera control: 11 viewpoints **
Modeler: 72 kf, 4285 tr
   not adding kf bc: time diff
```

Figure 4.4: Screenshot of the remote user's interface.

Click to change viewpoint

The camera control module continually stores new keyframes with their associated camera poses from the live video stream (if tracking quality is good enough). Keyframes that have become obsolete because they are close to a newer keyframe are automatically discarded.

When the user right-clicks into the view, we compute the 3D hit point, and subsequently find the camera whose optical axis is closest to this point (which may be the current camera as well). We then transition to this camera and adapt yaw and pitch such that the new view centers around the clicked-upon point. This allows the user to quickly center on a nearby point as well as quickly travel to a far away point with a single click. We note that a similar control is used for traveling in Google Street View and Microsoft Photosynth (cf. Nuernberger et al., 2014).

Saving & revisiting viewpoints

Further, the user can actively save a particular viewpoint to revisit it later (similar to the navigation investigated by Sukan et al. (2012)). Pressing `Alt` plus any number key saves the current viewpoint; pressing the respective number key alone revisits this view later. Small numbers along the top of the screen indicate which numbers are currently in use (see Figure 4.4).

4.2.6 Remote user's system: Annotation control

In addition to being able to control the viewpoint, the remote user can set and remove virtual annotations. Annotations are saved in 3D world coordinates, are shared with the local user's mobile device via the network, and immediately appear in all views of the world correctly anchored to their 3D world position (cf. Figures 4.2 and 4.4).

For this prototype, we implemented only simple, animated spherical markers. If annotations are outside the user's current field of view, an arrow appears along the border of the screen pointing towards the annotation (see Figure 4.4; also cf. Gauglitz et al., 2012a). Annotations are "pulsing" with 1 Hz and 15% amplitude to increase their visual saliency. Together with the independent viewpoint control as described above, the remote user can thus effectively direct the local user to elements outside the local user's current field of view.

The remote user sets a marker by simply left-clicking into the view (irrespective if "live" or "decoupled"). The depth of the marker is derived from the 3D model, presuming that the user wants to mark things on physical surfaces rather than in mid-air. Pressing the space bar removes all the annotations. More complex and/or automatic erasure management (Fussell et al., 2004; Jo and Hwang, 2013) could be integrated as desired.

These annotations were sufficient for our task (cf. user study tasks below), but other tasks may require more complex annotations. As discussed earlier, other works have experimented with 2D drawings (Ou et al., 2003; Kirk and Fraser, 2006; Gurevich et al., 2012) or transfer of hand gestures (Kirk and Fraser, 2006; Huang and Alem, 2013; Huang et al., 2013), which are undoubtedly more expressive but thus far have been used mostly with stationary cameras. More complex annotations like this could be integrated (now world-stabilized as well) as needed.

Setting annotations while tracking is lost

As explained in Section 4.2.1, the local user's system may lose tracking, in which case anchored annotations cannot be displayed. However, the remote user can decouple his/her camera from the live view and continue to set annotations anchored to the model, which are displayed to the local user as soon as tracking resumes.

Additionally, if the remote user's view is "live," we still enable the remote user to set annotations, which however cannot be anchored and thus are stored (and displayed) in 2D image-referenced coordinates. These annotations are set to disappear automatically after a few seconds. Thus, the local user is not forced to recover tracking first if the static annotations are deemed "good enough" in that particular moment.

4.2.7 Remote user's system: Renderer

Finally, the renderer renders the scene using the 3D model, the continually updated keyframes, the incoming live camera frame (including live camera pose), the virtual camera pose, and the annotations. In addition to a generally desirable high level of realism, a particular challenge rather unique to our application is the seamless transition to and from the live video. That is, as the virtual camera approaches the physical camera, the views should become identical, with no noticeable transition from "model view" to "live view." (Very recently, Tatzgern et al. (2014) presented a system which features similar transitions between a live video ("AR view") and synthesized views of a reconstructed environment (“VR view”).) We achieve this by using image-based rendering as follows.

The 3D model is rendered as a polygonal model, upon which the images of the live frame and closest keyframes are projected using projective texture mapping. As the virtual camera moves, the different textures are faded in and out by adapting their opacity. In areas which are modeled accurately, transitions are thus completely seamless, with lighting effects naturally blending in. More sophisticated image-based rendering techniques (e.g., Chaurasia et al., 2013) could be integrated as appropriate.

We note that the live view may extend beyond the area currently modeled in 3D. To ensure that these areas do not suddenly “disappear” immediately after transitioning away from the live view, we extend the model (using its average distance to the camera) with a proxy surface on which textures can be projected. To soften artifacts, we blur the parts of the projective textures that fall onto this part.

In effect, when the remote user’s virtual camera pose is identical (coupled) to the local user’s current physical camera pose, the rendered image is identical to the live video, and the transition to and away from it are seamless.

Due to the structure of our camera control, the camera is often at the location of a previously cached keyframe (albeit with possibly modified yaw, pitch, or field-of-view), which enables image-based rendering with high levels of fidelity. The 3D model is rarely visible as a polygonal model, thus modeling artifacts are rarely disturbing. However, the more accurate the model, the better the transitions can be rendered, and the more precisely annotations can be anchored.

4.3 Prototype 3

Prototype 3 builds on Prototype 2 and improves in particular the remote user's interface. The overall architecture, the local user's system as well as the architecture and underlying components of the remote user's system remained the same.

This prototype will be published in the proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST) 2014 (Gauglitz et al., 2014b).

Two of the drawbacks of the mouse-and-keyboard-based interface of Prototype 2 (Section 4.2) are (a) the need to memorize keyboard shortcuts, and (b) the level of indirection of the mouse-based interaction. In Prototype 3, we thus introduce a novel interface for the remote user, who now uses a touchscreen for all interactions with the system. In Section 4.3.2, we describe the main elements of the graphical user interface before focusing on two particular aspects — namely, the use of 2D drawings as world-stabilized annotations in 3D and gesture-based virtual navigation — in Sections 4.3.3 and 4.3.4, respectively.

4.3.1 Motivation for using a touchscreen rather than 3D input

One may argue that, for interaction in a three-dimensional space, one should use an interface which affords three-dimensional input and thus can, for example, create annotations in 3D.

Sodhi et al. (2013) described a prototype system which uses three-dimensional input for the purpose of remote collaboration, by reconstructing the remote user's hand in 3D and transferring this reconstruction and a 3D "motion trail" into the local user's space.

However, even if sensors that support unthethered, unobtrusive 3D input (e.g., high resolution active depth sensors) become commonplace, additional issues remain. First, unless such sensors are matched with an immersive 3D display that can synthesize images in any physical space (such as a stereo head-worn display), the space in which the input is provided and the space in which objects are visualized remain separate, in much the same way as is the case with a standard computer mouse. This has the effect that *relative* spatial operations are very natural and intuitive (e.g., moving the mouse cursor downwards/indicating a direction in 3D, respectively), but *absolute* spatial operations (e.g., pointing to an object, which is arguably very important in our context) remain indirect and require the user to first locate the mouse cursor/representation of the hand, respectively, and position it with respect to the object of interest. This can be well observed in the illustrations and discussion by Sodhi et al. (2013).

Second, even with such an immersive 3D display, haptic feedback is typically missing (Xin et al., 2008).

In contrast, touchscreens are not only ubiquitous today, but they afford direct interaction without the need for an intermediate representation (i.e., a mouse cursor), and provide haptic feedback during the touch. In our context, however, they have the downside of providing 2D input only. Discussing the implications of this limitation and describing and evaluating

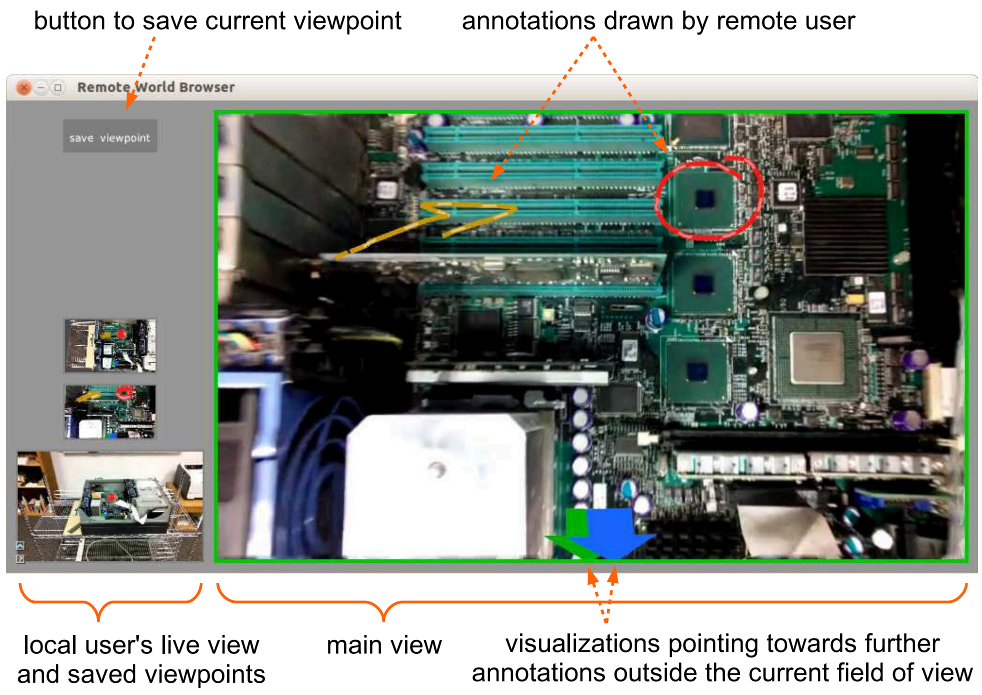

button to save current viewpoint
annotations drawn by remote user
Remote World Browser
save viewpoint
local user's live view and saved viewpoints
main view
visualizations pointing towards further annotations outside the current field of view

Figure 4.5: Screenshot of the remote user's touchscreen interface.

appropriate solutions in the context of live remote collaboration is an important part of this section.

4.3.2 Touchscreen interface elements

Figure 4.5 presents the elements of the graphical user interface. The main part of the screen shows the main view, in which annotations can be created (Section 4.3.3) and gesture-based virtual navigation takes place (Section 4.3.4).

A side pane contains, from top to bottom, (1) a button to save the current viewpoint, (2) small live views of saved viewpoints, and (3) the local user's live view (whenever the main view is not identical to it). A tap onto any of the views in the side pane causes the main view to transition to that respective viewpoint.

A two-finger tap onto the main view while it is coupled to the local user's live view freezes the current viewpoint. (As in Prototype 2, only the *viewpoint* is frozen; the live image is still projected into the view.)

When the user starts to draw an annotation while the main view is coupled to the (potentially moving) live view, the viewpoint is temporarily frozen in order to enable accurate drawing. In this case, the view automatically transitions back to the live view as soon as the finger is lifted.

Thus, all interface functions are immediately accessible on the screen, allowing quick access and, in contrast to Prototype 2, avoiding the need to memorize keyboard shortcuts.

4.3.3 2D drawings as annotations in 3D space

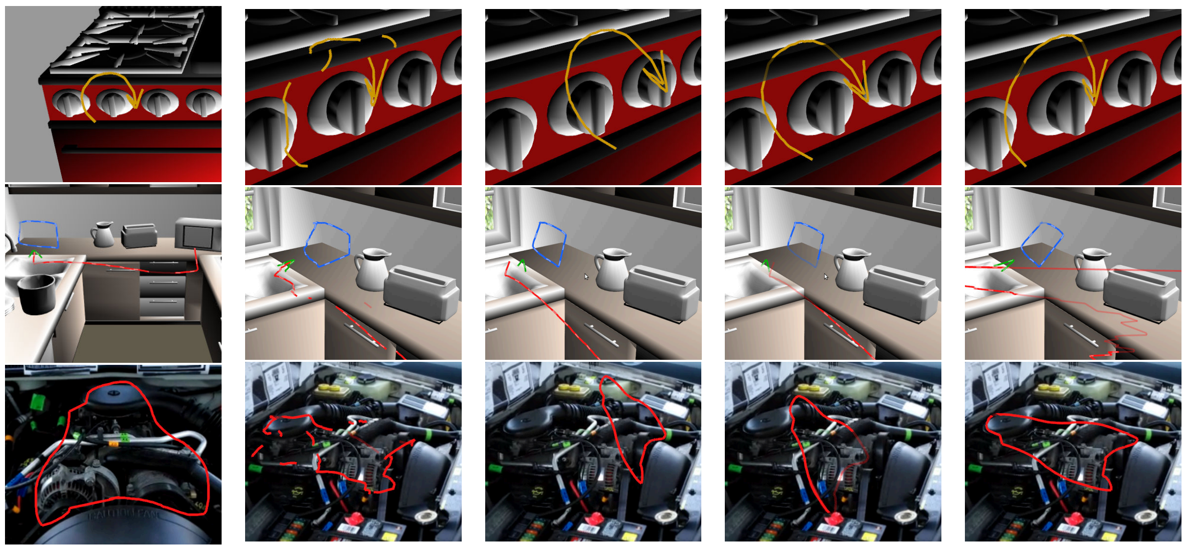
Using a single finger, the remote user can draw annotations into the scene; a few examples are shown in Figure 4.6(a). Since the camera is tracked with respect to the scene, these annotations automatically obtain a position in world coordinates. However, due to our use of a touchscreen, the depth of the drawing along the current viewpoint's optical axis is unspecified.

While drawings have been used for the purpose of remote collaboration (Ou et al., 2003; Kirk and Fraser, 2006; Fussell et al., 2004), they were created on a 2D surface as well as displayed in a 2D space (e.g., a live video), and it remains up to the user to mentally "unproject" them and interpret them in 3D. This is fundamentally different from creating annotations that are anchored and displayed in 3D space, that is, in AR.

In other areas such as computer-aided design (CAD), the interpretation of 2D drawings in 3D is more common (Tolba et al., 1999; Igarashi et al., 2007; Zeleznik et al., 2007; Xin et al., 2008). Another example for 3D interpretation of 2D input is the interactive image-based modeling by van den Hengel et al. (2007). However, the purpose (design/modeling vs. communication), intended recipient (computer vs. human collaborator) and, in most cases, scene (virtual model vs. physical scene) all differ fundamentally from our application, and the interpretation of 2D input is typically guided and constrained by task/domain knowledge. Thus, these techniques cannot immediately be applied here.

To our knowledge, freehand 2D drawings have not been used before as world-stabilized annotations unprojected into 3D space for live collaboration.

In principle, it is possible to ask the user to explicitly provide depth, for example by providing a second view onto the scene and having the user shift points along the unspecified dimension. This may be appropriate for professional tools such as CAD. However, in our case, we want to enable the user to quickly and effortlessly communicate spatial information, such as with hand gestures in face-to-face communication. Thus, we concentrate on ways to infer depth automatically. In this section, we discuss and evaluate several alternatives to do so.



(a) original view (b) spray paint (c) minimum depth (d) median depth (e) dominant plane

Figure 4.6: Depth interpretations of 2D drawings. In each row, (a) shows the viewpoint from which the drawing was created, and (b-e) show different depth interpretations from a second viewpoint (all interpretations have the same shape if seen from the original viewpoint). Annotation segments that fall behind object surfaces are displayed semi-transparently, which was deemed "very helpful" by 7 of 11 users (cf. Figure 5.12).

Depth interpretations

We start with the assumption that the annotation shall be in contact with the 3D surface in some way; i.e., annotations floating in mid-air are, for now, not supported.

Given an individual 2D input location $p = (x, y)$, a depth d can thus be obtained by unprojecting p onto the model of the scene. For a sequence of 2D inputs (a 2D drawing) $p_1, ..., p_n$, this results in several alternatives to interpret the depth of the drawing as a whole:

- **"Spray paint."** Each sample p_i gets assigned its own depth d_i independently; the annotation is thus created directly on the 3D surface as if spray painted onto it (Figure 4.6(b)).

- **Plane orthogonal to viewing direction.** The annotation is created on a plane orthogonal to the viewing direction. Its depth can be set to any statistic of $\{d_i\}$, for example, the minimum (to ensure that no part of the annotation lands behind surfaces; Figure 4.6(c)) or the median (Figure 4.6(d)).

- **Dominant surface plane.** Using a robust estimation algorithm such as RANSAC or Least Median of Squares, one can estimate the dominant plane of the 3D points formed by $\{p_i\}$ and the associated $\{d_i\}$, and then project the drawn shape onto this plane (Figure 4.6(e)).

It should be noted that all of these alternatives result in the same shape in the case of a planar surface orthogonal to the optical axis.

All of these approaches have merits; which one is most suitable will depend on the context and task. Spray paint appears to be the logical choice if the user intends to "draw" or "write onto" the scene, trace specific features, etc.[1]

For other types of annotations, planarity may be preferable. To refer to an object in its entirety, the user might draw an outline *around* the object. Drawings used as proxies for ges-

[1]Note that projective displays — used for remote collaboration for example by Gurevich et al. (2012) and appealing due to their direct, un-mediated overlay of annotations — can intrinsically only produce spray paint-like effects, unless not only the projector, but also the user's eyes are tracked and active stereo glasses (synced with the projector) are used in order to create the illusion of a different depth.

tures — for example, drawing an arrow to indicate a direction, orientation, or rotation (cf. Figure 4.6 top two rows) — are likely more easily understood if projected onto a plane.

Another aspect for consideration, especially in the case of models reconstructed via computer vision, is the sensitivity to noise and artifacts in the model. A spray-painted annotation may get unintelligibly deformed and the minimum depth plane may be shifted. The median depth and dominant plane are more robust as single depth measurements carry less importance.

4.3.4 Gesture-based virtual navigation

As discussed in Section 4.3.2, the user can freeze the viewpoint (Figure 4.7(a)), save the current viewpoint, and go to the live view or any previously saved view (Figure 4.7(b)), all with a single tap onto the respective region in the interface. In addition, we implemented gesture-based navigation controls (Figure 4.7(c)–(e)) which we will discuss in this section. As a distinguishing element from drawing annotations (Section 4.3.3), all navigation-based controls on the main view are triggered by *two* fingers.

The reasoning for using keyframe-based navigation (cf. Section 4.2.5) has remained unchanged: While we want to empower the remote user to explore the scene as freely as possible, only parts of the scene that have previously been observed by the local user's camera are known and can be rendered. However, it now has to be realized via touchscreen-based input.

Thus, our navigation shares an important characteristic with the work by Snavely et al. (2008): from a more or less sparse set of camera poses we want to find paths/transitions that

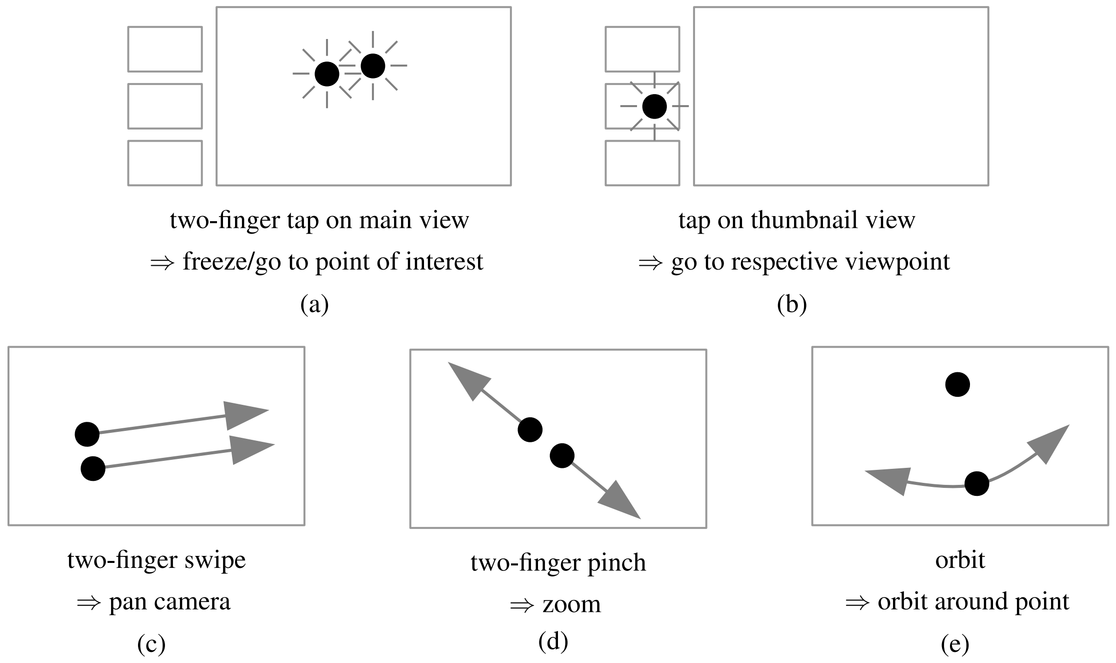


Figure 4.7: Navigation features.

can be mapped to specific input controls. However, in contrast to their work, we do not attempt to mine longer paths and suggest them to the user, but rather to find suitable transitions given a user's specific input.

For all gestures, we designed the controls such that the 3D scene points underneath the touching fingers follow (i.e., stay underneath) those fingers throughout the gesture (i.e., "contact trajectories match the scene transformations caused by viewpoint modifications" (Marchal et al., 2013)) as far as possible.

The software stack that we implemented to process the touchscreen and mouse input is depicted in Figure 4.8. A pointing device handler receives low-level mouse and touchscreen

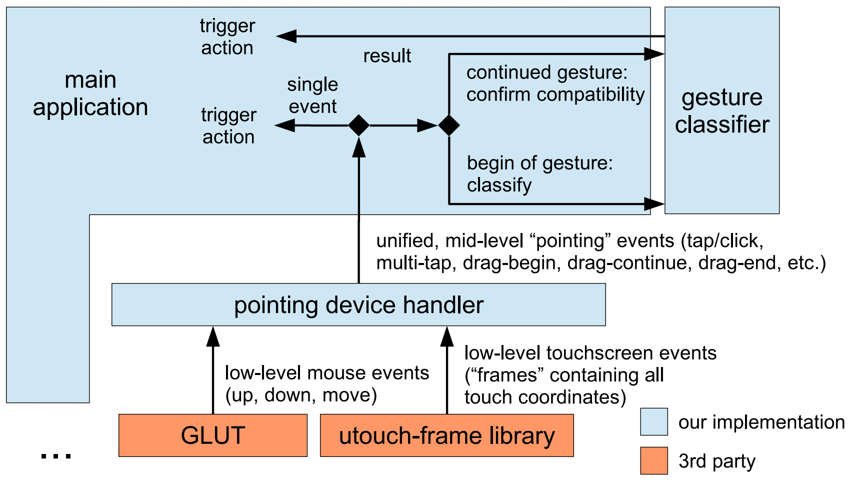

trigger action
result
main application
single event
trigger action
continued gesture: confirm compatibility
begin of gesture: classify
gesture classifier
unified, mid-level "pointing" events (tap/click, multi-tap, drag-begin, drag-continue, drag-end, etc.)
pointing device handler
low-level mouse events (up, down, move)
low-level touchscreen events ("frames" containing all touch coordinates)
GLUT
utouch-frame library
...
our implementation
3rd party

Figure 4.8: Software stack for processing events from pointing devices (mouse & touchscreen).

events (from GLUT and the Ubuntu utouch-frame library[2], respectively) and generates events that are unified across the devices and of slightly higher level (e.g., recognize "click"/"tap" from down+*(no move)*+up events, distinguish from drag-begin, etc.). These events are received by the main application. Multi-stage events (i.e., gestures) are sent on to a gesture classifier which classifies them or, after successful classification, validates the compatibility of the continuing gesture. The gesture classifier operates on relatively simple geometric rules (e.g., for swipe, the touch trails have to be of roughly the same length and roughly parallel), which worked sufficiently for our purposes, as our qualitative evaluation (cf. Section 5.4.3) confirmed.

[2]Open Input Framework Frame Library, https://launchpad.net/frame

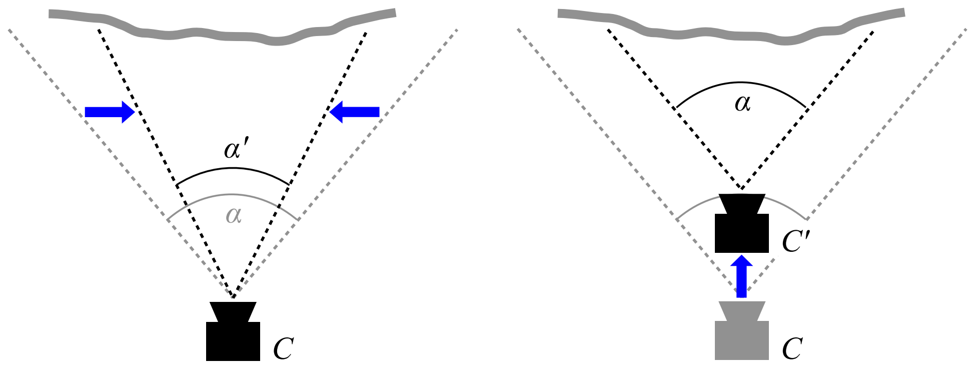


Figure 4.9: "Zooming in" via decreasing the field of view (left) and dollying forward (right).

Panning

By moving two fingers in parallel, the user can pan, i.e., rotate the virtual camera around its optical center (Figure 4.7(c)). As in Prototype 2, we constrain the panning to the angular extent of the known part of the scene and provide visual feedback in the form of an increasingly intense blue gradient along the respective screen border.

Transitional zoom

There are two different ways of implementing the notion of "zoom" in 3D: either via modifying the field of view (FOV) (Figure 4.9 left) or via dollying (i.e., moving the camera forward/backward; Figure 4.9 right). For objects at a given depth, both changes are equivalent; differences arise due to varying depths (i.e., parallax). Which motion is desired by the user may depend on the particular scene. Given the large overlap in effect (cf. optical flow diagrams by Marchal et al. (2013)), we wanted to avoid providing two separate controls.

In our context, with an incomplete and/or imperfect model of the environment, changing the FOV has the advantage that it is trivial to render, while rendering a view correctly after dollying the camera might be impossible due to occlusions. Therefore, Prototype 2 supported zooming (controlled by the scrollwheel) via change of FOV only. However, changing the FOV disallows exploration of the scene beyond a fixed viewpoint, and its usefulness is limited to a certain range by the resolution of the image.

We thus developed a novel hybrid approach: changing the FOV to allow for smooth, artifact-free, fine-grained control combined with transitioning to a suitable keyframe, if available, in front or behind the current location for continued navigation. We note that the availability of a keyframe automatically implies that it may be reasonable to move there (e.g., walk in this direction), while free dollying has to be constrained intelligently to not let the user fly through the physical surface when his/her intent may have been to get a close-up view. We implemented this hybrid solution as follows.

Given a two-finger "pinch" gesture (Figure 4.7(d)) with start points $s_{1,2}$ and end points $e_{1,2}$, we first calculate the change in FOV that corresponds to the change in distance $|s_1 - s_2|$ to $|e_1 - e_2|$. We then adapt yaw and pitch such that the scene point under $(s_1 + s_2)/2$ gets mapped to $(e_1 + e_2)/2$. (We disallow roll on purpose as it is rarely used or needed; cf. Marchal et al. (2013).)

To determine if transitioning to a different keyframe is in order, we determine the 3D world points $\{c_i^{\mathrm{3D}}\}_{i=1..4}$ which, with the modified projection matrix, get mapped into the corners of the view $\{c_i\}$. The ideal camera pose R is the one that projects $\{c_i^{\mathrm{3D}}\}$ to $\{c_i\}$ with its *native* projection matrix P (i.e., unmodified FOV). Therefore, we project $\{c_i\}$ using the camera pose of each of the stored keyframes, and select the camera k^* for which the points land closest to the image corners $\{c_i\}$; i.e.,

$$k^* = \arg\min_k \sum_{i=1}^{4} \left| P \cdot R_k \cdot c_i^{3\mathrm{D}} - c_i \right|^2 \tag{4.1}$$

To ensure that the transition is consistent with the notion of dollying, we only consider cameras whose optical axis is roughly parallel to the current camera's optical axis.

If k^* is not identical to the current camera, we thus transition to R_{k^*} and adapt FOV, yaw and pitch again as described above. A hysteresis threshold can be used to decrease erratic behavior.

In effect, the user can "zoom" through the scene as far as covered by available imagery via a single, fluid control, and the system automatically choses camera positions and view parameters (FOV, yaw, pitch) based on the available data and the user's input.

Orbiting with snap-to-keyframe

As a third gesture, we added orbiting around a given world point $p^{3\mathrm{D}}$. The corresponding gesture is to keep one finger (relatively) static at a point p and move the second finger in an arc around it (Figure 4.7(e)). The orbit center $p^{3\mathrm{D}}$ is determined by un-projecting p onto the scene and remains fixed during the movement; additionally, we maintain the gravity vector (as reported by the local user's device). The rotation around the gravity vector at $p^{3\mathrm{D}}$ is then specified by the movement of the second finger.

Again, we want to guide the user to keyframe positions if possible as the rendering from those positions naturally has the highest fidelity. Thus, once the user finishes the orbit, the camera “snaps” to the closest keyframe camera pose. (The idea of orbiting constrained to keyframes again resembles the approach by Snavely et al. (2008), where possible orbit paths are mined from the set of pictures and then suggested to the user for exploration of the scene, but differs in that we do not find paths beforehand, but find a suitable transition given the user’s input.)

Selection of the “snap” target pose. Given the list of available camera positions, we first filter out all cameras for which p^{3D} is not within the field of view. Among the remaining poses, we select the one which minimizes the following distance function with respect to the camera pose at which the user ended the orbit:

$$d(R_1, R_2) = \begin{cases} \frac{d_t(R_1,R_2)}{d_\measuredangle(R_1,R_2)} & \text{if } d_\measuredangle(R_1, R_2) > 0 \\ \infty & \text{otherwise} \end{cases}$$

where $d_t(\cdot, \cdot)$ is the translational distance between the camera origins, and $d_\measuredangle(\cdot, \cdot)$ is the dot product of the optical axes.

Preview visualization “snap” target. During an on-going orbit process, we provide two visualizations that indicate where the camera would snap to if the orbit ended at the current location (see Figure 4.10): First, we display the would-be target keyframe as a preview in a small inset in the bottom left corner of the main view. Second, we display a semi-transparent

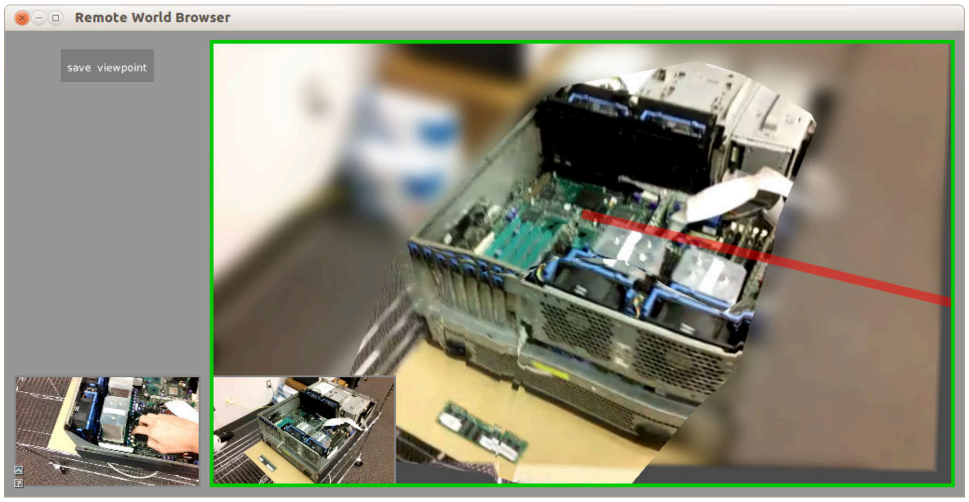


Figure 4.10: Screenshot during an orbit operation, with the two visualizations of the "snap" position: 1. the inset image in the bottom left corner of the main view previews the target image; 2. the red line connects the orbit point and the target camera origin.

red line from the orbit point p^{3D} to the would-be target camera origin. While less expressive than the image, this visualization has the advantage of being in-situ: the user does not have to take their eyes off the model that he/she is rotating. We opted for a minimal visualization (instead of, for example, visualizing a three-dimensional camera frustum) in order to convey the essential information but keep visual clutter to a minimum.

Chapter 5

Evaluation via User Studies

In this chapter, we describe three user studies to validate the usability of our prototype systems, compare them with more conventional interfaces, and gain insights on the design of particular interface features.

5.1 User Study 1

In user study 1, we compared Prototype 1 (Section 4.1) with one interface without any annotations and one interface with static markers. The study's scenario was that of a remote expert instructing and directing a novice local user in operating an airplane.

This study was first presented together with Prototype 1 in our paper in the proceedings of the ACM International Conference on Human-Computer Interaction with Mobile Devices and Services 2012 (Gauglitz et al., 2012a).

Various parameters of the study as reported below were refined during several pilot study trials with a total of twelve users.

5.1.1 Task & physical setup

We created a mock-up airplane cockpit by printing a high-resolution image of the interior of an airplane cockpit (Figure 5.1) on 3’×4’ paper and mounting it to a metal panel on the wall. To simulate a remote user in another location, we placed a room divider next to the poster and placed the remote user’s station on the other side. This allowed both users to communicate verbally by simply talking out loud while blocking any direct visual communication.

The device for the local user consisted of a MIMO 10.1” ‘iMo Monster’ Touchscreen (used as display only, touch disabled) with a Point Grey Firefly (34° horizontal field of view) mounted on the back to deliver 640×480 Bayer images at 30 Hz. To make the tablet more comfortable to hold, we added some rubber padding on both sides of the tablet and a strap to go around the user’s hand (cf. Figure 4.1).

The remote participant used a standard desktop PC interface with monitor, keyboard, and mouse. The system ran on a standard PC with an Intel i7 Core with 4 GB RAM running Ubuntu 10.10. All interfaces and further system components were implemented in C++, making use of the OpenCV and libCVD libraries for the vision system.

The task that each pair of participants performed was to identify and “operate” a series of control elements in our mock-up cockpit, in order to, e.g., “safely land the airplane.” The local user was assumed to be a novice, not knowing which elements to operate, and had to be instructed by the remote expert. “Operating” an element was simulated by placing a magnetic pin onto it. The pins were numbered so that the study administrator could later verify the

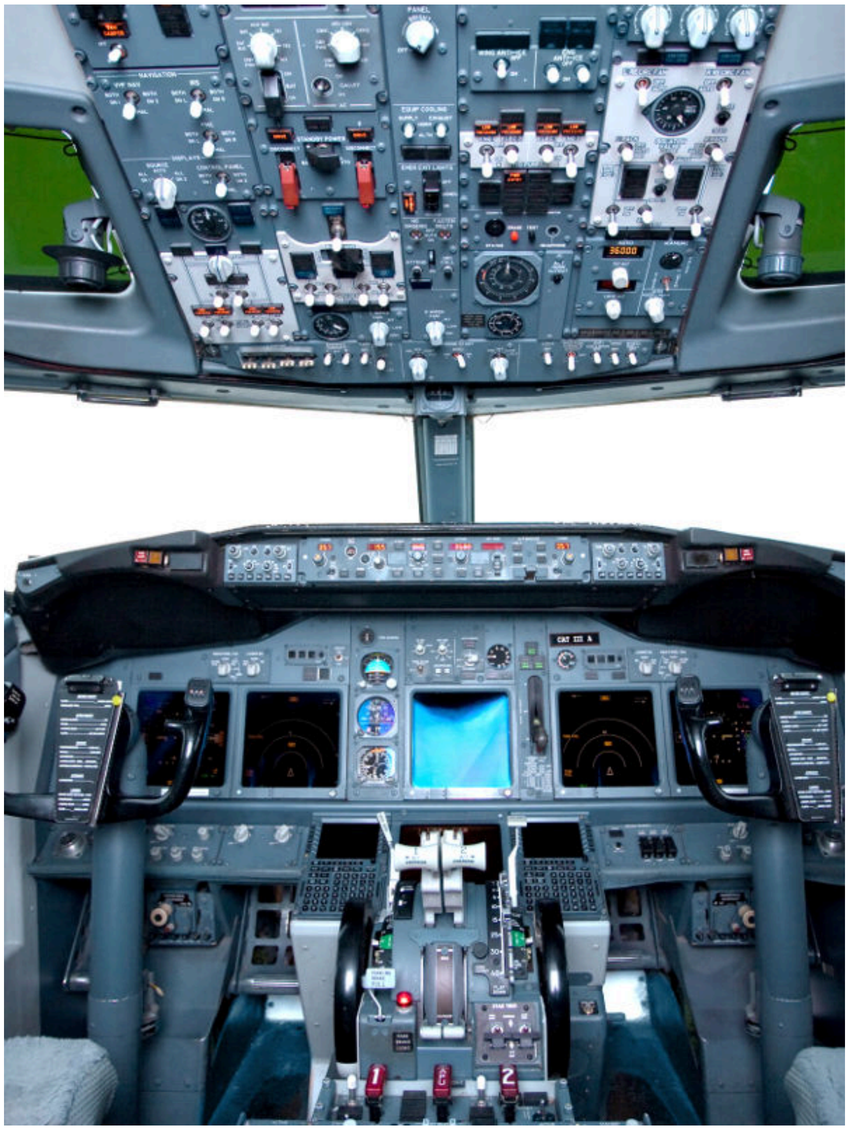
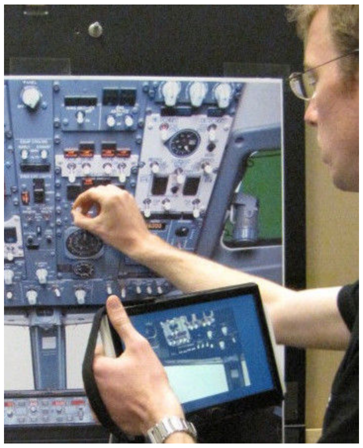


Figure 5.1: Our testbed: view of an airplane cockpit. This image was printed in size 3’×4’ and mounted to a metal panel on the wall. The resolution is high enough that most of the control element labels are readable. Original image file obtained from iStockphoto.com/Smaglov.


(a) local user

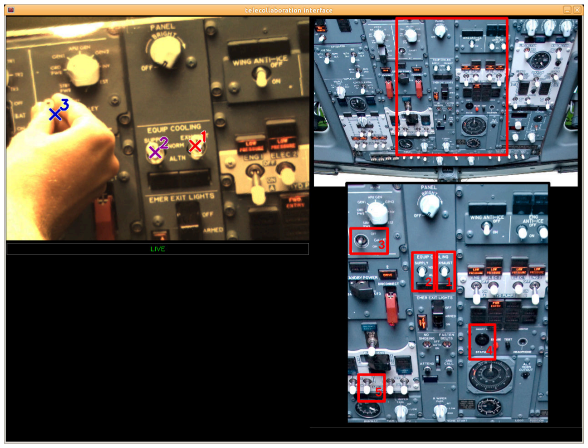

(b) remote user's interface

Figure 5.2: (a) Local user with tablet, putting a magnetic pin on one of the "buttons." (b) Overview of the remote user's interface with live video view on the top left and "expert knowledge" information (consisting of overview image on the top and detail on the bottom) on the right.

correct placement and order. We chose the testbed such that it would neither be too easy nor too difficult to reference the control elements verbally. Many of the elements had readable labels, visually distinguishable features, and/or could be described by their relative position in the cockpit. However, labels were relatively small, and some switches were in a series of alike-looking elements. It would be easy to design a testbed that would be much easier or much harder, but through pilot studies we arrived at this setup as a reasonable and, in particular, realistic compromise between the two extremes.

One participant played the role of the local user. This participant stood in front of the poster with the display device (as seen in Figure 5.2(a)) and was instructed by the remote user as to which control elements to "operate," i.e., place a pin onto.

The other participant played the role of the remote expert and was responsible for directing the local user as to which control element to operate. The remote user sat in front of a desktop monitor which showed the camera feed from the local user on the left side. On the right side, the remote user saw a detail of the cockpit with a sequence of buttons clearly marked and, above, an overview image in which the location of the detail was indicated (Figure 5.2(b)). These images simulated the "expert knowledge" that the remote user was assumed to possess. The remote user then communicated the locations of each element to the local user verbally and via the interface functions. In addition to directing the local user, the remote user was asked to monitor correct pin placement. (Users were allowed to remove and re-place incorrectly placed pins, and the remote user was instructed to inform the local user accordingly.)

Each page of the expert knowledge information indicated a sequence of five random buttons. When completed, the remote user could press a key to proceed to the next page. After five pages (25 buttons), we introduced a brief break in which the study administrator would take down the pins, comparing them with a control sheet and noting down errors (if any).

5.1.2 Conditions

With all interfaces, the participants were able to talk to each other without restriction.

- **Interface A: video only.** In this interface, the live video from the tablet's camera is streamed to the remote user's screen (i.e., one-way video), but the remote user does not have any means of providing visual feedback. This is similar to using a current tablet PC with rear-facing camera and standard video conferencing software.

- **Interface B: video + static markers.** In this interface, the remote user can click into the video view to create a marker — a colored ‘X’ — that is visible on both screens (the remote user’s screen as well as the local user’s tablet screen). However, the marker is static within image coordinates, so it appears to “swim away” from its original position if the tablet’s camera is moved. The remote user can set up to five of those markers, and clear them by pressing the space bar at any time. By pressing a number key between 1 and 5, the user can select the next marker to be set; without pressing number keys, the system rotates through markers 1 to 5. The next marker to be set is indicated by a small white number next to the mouse cursor. This condition is similar to the pointing in Fussell et al. (2004) in that it assumes a stationary camera.

- **Interface C: video + world-stabilized markers.** This interface is using Prototype 1 as described in Section 4.1. As with interface B, the remote user can set up to five markers by clicking into the video view (space bar to clear, number keys to select marker as in B). However, now the markers are stored in world-stabilized coordinates and thus stick to their original positions despite movement of the camera. With a right-click, the remote user can freeze his/her viewpoint as described in Section 4.1.

5.1.3 Experimental design

We used a within-subjects design with a single independent variable (interface type A, B, C) and a single dependent variable (total number of tasks completed). We also recorded the number of errors. The order of the interfaces was completely balanced for all six possible orderings, with each possible ordering traversed by four teams, and the tasks were randomly selected for each interface from a set of tasks.

Our hypotheses about the study's outcome were as follows:

H1.1 Users will complete more tasks with both interface B and C than with interface A.

H1.2 Users will complete more tasks with C than with B.

H1.3 Users will prefer interface C over both A and B.

H1.4 Users will feel more confident about their task performance with interface C.

5.1.4 Participants

We had a total of 48 participants in the main study, 18 to 39 years old (average 23.3), 27 male and 21 female, who worked together on 24 teams as detailed in Table 5.1.

All participants had normal or corrected vision. 40 reported they did not know the other participant, 4 had "met before," and 4 "knew each other well." 93% stated they had at least 10 years of experience speaking English. Each user was compensated for their time commitment of about one hour with a nominal amount of USD 10. We had two further participant teams

local user remote user	female female	female male	male female	male male
# of teams	6	4	5	9

Table 5.1: Gender distribution and participant teams.

whose data we did not include in the analysis. In one team, one individual had a form of color-blindness; in the other team, one participant did not adequately follow the study administrator's instructions during the training period.

5.1.5 Procedure

At the beginning of the study, each participant was given a color blind test and filled out a pre-study questionnaire with demographic and background information. The study administrator then verbally explained the scenario and the roles each participant would play. Next, the local user was given the hand-held display and adjustments to the strap were made so that the user was comfortable holding the device in one hand.

For each interface, the administrator explained the respective interface. The administrator was careful to only explain the individual features and to not recommend any particular strategy; instead, the users were explicitly told that it was up to them to determine how exactly they would make use of the features. Next, the users were given a training session of five minutes to get accustomed to the interface and develop a strategy for using it. During this training session, the administrator would correct any mistakes made by either user, and would also encourage the users to try out the different features if they had not already done so. After the training session, a measured session of seven minutes was started. Before starting this session, the administrator gave the stipulation that selecting the correct elements in the correct order was important and that participants should confirm with each other if in doubt, but that within that, users should work as fast as possible.

After each interface, users filled out a brief questionnaire asking about their experience with the interface and any physical discomfort they may have felt. While some local users indicated that their arms got tired, all of them explicitly confirmed verbally that they did not have any concerns about continuing the study.

Finally, the participants were asked to fill out a post-study questionnaire, asking them to compare the interfaces and to note any further comments on the study.

5.1.6 Results

Task performance

The stipulation to not make mistakes and confirm with each other when in doubt worked very well: 16 out of 24 teams completed all three trials without a single error, and only two teams made more than one error in one trial (in which users would select around 20 to 60 buttons). The errors were spread evenly among the three interfaces. We conclude that all users worked meticulously and thus that the comparison of the number of completed tasks is meaningful.

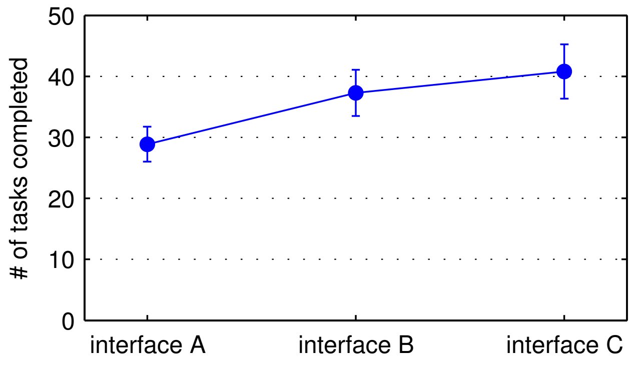


Figure 5.3: Main quantitative result: number of tasks completed as a function of the three different interfaces, shown are mean and 95% confidence intervals. Users were significantly faster with B than with A, and significantly faster with C than with A (cf. Table 5.2).

On average, participants completed 28.9 tasks with Interface A, 37.3 tasks with Interface B, and 40.8 tasks with Interface C (Figure 5.3). Analyzing the results, Mauchly's test did not show a violation of sphericity against interface ($W(2) = 0.94$, $p = 0.49$). With a one-way repeated measures ANOVA, we found a significant effect of interface on the number of tasks completed with $F(2,46) = 23.45$, $p < 0.001$, and $\eta^2_{\text{partial}} = 0.50$. Using Tukey's posthoc analysis (Table 5.2), we found that users completed significantly more tasks with both interfaces B and C than with interface A, thus confirming hypothesis H1.1. We did not find a significant difference between B and C, hence hypothesis H1.2 could not be confirmed despite the higher average number of completed tasks with interface C.

Interfaces	Difference	Lower	Upper	P Adj.	
B – A	8.42	4.08	12.75	< 0.001	*
C – A	11.92	7.58	16.25	< 0.001	*
C – B	3.50	-0.83	7.83	0.135	

Table 5.2: Tukey's posthoc analysis for all pairwise comparisons with a 95% family confidence interval. * indicates significant differences.

Questionnaires

In the post-study questionnaire, users were asked to rate their level of agreement to the statements "This interface helped me to solve the task" ('helpfulness'), "This interface made me feel confident that I was doing the task correctly" ('confidence'), and "I had difficulties using this interface" ('difficulties') for each interface, as well as rank them ("Which of the interfaces did you like best/would you choose to use for a real task?"). The results are aggregated in Figure 5.4.

Assuming that the scale of the ratings is linear and analyzing them with a mixed model ANOVA, with interface and user role as fixed effects and team ID as random effect to account for within-team correlations (note that the questionnaire responses are per user, while the task performance discussed above are measured per team), the following effects were found:[1]

[1]We note that a different model was used in the first publication of these results (Gauglitz et al., 2012a). The linear mixed model used here models the design of the study more closely, and the report here is more detailed. However, we emphasize that all effects reported in Gauglitz et al. (2012a) based on the previous analysis are confirmed by our new analysis.

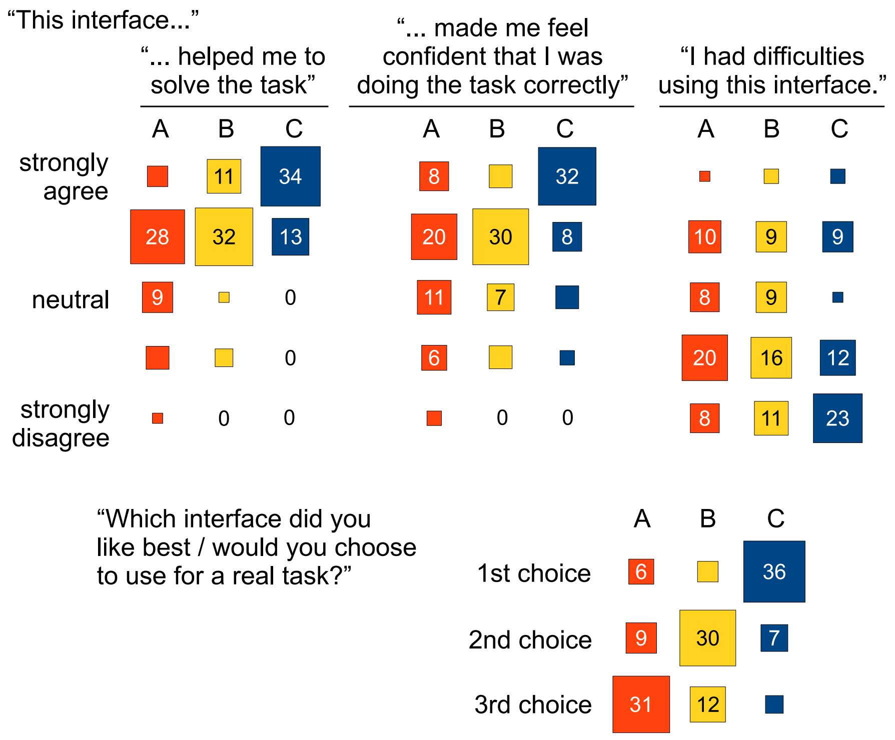


Figure 5.4: User responses from post-study questionnaire.

There is a significant effect of interface on the rating for 'helpfulness' (F(2,109) = 34.05, $p < 0.001$) and 'confidence' (F(2,109) = 15.55, $p < 0.001$). Post-hoc tests indicated that interface C was rated significantly better than both A and B for both questions; interface B was better than A for 'helpfulness', but not 'confidence.' Looking at the histograms (Figure 5.4) for 'helpfulness' and 'confidence', it is notable that users rated interfaces A and B well overall, but frequently reserved the highest rating for interface C alone: C received the highest rating with regards to 'helpfulness' from 72% of the users (compared with 8.5% for A and 23% for B) and the highest rating with regards to 'confidence' from 68% (compared with 17% for A and 11% for B).

Further, there was a signification interaction of interface and role on 'helpfulness' (F(2,109) = 4.09, p = 0.02): for the local users alone, there was no difference between interfaces A and B. The remote users were more 'confident' about task execution than the local users according to a borderline significant effect (F(1,109) = 4.01, p = 0.048), but the interaction between role and interface was not significant for this question (p = 0.07).

There were no significant differences in the answers to "I had difficulties using this interface" despite a trend towards better ratings for interface C (F(2,109) = 2.12, p = 0.13).

With respect to the ranking of the three interfaces, there is a significant difference in ranking according to Friedman's test for both user roles ($\chi^2(2) = 15.36$, p = 0.0005 (local user), and $\chi^2(2) = 25.58$, $p < 0.0001$ (remote user), respectively). Pairwise comparisons (with Bonferroni's correction applied) indicated that interface C —,selected by 79% of the users as their first choice — was ranked significantly better than both other interfaces in all cases.

Many users further confirmed their preference for interface C with comments: *"The interface with vision tracking was better and easier."*; *"I was very impressed with the tracking capabilities. The interface was very easy to understand"*; *"This interface is an order of magnitude better than the others"*; *"The tracked markers [were] much eas[ier] to give direction[s] with."*

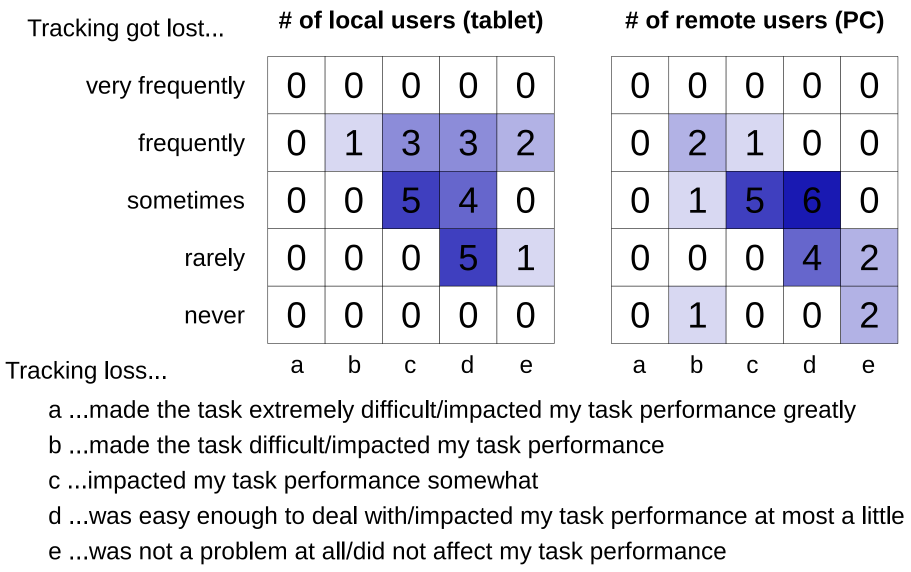
These results confirm hypotheses H1.3 and H1.4.

Very few users seemed to be distracted by the additional features and the multimodality of the task or commented to that effect, for example: *"It was actually easier to just talk it [out] rather than worry about clicking the mouse which left me distracted."*

Perception of tracking loss

Since our tracking system was not perfect and tracking loss would be clearly noticeable to participants, we were also interested in the users' subjective perception of the frequency and impact of tracking loss. The results are aggregated in Figure 5.5.

Overall, 61% of the users reported 'at most a little' or 'no' impact on task performance, another 29% 'some' impact. One user commented: *"The tracking did disappear once in awhile, but it was easy enough to communicate until the tracking reappeared. Overall, the tracking [...] was really helpful and effective."* Only one tablet user found that tracking loss had made the task difficult. It is interesting that three of the remote users had the impression that tracking 'never' got lost: it is very unlikely that this was indeed the case, but the interface (in particular, the freezing) might have made the tracking loss transparent to the remote user.



Tracking got lost...	# of local users (tablet)					# of remote users (PC)				
very frequently	0	0	0	0	0	0	0	0	0	0
frequently	0	1	3	3	2	0	2	1	0	0
sometimes	0	0	5	4	0	0	1	5	6	0
rarely	0	0	0	5	1	0	0	0	4	2
never	0	0	0	0	0	0	1	0	0	2
Tracking loss...	a	b	c	d	e	a	b	c	d	e

a ...made the task extremely difficult/impacted my task performance greatly
b ...made the task difficult/impacted my task performance
c ...impacted my task performance somewhat
d ...was easy enough to deal with/impacted my task performance at most a little
e ...was not a problem at all/did not affect my task performance

Figure 5.5: Users' perceptions of tracking loss. The matrices aggregate the results of two different questions on the intermediate questionnaire filled out immediately after using interface C.

As study 2 followed a similar pattern as study 1, we first describe study 2, then discuss the results of both studies together in Section 5.3.

5.2 User Study 2

In user study 2, we compared Prototype 2 (Section 4.2) with two baselines: a video+audio only interface and an interface with static annotations (also called "markers" throughout the study). Both the local and the remote users were study participants. The general design and method were, purposefully, very similar to study 1: we wanted to create a similar validation and comparison with baselines of the new prototype system.

This study will be published, together with Prototype 2, in the proceedings of the ACM Symposium on User Interface Software and Technology (UIST) 2014 (Gauglitz et al., 2014a).

We first conducted several pilot study trials with a total of 20 users (10 teams), during which we refined study parameters, overall procedure, and training procedure.

5.2.1 Task & physical setup

We chose a "car repair" task for the study. That is, the local user stood in front of a car, hood open, and received help from the remote user in "identifying the problem" (cf. Chen et al. (2013) for a similar scenario). The study took place outdoors, and while we used a relatively isolated location with no direct sunlight, several environment factors were beyond our control, including light conditions, passers-by, noise from a nearby street and a nearby airport, all of which added to the realism of our study. The remote user's PC, including network infrastructure, was mounted onto a metal cart on wheels and positioned adjacent to the car such that the participants could verbally communicate but not see each other.

To make sure that the individual tasks were roughly equivalent, quick enough to perform, independent of each user's dexterity, and not dangerous in any way, we used proxy tasks that would require similar communication between the users but little or no physical labor, such as locating individual elements and finding pieces of information. For example, instead of unscrewing a bolt, we asked the users to identify its size by testing it with a set of provided nuts, which requires the same communication between the users in order to identify the correct

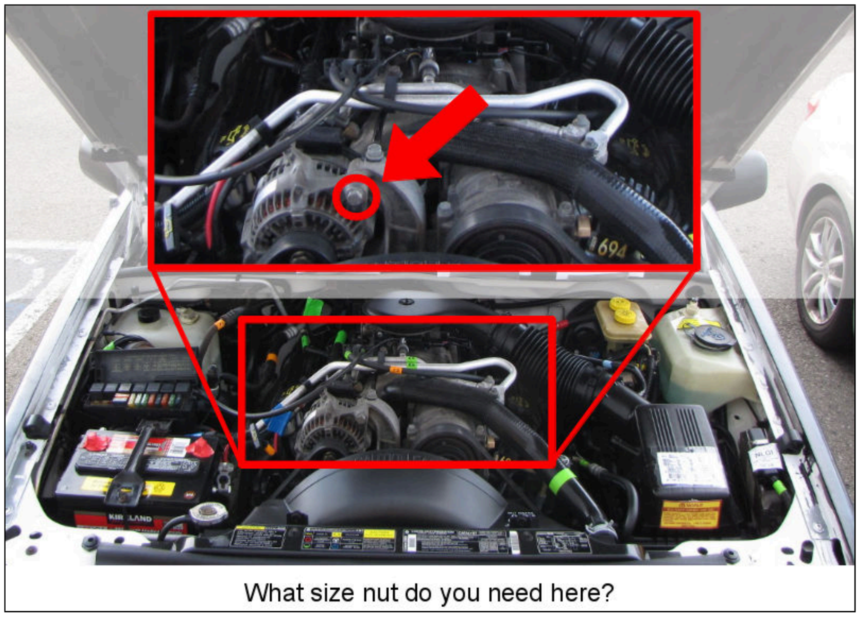


Figure 5.6: One example out of the 80 individual tasks. These instructions were provided to the remote user, who then needed to communicate them to the local user.

element. The remote user was given simulated expert knowledge as a list of diagrams with specific questions (e.g., size or label on a particular screw or cable, rating of a particular fuse, serial number of a part) as well as where its answer could be located (see Figure 5.6). The remote user then had to communicate this information to the local user, who would locate the requested information and write it down.

5.2.2 Conditions

As in study 1, in all conditions, the two users were able to talk to each other without restrictions.

- **Interface A: video only.**

- **Interface B: video + static markers.**
- **Interface C: Prototype 2 as presented in Section 4.2.**

Again, conditions B and C both allowed up to five concurrent markers in different colors, with further clicks re-setting the oldest marker.

5.2.3 Experimental design

We used a within-subjects design with one independent variable (interface type) and one dependent variable (task completion time). We also recorded the number of errors and obtained several user ratings via questionnaires. The order of the interfaces was completely balanced, with each of the six possible orderings traversed by five of the 30 teams. For each team, three lists with 15 tasks were created at random, with the remaining tasks reserved for training.

Our hypotheses about the study's outcome were as follows:

H2.1 Users will complete the task faster with interfaces B and C than with interface A.

H2.2 Users will complete the task faster with C than with B.

H2.3 Users will prefer interface C over both A and B.

5.2.4 Participants

60 users (18–30 years (mean 20.8), 29 female, 31 male) participated in the main study, working together in 30 teams as detailed in Table 5.3. Each user received a compensation of USD 10; all

local user remote user	female female	female male	male female	male male
# of teams	8	7	6	9

Table 5.3: Gender distribution and participant teams.

teams additionally competed for a bonus of USD 10 / 20 / 40 per person for the top 5 / second fastest / fastest error-free task performances for all three conditions combined.

In two teams not included in the numbers above, one user was color blind. It remains unclear whether this affected the task performance. However, we used color extensively (markers, labels in the car, etc.), and at least one of the users was unable to disambiguate a few of the elements. We thus decided not to include the data from these two trials in the analysis.

5.2.5 Procedure

Each participant completed a color blind test and a pre-study questionnaire with background information. The general setup and task was then explained in detail.

For each session, the study administrator explained the respective interface features in detail, then the users conducted several training tasks with the respective interface. Each user completed a minimum of five training tasks for each new feature and was asked explicitly if they were comfortable using the interface before proceeding to the timed session. After each session, the users filled out an intermediate questionnaire rating this interface. Lastly, each user filled out a post-study questionnaire and received their compensation.

During the pilot study trials, it quickly became clear that not all of the camera control features that our prototype (interface C) featured were necessary for this particular environment and task, and that the limited amount of time prohibited explaining and training the user on each of them. We thus concentrated the training on a subset of features that appeared to be most useful in this context, namely, the freezing of the camera, and the saving/revisiting of viewpoints. However, the other features (panning, zooming, change viewpoint via click) were still available and were occasionally discovered and used by participants.

Like any monocular SLAM system, our system requires a stereo initialization (i.e., a distinct camera movement) before it tracks robustly. We address this particular limitation in Chapter 6. For the time being, to ensure a consistently high quality of initialization, we conducted this initialization step (until the modeler had started to extract 3D surface) before handing the device to the local user.

5.2.6 Results

Task performance

Overall, 98.5% of the tasks were answered correctly (21 errors at a total of $30 \times 3 \times 15$ tasks). Analyzing the number of errors, Mauchly's test indicated that the assumption of sphericity against interface had not been violated ($W(2) = 0.95$, $p = 0.50$), and no significant effect of interface on the number of errors was found using a one-way repeated measures ANOVA ($F(2,58) = 0.47$, $p = 0.63$, and $\eta^2_{\text{partial}} = 0.016$). Additionally, we note that 5 of the 21 errors

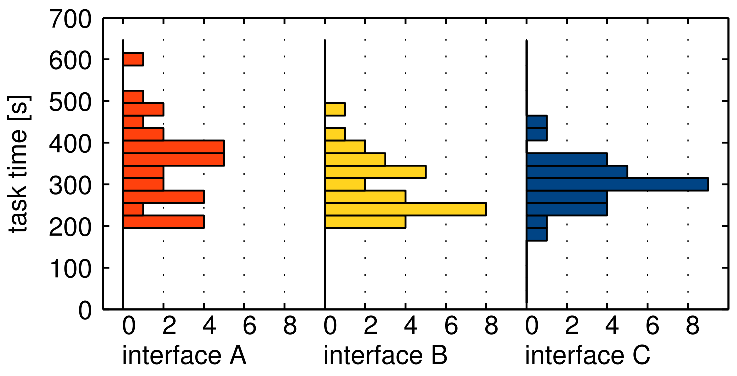


Figure 5.7: Histogram of task times per interface.

were made on two particular subtasks in which the participant may have misread the expert knowledge diagram. As in study 1, we conclude that all users worked meticulously and thus that the comparison of the task times is meaningful.

Analyzing the task time, Mauchly's test indicated that the assumption of sphericity had not been violated (W(2) = 0.99, p = 0.93). With a one-way repeated measures ANOVA, we found a significant effect of interface on task completion time with F(2,58) = 6.94, p = 0.0020, and $\eta^2_{\text{partial}} = 0.19$. Post hoc comparisons using Tukey's HSD test indicated that users were significantly faster with both interfaces B (M = 313.6, SD = 69.6) and C (M = 317.5, SD = 57.6) than with interface A (M = 364.7, SD = 96.7), thus supporting hypothesis H2.1. No significant difference was found between B and C; hence, hypothesis H2.2 was not supported.

Questionnaires

In the intermediate questionnaires filled out immediately after each session, the users were asked to rate their level of agreement to the statements, “This interface helped me to solve the task” (‘helpfulness’), “This interface made me feel confident that I was doing the task correctly” (‘confidence’), and “I had difficulties using this interface” (‘difficulties’). The responses are aggregated in Figure 5.8.

We note two differences in methodology compared to the respective questions in study 1: First, we used a 7-point scale rather than a 5-point scale as before, in order to allow for more nuanced ratings. Second, in study 1, these questions and the ranking of the interfaces appeared together on the post-study questionnaire. Due to this, the users may have taken their ranking into account when filling out the ratings on the individual questions. To seperate the ratings and the ranking from each other, the ratings were now obtained in the intermediate questionnaries, that is, before the next interface was introduced.

Analyzing the ratings with a mixed model ANOVA, with interface and user role as fixed effects and team ID as random effect to account for within-team correlations as in the analysis of study 1, the following effects were found:[2]

There is a significant effect of interface on the rating for all of the above questions (in the above order: $F(2,145) = 18.89$, $p < 0.001$; $F(2,145) = 20.69$, $p < 0.001$; $F(2,145) = 4.05$,

[2]We note that a different model was used in the publication of these results in Gauglitz et al. (2014a). The linear mixed model used here makes more assumptions about the nature of the data, but models the design of the study more closely, and the report here is more detailed. However, we emphasize that all statistical effects reported in Gauglitz et al. (2014a) based on the previous analysis are confirmed by the analysis here.

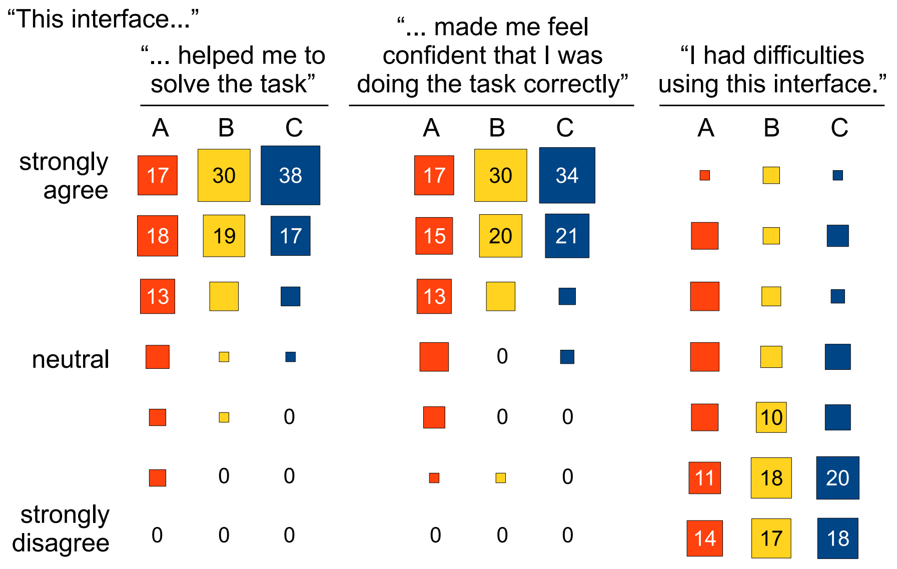


Figure 5.8: Results from intermediate questionnaires: interface ratings.

p = 0.02). Post-hoc tests indicated that on the first two questions, interfaces B and C were rated better than A; for 'difficulties,' interface C was rated better than A.

There were no significant effects of user role or the interaction of interface and user role on 'helpfulness' and 'confidence.' With regards to 'difficulties,' user role had a significant effect — local users found it less difficult than remote users — (F(1,145) = 30.51, $p < 0.0001$), and there was a significant interaction of interface and user role (F(2,145) = 3.27, p = 0.04), indicating that (a) the difference between the interfaces was mainly due to the local users' ratings (with no difference present for the remote users alone), and (b) the local users reported less difficulties than the remote users for both interfaces B and C, but no difference between the roles was found for interface A.

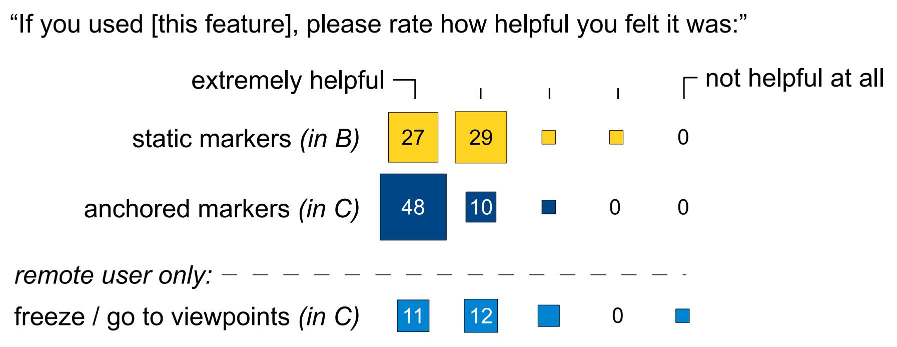


Figure 5.9: Results from intermediate questionnaires: individual features.

Users were further asked to rate the helpfulness of individual features on a 5-point scale from "extremely helpful" to "not helpful at all" (Figure 5.9). The anchored markers in interface C were perceived as more helpful than the static markers in interface B (F(1,87) = 17.30, p = 0.0001), with 80% of the users perceiving the former as "extremely helpful." There was no effect of user role or interaction. The camera control features were perceived as "extremely" or "very" helpful by 77% of the remote users.

In the post-study questionnaire, users were asked to rank the three interfaces ("Which interface did you like best / would you choose to use for a real task?"). There is a significant difference in ranking according to Friedman's test for both user roles ($\chi^2(2) = 41.86$, $p < 0.0001$ (local user), and $\chi^2(2) = 32.34$, $p < 0.0001$ (remote user), respectively). Pairwise comparisons (with Bonferroni's correction applied) indicated that all pairwise differences are significant. 80% of the users selected interface C as their first choice (cf. Figure 5.10), supporting H2.3. The preference for C is 83% among tablet users and 77% among PC users, but this difference was not significant according to Wilcoxon's paired signed rank test (V = 29, p = 0.45).

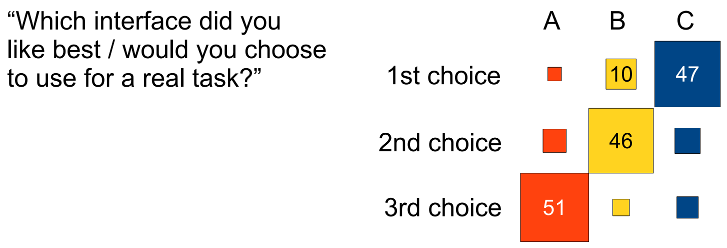


Figure 5.10: Results from post-study questionnaire: interface preference.

Observations and open-ended questions on the questionnaires revealed several interesting details, most notably an inadvertent side effect of the world-stabilization of markers: Since the markers in interface C has a constant world size, they ended up being quite large when the user got very close to the object, and were thus too large for some tasks, as described by this user: *"Overall, the markers and viewpoints were extremely helpful. However, [...] for the tasks where text needed to be located, the marker was much bigger than the words."* Having constant screen size, the markers in B did not have this side effect.

Several users expressed their preference for interface C, but commented that too many keys had to be memorized: *"[C] was more useful [...] but it was sometimes difficult to remember which buttons would navigate to which screen shots, what to press to unfreeze the pane, and so on. However, I imagine it would get much easier with additional practice"*; *"[C] was by far the most helpful user interface, but it was a bit difficult to use due to the [number of] buttons."*

5.3 Discussion of Studies 1 and 2

To summarize the results of studies 1 and 2, in both studies, an overwhelming majority of the participants (in both roles) preferred our respective prototype C over both baselines (H1.3 and H2.3 supported); users performed significantly faster with it than with the respective video-only baseline A (H1.1 and H2.1 supported); but no significant difference in task performance was found in comparison with the respective static marker condition B (H1.2 and H2.2 not supported).

Overall, most participants seemed very engaged with the task in both studies, and many explicitly commented that they liked the experience. In both studies, with both interfaces B and C, most teams effectively used multimodal communication, by indicating elements with markers as well as providing verbal explanations at the same time.

5.3.1 Use of features

The use of the interface features varied among teams. In both studies, almost all teams used markers for the overwhelming majority of tasks. Most used the freeze feature (in study 1) and save/go-to-viewpoint feature (in study 2), though some remote users consciously decided against it. In study 1, only a few people used the number keys to change the sequence of markers.

5.3.2 Task performance with interface C vs. interface B

Given the overall very promising user ratings of the respective interface C and few usability problems, the question remains why no significant difference was found between C and B in terms of task performance (i.e., number of tasks completed in study 1 and time to complete the set of tasks in study 2). We observed indications for several possible reasons that may have reduced the differences between the interfaces in general and/or counteracted potential benefits of interface C in particular. They largely fall into three categories: limitations of our respective interface, nature of the chosen task, and user study artifacts.

Limitations of our respective prototype (i.e., interface C):

- **Imperfect tracking.** We emphasize that our prototypes are built upon visual tracking, which is naturally imperfect, but being compared against interfaces which (in our studies) were technically failsafe. Although not perceived as a large problem by most users in study 1 (cf. Figure 5.5), occasional or more frequent tracking loss may have impacted the task performance. One noteworthy circumstance is that, in both systems, tracking got lost if users moved very close to the object surface (due to fewer trackable features in the field of view), so that users had to step back a little to recover tracking: *"I found that when I [...] needed to 'zoom in' to physically put the push pins in the correct spots, the tracking marks sometimes did not show up [...] until I zoomed out again... this slowed* *things down a bit"* (study 1). Improving tracking for this condition in particular would be beneficial.

- **Suboptimal interface design.** While the interface received very good ratings, and there was no indication that it was perceived as more difficult to use than the baselines (cf. Figures 5.4 and 5.8), individual elements were found to be suboptimal. In particular, in Prototype 2, several users had commented that memorizing the keys was a burden, and the size of the markers was unsuitable for some tasks (cf. the discussion of this aspect in Section 5.2.6). We note that both of these issues are resolved in Prototype 3.

Nature of the task:

- Since the local user had to wait for instructions from the remote expert for each step, it was relatively easy (although, as some users in study 1 commented, more strenuous) to hold the tablet still while the remote expert referenced elements. A task that required more autonomous work from the local user would likely show a greater benefit of the decoupling of views.

User study artifacts:

- **Statistical power** (study 1 in particular). As evident from Figure 5.3 and Table 5.2, in study 1, the average task performance with C was better than with B; the p value of 0.135 suggests that with higher statistical power (i.e., more participants), C might have been shown to have a significant benefit in task performance as well (even with all other factors unchanged).

- **Training effect & small task environment.** As users started, the difficulty of verbally giving spatial and directional instructions — including confusion regarding “left” and “right” (relative to the car’s driving direction or the user’s perspective?) and “up” and “down” (in world or screen coordinates?), as well as misidentification of elements — was very apparent, which supports the need for spatial annotations. However, as an artifact of the training, the within-subjects design, and the small task environment, users quickly became experts in giving and understanding directions for this particular task, possibly more so than becoming experts in using the interfaces. Oftentimes, users came up with names for different parts of the object, which did not have to be technically accurate in order to be effective (the most creative moniker we heard may have been “rainbow keyboard” for the fuse box of the car in study 2). As one user commented when asked about the usefulness of the camera control (in study 2): *“[It] wasn’t very useful since I already knew the layout of the engine from previous tasks.”* For real applications, we might assume the opposite: users become familiar with the interfaces available to them and communicate with new partners in new environments. To account for this, significantly different environments could be used for each training and interface session; however, this may be challenging in terms of practicality and ensuring fair conditions across all sessions.

- **Simulated expert knowledge & proxy tasks** (study 2 in particular). Due to the increasingly familiar environment, and additionally motivated by the rewards in study 2, the tasks became very fast-paced. Thus, the remote user's mental load of understanding the next instructions (cf. Figure 5.6), aligning them with his/her mental model of the scene, and then with his/her view of the scene, was oftentimes slower than the actual task execution, which required little physical labor and could thus be completed very quickly. As the time-critical path shifted from the local to the remote user, a potential benefit of the virtual camera control — namely, that the remote user could browse and familiarize him/herself with the environment (e.g., for the next task) — became less relevant, while a potential downside (increased complexity) became more relevant. One user commented (in study 2): *"Using the multiple views was a little more hectic on the 'expert's' side, but only because we were trying to complete the tasks as quickly as possible. In a situation in which this technology would be used, this feature would no doubt be very helpful for quality and efficiency of task completion."* We note that this is also an artifact of the study setup: in a real application, no simulated expert knowledge has to be understood, and the local user's task may require more physical labor or travel.

- **Limited training time.** While the user's ratings suggest that users did not perceive the interface as more difficult to use (cf. Figures 5.4 and 5.8), it may require more training than was alotted here for optimal use. Individual users commented that C might benefit from longer-term use: *"I would utilize B [...]. However, with more time to adapt, I* *probably would utilize C”* (study 1); *“I thought using the markers and the freeze was more difficult [...]. I think with another trial’s worth of practice it would have less effect.”* (study 1); *“... I imagine [using interface C] would get much easier with additional practice”* (study 2).

- **Limited amortization of one-time time investments due to short task duration** (study 2). In study 2, with interface C, some users took extra time to carefully position the camera to save viewpoints for later use, but the amortization of this time investment was limited by the relatively short task duration.

We further note that due to our use of a magic lens tablet, our AR experience was indirect. Some users explicitly commented on this: *“I had to see where the X was on the screen then find it again in ‘real life.’ That transition added a non-trivial delay”* (study 1). Due to this indirection and since they could hold the tablet in one hand, putting down pins with the other, interfaces B and C are effectively more similar to each other than with a direct view (e.g., with an HWD or projective display).

Given that certain aspects of the interface mentioned above have specifically been addressed with the design of Prototype 3, and that several of the other observations hint at artifacts of the study setup (rather than fundamental limitations of the prototypes or interfaces), we believe that we have gathered strong evidence for the benefits of our paradigm, in particular given that underlying technologies (such as SLAM) will continue to improve.

5.4 User Study 3

In contrast to studies 1 and 2, study 3 does not aim at evaluating the interaction with a system as a whole, but rather shed light on the design of individual features. In particular, we wanted to obtain insights on the interpretation of 2D drawings in 3D (Section 4.3.3) and the suitability of gesture-based virtual navigation for constrained spaces (Section 4.3.4), both of which differ significantly compared to their use in existing work. Rather than a comparative, task performance-based study, study 3 was thus designed as a qualitative study.

This study will be published, together with Prototype 3, in the proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST) 2014 (Gauglitz et al., 2014b).

5.4.1 Participants

We asked eleven users to interact with the system following a structured protocol and provide feedback on particular design elements. They were compensated for their time commitment of about 50 minutes with US-$ 10. Of these study participants, five were female; the age range was 18–22 years. Four stated that they were "somewhat familiar" with interactive 3D software (e.g., 3D modelers). All had used small-scale touchscreens (e.g., smartphones) on a daily basis, but only one had used large-scale touchscreens more than "occasionally to rarely." Five of the participants had participated in study 2 (Section 5.2) and were thus also asked to comment on the differences compared to the earlier design.

5.4.2 2D drawings in 3D

We asked the users to communicate a piece of information to a hypothetical partner via a drawing. For example, for Figure 4.6(a) top to bottom: “In which direction do you have to turn the knob?”; “Where should the microwave be placed?”; “Where is the motor block?”.

We tried to cover a range of different questions to prompt a variety of drawings. We also used scenes in which the original viewpoint was roughly perpendicular to the object’s main surface (Figure 4.6 top) as well as slanted surfaces (Figure 4.6 bottom two rows). Further, to be able to distinguish conceptual issues from sensitivity to noise, we used both virtual models (Figure 4.6 top two rows) and models created by our system via SLAM and approximate surface modeling (Figure 4.6 bottom). We avoided situations in which all variants result in the same or nearly the same shape, that is, single planar surfaces.

We then moved to a different viewpoint and asked the users which of the four depth interpretations (Figure 4.6(b-e)) was most suitable.

We emphasize that we did not tell the users *what* they should draw, and thus the drawn shapes varied appreciably in nature. For example, to refer to a particular object, some users traced the object’s outline (similar to Figure 4.6 bottom), some circled it loosely, and others drew an arrow pointing towards it.

Arrow heads and indicating direction. To indicate a direction of rotation (e.g., Figure 4.6 top row), all users used arrow-like drawings. However, the type of arrow head varied, as detailed in Table 5.4. Initially, roughly one quarter of all arrows were drawn without head; that

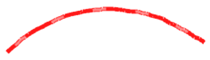
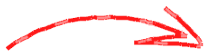
arrow head	none	attached	detached
example			
initially:	24.6%	23.1%	52.3%
with animation:	72.3%	15.4%	12.3%

Table 5.4: Usage of different styles of arrow heads initially and after an animation visualizing the drawings' direction was introduced.

is, the shape itself does not actually convey the direction. Users evidently assumed that the direction of the drawing would implicitly be communicated. As the drawings mimic gestures (and one would rarely add an arrow head at the end of a hand gesture motioned in mid-air), this assumption is reasonable.

We had anticipated this variant due to prior observations and thus implemented an animated visualization to do so: here, the line contains lighter dashes which flow in the direction in which the line was drawn.[3] Thus far, we use a fixed speed of flow, but one could extend this visualization by additionally visualizing the speed of drawing via the speed of flow.

After this animation was introduced after the first set of tasks, almost three quarter of all arrows were drawn without head (cf. Table 5.4). The animation was rated as "very helpful" or "helpful" by 10 of 11 users; nobody perceived it as distracting (even though it was not necessary for some of the drawings) (cf. Figure 5.12).

[3]The design was inspired by the visual appearance of animated Line Integral Convolution (Cabral and Leedom, 1993) and can loosely be thought of as a coarse approximation for it for a single dimension and constant speed.

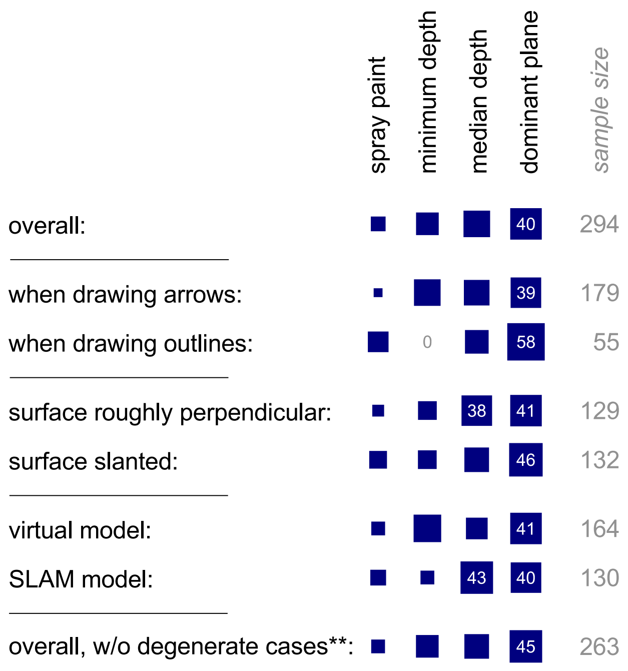


Figure 5.11: Users' preference among the four different depth interpretations, broken down by various subsets. Per each row, the areas of the squares and the numbers indicate the preference for a particular variant in percent.

User preference among depth interpretations. Figure 5.11 details the users' preference among the depth interpretations, broken down by various subsets. Overall, in every category, users tended to prefer the planar projections in general, and the dominant plane version in particular, which was the most frequently chosen variant in every category but one.

Limitations. One limitation of our current implementation is that drawings are treated as separate annotations as soon as the finger is lifted, which means that they get moved onto separate planes by the three planar projection methods (cf. Figure 4.6(c-e) middle row). Individual segments that are created in very close succession — such as a detached head for an arrow,

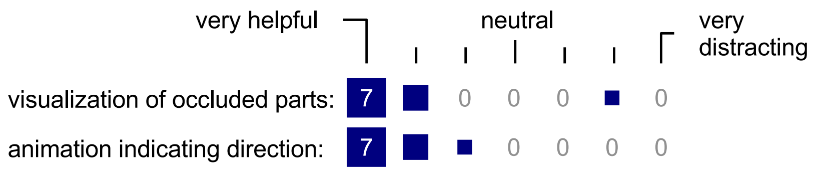


Figure 5.12: Ratings of two visualization options (# of users).

which was by far the most commonly used style of arrow before we introduced the animation indicating direction (cf. Table 5.4) — should ideally be treated as one entity.

Further, the "dominant plane" method is susceptible to extreme artifacts if the points lie on or close to a single line, such as seen in Figure 4.6(e) middle row. When removing the cases in which this occurred from consideration, the general preference for the dominant plane version increases further (last row in Figure 5.11). In future work, these degenerate configurations should be detected and one of the other variants (e.g., median depth) used instead.

Naturally, there are several cases in which none of these options will work satisfactorily, such as when trying to create an annotation in mid-air or behind physical surfaces. Of course, an interface may also offer to change the type of depth inference used, e.g., for advanced users. Even for 2D only, some interfaces offer a whole array of drawing tools (Gurevich et al., 2012); however, it has been reported that users use freehand drawing most often (Gurevich et al., 2012). With this consideration, the quest here is to find out what the default setting should be, such that gestures used in face-to-face communication can be emulated as efficiently as possible.

Given our results, we suggest that a planar projection will likely serve this purpose best, in particular if the implementations described here are improved upon. Three areas for improvement are: 1. group segments created in close succession; 2. avoid degenerate cases for the case of dominant plane estimation; 3. combine the methods and choose, for example, the dominant plane if the fraction of supporting inliers is large, and median depth plane otherwise. These ideas are left to future work.

5.4.3 Gesture-based navigation

We asked the users to try out the individual navigation controls and rate them. The results are aggregated in Figure 5.13.

After having been told only that two-finger gestures would control the camera (since one-finger gestures draw annotations), and asked to pan, all users intuitively guessed the gesture (i.e., swipe) correctly — unsurprisingly, perhaps, given their exposure to touchscreen interfaces — and were able to control the camera as suggested; the control received high ratings in terms of both intuitiveness and ease of use. Additionally, all users correctly identified what the intensifying blue gradient communicated; this visualization was rated as "very helpful" by 7 of 11 users, and as "helpful" by the others (cf. Figure 5.13).

For zooming, again all users intuitively guessed the gesture (i.e., pinch) correctly and were able to control the camera as suggested. We let them try out zoom first with the transitional component disabled, i.e., only the FOV was changed. This control was given the highest rating

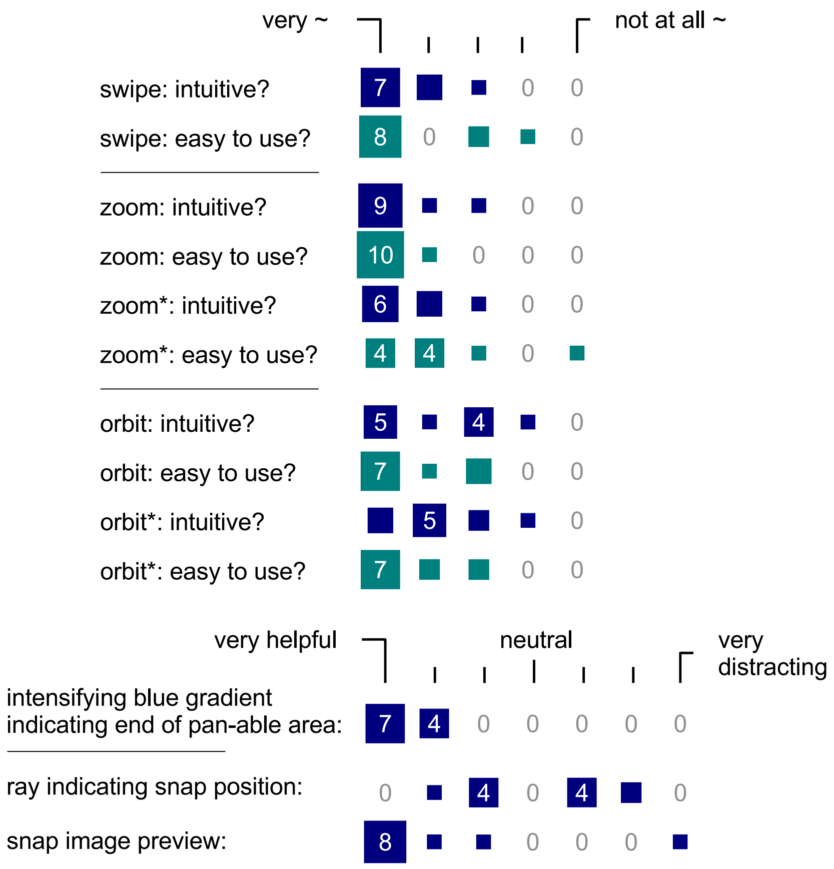


Figure 5.13: Ratings of various navigation elements (areas of squares are proportional to number of users).

on a 5-point scale for intuitiveness and ease of use by 9 and 10 out of 11 users, respectively. With transitional component enabled (labeled “zoom*” in Figure 5.13), the control still received high ratings, though clearly lower than without. Several users appreciated the fact that it transitioned to other frames and thus allowed to zoom further, however, the decreased smoothness and possibility of getting “stuck” were noted as downsides. The idea of using a hysteresis threshold to decrease artifacts was a result of this feedback (i.e., it was not implemented at the time of this evaluation).

For orbiting, we first introduced the control on a *virtual* model without any constraints/snap-to-keyframe. While the ratings for intuitiveness are lower than for the other methods, the ratings for ease of use are similar. With snap-to-keyframe, on a model reconstructed from images (labeled “orbit*” in Figure 5.13), the ratings are very similar, suggesting that the constraint was not irritating. The two visualizations received very different ratings, however: while the preview image was rated as “very helpful” by 8 of 11 users, the red line was perceived as “(somewhat or slightly) helpful” by only half the users, and as “(somewhat or slightly) distracting” by the other half. Conversations made clear that despite explanations, not all users understood and/or saw value in this visualization.

We suggest that the consistently high ratings indicate that the controls are designed appropriately. By design, our controls are dependent on the viewpoint distribution; an area that deserves further investigation is how to ensure that the user cannot get “stuck” in the case of unfavorable viewpoint distributions.

Chapter 6

Model Estimation and Selection towards Unconstrained Tracking and Mapping

A key component of our approach (as described in Chapter 3) is the ability to track and map the environment from the video stream (or, more generally speaking, the input that the sensors provide).

This is a research problem which has received considerable attention from the robotics, computer vision, and augmented reality communities. Two important characteristics of a tracking and mapping (T&M) system are the type of camera motion and the geometry of the environment that it supports. For example, a T&M system may assume a planar environment (as done in Prototype 1 (Section 4.1) and, for example, Pirchheim and Reitmayr, 2011) or a camera that is rotating around its optical center (DiVerdi et al., 2009; Wagner et al., 2010). SLAM systems (such as Davison et al., 2007; Eade and Drummond, 2008; Klein and Murray, 2007; Newcombe et al., 2011a, as well as the one employed in Prototype 2 (Section 4.2)) can deal with environments of arbitrary geometry and any camera motion that induces parallax. However, with few exceptions (Civera et al., 2008b), they do not support rotation-only camera motion: Their mapping is intrinsically built upon triangulation of features; thus, they require that each feature be observed from two distinct camera locations and may produce degenerate maps or fail completely if the camera rotates from one part of the environment to another.

Therefore, most monocular visual SLAM systems need to be initialized with a distinct "traveling" movement of the camera *for each newly observed part of the environment*, and the required travel distance is directly proportional to the distance to the environment. This restriction is acceptable for vehicle navigation or if building a model of the environment is the user's main intent. However, it is a major limitation for the use of SLAM systems in AR in general and in our framework in particular, where the environment modeling is assumed to be done in the background and ideally transparent to the user, who should not be required to move a certain way in order to make the system work. Moreover, rotation-only "looking around" is a very natural motion and may occur in many application scenarios (Langlotz et al., 2011; Wagner et al., 2010), particularly in ours (imagine a local user wanting to give an overview of his/her current environment).

Thus, the paradigm for modeling the environment should be to make the best possible use of all data that can be casually collected and to enable viewing and placement of annotations for as much time as possible. In particular, this means not forcing the user to concentrate on model building, and not discarding all frames that stem from rotation-only movements (as in most monocular SLAM systems) or translations (as with panorama mapping).

In this chapter, we present an approach and proof-of-concept implementation that fulfills these criteria: We describe a real-time tracking and mapping system that explicitly supports

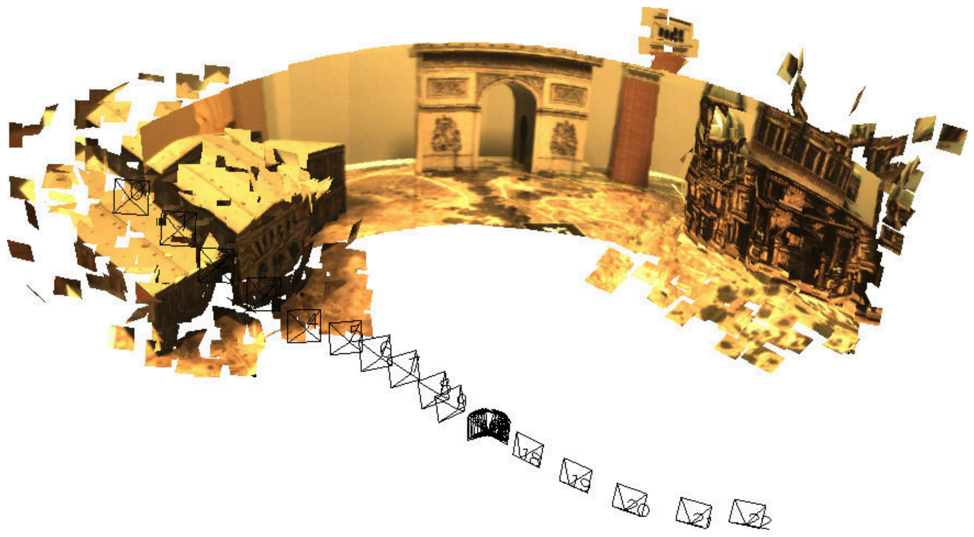


Figure 6.1: Our system supports both parallax-inducing and rotation-only camera motion. In the former case, it acts as a SLAM system and models 3D structure in the environment; in the latter case, it acts as a panorama mapper. It seamlessly switches between the two modes, thus being able to track and map through arbitrary sequences of parallax-inducing and rotation-only camera movements and — fully automatically and in real time — creating combinations of 3D structure and panoramic maps, as shown here. © IEEE 2014.

both parallax-inducing and rotation-only camera motions in 3D environments (Figure 6.1), does not need a separate initialization step, and continues to collect data despite intermittent tracking loss. In the case of intermittent tracking loss, it creates several disjoint maps which are later merged if possible. One key element of our approach is the use of the 'Geometric Robust Information Criterion' (GRIC) by Torr (2002) (adapted to support large search regions (Section 6.5.1) and the absolute pose models (cf. Section 6.5.2)) to decide whether the current camera motion can best be represented as a parallax-inducing motion or a rotation-only motion.

Here, the GRIC score is one representative of a class of metrics, sometimes called information criteria, that have been proposed to assess the fitness of a particular model given the data. In SfM, the GRIC score has been applied particularly to detect homographies in order to *avoid* them during keyframe selection (Pollefeys et al., 2002; Repko and Pollefeys, 2005). In contrast, we use the GRIC score to select between two models, but we use the data in either case.

An earlier iteration of the work described in this chapter was presented at the IEEE International Symposium for Mixed and Augmented Reality 2012 (Gauglitz et al., 2012b); the present version, presenting a significant generalization of the work over the first publication, was published in the IEEE Transactions on Visualization and Computer Graphics (Gauglitz et al., 2014c); © IEEE 2012/2014, reprinted with permission.

6.1 Alternative Approaches

In theory, using stereo cameras eliminates the problem of requiring the camera to travel, since the baseline required to triangulate features is built-in. In practice, however, using stereo cameras is only a partial remedy, since the baseline has to be significant in relation to the distance to the environment in order to reliably estimate depth. Thus, a wearable stereo system would be unable to map a building across the street without requiring the user to provide additional baseline by traveling (while a panorama system, though unable to provide depth, would produce very usable information).

Similarly, systems based on active depth sensors (Lieberknecht et al., 2011; Newcombe et al., 2011b) do not require the sensor to move. However, they have other inherent limitations such as limited range, inability to work in sunlight, power consumption, and need for additional special hardware.

We are thus interested in addressing this problem for monocular vision.

As briefly mentioned in Section 2.4, Civera et al. (2008a) have proposed the use of an alternative, six-dimensional parametrization for filter-based SLAM systems which supports rotation-only motions, but does so at a high computational cost. As a result, the number of features that can be tracked in real time, which is typically already smaller for filter-based SLAM than for keyframe-based SLAM,[1] is further decreased. In the case of long sequences of camera rotation, many of those computation cycles are spent filtering data where no gain is to be expected (namely, on re-estimating the (still undefined) feature depth). Conceptually, assuming that feature depths are likely to be correlated, whether or not existing features can be triangulated using a new camera view depends more on the camera movement than on each individual feature, and could thus be decided once per frame. In Civera et al. (2008a,b), however, the question is answered for each feature individually.

Therefore, we consider it an advantage of our approach that we explicitly switch to panoramic mapping if supported by the observations, thus taking advantage of some of the advantages that panoramic mapping offers, such as a robust outlier-resilient model (homography) and a straightforward mapping of the entire frame instead of sparse features, both of which are especially important for AR. On the other hand, the approach of Civera et al. (2008b) may be

[1]For an interesting analysis of the relative computational cost of the two approaches see Strasdat et al. (2010).

preferable when modeling the transition between two types of movement or when only some of the features exhibit parallax.

While admitting features without depth could in principle be adopted for keyframe-based SLAM (in Klein and Murray (2009), this approach is employed to admit features before their depths are known), the ability to rely on them exclusively would require fundamental and possibly costly changes to the underlying mapping and bundle adjustment.

Concurrent to our second publication (Gauglitz et al., 2014c) on this topic, Pirchheim et al. (2013) have published an approach to the same problem. On a high level, both approaches are similar: both distinguish rotation-only and parallax-inducing motion and switch between panorama mapping and SfM mapping accordingly. Based on an existing SLAM system, the implementation presented by Pirchheim et al. (2013) was demonstrated to operate in real time on a mobile device. However, their approach is significantly less general: Their system supports embedding of panoramas into an existing SfM map to support temporary rotation-only sequences, but assumes that the system has been initialized with a standard stereo initialization. It is effectively similar to the model-based part of our system (i.e., the left branch in Figure 6.4) and thus, in contrast to our approach, supports neither starting with rotation nor tracking through arbitrary sequences of rotation-only and parallax-inducing motions.[2]

[2]We note that Pirchheim et al. (2013) include a detailed theoretical comparison of their approach with the first iteration of our work (Gauglitz et al., 2012b), but several aspects of this comparison no longer apply to the second iteration (as described here and in Gauglitz et al. (2014c)).

6.2 System Overview

Our concept borrows two key ideas from Klein and Murray's PTAM system (Klein and Murray, 2007, 2008, 2009), namely, the central role of keyframes and the splitting of tracking and mapping into two parallel threads.

The split and design of the threads follows two guidelines: (1) the tracking thread should be as fast as possible and leave all tasks that are not imminent in order for the next frame to be processed to the mapping thread, which runs asynchronously in the background; (2) the first steps in the pipeline should not be dependent on the motion model (parallax-inducing vs. rotation-only) that will be selected, in order to minimize redundant computations.

Figure 6.2 presents a conceptual overview of the system's operation. Briefly summarized, the system operates as follows: The tracking thread receives a new frame and locates features via template-based matching, which stem either from the previous frame or, if a map of the environment has already been built, were projected from the map into the frame. From these feature correspondences, it then estimates both a model for parallax-inducing camera motion as well as a model for rotation-only motion, and selects the model that better represents the data via comparing their GRIC scores. Under certain conditions, a *keyframe* is inserted. Consecutive keyframes of the same kind (SfM or panorama) are collected into a *keyframe group*. The mapping thread operates on these sets of collected keyframes and creates an appropriate environment model (SfM or panorama).

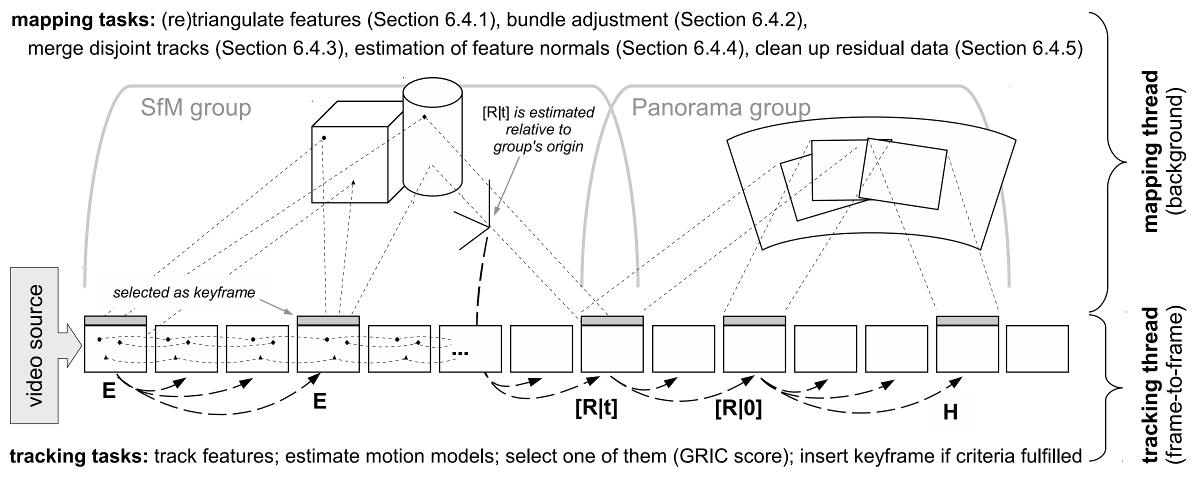

mapping tasks: (re)triangulate features (Section 6.4.1), bundle adjustment (Section 6.4.2),
merge disjoint tracks (Section 6.4.3), estimation of feature normals (Section 6.4.4), clean up residual data (Section 6.4.5)
SfM group
[R|t] is estimated relative to group's origin
Panorama group
mapping thread (background)
video source
selected as keyframe
E
E
[R|t]
[R|0]
H
tracking thread (frame-to-frame)
tracking tasks: track features; estimate motion models; select one of them (GRIC score); insert keyframe if criteria fulfilled

Figure 6.2: Conceptual overview of the system's operation. © IEEE 2014.

The tracking and mapping threads are described in detail in Sections 6.3 and 6.4, respectively. Section 6.5 describes the problem of model selection and the GRIC score.

6.2.1 Two-view relations and model-based poses

Conceptually, two two-view relations (namely, the essential matrix E and the homography H) are all-encompassing in that they describe all cases of camera movements that can occur in a static environment. Consequently, our initial implementation used only these two models (Gauglitz et al., 2012b). However, this implementation does not make optimal use of the environment model that is built in the background; by sequentially estimating E from potentially small baselines, tracking remains relatively brittle and jittery (cf. Figure 6.11).

Therefore, we have extended our concept to include both two-view relations (for initialization, after tracking loss, or when rotating into uncharted territory) as well as two model-based absolute poses $[R|t]$ and $[R|0]$[3] (whenever the camera observes a part of the environment for which a 3D model has been built). It should be noted that, in each frame, we still estimate only two models (the two-view relations *or* the absolute pose models); thus, the computational load has not increased.

The integration of absolute pose models has two further advantages: First, it allows to distinguish rotation-only movement from planar environments (which is difficult in the model-free case, since in both cases, the resulting transformations are described by the same relation, namely, a homography). Second, it decreases the risk of fragmentation of the model into connected sub-maps of alternating types (which a linear sequence of E's and H's may produce, as discussed in Gauglitz et al. (2012b)), since the system can connect incoming frames directly to an existing model (rather than only to the previous keyframe).

Why is estimating $[R|0]$ necessary? Unlike in the case of E vs. H, when a model is available, the absolute pose $[R|t]$ is well-defined and can be estimated irrespective of the (relative) camera motion. Thus, it is less obvious why estimating $[R|0]$ and the subsequent model selection step is necessary. We do so for the following reason: Consider the case that the camera observes a known scene (thus using model-based tracking), then rotates towards unobserved parts of the scene. Due to the rotation-only movement, no new features can be triangulated. If $[R|t]$ is the only model-based pose that is estimated, and has priority as long as enough triangulated features are visible, the system will switch to H only when very few (if any) of the existing

[3]More precisely, $[R|t]_{\text{prev}} \circ [R|(0,0,0)^T]$, where $[R|t]_{\text{prev}}$ is the (fixed) pose of the previous keyframe.

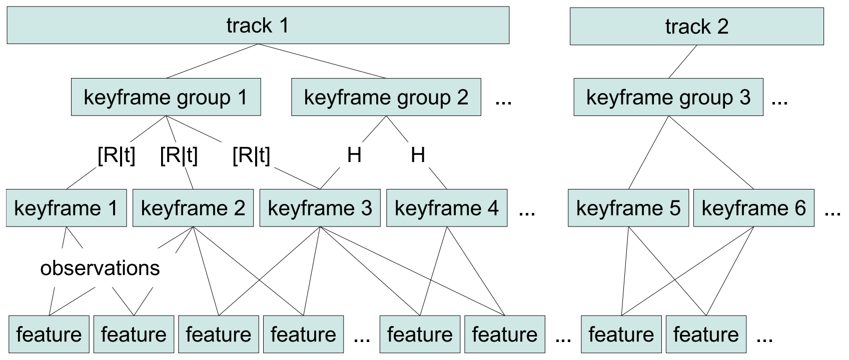


Figure 6.3: Schematic overview of the main data structures used to store the emerging map(s). © IEEE 2014.

features are left, risking tracking loss in between and generating a panorama that has very little overlap with the existing model. By estimating $[R|0]$ and thus explicitly switching to panorama mode (if $[R|0]$ proves to better represent the data), we can start building the panorama immediately, ensuring that it is well-connected to the existing structure by the time the system switches to H, seamlessly continuing the panorama as it extends into newly observed parts of the scene.

6.2.2 Data structures

Figure 6.3 visualizes the main data objects that store the current system state and the emerging map as well as their relations.

The most central element is the keyframe group: each keyframe group governs one submap consisting of a set of keyframes which are either all linked by 3D pose information (SfM group) or all linked by homographies (panorama group). The keyframe group also determines the frame of reference, with respect to which all pose information is stored. A keyframe may be part of a panorama group as well as a SfM group (e.g., keyframe 3 in Figure 6.3), in which case it gets assigned a pose in both groups. The set of keyframes and keyframe groups can be thought of as a graph, similar to the representations by Eade and Drummond (2008) and Klopschitz et al. (2010). Initially, this graph is linear, consisting of a chain of E's and H's modeling the (linear) video stream. As soon as the system has built a model of the environment and is using model-based tracking however, new keyframes are linked directly to the model, creating a branching graph.

When tracking is lost, all links to the current map are lost, and the tracker starts a new track. Initially, the new track is completely unconnected to the previous data, but can later be merged (if there is some overlap in what is observed during both tracks) as explained in Section 6.4.3.

6.3 Tracking

A flowchart of the tracking thread is presented in Figure 6.4. This section describes the main operations in detail.

When a new keyframe is added, new keypoints are detected using a corner detector (such as FAST (Rosten and Drummond, 2006) or Shi-Tomasi (Shi and Tomasi, 1994)) in all areas not already covered by keypoints. We enforce a spatially well-distributed set of keypoints, which was shown to improve tracking robustness (Gauglitz et al., 2011; Gruber et al., 2010b),

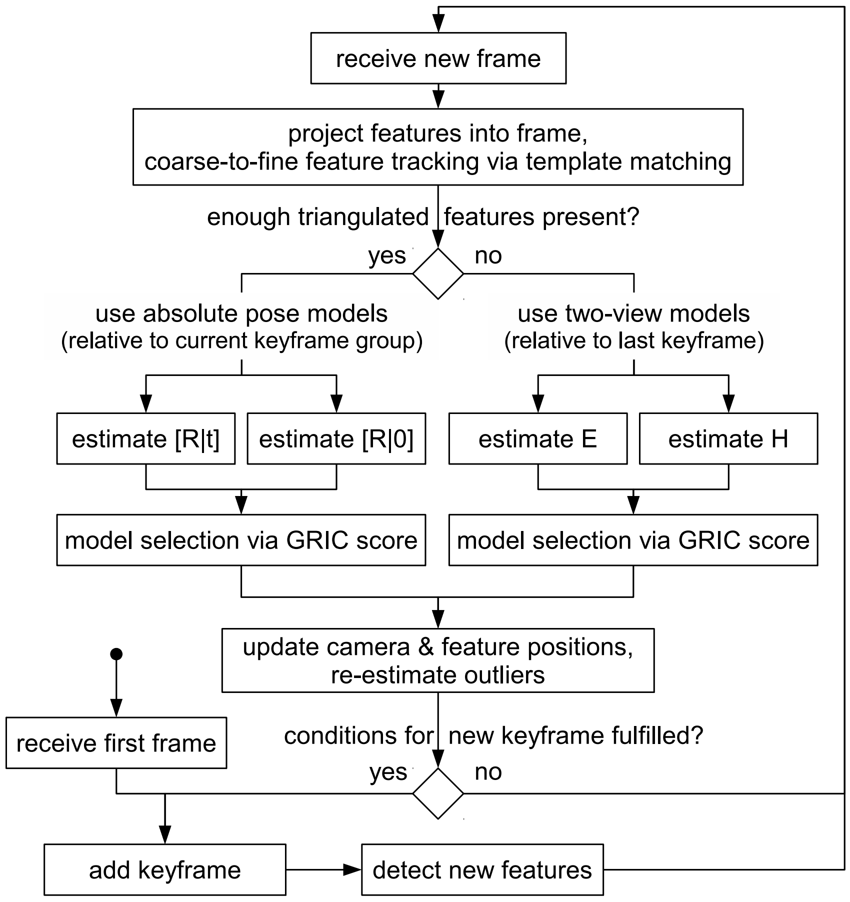

receive new frame
project features into frame, coarse-to-fine feature tracking via template matching
enough triangulated features present?
yes
no
use absolute pose models (relative to current keyframe group)
use two-view models (relative to last keyframe)
estimate [R|t]
estimate [R|0]
estimate E
estimate H
model selection via GRIC score
model selection via GRIC score
update camera & feature positions, re-estimate outliers
receive first frame
conditions for new keyframe fulfilled?
yes
no
add keyframe
detect new features

Figure 6.4: Flowchart of the tracking thread. © IEEE 2014.

by overlaying a simple rectangular grid over the image and selecting the strongest keypoint in each cell.

To project existing features into the frame and predict their appearance, as much information about the feature as possible is used. That is, if the 3D position and the normal (cf. Section 6.4.4) of the feature are available, it is sampled from the original keyframe using a fully qualified homography (representing a 3D plane rotation). If only the 2D location of the feature is known, the patch is sampled in 2D.

6.3.1 Coarse-to-fine feature tracking

Frame-to-frame feature correspondences are created using a multi-level (i.e., coarse-to-fine), active search patch tracker with normalized cross-correlation (NCC)-based template matching. On the full-resolution image, the feature location is refined to subpixel accuracy by using a quadratic fit to neighboring scores.

This is similar to the keypoint tracking by other systems (Klein and Murray, 2007; Wagner et al., 2010, 2009); however in contrast to these system, the multi-level tracking is executed on a per-feature basis (instead of interleaved with the pose estimation), since we do not know which type of camera motion to expect (and thus which model to enforce) until after the tracking step. We have designed, but not yet fully integrated, a more sophisticated approach that retains the advantages of the interleaved pose estimation, and discuss this approach in Section 6.7.2.

Model	Estimation algorithm
Essential matrix E	MAPSAC (Torr, 2002) with five-point algorithm (Nistér, 2004)
Homography H	MAPSAC (Torr, 2002)
Abs. pose $[R\|t]$	Gradient descent with M-estimator (cf. Klein and Murray, 2007)
Abs. orientation $[R\|0]$	Gradient descent with M-estimator (cf. Wagner et al., 2010)

Table 6.1: Estimation algorithms for the four models considered here. In each frame, only two models are estimated. © IEEE 2014.

6.3.2 Model estimation and outlier re-estimation

After all features are located in the full-resolution image, the motion model is estimated. If a 3D model of the observed scene is available (that is, there exists a SfM group which contains sufficiently many of the successfully tracked features), we estimate $[R|t]$ and $[R|0]$ using iterative gradient descent of an M-estimator (cf. Table 6.1 bottom). Otherwise, we instead estimate both a homography H and an essential matrix E between the previous keyframe and the current frame using MAPSAC (cf. Table 6.1 top).

The probability density function that is assumed for inliers and outliers is an important part of the model selection process. Here, we make the common assumption that inliers are distributed normally with measurement error σ, and outliers are distributed uniformly across the search region. Thus, after model estimation, the measurement error σ and inlier ratio γ, which are needed for the model selection step, are estimated using expectation-maximization.

Next, the GRIC score is computed for both models, and the better model (i.e., the one with lower GRIC score) is selected. The model selection and the GRIC score will be explained in more detail in Section 6.5. If E is determined to be the better fit, it is decomposed into a relative pose $[R_{\text{rel}}|t_{\text{rel}}]$.

In an attempt to retain individual features as long as possible, outliers are re-estimated after the model selection. This is trivial in the cases of H, $[R|t]$, and $[R|0]$, since the model defines a unique mapping for each point. In the case of E, each outlier is re-estimated by running the NCC-based template matching again on a thin rectangular matching area that was rotated to align with the epipolar line. Features that prove unreliable (i.e., that repeatedly are outliers and need to be re-estimated) are removed.

If no keyframe is added (cf. next section), processing of this frame is completed.

6.3.3 Inserting a new keyframe

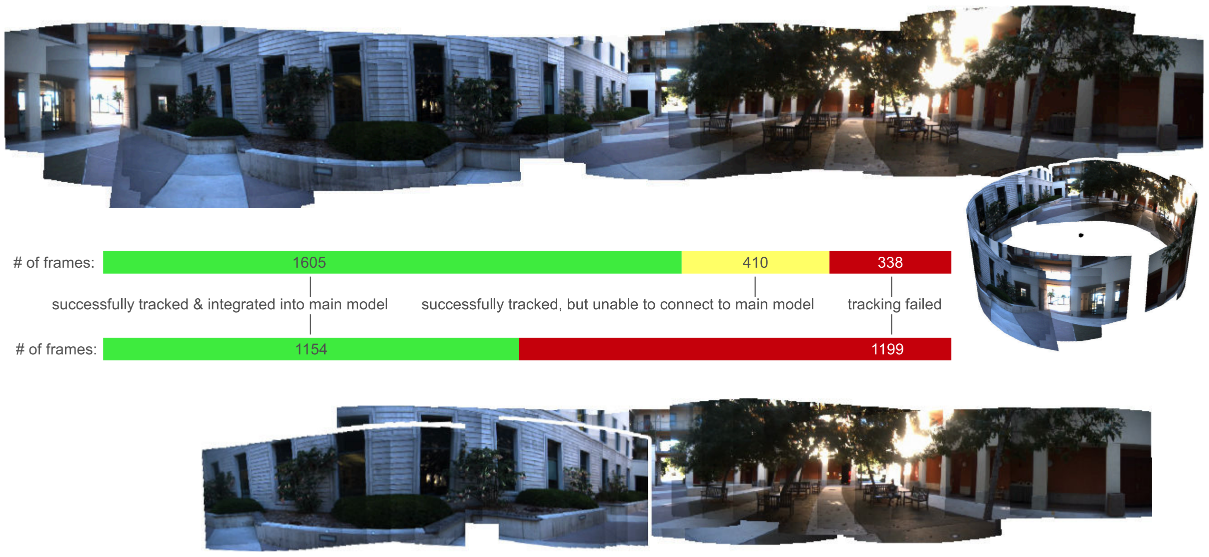
The current frame is added as a new keyframe k_{new} when several conditions (similar to the ones suggested by Klein and Murray (2007)) are met: (1) tracking quality is good (as determined by the fraction of inliers that MAPSAC finds); (2) enough time has passed since the last keyframe insertion; (3) in the case of rotation-only motion (H and $[R|0]$), when the median 2D distance that the keypoints “traveled” since the last keyframe is large enough, and in the case of parallax-inducing motion (E and $[R|t]$), when the median feature triangulation angle is large enough.

If the estimated motion model connects k_{new} to an existing keyframe group of the respective type (i.e., a SfM group for E or $[R|t]$, and a panorama group for H or $[R|0]$), k_{new} gets merged into the existing group (detailed below for the case of E, and straightforward in all other cases).

Otherwise, a new keyframe group consisting of k_{new} and the previous keyframe k_{prev} gets created. Insertion of a new group marks the beginning of a new sub-map, but it does not invalidate any of the existing map data: particularly during model-based tracking, where both feature and camera positions are known in 3D, the tracker can still refer to all currently registered features and project them into incoming frames. Note that a switch from $[R|0]$ to $[R|t]$ does *not* cause creation of a new SfM group: coming from $[R|0]$ implies that 3D features are visible and thus that there is an SfM group to which the new keyframe is connected even after intermittent rotation-only motion.

Lastly, new features are detected in all uncovered image regions by applying the same grid as in the first frame and choosing new features for each cell that is not covered by currently tracked features.

Merging essential matrices. While E can be decomposed into relative pose information $[R_{\text{rel}}|t_{\text{rel}}]$ between the new frame and the previous keyframe k_{prev}, the scale of t_{rel} is arbitrary. Before it can be integrated into an existing SfM group, a common scale has to be found. In order to do so, we use the set of all features that have been observed (and thus have a triangulated position) in both the existing SfM group as well as with respect to $[R_{\text{rel}}|t_{\text{rel}}]$, and calculate the ratios of their distances to k_{prev} in both coordinate systems. We then take the median of those ratios as a robust measure of the scale between the two coordinate systems and scale t_{rel} accordingly.



# of frames:
1605
410
338
successfully tracked & integrated into main model
successfully tracked, but unable to connect to main model
tracking failed
# of frames:
1154
1199

Figure 6.5: Merging of tracks (top) vs. relocalization (bottom). The rotation-only input video for the data shown here contains several rapid camera motions (cf. description of the dataset in Coffin et al. (2011)), resulting in intermittent tracking loss. After each loss, our system starts a new partial panorama, and, while the incoming live frames are being processed, stitches them together in the background. The final panorama (top and right) was stitched together from 11 partial panoramas. In comparison, with the same tracking system but using relocalization, only the model at the bottom gets created. Here, two pairs of keyframes were estimated to have a small non-zero baseline, thus two parts of the panorama — displayed with a white border — are offset slightly from the rotation axis. © IEEE 2014.

6.3.4 Relocalizing vs. starting a new track

When tracking gets lost — i.e., when the number of inliers for the selected model falls below a set threshold — the standard strategy employed by most T&M systems (e.g., Klein and Murray, 2007; Pirchheim and Reitmayr, 2011; Wagner et al., 2010) is to continuously try to *relocalize* the camera with respect to the current map with each new frame until successful. However, this means that tracking and mapping are suspended and no data is collected until relocalization is successful.

Here, we employ an alternative strategy proposed by Eade and Drummond (2008): instead of trying to relocalize, we start a new track immediately, and leave it to the background thread to merge tracks if possible (cf. Section 6.4.3). The benefit of this method, illustrated in Figures 6.5 and 6.12, is that the system continues to collect data even after tracking failure occurs, and, if the tracks are later found to overlap, merges the created maps. If they do not overlap, the maps remain separate. (Note that in this case, a recovery-based system would never recover.)

6.4 Mapping

The mapping thread runs in parallel to the tracking thread and is responsible for the following tasks:

1. triangulate new features (Section 6.4.1),
2. run bundle adjustment (Section 6.4.2),
3. merge disjoint tracks (Section 6.4.3),
4. estimate feature normals (Section 6.4.4),
5. clean up residual data (Section 6.4.5).

These tasks are allowed to be more computationally intensive than the tracker's tasks, since the system does not depend on them in order to process the next frame.

Each of the tasks gets assigned a priority which depends on the time it was last executed and the system's current state. In each iteration, the task with highest priority is executed. For example, after a new keyframe is inserted, triangulation of new features becomes important; if tracking got lost and thus a new track is started, merging of tracks is prioritized to enable quick merging of tracks. Thus, assuming that the two threads run on two independent cores, the latter can happen as quickly as conventional relocalization.

6.4.1 Triangulating features

A feature that was observed in at least two keyframes within the same SfM group, but not bundle adjusted yet, is triangulated and gets assigned a 3D location with respect to this group's frame of reference. When the tracker adds a new keyframe with a new observation of a feature f, f is re-triangulated using all information when the mapper cycles through this step the next time.

6.4.2 Bundle adjustment

SfM groups with at least three keyframes get passed through a standard bundle adjuster (Lourakis and Argyros, 2009) that globally optimizes all keyframe (i.e., camera) poses and feature positions in this group's frame of reference. If there are multiple such groups, the group with the newest keyframe (that is, the keyframe that got adjusted the least number of times) is processed.

More sophisticated bundle adjustment strategies with local and global adjustment steps (Klein and Murray, 2007) could be integrated as needed.

6.4.3 Merging disjoint tracks

When tracking gets lost, the tracker immediately starts a new, independent *track*, rather than continuously trying to relocalize with respect to the existing map. In doing so, the tracker continues to collect and 'stitch together' data even though the spatial relation to the first map is initially unknown.

The algorithm that merges tracks is similar to the keyframe-based recovery by Klein and Murray (2008) (also used in Pirchheim and Reitmayr (2011); Wagner et al. (2010), among others): as observed by Eade and Drummond (2008), recovery, loop closure, and (here) merging of tracks are effectively similar to each other; the main difference lies in when the algorithm is executed and how its result is used.

Whenever a new keyframe is taken, the system stores a downsampled, blurred copy of the image (here: 80×60 pixels, blurred with a Gaussian with $\sigma = 1.5$px), dubbed small blurry image (SBI).

Merging of tracks is implemented as follows: The algorithm chooses a keyframe k_1 and computes the normalized cross-correlation (NCC) of its SBI with the SBI of all other keyframes. Keyframes on the same track as k_1 are omitted, as are keyframes to which a previous merge attempt failed. The keyframe k_2 with the highest NCC score is selected, and the SBIs of k_1 and k_2 are aligned to each other using inverse compositional image alignment (Baker and Matthews, 2002) of an affine homography H_A. The features of k_1 are then projected into k_2 using H_A, and a regular "tracking" step (cf. Section 6.3) is executed.

If the tracking step fails, k_2 is "blacklisted" in k_1 as a failed attempt (so that the algorithm does not attempt the same combination again), and k_1 stores a timestamp of when this merge attempt occurred. The next time the algorithm tries to merge tracks, the keyframe that has not been chosen as k_1 the longest is chosen as k_1.

If the tracking step succeeds in estimating a model that is supported by a sufficient fraction of feature correspondences, the two tracks are considered successfully merged. Several different cases have to be considered in actually merging the environment models, depending on the type of transition connecting the two tracks, and whether or not k_1 and k_2 are already part of a keyframe group of the respective type. If available, the transition is preferred that would not introduce a new keyframe group. (For example, if k_1 and k_2 are both part of a panorama group, and could be connected with either H or E, H is preferred.) Adding of a group (if needed) as well as merging of panorama groups is straightforward (the latter only requires concatenating the homographies accordingly). To merge SfM groups, features that are observed in both groups are needed. To generate those, we track the features that are connecting k_1 and k_2 one frame "into" k_2's group via epipolar search. Then, a common scale is computed as described in Section 6.3.3.

The benefit of merging of tracks is visualized in the case of panorama data in Figure 6.5. In this particular case, the map of the merged tracks consists of an almost complete horizontal panorama, to which 1605 frames (68% of the input data) are registered. Another 410 frames are registered to partial panoramas (not shown) which the system was unable to connect to the main model (whether these should be counted as success or failure depends on the application). In comparison, with the same tracking system but using relocalization, while the system is able to recover and expand the initial map several times, only 1154 frames (49%) are tracked; all other data (which "passed by" while the system unsuccessfully tried to relocalize) are discarded. A similar result for 3D data is presented in Figure 6.12.

6.4.4 Estimation of feature normals

To properly predict a feature's appearance from an assumed camera pose, one needs to know not only its 3D position, but also its local structure. Assuming that sufficiently many features are located on locally approximately planar surfaces, this structure can be defined by a normal vector n, and the change in appearance between two views of the feature is given by a homography $H_\perp$, which can be expressed in terms of n, the relative camera pose $[R_\text{rel}|t_\text{rel}]$, and the feature's 3D position x (Molton et al., 2004; Wuest et al., 2008):

$$H_\perp = R_\text{rel} + \frac{t_\text{rel} \cdot n^T}{n^T x} \quad \Leftrightarrow \quad H_\perp = R_\text{rel}(n^T x \cdot I_{3\times 3} + t_\text{rel} \cdot n^T) \tag{6.1}$$

(Note that $H_\perp$ is a homogeneous quantity, i.e., its scale is arbitrary.) The two views will be reasonably similar, as otherwise the tracker would have failed to locate the feature. Yet, small differences between them allow to estimate $H_\perp$ and thus n using image alignment (Molton et al., 2004; Wuest et al., 2008).

If we assume n to be the only unknown, $H_\perp$ has only two degrees of freedom, and it is possible to parametrize it accordingly (Molton et al., 2004). However, as Molton et al. (2004) note, noise in other variables will cause the projections to not coincide precisely, such that one has to allow for at least two additional degrees of freedom for shift. We have had more success with using an affine homography (as in Wuest et al., 2008) with a straightforward six-degree-of-freedom parametrization, which we use in a standard inverse-compositional image alignment framework (Baker and Matthews, 2002), and afterwards extracting n from the over-determined system given by Equation (6.1) using singular value decomposition.

Executing the image alignment step for each feature is fairly expensive; however, it should be noted that it can be run feature-by-feature in the background and is highly parallelizable, and none of the other steps are immediately dependent on it. (While no normal estimate is available, the tracker will continue to use a 2D projection of the feature.)

To maintain and refine the normal vector n over time, instead of feeding it through a Kalman filter (Molton et al., 2004; Wuest et al., 2008), we simply collect independent estimates of n from random pairs of views, the average of which is taken as the currently assumed value of n. This allows us to adapt elastically to the amount of processing power available. For well-textured and indeed planar features, the set of estimates is highly consistent, while less textured and highly non-planar features cause the image alignment to fail to converge or produce a set of wildly diverging estimates. This information could be used to further characterize the features and potentially remove them; this idea is left to future work.

In our current implementation, apart from the added value of having a richer environment model, estimation of normals extended the lifetime of individual features (that is, the number of frames that they were correctly tracked) by a moderate but noticeable amount, as shown in Figure 6.6.

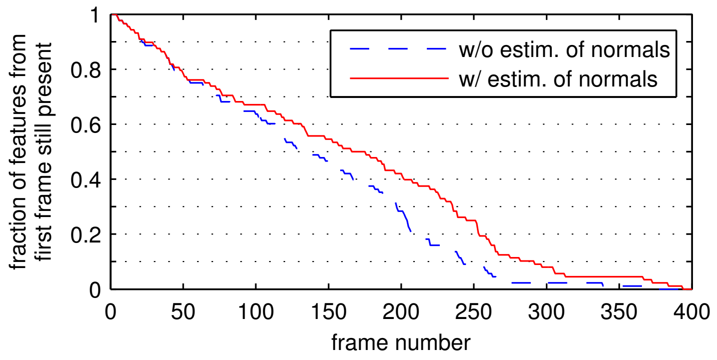


Figure 6.6: Effect of normal estimation on lifespan of features. The data in this figure stems from the first part of the video to Figure 6.1, in which the camera slowly descends around the building on the left. © IEEE 2014.

6.4.5 Cleaning up residual data

When a new track is started and thus a new keyframe is created (Section 6.3.4), but tracking gets lost again immediately after that, this new keyframe (and all features found in it) are not connected to any other data and provide very little useful information. To make sure that this kind of residual data does not accumulate, the mapping thread goes through the set of keyframes, identifies isolated keyframes and removes them.

6.5 Model Selection

If observed data may have arisen from several different models, one faces the problem of selecting the right model in addition to the common problem of estimating the model parameters. Model selection is a complex problem in particular if the models in question are of fundamen-

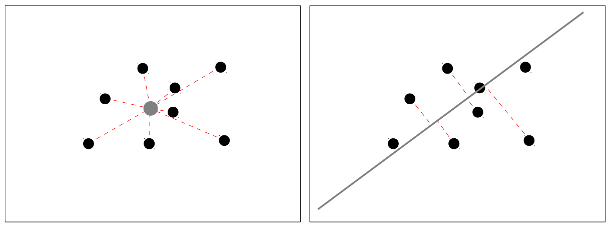


Figure 6.7: Illustration why model selection is difficult: Fitting a 2D point and a 2D line to a set of noisy measurements. Even though both models have the same number of degrees of freedom (2), the dimensionality of the error is different (2 vs. 1), and the sum of errors is guaranteed to be smaller in the case of the line, regardless of how "point-like" the distribution may appear. Cf. Torr et al. (1999), Fig. 16. © IEEE 2014.

tally different kind, as is the case here: a homography is a bijective 2D map, and thus the observed residual between an estimated and measured feature location is two-dimensional, whereas the essential matrix maps a 2D point to a line in 2D, and thus the residual is one-dimensional (perpendicular to the line). This is analogous to trying to determine whether an observed set of 2D points can best be represented by a point or a line (Figure 6.7). It becomes apparent that residuals alone are not sufficient to select the model.

For this reason, several different metrics have been proposed (Akaike, 1974; Rissanen, 1978; Schwarz, 1978; Torr et al., 1999; Torr, 2002). Here, we use the Bayesian formulation of the 'Geometric Robust Information Criterion' (GRIC) by Torr (2002), which is based on the Bayesian probability that a given model generated the observed data. In this section, we first describe the GRIC score for the case of the two-view relations E and H, then its application to the estimation of the absolute poses $[R|t]$ and $[R|0]$. We use the same notation and variable names as Torr (2002). The subscript m is used to indicate that a quantity is dependent on the models that are being compared. $\log(x)$ denotes the natural logarithm.

6.5.1 Generalized GRIC score for two-view relations

The generic formula of Torr's Bayesian GRIC score is

$$\text{GRIC}_m = -2\mathcal{L}_{\text{MAP},m} + k_m \log n \tag{6.2}$$

(cf. (Torr, 2002, Eq. 46)) where k_m is the number of parameters of the model, n is the number of observations, and $\mathcal{L}_{\text{MAP},m}$ denotes the maximum a-posteriori log likelihood of model m given the data, which is based on the probability density functions (PDFs) that the data are assumed to stem from. (Note that the GRIC score is a *cost* function, i.e., lower score indicates better model fit.)

For the problem at hand, we assume a mixture of inliers affected by Gaussian noise, and outliers sprinkled uniformly in the search region. Specifically (cf. (Torr, 2002, Eq. 15)):

$$\mathcal{L}_{\text{MAP},m} = \sum_i \log(\gamma_i \cdot p_{\text{in}} + (1-\gamma_i) \cdot p_{\text{out}}) \tag{6.3}$$

$$\text{with} \quad p_{\text{in}} = \frac{\sqrt{2\pi\sigma^2}^{d_m - D}}{c_m} \cdot \exp\left(-\frac{e_{i,m}^2}{2\sigma^2}\right) , \tag{6.4}$$

$$p_{\text{out}} = 1/v , \tag{6.5}$$

$\gamma_i \in \{1, 0\}$ indicates if correspondence i is an inlier, $e_{i,m}$ are the individual reprojection errors, σ is their standard deviation, D is the dimensionality of each datum (here: a pair of 2D points, i.e., $D = 4$), d_m is the dimensionality of the model manifold, $D - d_m$ is the dimensionality of the error (for H, $D - d_m = 2$, while for E, $D - d_m = 1$ (perpendicular to the epipolar constraint, in analogy to Figure 6.7)), and c_m and v are the volumes of the spaces from which inliers and outliers, respectively, arise (cf. Torr, 2002).

Let γ denote the expected inlier ratio $E(\gamma_i)$, and maximize over γ_i:

$$\Rightarrow\ \text{GRIC}_m = -2\sum_i \log(\max\{\gamma \cdot p_{\text{in}}; (1-\gamma)p_{\text{out}}\}) + k_m \log n$$

$$= \sum_i \min\{\underbrace{-2\log(\gamma \cdot p_{\text{in}})}_{(*)}; \underbrace{-2\log((1-\gamma)p_{\text{out}})}_{(**)}\}) + k_m \log n \quad (6.6)$$

$$(*) = -2\log\left(\gamma \frac{\sqrt{2\pi\sigma^2}^{d_m - D}}{c_m} \cdot \exp\left(-\frac{e_{i,m}^2}{2\sigma^2}\right)\right) \quad (6.7)$$

$$= \frac{e_{i,m}^2}{\sigma^2} + (D - d_m)\log 2\pi\sigma^2 + 2\log\frac{c_m}{\gamma} \quad (6.8)$$

$$(**) = -2\log\left(\frac{1-\gamma}{v}\right) \quad (6.9)$$

$$= T_m + (D - d_m)\log 2\pi\sigma^2 + 2\log\frac{c_m}{\gamma} \quad (6.10)$$

$$\text{where} \quad T_m = 2\log\left(\frac{\gamma}{1-\gamma} \cdot \frac{v}{c_m}\right) - (D - d_m)\log 2\pi\sigma^2\ . \quad (6.11)$$

Thus, the final formula is given by:

$$\text{GRIC}_m = \sum_i \rho_2\left(\frac{e_{i,m}^2}{\sigma^2}\right) + n\left((D - d_m)\log 2\pi\sigma^2 + 2\log\frac{c_m}{\gamma}\right) + k_m \log n \quad (6.12)$$

with $\rho_2(x) = \min\{x, T_m\}$.

Difference to Torr's formula

It should be noted that our formula differs from that given by Torr (Torr, 2002, Eq. 48), even though both are based on PDFs of the form of Equations (6.4) and (6.5). The difference is that in Torr's formula, it is assumed that the inliers are uniformly distributed in disparity across the entire search region, which is not the case for large active search regions. (Torr (2002) explicitly warns that "care should be taken to rederive [this quantity] according to the exact distribution [...] in different scenarios."). To explain, we first elaborate on a few of the quantities that are included in the calculation of the score (cf. Equations (6.11) and (6.12) and Table 6.2).

The GRIC score takes into account the volumes of the spaces in which certain measurements may occur. For example, assuming that each image has dimensions $L \times L$, then any individual 2D point measurement may occur on the volume $L \times L$. In an application that uses constrained search regions to establish correspondences (like ours), an arbitrary correspondence (i.e., pair of points) may occur on the volume $v = L \times L \times S \times S$, where S is the size of the search region. However, a correspondence that is an *inlier* to a (given) bijective 2D map such as a homography is distributed on the volume $c_H = L \times L$ (because the second point is uniquely fixed by the bijective map). If the model is a non-bijective relation such as an essential or fundamental matrix, which constrain a point to lie on a line but do not fix its position along the line, the volume becomes $c_E = L \times L \times R$, where R is the range of the disparity along which the feature match is expected to occur.

Δ is limited to be no greater than S, as otherwise the match will not be found and the correspondence will become an outlier. For simplicity, Torr sets $\Delta = S$, which causes several terms to cancel each other in the derivation of the final GRIC score. However, this effectively assumes that the inliers are uniformly distributed in disparity in the entire search region. For large S, this is not the case: The search region is large because we want tracking to be robust to fast camera movement (including camera rotations), but we expect the (2D) movement of all features to be strongly correlated; the range of disparity is still likely to be only a few pixels.

Thus, setting $\Delta = S$ causes the likelihood for E to decrease (because the observations do not match the model's assumption) and creates a significant bias towards selecting H. It should be noted that this should not be considered a principle flaw in Torr's formula: most any modeling process makes assumptions about the data in a trade-off of accuracy and model complexity; and here, one particular assumption does not hold well in our case.

Evidence that this bias is significant and can lead to incorrect model selection

Assume that a traveling camera observes a scene which contains a near-planar object on which many (but not all) correctly and accurately tracked feature correspondences lie. Those correspondences will be inliers to both E and H, with $e_i \approx 0$. Assume that there are further accurately tracked features on other surfaces (will be inliers to E, but outliers for H), and, optionally, several spurious correspondences.

With Torr's formula (Torr, 2002, Eq. 48), the GRIC score for either model m then is

$$\text{GRIC}_m \approx (n - n_{\text{in},m}) \cdot T_m + n d_m \log \tfrac{S^2}{2\pi\sigma^2} + k_m \log n + \text{const} \tag{6.13}$$

where $n_{\text{in},m}$ is the number of inliers for model m. We are interested in the threshold when the scores for the essential matrix E and the homography H are equal:

$$\begin{aligned} &(n - n_{\text{in},E}) \cdot T_E + n d_E \log \tfrac{S^2}{2\pi\sigma^2} + k_E \log n = \\ &(n - n_{\text{in},H}) \cdot T_H + n d_H \log \tfrac{S^2}{2\pi\sigma^2} + k_H \log n \end{aligned} \tag{6.14}$$

Assuming that $(k_E - k_H) \log n << (n - n_{\text{in},m}) \cdot T_m$:

$$\Rightarrow \quad \frac{n_{\text{in},E}}{n_{\text{in},H}} = \frac{2\,\log \frac{\gamma}{1-\gamma} + (D - d_H) \cdot \log \frac{S^2}{2\pi\sigma^2}}{2\,\log \frac{\gamma}{1-\gamma} + (D - d_E) \cdot \log \frac{S^2}{2\pi\sigma^2}} \tag{6.15}$$

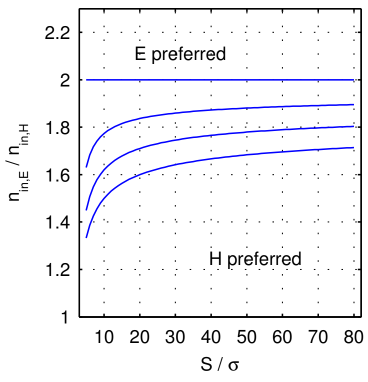
With $D - d_E = 1$ and $D - d_H = 2$ (cf. Table 6.2):

$$\Rightarrow \quad \frac{n_{\text{in},E}}{n_{\text{in},H}} = \frac{2\,\log \frac{\gamma}{1-\gamma} + 2\log\left(\frac{S^2}{2\pi\sigma^2}\right)}{2\,\log \frac{\gamma}{1-\gamma} + 1\log\left(\frac{S^2}{2\pi\sigma^2}\right)} \tag{6.16}$$

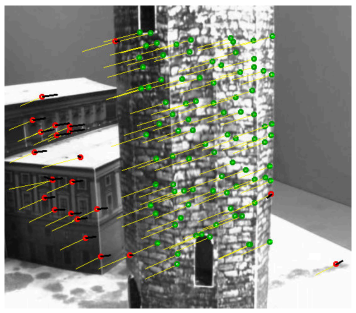
This ratio is plotted for a range of reasonable values for S, σ, γ in Figure 6.8(a). It can be seen that the range of $n_{\text{in},E}/n_{\text{in},H}$ for which the algorithm will prefer H (i.e., the area below the line) increases with the search region S, up to the point that the algorithm will, for very large S and γ close to 0.5, prefer H even if E produces almost twice as many inliers ($n_{\text{in},E}/n_{\text{in},H} = 2$).

Figure 6.8(b,c) illustrates a real-world case in which exactly this is the case: GRIC$_H$ $<$ GRIC$_E$ even though E models all correspondences correctly (including rejection of three spurious correspondences), while H discards a sizable set of correct correspondences as outliers. It could be argued that this result comes about because γ (the assumed fraction of inliers) is not chosen optimally for this particular frame — i.e., the model is "looking for" more outliers than there are — however, as we will discuss in Section 6.7.1, estimating γ accurately is challenging itself, and the actual inlier rate can vary greatly from one from to the next. Thus, it seems prudent to not rely on an accurate estimate of γ and address the non-matching assumption $\Delta = S$ independently.

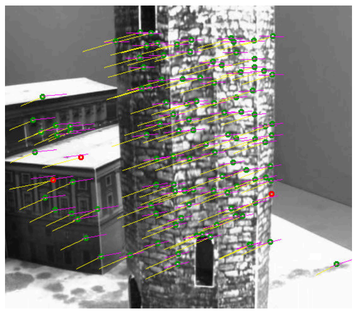
We thus re-derived the GRIC score for the general case of Δ independent of S. For completeness, we include a proof that our formula is equivalent to Torr's formula ((Torr, 2002, Eq. 48)) for the special case $\Delta = S$.



(a) preferred model w.r.t. inlier ratio



(b) tracking with H



(c) tracking with E

Figure 6.8: (a) Model selection (according to Torr's GRIC score) as a function of the inlier ratio $n_{\text{in},E}/n_{\text{in},H}$, expressed as a function of S/σ along the x-axis and $\gamma = \{0.5,\ 0.6,\ 0.7,\ 0.8\}$ (lines from top to bottom). Below the respective line, the GRIC score will prefer H, even if E produces more inliers. (b) Tracked feature correspondences in scene classified by H, (c) the same correspondences classified by E. Inliers are shown in green, outliers in red. Even though E models all correct correspondences correctly and H misclassifies 19 correct correspondences as outliers, GRIC$_H$ $= 1633.83 <$ GRIC$_E$ $= 1919.53$ and the algorithm would chose H over E. (Here: $n = 112$, $S = 40$, $\sigma = 1$, $\gamma = 0.5$.) The more general score derived in Section 6.5.1 decreases this problem. © IEEE 2012.

Proof that the generalized GRIC Score is equivalent to Torr's formula for $\Delta = S$

For the special case of $\Delta = S$, note that $U := (v/c_m)^{\frac{1}{D-d_m}} = S$ and $U' := c_m/U^{d_m} = L^2/S^2$ (i.e., both are *independent* of the model m) for all models considered here. Starting with Equation (6.11),

$$T_m = 2 \log\left(\frac{\gamma}{1-\gamma}\right) + 2 \log\left(U^{D-d_m}\right) - (D - d_m) \log 2\pi\sigma^2 \tag{6.17}$$

$$= 2 \log\left(\frac{\gamma}{1-\gamma}\right) + (D - d_m) \log\left(\frac{U^2}{2\pi\sigma^2}\right) \tag{6.18}$$

which, with $\lambda_1 := \log(U^2/(2\pi\sigma^2))$, is equivalent to Torr's definition of T (Torr, 2002, Eq. 18). Further, starting with Equation (6.12),

$$\text{GRIC}_m = \sum_i \rho_2\left(\frac{e_{i,m}^2}{\sigma^2}\right) + A_m + k_m \log n \tag{6.19}$$

$$\text{with } A_m = n\left((D - d_m) \log 2\pi\sigma^2 + 2 \log \frac{c_m}{\gamma}\right) \tag{6.20}$$

$$= n\left((D - d_m) \log 2\pi\sigma^2 + 2 \log \frac{c_m}{\gamma} + d_m \log U^2 - d_m \log U^2\right) \tag{6.21}$$

$$= n\left(\lambda_1 d_m + D \log 2\pi\sigma^2 + 2 \log \frac{c_m}{\gamma} - d_m \log U^2\right) \tag{6.22}$$

$$= \lambda_1 n d_m + \log\left(\frac{(2\pi\sigma^2)^D}{\gamma^2} \cdot \left(\frac{c_m}{U^{d_m}}\right)^2\right) \tag{6.23}$$

$$= \lambda_1 n d_m + \log\left(\frac{(2\pi\sigma^2)^D}{\gamma^2} \cdot U'^2\right) \tag{6.24}$$

$$= \lambda_1 n d_m + \text{const} \tag{6.25}$$

$$\Rightarrow \quad \text{GRIC}_m = \sum_i \rho_2\left(\frac{e_{i,m}^2}{\sigma^2}\right) + \lambda_1 n d_m + k_m \log n + \text{const}$$

Model m	k_m	$D - d_m$	c_m	v
E	5	1	$L \times L \times \Delta$	$L \times L \times S \times S$
H	8	2	$L \times L$	$L \times L \times S \times S$
$[R\|t]$	6	2	V	$V \times S \times S$
$[R\|0]$	3	2	V	$V \times S \times S$

Table 6.2: GRIC score parameters for the four models considered here. $L \times L$ is the area of one camera frame, $S \times S$ is the area of the constrained search region, Δ is the range of disparity. See Section 6.5.2 for discussion of V. © IEEE 2014.

which is equivalent to (Torr, 2002, Eq. 48). (Note that all terms that are not dependent on m are considered constant in this context, even if they change from frame to frame.)

6.5.2 GRIC score for absolute pose models

With appropriate model-specific parameters as given in Table 6.2, Equations (6.11) and (6.12) are applicable to $[R|0]$ and $[R|t]$ as well, since the PDFs on which they are based follow the shape of Equations (6.4) and (6.5). Arguably, this case is actually less complex than the two-view relation case, since the dimension of the error $D - d_m$ is the same for both models (see Table 6.2).

Two aspects are worth noting: First, in analogy to the two-view case, c_m and v are defined as the volumes of the spaces from which observations arise. This appears to necessitate to define the "volume" V of the 3D model, and to raise the question whether one assumes it to be part of the noisy estimation process, or (in comparison with the current measurement error) to be noise-free. However, a look at the GRIC score equations (Equations (6.11) and (6.12)) and Table 6.2 reveals that, in the comparison of $[R|t]$ vs. $[R|0]$, only the quotient c_m/v and difference $D - d_m$ are needed, and the values of V and D fall out as constant terms.

Second, in the case of E vs. H, one parameter to the GRIC score is the range of disparity Δ that we expect for E. Conveniently, this parameter gives some control over how the process handles motions that are very close to rotation-only, i.e., have a very small baseline and thus very little parallax: the bigger Δ, the more likely H is selected in those cases.

When comparing $[R|t]$ and $[R|0]$, this free parameter does not exist: all features are mapped to a unique image point. Further, especially for a good model and slow camera motion, the measurement noise σ becomes very small, and we observed that consequently, the GRIC score becomes extremely accurate in discerning even small shifts in baseline. Thus, $[R|0]$ is rarely selected (despite the smaller value of k_m, which gives it a slight advantage in case of close-to-equal error sums). Probabilistically, this makes perfect sense: the likelihood for $||t||$ exactly equal to 0 is virtually non-existent, and whenever the error distribution hints to even the slightest shift, $[R|t]$ should be selected.

Practically, however, motions with very small t — imagine a camera mounted on a tripod with the optical center and the rotation axis in slight misalignment — are arguably better modeled as a panorama: For a panorama, a small shift of the optical center causes minor distortion artifacts, but no fatal harm to tracking and mapping. (DiVerdi et al. (2009) analyze this type of error in detail.) For structure-from-motion, it results in features with unreliable depth estimates from extremely ill-conditioned triangulations, and, as soon as only those features are visible, likely tracking failure.

Therefore, we adapt the model that is estimated as follows: instead of estimating $[R|0]$, we estimate $[R|t \leq t_{\text{max}}]$ (more precisely, $[R|t]_{\text{prev}} \circ [R|t] \mid ||t|| \leq t_{\text{max}}$). Practically, we implemented this by estimating $[R|t]$, but constraining $||t||$ to $[0; t_{\text{max}}]$ after each gradient descent iteration. t_{max} can be understood as a soft threshold regulating how much baseline shift the system shall observe before preferring $[R|t]$ over $[R|0]$, analogous to Δ for E vs. H. By dividing by the distance to the environment (which is available, since we are in the model-based condition), t_{max} can be transformed into a scale-independent feature observation angle α_{max}, which we use as input parameter to the system.

6.6 Evaluation

We implemented our system prototype in C++, making use of the OpenCV[4], TooN[5] and libCVD[6] libraries. Our system runs in real time (at about 20-25 ms per frame) on a commodity PC without specific optimizations. Typical timings are presented in Figure 6.9. They are, of course, strongly dependent on the hardware and parameter configuration (e.g., number of keypoints per frame) and presented here only as a coarse reference point.

Currently, our implementation is not optimized to run in real time on a mobile device. However, the most expensive parts (in particular, the tracking with NCC-based matching, cf. Figure 6.9) are computationally similar to T&M systems that were shown to operate in real

[4] `http://opencv.willowgarage.com/`
[5] `http://www.edwardrosten.com/cvd/toon.html`
[6] `http://www.edwardrosten.com/cvd/`

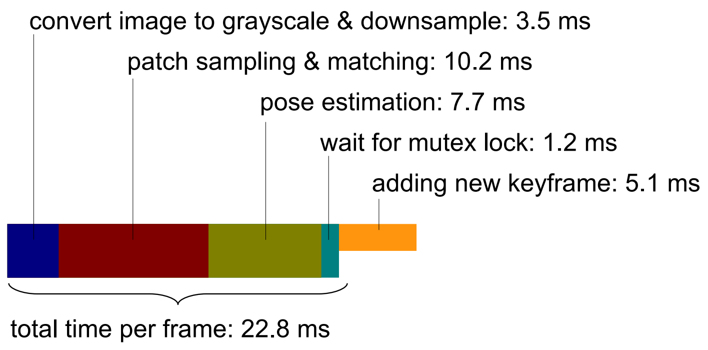


Figure 6.9: Breakdown of timings in tracking thread. These times were taken on a commodity PC (Intel i7 Core, 4 GB RAM, Ubuntu 12.04), without specific optimizations or the use of a GPU. All times are averaged per frame over the entire sequence. Since creating a new keyframe is not executed every frame, its contribution to the average time is minimal. © IEEE 2014.

time on such devices several years ago (Klein and Murray, 2009; Wagner et al., 2010). Thus, we argue that with appropriate algorithmic and device-specific optimizations, running an implementation of our concept on a mobile device is feasible.

We tested our system on a variety of video sequences, including rotation-only sequences from a panorama dataset by Coffin et al.[7] (Coffin et al., 2011), sequences from the "City of Sights" repository[8] (Gruber et al., 2010a), as well as further self-recorded videos, using models from the "City of Sights" as backdrop. Results from those videos are presented in Figures 6.1, 6.5, 6.10, 6.11, and 6.12.

[7]`http://tracking.mat.ucsb.edu`
[8]`http://cityofsights.icg.tugraz.at`

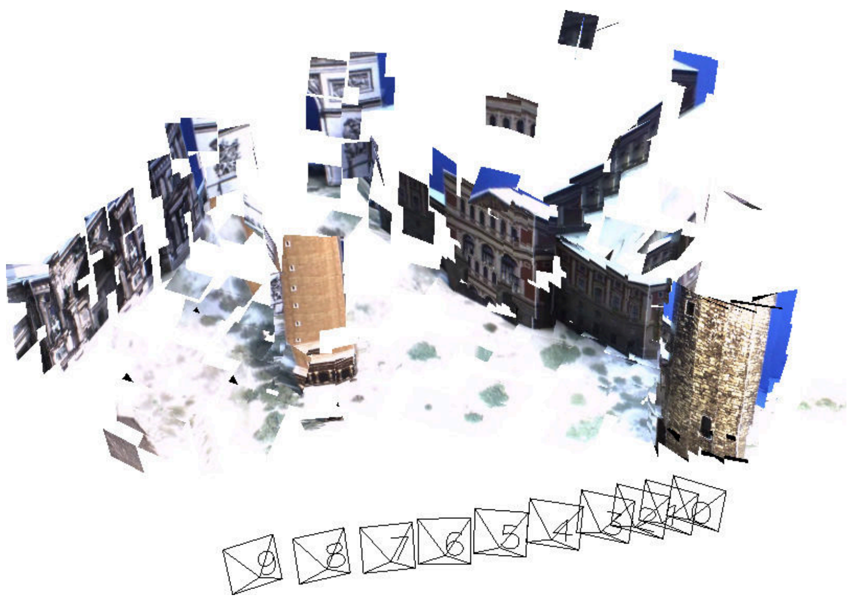


Figure 6.10: Our system acting as a SLAM system, reconstructing the scene in 3D and recovering the camera trajectory fully automatically. Each reconstructed 3D feature is visualized as a small image patch (sampled from the frame in which it was first observed), oriented according to its estimated normal vector. © IEEE 2014.

6.6.1 Tracking accuracy

To evaluate the benefits of integrating model-based tracking, we ran our system with and without model-based tracking (i.e., for the latter case, the left branch in Figure 6.4 is disabled) on 800 frames of video from the "City of Sights" repository for which accurate ground truth of the camera trajectory is provided. The 3D model that our system generates from this video sequence (*with* model-based tracking) is shown in Figure 6.10.

Since the reconstructions including camera locations are arbitrary up to a similarity transform, we first aligned the point cloud of the camera locations for both conditions to the ground truth by aligning the first camera position, and choosing scale and rotation such that the overall

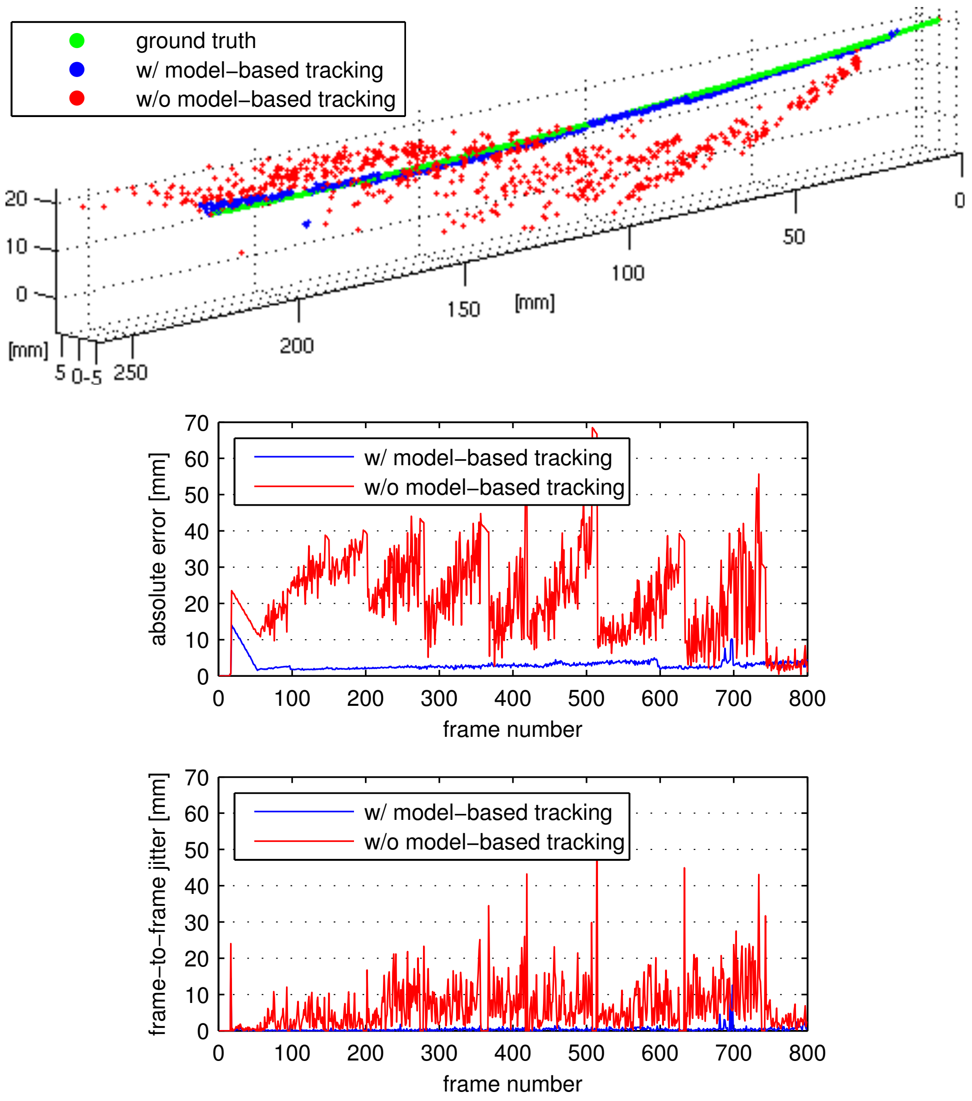


Figure 6.11: Tracking accuracy with and without model-based tracking. The point clouds are aligned to the ground truth as described in the text. From top to bottom: camera locations in 3D, absolute error per frame, jitter (i.e., camera displacement from one frame to the next) per frame. © IEEE 2014.

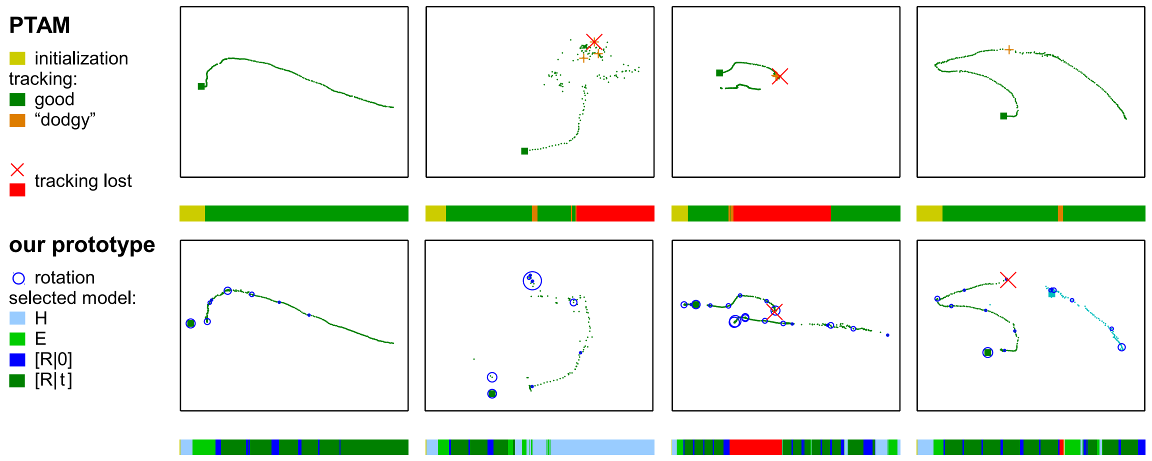


Figure 6.12: Qualitative comparison of camera trajectories for PTAM (top) and our prototype (bottom) on selected videos, highlighting a few particular characteristics. Each video is 300-500 frames long, the camera travels a few feet along a scene similar to the one in Figure 6.10. The trajectories from both systems were aligned to each other as described in Section 6.6.1. The color bar underneath each panel shows per-frame tracking/model selection information over time. A blue circle indicates that our prototype selected a rotation-only model (note that this does not have a permanent effect (i.e., there is no effect on the map) unless a keyframe is added at that point). The circle's radius is logarithmically proportional to the length of the rotation-only sequence. © IEEE 2014.

error was minimal (in a least-squares sense). The results are presented in Figure 6.11, showing that with model-based tracking, tracking is highly accurate and very smooth.

6.6.2 Qualitative comparison with PTAM

In Figure 6.12, we present a qualitative comparison of our prototype with PTAM[9] (Klein and Murray, 2007, 2008) on four representative videos:

[9]`http://www.robots.ox.ac.uk/~gk/PTAM/`

For slow, smooth parallax-incuding camera movement (Figure 6.12 far left), both systems produce very similar trajectories. On a parallax-inducing movement followed by extended rotation-only movement (Figure 6.12 second from left), PTAM loses track and unsuccessfully attempts to recover for the remainder of the sequence, while our prototype keeps track throughout, attaching a panorama to the triangulated model. In Figure 6.12 second from right, both systems lose track due to occlusion. PTAM successfully recovers as the camera moves back to the known part of the scene, but misses out on a long stretch, while our prototype starts a new track and successfully connects the two; thus, its final map covers a larger part of the environment. Finally, on a parallax-inducing movement with relatively quick movement in the middle (Figure 6.12 far right), PTAM is able to maintain tracking, while our prototype loses track. It automatically starts a new track (shown in light blue, *aligned separately to PTAM's track*), but is unable to connect the two tracks.

Thus, the results demonstrate the conceptual advantages of supporting rotation-only movement and linking of tracks instead of recovery, but they also underscore that our current implementation is to be regarded a proof-of-concept prototype, and certain aspects are left unoptimized. In particular, tracking during parallax-inducing motion is not as robust as PTAM.

6.7 Discussion: Aspects for Further Investigation, Applications & Limitations

6.7.1 On estimating the probability density function

Optimally estimating both a motion model and the parameters of the probability density function (PDF) of the measurement error (e.g., standard deviation σ of the measurement error and inlier ratio γ) is a chicken-and-egg problem: either result requires the other. In many T&M systems, knowing the PDF exactly is not crucial: with appropriate cost functions, pose estimation algorithms (both via sample consensus and M-estimators) are reasonably robust to noise in σ and γ, and "bad" values may merely be an indicator for inevitable tracking failure. Model selection however gets more accurate the better the estimate of the PDF is.

If the observed error distribution matches the expected shape (i.e., Equations (6.4) and (6.5)), the described process via expectation-maximization (Section 6.3.2) works well and appears to produce accurate estimates of its parameters. If however the observed distribution does not match the expected shape, for example due to questionable tracking, expectation-maximization appears to intensify the problem by producing unusable values. This effect can be limited by ad-hoc measures such as filtering over several frames and clamping of the values, but we suggest that further investigation is warranted to find an optimal and principled solution balancing accurate estimates and leniency for tracking.

6.7.2 On improving coarse-to-fine matching

The standard approach to coarse-to-fine matching (approximately as employed by Klein and Murray (2007); Wagner et al. (2010, 2009)) is to 1. locate features in a coarse (downsampled) image, 2. estimate the camera pose, 3. reproject features into the higher resolution image using the new pose estimate, 4. refine their position (in a tightly constrained area), and lastly 5. refine the camera pose using the refined features.

Unfortunately, this approach cannot be applied directly here, because we do not know which type of camera motion to expect (and thus which model to enforce) until after the tracking step. As described in Section 6.3.1, thus far, we have simply omitted the intermediate pose estimation (step 2). This however results in a higher risk of losing individual features, as they must be found independently in an effectively larger search region.

There are several alternative approaches in which this could be solved, for example, executing steps 2 to 5 for each motion model under consideration, or selecting the model after step 2. We have designed another, more sophisticated approach, which retains the advantages of the interleave pose estimation at little additional cost: We propose to estimate a *superset model* which admits all features that are inliers to *either* motion model under consideration (thus, the tracking step remains agnostic to the type of model that is later selected), yet constrains the features such that many outliers can be identified. For model-based tracking, $[R|t]$ is this superset model, as it trivially encompasses $[R|0]$.

For the two-view relations H and E, the existence of such a superset model is less obvious. However, a closer look reveals that indeed, E fulfills these criteria: while the epipolar geometry is, strictly speaking, not defined in the case of rotation-only movement, an algorithm used to compute E will return *some* result $\breve{E}$, and true inliers for *either* H or E will fulfill $x_2^T \cdot \breve{E} \cdot x_1 = 0$ (cf. in particular Hartley and Zisserman, 2004, Section 11.9.3). Thus, while $\breve{E}$ may lack a geometric interpretation (in the case of rotation-only movement), it does provide a necessary condition for any correct correspondence. Assuming that outliers are statistically independent, the chances for a random outlier to satisfy $x_2^T \cdot \breve{E} \cdot x_1 \approx 0$ are small.

We have implemented and verified this approach on several videos (including rotation-only sequences, which, for $\breve{E}$, is the most unfavorable case). However, we have not integrated it completely with the rest of our system, so the evaluation of potential robustness and performance improvements is left for future work. One particular challenge is to robustly estimate the PDF parameters (cf. Section 6.7.1) in the now effectively much smaller active search regions.

6.7.3 On tracking robustness

We emphasize that our current implementation is to be regarded a proof-of-concept implementation. As demonstrated in Section 6.6.2, tracking robustness during general motion is not yet comparable to systems such as PTAM (Klein and Murray, 2007) or DTAM (Newcombe et al., 2011a).

Techniques that were proven to increase robustness or scalability — such as estimation of in-plane rotation before the pose estimation (Klein and Murray, 2008), more advanced map feature management including filtering of outliers, proactive search for new features in existing frames (Klein and Murray, 2007), and more sophisticated bundle adjustment strategies — will need to be integrated as appropriate.

6.7.4 On merging of maps

As discussed in Section 6.4.3, the algorithm we use to detect overlap between disjoint tracks and thus initiate their merging is basically the same as what is commonly employed for relocalization, which implies that similar detection performance can be expected. An interesting area for future research is to exploit the fact that we actually have more data than what is available with the relocalization strategy: instead of one individual new frame, we have a new map consisting of multiple frames; thus, overlap detection could potentially yield higher detection performance than relocalization. This could be exploited even within the SBI-based recovery framework: For example, Kim et al. (2009) have explored the use of "virtual keyframes," i.e., renderings of the model from strategically distributed viewpoints, rather than actual keyframes. As we now seek to connect two models, this strategy could be applied on both sides.

Similarly, the overlap between the tracks may encompass an area larger than a single frame. While we currently use only features from one pair of keyframes to merge the maps, the registration could be improved by explicitly calculating and then using the entire extent of the overlap. Note however that a much larger overlap is unlikely, since otherwise it would likely have been detected earlier.

6.7.5 On applications & limitations of the hybrid map

It is clear that hybrid maps consisting of both SfM and panorama data (such as Figure 6.1 or the map produced from Figure 6.12 second from left) do not possess all properties that one would look for in an ideal environment map. Most obviously, the panorama part lacks depth information, but furthermore, the individual SfM reconstructions do not share a common scale, since the rotation-only movement cannot propagate scale. Thus, this type of data may not suit the needs of all applications. We emphasize that this is not a limitation of our design, but a limitation inherent to the input data; the alternative is not to create a fully three-dimensional model, but to lose tracking altogether. If a fully three-dimensional model of the environment is required, one has to either employ additional and/or active sensors, or put constraints on the camera movement. However, even if one aims for a fully three-dimensional model, it is arguably better to collect the panoramas than to discard them. For example, Pan et al. (2011) describe how to first collect panoramas, and then treat them as single images with wide field of view for 3D reconstruction afterwards.

We note that the virtual navigation techniques discussed in Sections 4.2.5 and 4.3.4 are perfectly suited for this data: Each camera position tied to the model provides a keyframe for rendering and navigation, the structural components can be exploited during the model building process, and (partial) panoramas extend the range in which the user can pan.

6.7.6 On model selection and scene segmentation

While we currently select only one model for the entire frame — which is, theoretically, the correct thing to do, since the motion refers to the camera and thus to the entire frame — there may be cases especially in outdoor scenes in which the foreground exhibits enough parallax to be modeled in 3D, while the background exhibits little parallax and might benefit from the stable, dense mapping that homographies offer. This leads to an interesting problem in which image segmentation and environment modeling interact.

Chapter 7

Conclusions

This chapter concludes the dissertation. We briefly summarize the presented work, review the contributions of this dissertation, discuss limitations of the described systems, and finally offer an outlook on opportunities for future work.

7.1 Summary

In this dissertation, we described a framework to integrate the physical environment into video-mediated remote communication using unobtrusive, mobile AR (Chapter 3). Our framework does not require preparation of the environment or specialized equipment, and is compatible with a wide range of hardware, including systems that are already ubiquitous (e.g., smartphones) as well as more complex immersive systems. We further described the implementation of three prototypes with increasing complexity and flexibility (both in terms of technical capabilities as well as user interfaces) (Chapter 4), as well as several user studies to validate their usability, evaluate individual interface components, and establish their benefits over more conventional interfaces (Chapter 5). Lastly, we addressed one particular technical limitation of current visual environment modeling which becomes apparent in this context in particular, and described a conceptual solution and proof-of-concept implementation for it. Namely, we described a system that is able to track and map through motion sequences that neither conventional six-degree-of-freedom SLAM systems nor panoramic mapping systems can process (Chapter 6), which we believe is an important conceptual step towards the vision of fully transparent tracking and mapping for “Anywhere Augmentation” (Höllerer et al., 2007).

Even though our prototypes rely on noticeably imperfect computer vision-based tracking (while being compared to more conventional interfaces which were failsafe), and users received a very limited amount of training time with the new technology, they were shown to be overwhelmingly preferred by users (by 79% and 80%, respectively) in two extensive comparative user studies, and, in both studies, the task performance was significantly better compared with a video-only interface.

However, in both studies, no task performance difference was found in comparison with a (technically much simpler) “static marker” interface. Possible reasons have been discussed in detail in Section 5.3; they mainly fall into three categories: (1) limitations of our respective prototype (e.g., imperfect tracking), (2) nature of the chosen task, and (3) artifacts of the user study (e.g., training effect, simulated expert knowledge, and proxy tasks), that is, effects that would not be present in an actual application. Certain aspects in the first category have already been remediated with the design of the third prototype.

7.2 Contributions

The vision for this work is to enable unobtrusive, mobile remote collaboration that allows the remote user to intuitively and directly interact with the remote environment, and thus collaborate on physical tasks. This vision itself is not new — we have pointed out related work that shares it in Chapter 2 — however, we believe that this dissertation has brought it significantly closer to realization.

We provided contributions on multiple levels: Conceptually, we provided a framework to formalize our approach which is explicit enough to be used as a guide for implementation, yet flexible enough to to compatible with a wide array of (hardware and software) configurations. Practically, many individual features of the presented systems are novel and — despite the concurrent publication of closely related work (such as Sodhi et al., 2013; Jo and Hwang, 2013) which share certain aspects — unique. Perhaps most notably among them is our camera control, which supports both a coupled and a decoupled camera, including seamless transitions between the two (Section 4.2.5), as it is an important factor in a bigger picture:

First, while the system is complex under the hood, one important advantage of our approach is that it seamlessly extends the ubiquitous videoconferencing paradigm: it adds to it, but it does not take anything away. If desired, the user can fall back to simply watching the live video feed. In combination with the compatibility with low-cost, existing hardware, this significantly lowers the barrier for introduction and user acceptance: the technology can be added to existing solutions without disrupting existing workflows or habits. It may prove invaluable in some situations, and it does not hamper other situations in which it might not be needed.

Second, our work bridges and creates a synergy between video conferencing (Skype, Apple FaceTime, Google Hangouts, etc.) and remote world exploration (Microsoft Photosynth, Quicktime VR (Chen, 1995), Google StreetView, Uyttendaele et al. (2004), etc.) in a unique coherent interface. This synergy is a direct result of our approach and certain design decisions, particularly our camera control that allows both a coupled and a decoupled camera. We believe that this synergy bears significant potential which our systems have only started to tap into. (We will outline a particular aspect for future work that this synergy entails in Section 7.4.3.)

Our systems are designed to put the user's needs first: they should support the user's communication as good as possible, rather than forcing them to communicate in a certain way. This is important in particular when working with technologies that may fail, as is the case here for visual tracking and modeling; effects of failure need to be mitigated with the user's needs in mind. This paradigm manifests itself in individual features which might seem minor from a technical point of view, but were found to be important through extensive practical testing. Particular examples are that (1) in Prototype 1, tracking automatically resets, depending on the situation, rather than bothering the user to recover it (Section 4.1.3), (2) in Prototype 2, the user may set (static) annotations even if tracking is lost (Section 4.2.6), and (3) in Prototype 3, the video freezes automatically when drawing an annotation (Section 4.3.2).

Further individual contributions include establishing the need to visualize direction of drawings (Section 5.4.2) as well as the discussion on the interpretation of 2D annotations in 3D and study-based design recommendations regarding it (Sections 4.3.3 and 5.4.2). The concept proposed in Chapter 6 is the first approach to keyframe-based tracking and mapping that can cope with rotation-only as well as parallax-inducing motion.

Our user studies demonstrate the level of maturity and usability of our systems, and more importantly, have demonstrated that the proposed concept is promising and has the potential to greatly improve video-mediated communication and broaden its applicability. We have demonstrated that achieving benefits requires neither futuristic, specialized, or fully immersive hardware interfaces nor special training, but is achievable with low-cost hardware and little training, making it suitable for widespread adoption. (This does not imply, however, that more immersive systems could not offer additional benefits.) We have also discussed artifacts that posed challenges in conducting the user studies in detail, and hope that researchers interested in conducting futures studies in this area will find our report helpful to inform their study designs.

7.3 Limitations of the Current Implementation

In the iterations from the initial proof-of-concept implementation (Prototype 1) to the last instantiation (Prototype 3), our systems have become increasingly more flexible and technical limitations have been removed. However, a few assumptions and technical limitations remain, which we discuss here.

7.3.1 Level of detail of the model

Building upon a tetrahedralization of a sparse point cloud, the modeling approach taken in Prototypes 2 and 3 is very fast (as demonstrated in Section 4.2.4), but results in a model that does not exhibit the level of detail that is achievable with, for example, dense voxel-based volumetric fusion (Curless and Levoy, 1996; Newcombe et al., 2011b; Pradeep et al., 2013). While the existence of a 3D model is quintessential for the anchoring of annotations and rendering, a relatively coarse level of detail is, arguably, acceptable, as the keyframe-based navigation and rendering de-emphasize modeling artifacts.

However, techniques to create a more detailed model exist. Conceptually as well as implementation-wise, our approach is modular: none of the other components depend on the particular modeling algorithm. Thus, other algorithms could be plugged in as needed. This includes algorithms to densify the point cloud (Furukawa and Ponce, 2010) while keeping the overall approach the same; using other, computationally more intensive vision-based algorithms (Pradeep et al., 2013); or (where permissible by the application) active depth sensor-based modeling (Newcombe et al., 2011b).

7.3.2 Static scene

Somewhat more fundamentally, the system assumes that the scene is largely static. This assumption can be found in several of the components and needs to be remediated in different ways. The first component is the SLAM system on the local user's side. However, we found it to be reasonable robust to occlusions and partial and/or gradual changes in the scene. Further, there are SLAM systems that have been specifically designed to handle (semi-)dynamic scenes, most recently for example Tan et al. (2013).

Other components that are affected by this assumption — and where a solution is arguably less readily available — include the modeler, the keyframe-based navigation and keyframe-based rendering. Here, a solution will likely require an active detection and invalidation of changed regions, likely fed in by the tracker. Alternatively or in conjunction to the technical solution, the user interface can be designed to account for a potentially changing environment. This could entail visualizing how recently a specific part of the environment has been observed (e.g., by slowly decreasing the saturation of the imagery) and thus communicating to the user that certain observations may be outdated.

7.3.3 Stereo initialization

Thus far, like almost all monocular SLAM systems, our prototype requires a stereo initialization — i.e., a distinct camera movement — before it tracks robustly. This is done quickly and thus not a problem if the user is aware of the requirement; however, we feel that it is an obstacle for truly transparent and user-friendly operation. A more flexible approach with automatic model estimation and selection towards unconstrained tracking and mapping was presented in Chapter 6. However, this solution has not reached the robustness and maturity needed for transparent operation. The research community is also investigating ways to reduce the burden of initialization in other ways (Mulloni et al., 2013).

7.3.4 Occlusion of annotations on local side

Currently, the 3D model is available only on the remote user's side. Thus, virtual annotations can be correctly occluded by the physical scene by the remote user's renderer, but not by the local user's renderer. While other depth cues (most notably, parallax) still indicate the annotation's location, it would be desirable to allow for occlusions. The remote system could send either the entire model geometry or alternatively some sort of local visibility information per annotation back to the local device. Designing an elegant, bandwidth-efficient solution for either approach is an interesting aspect for future work.

7.4 Opportunities for Future Research

Opportunities for future research — within the proposed concept and beyond — are plentiful. Future work may address the limitations described in the previous section. Specific aspects in the context of our work on model estimation and selection (Chapter 6) that warrant further investigation have been discussed in Section 6.7. We outline further opportunities prompted by our work in this section.

7.4.1 Other types of AR displays

Even Prototype 3 as the culmination of our series of implementations does not exhaust the potential of the proposed framework, which leaves ample opportunity for future work: Guided by Figure 3.1, several other components may be substituted with other implementations and evaluated.

One particular example is the AR display for the local user. In all of the systems described here, we used a "magic lens" tablet as the local user's AR display. However, the proposed concept is compatible with other display types, and it would be interesting to evaluate other choices such as pico-projectors or head-worn displays. Both have been explored in the literature for remote collaboration (e.g., Gurevich et al. (2012) and Huang and Alem (2013), respectively), but not in combination with the other features of our approach (in particular, world-stabilized annotations and virtual scene navigation), which may alter potential benefits significantly.

As briefly alluded to in Section 5.3.2, we speculate that the benefits of decoupled views and world-stabilized annotations are, in fact, more apparent when using a head-worn or projective display: The hand-held device functions as a separate object that creates a level of indirection in the AR experience — the annotations appear on the screen rather than "in the real world" — and that the local user can consciously move according to the remote user's instructions (most notably, holding it still if required). In contrast, a head-worn display or (head or body-worn) pico-projector is subjected to the user's natural head and body movement, making the need for tracked annotations and "view stabilization" for the remote user quite apparent. (Bauer et al. (1999), who used a head-worn display and non-tracked annotations, noted that “initial observations indicated that movements of the head-mounted camera made it very difficult to use the telepointer without image freezing.”)

Examples of other components to be investigated include the integration of cues such as gestures (Sodhi et al., 2013) or gaze.

7.4.2 Further work on live navigation of remote environments

On the topic of live navigation of remote environments, we have presented two interfaces for virtual navigation in this dissertation (Sections 4.2.5 and 4.3.4, respectively). In parallel, Sodhi et al. (2013) and Jo and Hwang (2013) have presented prototypes using physical navigation.

In Sections 4.2.5 and 4.3.1, we explained our reasoning for opting for virtual navigation rather than physical navigation for the prototypes presented here. However, these interfaces should be seen as a starting point; this area is very new and warrants further investigation under consideration of the specific requirements of live collaboration. This includes “smart” camera control based on semantic (scene or task) knowledge, comparative evaluation of different methods of virtual navigation as well as in comparison to physical navigation.

7.4.3 Integration of large-scale maps and further digital data

As described in the introduction, the main motivation of this work was to better integrate the physical world into remote collaboration, as we felt that this was an element crucially lacking in current solutions. (This gave rise to the title of our first relevant publication (Gauglitz et al., 2012a).) As such, *digital* data aside from the live video (and the model built from it) did not play a big role; and while AR is the underlying paradigm to enable our approach, the actual augmentations we used are rather minimal.

Looking forward, AR also offers the opportunity to embed rich digital data in this context. We mention two examples: First, the remote user may have access to additional data, for example a manual of the object in need of repair in the remote assistance scenario. Current video conferencing solutions oftentimes provide separate channels to exchange digital data, such as text snippets or files, but these remain wholly unconnected to the scene. Future AR-based systems should allow the remote user to take digital data such as this — for example, a specific diagram or description from the manual — and not only share it with the local user, but do so by placing it at the relevant place in the scene.

Second, the described synergy between video conferencing and remote world exploration suggests the integration of another type of data: When the physical location of the local user's environment is or becomes known — this may happen automatically via sensors such as GPS, via image-based localization, or be provided by one of the users — the system could automatically extend the (live, but small-scale) model from the local user with existing, large-scale maps or models of the environment, in a way that allows the remote user to seamlessly navigate both together, and, for example, provide directions by selecting targets which the local user has not even observed yet.

7.4.4 Extension to more than two users and other roles

The conceptual description, as well as the user study tasks, in this dissertation focused on a scenario with two users in clearly defined roles: a local user requesting assistance from a remote expert. However, much of the work, including the framework and the presented systems, are applicable (or can be extended) to a much wider range of scenarios regarding both roles and numbers of users.

With regards to roles of the users, we focused on a local worker-remote expert scenario here for ease of description; however, the system is agnostic to the actual task at hand or distribution of (domain) knowledge. For example, an alternative scenario is that a realtor or event planner wants to show a property/venue to a remote customer: here, the local user is the 'expert.' Following our paradigm, the customer can browse the space, ask questions and refer to specific features in the environment while doing so.

With regards to number of users, we note that the presented framework naturally extends to more than one local user as well as more than one remote user. In the case of multiple local users, all of their sensors can contribute data to a central model building process, and each local user has their own view of the shared space including shared annotations. Likewise, each remote user has their own view of the shared space and can create annotations. While our framework naturally scales to these scenarios, new interface questions arise which will have to be investigated, including visualization of the other collaborator's visual attention focus (if desired), ownership of shared annotations, and/or the need for private (not-shared) annotations.

Bibliography

Adcock, M., Anderson, S., and Thomas, B. RemoteFusion: Real time depth camera fusion for remote collaboration on physical tasks. In *Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry (VRCAI)*, pages 235–242, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2590-5. doi: 10.1145/2534329.2534331.

Akaike, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

Alem, L., Tecchia, F., and Huang, W. HandsOnVideo: Towards a gesture based mobile AR system for remote collaboration. In Alem, L. and Huang, W., editors, *Recent Trends of Mobile Collaborative Augmented Reality Systems*, pages 135–148. Springer, New York, 2011. ISBN 978-1-4419-9845-3.

Baker, S. and Matthews, I. Lucas-Kanade 20 years on: A unifying framework, part 1. Technical Report CMU-RI-TR-02-16, Robotics Institute, Carnegie Mellon University, Pittsburgh, USA, July 2002.

Baudisch, P. and Rosenholtz, R. Halo: a technique for visualizing off-screen objects. In *Proceedings of the ACM SIGCHI Annual Conference on Human Factors in Computing Systems (CHI)*, pages 481–488. ACM, 2003.

Bauer, M., Kortuem, G., and Segall, Z. "Where are you pointing at?" A study of remote collaboration in a wearable videoconference system. In *Proceedings of the 3rd International Symposium on Wearable Computers (ISWC)*, pages 151–158, 1999. doi: 10.1109/ISWC.1999.806696.

Billinghurst, M., Bowskill, J., Jessop, M., and Morphett, J. A wearable spatial conferencing space. In *Proceedings of the 2nd International Symposium on Wearable Computers (ISWC)*, pages 76–83, October 1998a. doi: 10.1109/ISWC.1998.729532.

Billinghurst, M., Weghorst, S., and Furness, T. Shared space: An augmented reality approach for computer supported collaborative work. *Virtual Reality*, 3:25–36, 1998b. ISSN 1359-4338. 10.1007/BF01409795.

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

Butz, A., Höllerer, T., Feiner, S., MacIntyre, B., and Beshers, C. Enveloping users and computers in a collaborative 3D augmented reality. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, pages 35–44, Washington, DC, USA, 1999. ISBN 0-7695-0359-4.

Cabral, B. and Leedom, L. C. Imaging vector fields using line integral convolution. In *Proceedings of the 20th ACM SIGGRAPH Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 263–270, New York, NY, USA, 1993. ACM. ISBN 0-89791-601-8. doi: 10.1145/166117.166151.

Chastine, J., Nagel, K., Zhu, Y., and Hudachek-Buswell, M. Studies on the effectiveness of virtual pointers in collaborative augmented reality. In *Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI)*, pages 117–124, March 2008. doi: 10.1109/3DUI.2008.4476601.

Chaurasia, G., Duchene, S., Sorkine-Hornung, O., and Drettakis, G. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics*, 32(3):30:1–30:12, July 2013. ISSN 0730-0301. doi: 10.1145/2487228.2487238.

Chen, S. E. QuickTime VR: An image-based approach to virtual environment navigation. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 29–38, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. doi: 10.1145/218380.218395.

Chen, S., Chen, M., Kunz, A., Yantaç, A. E., Bergmark, M., Sundin, A., and Fjeld, M. SEMarbeta: Mobile sketch-gesture-video remote support for car drivers. In *Proceedings of the 4th Augmented Human International Conference*, pages 69–76. ACM, 2013.

Christie, M. and Olivier, P. Camera control in computer graphics: Models, techniques and applications. In *ACM SIGGRAPH ASIA 2009 Courses*, SIGGRAPH ASIA '09, pages 3:1–3:197, New York, NY, USA, 2009. ACM. doi: 10.1145/1665817.1665820.

Civera, J., Davison, A., and Montiel, J. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, 2008a.

Civera, J., Davison, A., and Montiel, J. Interacting multiple model monocular SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3704–3709, 2008b.

Coffin, C., Ventura, J., and Höllerer, T. A repository for the evaluation of image-based orientation tracking solutions. In *Proceedings of the 2nd International Workshop on AR/MR*

Registration, Tracking and Benchmarking (TrakMark 2011), held in conjunction with the IEEE International Symposium on Mixed and Augmented Reality (ISMAR) 2011, 2011.

Curless, B. and Levoy, M. A volumetric method for building complex models from range images. In *Proceedings of the ACM SIGGRAPH Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 303–312, 1996. ISBN 0-89791-746-4. doi: 10.1145/237170.237269.

Davison, A. J., Mayol, W. W., and Murray, D. W. Real-time localisation and mapping with wearable active vision. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 18–27, Tokyo, Japan, October 2003.

Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6): 1052–1067, 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1049.

DiVerdi, S., Wither, J., and Höllerer, T. All around the map: Online spherical panorama construction. *Computers & Graphics*, 33(1):73–84, 2009.

Eade, E. and Drummond, T. Unified loop closing and recovery for real time monocular SLAM. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2008.

Fischler, M. A. and Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24 (6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692.

Furukawa, Y. and Ponce, J. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010. ISSN 0162-8828. doi: 10.1109/TPAMI.2009.161.

Fussell, S. R., Kraut, R. E., and Siegel, J. Coordination of communication: Effects of shared visual context on collaborative work. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 21–30, New York, USA, 2000. ACM. ISBN 1-58113-222-0. doi: 10.1145/358916.358947.

Fussell, S. R., Setlock, L. D., and Kraut, R. E. Effects of head-mounted and scene-oriented video systems on remote collaboration on physical tasks. In *Proceedings of the ACM SIGCHI conference on Human factors in computing systems (CHI)*, pages 513–520, New York, USA, 2003a. ACM. ISBN 1-58113-630-7. doi: 10.1145/642611.642701.

Fussell, S. R., Setlock, L. D., Parker, E. M., and Yang, J. Assessing the value of a cursor pointing device for remote collaboration on physical tasks. In *Extended Abstracts, ACM SIGCHI Annual Conference on Human Factors in Computing Systems (CHI)*, pages 788–789, New York, USA, 2003b. ACM. ISBN 1-58113-637-4. doi: 10.1145/765891.765992.

Fussell, S. R., Setlock, L. D., Yang, J., Ou, J., Mauer, E., and Kramer, A. D. I. Gestures over video streams to support remote collaboration on physical tasks. *Human-Computer Interaction*, 19:273–309, September 2004. ISSN 0737-0024. doi: 10.1207/s15327051hci1903_3.

Gauglitz, S., Foschini, L., Turk, M., and Höllerer, T. Efficiently selecting spatially distributed keypoints for visual tracking. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Brussels, Belgium, September 2011. doi: 10.1109/ICIP.2011.6115832.

Gauglitz, S., Lee, C., Turk, M., and Höllerer, T. Integrating the physical environment into mobile remote collaboration. In *Proceedings of the ACM SIGCHI International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI)*, San Francisco, USA, September 2012a. doi: 10.1145/2371574.2371610.

Gauglitz, S., Sweeney, C., Ventura, J., Turk, M., and Höllerer, T. Live tracking and mapping from both general and rotation-only camera motion. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Atlanta, USA, November 2012b. doi: 10.1109/ISMAR.2012.6402532.

Gauglitz, S., Nuernberger, B., Turk, M., and Höllerer, T. World-stabilized annotations and virtual scene navigation for remote collaboration. In *Proceedings of the 27th ACM Symposium on User Interfaces Software and Technology (UIST)*, Honolulu, Hawaii, USA, October 2014a. doi: 10.1145/2642918.2647372. To appear.

Gauglitz, S., Nuernberger, B., Turk, M., and Höllerer, T. In touch with the remote world: Remote collaboration with augmented reality drawings and virtual navigation. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology (VRST)*, Edinburgh, UK, November 2014b. To appear.

Gauglitz, S., Sweeney, C., Ventura, J., Turk, M., and Höllerer, T. Model estimation and selection towards unconstrained real-time tracking and mapping. *IEEE Transactions on Visualization and Computer Graphics*, 20(6):825–838, June 2014c. ISSN 1077-2626. doi: 10.1109/TVCG.2013.243.

Gelb, D., Subramanian, A., and Tan, K.-H. Augmented reality for immersive remote collaboration. In *Proceedings of the IEEE Workshop on Person-Oriented Vision (POV)*, pages 1–6, January 2011. doi: 10.1109/POV.2011.5712368.

Gergle, D., Kraut, R., and Fussell, S. Action as language in a shared visual space. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 487–496, 2004.

Gruber, L., Gauglitz, S., Ventura, J., Zollmann, S., Huber, M., Schlegel, M., Klinker, G., Schmalstieg, D., and Höllerer, T. The City of Sights: Design, construction, and measurement of an augmented reality stage set. In *Proceedings of the 9th IEEE International Symposium* *on Mixed and Augmented Reality (ISMAR)*, pages 157–163, Seoul, Korea, October 13-16 2010a.

Gruber, L., Zollmann, S., Wagner, D., Schmalstieg, D., and Höllerer, T. Optimization of target objects for natural feature tracking. In *Proceedings of the 20th International Conference on Pattern Recognition (ICPR)*, pages 3607–3610, Istanbul, August 2010b. doi: 10.1109/ICPR.2010.880.

Gurevich, P., Lanir, J., Cohen, B., and Stone, R. TeleAdvisor: a versatile augmented reality tool for remote assistance. In *Proceedings of the ACM SIGCHI Annual Conference on Human Factors in Computing Systems (CHI)*, pages 619–622. ACM, 2012.

Gustafson, S., Baudisch, P., Gutwin, C., and Irani, P. Wedge: clutter-free visualization of off-screen locations. In *Proceedings of the ACM SIGCHI Annual Conference on Human Factors in Computing Systems (CHI)*, pages 787–796. ACM, 2008.

Güven, S., Feiner, S., and Oda, O. Mobile augmented reality interaction techniques for authoring situated media on-site. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 235–236, 2006. ISBN 1-4244-0650-1. doi: 10.1109/ISMAR.2006.297821.

Hachet, M., Declec, F., Knodel, S., and Guitton, P. Navidget for easy 3D camera positioning from 2D inputs. In *Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI)*, pages 83–89, 2008. doi: 10.1109/3DUI.2008.4476596.

Hanson, A. J. and Wernert, E. A. Constrained 3D navigation with 2D controllers. In *Proceedings of the 8th Conference on Visualization*, 1997. ISBN 1-58113-011-2.

Hartley, R. and Zisserman, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. ISBN 0521540518.

Höllerer, T., Feiner, S., Terauchi, T., Rashid, G., and Hallaway, D. Exploring MARS: developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computers & Graphics*, 23(6):779 – 785, 1999. ISSN 0097-8493. doi: 10.1016/S0097-8493(99)00103-X.

Höllerer, T., Wither, J., and DiVerdi, S. "Anywhere Augmentation": Towards mobile augmented reality in unprepared environments. In Gartner, G., Cartwright, W., and Peterson, M. P., editors, *Location Based Services and TeleCartography*, Lecture Notes in Geoinformation and Cartography, pages 393–416. Springer, Berlin/Heidelberg, 2007.

Hoppe, C., Klopschitz, M., Donoser, M., and Bischof, H. Incremental surface extraction from sparse structure-from-motion point clouds. In *Proceedings of the British Machine Vision Conference (BMVC)*, Bristol, UK, 2013.

Huang, W. and Alem, L. HandsInAir: A wearable system for remote collaboration on physical tasks. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 153–156. ACM, 2013. ISBN 978-1-4503-1332-2. doi: 10.1145/2441955.2441994.

Huang, W., Alem, L., and Tecchia, F. HandsIn3D: Augmenting the shared 3D visual space with unmediated hand gestures. In *SIGGRAPH Asia 2013 Emerging Technologies*, pages 10:1–10:3. ACM, 2013. ISBN 978-1-4503-2632-2. doi: 10.1145/2542284.2542294.

Igarashi, T., Matsuoka, S., and Tanaka, H. Teddy: A sketching interface for 3D freeform design. In *ACM SIGGRAPH 2007 Courses*, New York, NY, USA, 2007. ACM. ISBN 978-1-4503-1823-5. doi: 10.1145/1281500.1281532.

Jankowski, J. and Hachet, M. A survey of interaction techniques for interactive 3D environments. In *Eurographics – STAR*, pages 65–93, 2013.

Jo, H. and Hwang, S. Chili: Viewpoint control and on-video drawing for mobile video calls. In *Extended Abstracts, ACM SIGCHI Annual Conference on Human Factors in Computing Systems (CHI)*, pages 1425–1430. ACM, 2013. ISBN 978-1-4503-1952-2. doi: 10.1145/2468356.2468610.

Kim, S., Coffin, C., and Höllerer, T. Relocalization using virtual keyframes for online environment map construction. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 127–134, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-869-8. doi: 10.1145/1643928.1643958.

Kim, S., Lee, G. A., Sakata, N., Vartiainen, E., and Billinghurst, M. Comparing pointing and drawing for remote collaboration. In *Extended Abstracts, IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 1–6, 2013. doi: 10.1109/ISMAR.2013.6671833.

Kirk, D. and Fraser, D. S. Comparing remote gesture technologies for supporting collaborative physical tasks. In *Proceedings of the ACM SIGCHI Annual Conference on Human Factors in Computing Systems (CHI)*, pages 1191–1200, 2006. ISBN 1-59593-372-7. doi: 10.1145/1124772.1124951.

Kirk, D. S. *Turn It This Way: Remote Gesturing in Video-Mediated Communication*. PhD thesis, University of Nottingham, 2006.

Klein, G. and Murray, D. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

Klein, G. and Murray, D. Improving the agility of keyframe-based SLAM. In *Proceedings of the 10th European Conference on Computer Vision (ECCV)*, pages 802–815, Marseille, France, October 2008.

Klein, G. and Murray, D. Parallel tracking and mapping on a camera phone. In *Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 83–86, October 2009. doi: 10.1109/ISMAR.2009.5336495.

Klopschitz, M., Irschara, A., Reitmayr, G., and Schmalstieg, D. Robust incremental structure from motion. In *Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2010.

Kraut, R., Miller, M., and Siegel, J. Collaboration in performance of physical tasks: Effects on outcomes and communication. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 57–66, 1996.

Kurata, T., Sakata, N., Kourogi, M., Kuzuoka, H., and Billinghurst, M. Remote collaboration using a shoulder-worn active camera/laser. In *Proceedings of the 8th International Symposium on Wearable Computers (ISWC)*, pages 62–69, Washington, DC, USA, 2004. ISBN 0-7695-2186-X. doi: 10.1109/ISWC.2004.37.

Kurillo, G., Bajcsy, R., Nahrsted, K., and Kreylos, O. Immersive 3D environment for remote collaboration and training of physical activities. In *Proceedings of IEEE Virtual Reality (VR)*, pages 269 –270, 2008. doi: 10.1109/VR.2008.4480795.

Kuzuoka, H., Oyama, S., Yamazaki, K., Suzuki, K., and Mitsuishi, M. GestureMan: A mobile robot that embodies a remote instructor's actions. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 155–162, New York, NY, USA, 2000. ISBN 1-58113-222-0. doi: 10.1145/358916.358986.

Ladikos, A., Benhimane, S., Appel, M., and Navab, N. Model-free markerless tracking for remote support in unknown environments. In *Proceedings of the International Conference on Computer Vision Theory and Applications*, 2008.

Langlotz, T., Degendorfer, C., Mulloni, A., Schall, G., Reitmayr, G., and Schmalstieg, D. Robust detection and tracking of annotations for outdoor augmented reality browsing. *Computers & Graphics*, 35(4):831–840, 2011. ISSN 0097-8493. doi: 10.1016/j.cag.2011.04.004.

Lanir, J., Stone, R., Cohen, B., and Gurevich, P. Ownership and control of point of view in remote assistance. In *Proceedings of the ACM SIGCHI Annual Conference on Human Factors in Computing Systems (CHI)*, pages 2243–2252, 2013. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2481309.

Lee, T. and Höllerer, T. Viewpoint stabilization for live collaborative video augmentations. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 241–242, Washington, DC, USA, 2006. ISBN 1-4244-0650-1. doi: 10.1109/ISMAR.2006.297824.

Li, J., Wessels, A., Alem, L., and Stitzlein, C. Exploring interface with representation of gesture for remote collaboration. In *Proceedings of the 19th Australasian conference on Computer-Human Interaction (OZCHI)*, pages 179–182, 2007. ISBN 978-1-59593-872-5. doi: 10.1145/1324892.1324926.

Lieberknecht, S., Huber, A., Ilic, S., and Benhimane, S. RGB-D camera-based parallel tracking and meshing. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 147–155, October 2011. doi: 10.1109/ISMAR.2011.6092380.

Lourakis, M. A. and Argyros, A. SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software*, 36(1):1–30, 2009. doi: 10.1145/1486525.1486527.

Lovegrove, S. and Davison, A. J. Real-time spherical mosaicing using whole image alignment. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 6313, pages 73–86, 2010. ISBN 978-3-642-15557-4. doi: 10.1007/978-3-642-15558-1_6.

Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

Maimone, A. and Fuchs, H. Encumbrance-free telepresence system with real-time 3D capture and display using commodity depth cameras. In *Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 137–146, October 2011. doi: 10.1109/ISMAR.2011.6092379.

Maimone, A., Yang, X., Dierk, N., State, A., Dou, M., and Fuchs, H. General-purpose telepresence with head-worn optical see-through displays and projector-based lighting. In *Proceedings of IEEE Virtual Reality (VR) 2013*, Orlando, USA, March 2013.

Marchal, D., Moerman, C., Casiez, G., and Roussel, N. Designing intuitive multi-touch 3D navigation techniques. In Kotzé, P., Marsden, G., Lindgaard, G., Wesson, J., and Winckler, M., editors, *Human-Computer Interaction – INTERACT 2013*, volume 8117 of *Lecture Notes in Computer Science*, pages 19–36. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40482-5. doi: 10.1007/978-3-642-40483-2_2.

Molton, N., Davison, A., and Reid, I. Locally planar patch features for real-time structure from motion. In *Proceedings of the 15th British Machine Vision Conference (BMVC)*, 2004.

Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P. Real time localization and 3D reconstruction. In *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 363–370, 2006. doi: 10.1109/CVPR.2006.236.

Muja, M. and Lowe, D. G. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application (VISSAPP)*, pages 331–340. INSTICC Press, 2009.

Mulloni, A., Ramachandran, M., Reitmayr, G., Wagner, D., Grasset, R., and Diaz, S. User friendly SLAM initialization. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 153–162, October 2013. doi: 10.1109/ISMAR.2013.6671775.

Newcombe, R., Lovegrove, S., and Davison, A. DTAM: Dense tracking and mapping in real-time. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2320–2327, 2011a.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. KinectFusion: Real-time dense surface mapping and tracking. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011b.

Nistér, D. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.

Nuernberger, B., Gauglitz, S., Höllerer, T., and Turk, M. Investigating viewpoint visualizations for click & go navigation. In *Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI)*, 2014.

Oda, O., Sukan, M., Feiner, S., and Tversky, B. Poster: 3D referencing for remote task assistance in augmented reality. In *Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI)*, pages 179–180, March 2013. doi: 10.1109/3DUI.2013.6550237.

Ou, J., Fussell, S. R., Chen, X., Setlock, L. D., and Yang, J. Gestural communication over video stream: supporting multimodal interaction for remote collaborative physical tasks. In *Proceedings of the 5th International Conference on Multimodal interfaces (ICMI)*, pages 242–249, 2003. ISBN 1-58113-621-8. doi: 10.1145/958432.958477.

Pan, Q., Arth, C., Reitmayr, G., Rosten, E., and Drummond, T. Rapid scene reconstruction on mobile phones from panoramic images. In *Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 55–64, 2011. doi: 10.1109/ISMAR.2011.6092370.

Pirchheim, C., Schmalstieg, D., and Reitmayr, G. Handling pure camera rotation in keyframe-based slam. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 229–238, October 2013. doi: 10.1109/ISMAR.2013.6671783.

Pirchheim, C. and Reitmayr, G. Homography-based planar mapping and tracking for mobile phones. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 27–36, 2011. doi: 10.1109/ISMAR.2011.6092367.

Pollefeys, M., Verbiest, F., and Van Gool, L. Surviving dominant planes in uncalibrated structure and motion recovery. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 613–614, 2002.

Pradeep, V., Rhemann, C., Izadi, S., Zach, C., Bleyer, M., and Bathiche, S. Monofusion: Realtime 3D reconstruction of small scenes with a single web camera. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 83–88, 2013. doi: 10.1109/ISMAR.2013.6671767.

Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L., and Fuchs, H. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the ACM SIGGRAPH Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 179–188, 1998. ISBN 0-89791-999-8. doi: 10.1145/280814.280861.

Regenbrecht, H., Lum, T., Kohler, P., Ott, C., Wagner, M., Wilke, W., and Mueller, E. Using augmented virtuality for remote collaboration. *Presence: Teleoperators and Virtual Environments*, 13:338–354, July 2004. ISSN 1054-7460. doi: 10.1162/1054746041422334.

Reitmayr, G. and Schmalstieg, D. Mobile collaborative augmented reality. In *Proceedings of the IEEE and ACM International Symposium on Augmented Reality (ISAR)*, pages 114–123, 2001. doi: 10.1109/ISAR.2001.970521.

Reitmayr, G., Eade, E., and Drummond, T. Semi-automatic annotations in unknown environments. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 67–70, Nara, Japan, November 13–16 2007.

Repko, J. and Pollefeys, M. 3D models from extended uncalibrated video sequences: Addressing key-frame selection and projective drift. In *Proceedings of the 5th IEEE International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 150–157, 2005.

Rissanen, J. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

Rosten, E. and Drummond, T. Machine learning for high-speed corner detection. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, volume 1, pages 430–443, May 2006. doi: 10.1007/11744023_34.

Sadagic, A., Towles, H., Holden, L., Daniilidis, K., and Zeleznik, B. Tele-immersion portal: Towards an ultimate systhesis. In *Proceedings of the 4th Annual International Workshop on Presence of Computer Graphics and Computer Vision Systems*, 2001.

Schwarz, G. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

Shi, J. and Tomasi, C. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994. doi: 10.1109/CVPR.1994.323794.

Snavely, N., Seitz, S., and Szeliski, R. Photo tourism: Exploring photo collections in 3D. *ACM Transactions on Graphics*, 25(3):835–846, 2006.

Snavely, N., Garg, R., Seitz, S. M., and Szeliski, R. Finding paths through the world's photos. In *Proceedings of the ACM SIGGRAPH Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 15:1–15:11, New York, NY, USA, 2008. ISBN 978-1-4503-0112-1. doi: 10.1145/1399504.1360614.

Sodhi, R. S., Jones, B. R., Forsyth, D., Bailey, B. P., and Maciocci, G. BeThere: 3D mobile collaboration with spatial input. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 179–188, 2013. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2470679.

Stafford, A., Piekarski, W., and Thomas, B. Implementation of god-like interaction techniques for supporting collaboration between outdoor AR and indoor tabletop users. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 165–172, 2006. ISBN 1-4244-0650-1. doi: 10.1109/ISMAR.2006.297809.

Strasdat, H., Montiel, J., and Davison, A. Real-time monocular SLAM: Why filter? In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2657–2664, 2010.

Sukan, M., Feiner, S., Tversky, B., and Energin, S. Quick viewpoint switching for manipulating virtual objects in hand-held augmented reality using stored snapshots. In *Proceedings of the 11th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2012.

Tan, D. S., Robertson, G. G., and Czerwinski, M. Exploring 3D navigation: combining speed-coupled flying with orbiting. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 418–425, New York, NY, USA, 2001. ISBN 1-58113-327-8. doi: 10.1145/365024.365307.

Tan, W., Liu, H., Dong, Z., Zhang, G., and Bao, H. Robust monocular SLAM in dynamic environments. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 209–218, October 2013. doi: 10.1109/ISMAR.2013.6671781.

Tang, J. C. and Minneman, S. L. VideoDraw: a video interface for collaborative drawing. *ACM Transactions on Information Systems*, 9:170–184, April 1991. ISSN 1046-8188. doi: 10.1145/123078.128729.

Tatzgern, M., Grasset, R., Kalkofen, D., and Schmalstieg, D. Transitional augmented reality navigation for live captured scenes. In *Proceedings of IEEE Virtual Reality (VR)*, Minnesota, USA, March 2014.

Tolba, O., Dorsey, J., and McMillan, L. Sketching with projective 2D strokes. In *Proceedings of the 12th ACM Symposium on User Interface Software and Technology (UIST)*, pages 149–157, New York, NY, USA, 1999. ISBN 1-58113-075-9. doi: 10.1145/320719.322596.

Torr, P. H., Fitzgibbon, A. W., and Zisserman, A. The problem of degeneracy in structure and motion recovery from uncalibrated image sequences. *International Journal of Computer Vision (IJCV)*, 32:27–44, 1999. ISSN 0920-5691. doi: 10.1023/A:1008140928553.

Torr, P. Bayesian model estimation and selection for epipolar geometry and generic manifold fitting. *International Journal of Computer Vision (IJCV)*, 50(1):35–61, 2002.

Uyttendaele, M., Criminisi, A., Kang, S. B., Winder, S., Szeliski, R., and Hartley, R. Image-based interactive exploration of real-world environments. *IEEE Computer Graphics and Applications*, 24(3):52–63, May 2004. ISSN 0272-1716. doi: 10.1109/MCG.2004.1297011.

van den Hengel, A., Dick, A., Thormählen, T., Ward, B., and Torr, P. H. S. VideoTrace: Rapid interactive scene modelling from video. *ACM Transactions on Graphics*, 26(3), July 2007. ISSN 0730-0301. doi: 10.1145/1276377.1276485.

Wagner, D., Schmalstieg, D., and Bischof, H. Multiple target detection and tracking with guaranteed framerates on mobile phones. In *Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 57–64, October 2009. doi: 10.1109/ISMAR.2009.5336497.

Wagner, D., Mulloni, A., Langlotz, T., and Schmalstieg, D. Real-time panoramic mapping and tracking on mobile phones. In *Proceedings of IEEE Virtual Reality (VR)*, March 2010.

Wellner, P. and Freeman, S. The DoubleDigitalDesk: Shared editing of paper documents. Technical Report EPC-93-108, EuroPARC, 1993.

Wuest, H., Wientapper, F., and Stricker, D. Acquisition of high quality planar patch features. *Advances in Visual Computing*, pages 530–539, 2008.

Xin, M., Sharlin, E., and Sousa, M. C. Napkin Sketch: Handheld mixed reality 3D sketching. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 223–226, 2008. ISBN 978-1-59593-951-7. doi: 10.1145/1450579.1450627.

Zeleznik, R. and Forsberg, A. Unicam – 2D gestural camera controls for 3D environments. In *Proceedings of the ACM Symposium on Interactive 3D Graphics (I3D)*, pages 169–173, 1999. ISBN 1-58113-082-1. doi: 10.1145/300523.300546.

Zeleznik, R. C., Herndon, K. P., and Hughes, J. F. SKETCH: An interface for sketching 3D scenes. In *ACM SIGGRAPH 2007 Courses*, 2007. ISBN 978-1-4503-1823-5. doi: 10.1145/1281500.1281530.