Freeform 3D Interactions in Everyday Environments

David Kim

School of Computing Science

Newcastle University

A thesis submitted for the degree of

Doctor of Philosophy

June 2014

This thesis is dedicated to my loving parents

for their love and endearing support

Acknowledgements

I wouldn't be here writing about any of what is going to follow in the next 200 pages, if it wasn't for the people who have inspired, supported and believed in me and who have given me the opportunity to work with them in the last few years. First, I would like to thank my supervisor Patrick Olivier, who essentially allowed me to start a career path as a researcher. Although being one of the most busiest persons I know, he always had time for me.

A very special thank to Shahram Izadi, who has been a key figure in my research journey and played many important roles beyond being my mentor, programming partner and manager at Microsoft Research. He allowed me to join his group and work with leading experts in computer science on a daily basis. He is one of the very few people who I've seen leading by example and creating very fruitful collaborative environments with a strong vision. He has guided and encouraged me to go beyond my limits so many times. Thanks for always creating opportunities and being around (always!), and especially for debugging my code so many times, even when it means being stuck in the office after midnight!

I would also like to thank Otmar Hilliges for introducing me to this research area when I was an undergraduate student at Ludwig-Maximilian University in Munich and introducing me to Patrick and Shahram.

Thanks to everyone in Culture Lab, Newcastle University. Jonathan Hook and Paul Dunphy for being great flatmates and helping me getting settled in the UK. Anja Thieme and Ko-Le Chen for always sharing their positivity with me. Jürgen Wagner for always being a ray of sunshine and teaching

us how to be strong, you are missed. Dan Jackson, Stephen Lindsey, John Shearer, Phil Heslop, Dave Green, Tom Bartindale for always offering me a helping hand.

Thanks to my colleagues and collaborators, who I really enjoy working with at Microsoft Research Cambridge, Cem Keskin, Christoph Rhemann, Sean Fanello, Christopher Zach, Alex Butler, Steve Hodges, Andrew Fitzgibbon, Pushmeet Kohli and Jamie Shotton. And for the very inspiring and enlightening hallway chats! Dave Molyneaux, Malte Weiß, Iason Oikonomidis, Matthias Nießner, Jiawen Kevin Chen, Dennis Bautembach, Richard Newcombe and Jakub Dostal, it has been my greatest pleasure working with you on so many amazing projects!

Thanks also to the Applied Sciences Group at Microsoft in Redmond especially Vivek Pradeep and Stevie Bathiche. Thanks for taking me on for a great productive summer in Redmond. Paul Dietz, Pablo Sala, Tristan Trutna, Tim Large, Karlton Powel, Nigel Keam and John Weiss for never hesitating to give me a hand.

My best friend Wladimir Barski, thanks for always being there for me mate! Sanghee Park, who really believed in me and encouraged me to follow my dreams. Most importantly, I would like to thank my mum Young-Nam, dad Chul-Ho, sister Ah-Rum and brother Paul who supported me on so many levels. You are the best!

Abstract

Personal computing is continuously moving away from traditional input using mouse and keyboard, as new input technologies emerge. Recently, natural user interfaces (NUI) have led to interactive systems that are inspired by our physical interactions in the real-world, and focus on enabling dexterous freehand input in 2D or 3D. Another recent trend is Augmented Reality (AR), which follows a similar goal to further reduce the gap between the real and the virtual, but predominately focuses on output, by overlaying virtual information onto a tracked real-world 3D scene.

Whilst AR and NUI technologies have been developed for both immersive 3D output as well as seamless 3D input, these have mostly been looked at separately. NUI focuses on sensing the user and enabling new forms of input; AR traditionally focuses on capturing the environment around us and enabling new forms of output that are registered to the real world. The output of NUI systems is mainly presented on a 2D display, while the input technologies for AR experiences, such as data gloves and body-worn motion trackers are often uncomfortable and restricting when interacting in the real world.

NUI and AR can be seen as very complimentary, and bringing these two fields together can lead to new user experiences that radically change the way we interact with our everyday environments. The aim of this thesis is to enable real-time, low latency, dexterous input and immersive output without heavily instrumenting the user. The main challenge is to retain and to meaningfully combine the positive qualities that are attributed to both NUI and AR systems.

I review work in the intersecting research fields of AR and NUI, and explore freehand 3D interactions with varying degrees of expressiveness, directness and mobility in various physical settings. There a number of technical challenges that arise when designing a mixed NUI/AR system, which I will address is this work: What can we capture, and how? How do we represent the real in the virtual? And how do we physically couple input and output? This is achieved by designing new systems, algorithms, and user experiences that explore the combination of AR and NUI.

Main Publications

David Kim, Shahram Izadi, Jakub Dostal, Christoph Rhemann, Cem Keskin, Christopher Zach, Jamie Shotton, Tim Large, Steven Bathiche, Matthias Nießner, D. Alex Butler, Sean Fanello, Vivek Pradeep. 2014. **RetroDepth: 3D silhouette sensing for high-precision input on and above physical surfaces**. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (**CHI** '14). ACM, New York, NY, USA. (full paper)

David Kim, Otmar Hilliges, Shahram Izadi, Alex D. Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. 2012. **Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor**. In Proceedings of the 25th annual ACM symposium on User interface software and technology (**UIST** '12). ACM, New York, NY, USA, 167-176. (full paper)

Otmar Hilliges, **David Kim**, Shahram Izadi, Malte Weiß, and Andrew Wilson. 2012. **HoloDesk: direct 3d interactions with a situated see-through display**. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (**CHI** '12). ACM, New York, NY, USA, 2421-2430. (full paper)

Shahram Izadi, **David Kim**, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. **KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera**. In Proceedings of the 24th annual ACM symposium on User interface software and technology (**UIST** '11). ACM, New York, NY, USA, 559-568. (full paper)

David Molyneaux, Shahram Izadi, **David Kim**, Otmar Hilliges, Steve Hodges, Xiang Cao, Alex Butler, and Hans Gellersen. 2012. **Interactive environment-aware handheld projectors for pervasive computing spaces**. In Proceedings of the 10th international conference on Pervasive Computing (**Pervasive**'12). Springer-Verlag, Berlin, Heidelberg, 197-215. (full paper)

Other Publications

Christian Weichel, Manfred Lau, **David Kim**, Nicolas Villar, Hans Gellersen. 2014. **MixFab: a mixed-reality environment for personal fabrication**. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (**CHI** '14). ACM, New York, NY, USA. (full paper)

D. Alex Butler, Shahram Izadi, Otmar Hilliges, David Molyneaux, Steve Hodges, and **David Kim**. 2012. **Shake'n'sense: reducing interference for overlapping structured light depth cameras**. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (**CHI** '12). ACM, New York, NY, USA, 1933-1936. (short paper)

Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, **David Kim**, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. **KinectFusion: Real-time dense surface mapping and tracking**. In Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality (**ISMAR** '11). IEEE Computer Society, Washington, DC, USA, 127-136. (full paper)

Alex Butler, Otmar Hilliges, Shahram Izadi, Steve Hodges, David Molyneaux, **David Kim**, and Danny Kong. 2011. **Vermeer: direct interaction with a 360 viewable 3D display**. In Proceedings of the 24th annual ACM symposium on User interface software and technology (**UIST** '11). ACM, New York, NY, USA, 569-576. (full paper)

David Kim, Paul Dunphy, Pam Briggs, Jonathan Hook, John W. Nicholson, James Nicholson, and Patrick Olivier. 2010. **Multi-touch authentication on tabletops**. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (**CHI** '10). ACM, New York, NY, USA, 1093-1102. (full paper)

Contents

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

In 1961 while sitting in a conference session on computer graphics, Douglas Engelbart reflected on how to improve the channel of communication between user and computer. At the time, computing interfaces predominately focused on the command line, and were tailored towards expert users. Engelbart envisioned a new, accessible or "user-friendly" paradigm for human computer interaction (HCI), where computing interfaces could be operated by novice and experts, young and old alike. He observed that by creating a device with two wheels perpendicular to each other, the rotation of each wheel could be sensed, and mapped to a translation along one axis. The computer could track the combined 1D motions to move a 2D cursor on the display accordingly. He recorded the idea in his notebook [Doug Engelbart Institute, 2014]. Two years later along with his colleague Bill English, Engelbart built the first computer mouse [Engelbart, 1970].

Over the next half century, researchers have explored this very same question: how to improve the channel of communication between user and computer, with the goal of ultimately making computing more "user friendly". We have seen many successful (and some less successful) concepts, research prototypes and commercial products over the years (see also Figure 1.1): At around the same time as the invention of the computer mouse, Ivan Sutherland and colleagues Ron Baecker, Wesley Clark, W.R. "Bert" Sutherland, and Fontaine Richardson, at MIT's Lincoln Labs [Buxton et al., 2005] researched concepts such as pen-based computing, with a ground breaking

project named SketchPad [Sutherland, 1963], as well as visual programming languages [Sutherland, 1966], 2D computer animation [Baecker, 1969], and other genre-defining ideas. This work seeded the rise of graphical user interface (GUI). Chuck Thacker, Butler Lampson, Alan Kay and colleagues were clearly influenced by both the seminal work of Engalbart and Lincoln Labs, when they invented the first personal computer – the Xerox Star – giving rise to the first GUI that employed the notion of Windows, Icons, Menus, Pointer (WIMP). A concept that lives in most desktop computers today, and was the direct inspiration for early versions of the Macintosh and Windows operating systems [Hinckley, 1996].

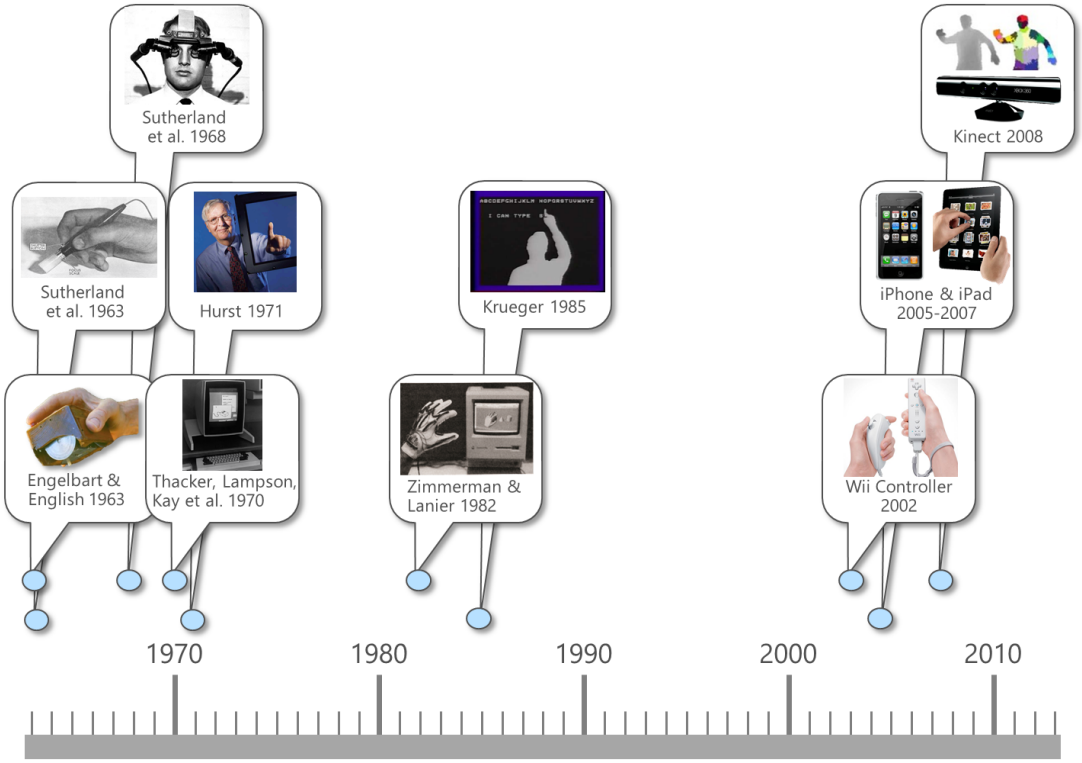

Figure 1.1: User interface technologies developed over the past half century that have radically changed how people interact with computers.

1.1.1 The rise of Natural User Interfaces

Whilst the mouse, GUI, and WIMP solidified the notion of *desktop computing*, researchers over the last half century have also explored computer interaction *beyond the*

desktop. The motivation again is to create user interfaces (UIs) that are "user-friendly" and improve the communication between the user and computer. Whilst desktop computing borrowed metaphors from the real-world (hence the name "desktop"), these new systems more explicitly recreated aspects of physical interaction in the digital domain. A particular focus of this work was on new *input* paradigms as opposed to *output*.

The first important invention in this space was the notion of *direct touch computing* in 1971, with the birth of the first resistive touch digitizer [Colwell and Hurst, 1975]. Whilst supporting only a single point of 2D input, the device heralded a more *direct* way of reaching out and touching digital content for interaction. Further inventions such as mutual capacitance touch sensors emerged at CERN in the late 70s, and later the first multi-touch digitizers were invented at the University of Toronto [Bill Buxton, 1994; Lee et al., 1985] (see Bill Buxton [2013] for important work in the field of multi-touch that followed). Since then, multi-touch input has become the de-facto method for interaction on personal devices such as tablets and phones, popularized by consumer devices such as the iPad and iPhone [Apple, 2014].

The popularity of multi-touch can in large be attributed to two main factors: First it replaces the mouse, an indirect mediator for input, with the ability to *directly* couple input with output. Second, it leverages the dexterity of our hands for input, in particular, harnessing higher degrees-of-freedom (DoFs) than is traditionally available through the mouse. For this reason, many researchers began referring to these interfaces as *natural* (particularly when compared to their traditional desktop counterparts). Despite being used in other contexts such as wearable computing [Starner et al., 1997b], the term "natural" struck a chord with multi-touch researchers, as it nicely captured the quality of these interfaces: removing the need for a dedicated indirect input devices for interaction, and borrowing more from physical interaction when manipulating digital objects.

In 1985, Myron Krueger developed a series of interactive art exhibits that responded to the movement and gesture of the viewer through an elaborate system of sensing floors, interactive tables, and video cameras. Audience members could directly interact with the video projections within a shared environment. Krueger leveraged cameras and custom signal processing hardware, to enable users to interact via silhouettes of their whole bodies and hands, without the need for specific input devices. It was one of the first truly unencumbered user experiences that leveraged the physicality of real-world input in the digital world. Whilst input was sensed in 2D, users were free to move

around the entirety of the 3D physical space to interact. At around this time, other modalities such as speech and eye gaze were being explored for user input. Chris Schmandt and colleagues at MIT invented a seminal hand pointing and speech UI for manipulating digital content called 'put that there' [Bolt, 1980]. Richard Bolt and others later extended this system to replace pointing with eye gaze input [Bolt, 1984].

This ground breaking work of Krueger, Schmandt, Bolt, alongside the birth of multi-touch, cemented the notion of the *Natural User Interface* (NUI). Put simply, NUI means capturing the qualities of physical interaction when interacting digitally. Typically this means moving away from using a dedicated input device, such as mouse, towards *unencumbered* input leveraging hands and body motion, gesture, speech and eye gaze. Whilst NUI is not conceptually new, it has only recently come to the forefront, perhaps exemplified by the first and second generation Xbox Kinect [Microsoft, 2014], which incorporated whole body tracking (and later head tracking and hand gestures) and speech for input.

Despite these recent advances, NUI still has major technical challenges, in particular robustly sensing the user, and their relationship with the environment, as well as the design of metaphors and UIs that can leverage these rich modalities of input.

1.1.2 The rise of Virtual and Augmented Reality

Only five years after the invention of the mouse and GUI, Ivan Sutherland and Bob Sproull at Lincoln Labs invented a new mode of display. Referred to as the Sword of Damocles this is widely considered to be the first virtual reality (VR) and augmented reality (AR) head-mounted display (HMD) system. The device was primitive both in terms of UI and realism, and the graphics comprising the virtual environment were simple wireframe rooms. The system displayed output from a computer program in the binocular display. The perspective that the software showed the user depended on the user's head pose. The weight of the HMD, and the need to track the head movements necessitated the HMD being attached to a mechanical arm suspended from the ceiling of the lab. The formidable appearance of the mechanism inspired its name.

Despite the primitive hardware of the Sword of Damocles, this system captured the essence of AR: spatially-aware virtual graphics that are overlaid onto the real world. Since then much research on *spatial AR* has emerged (see Chapter 2) with the aim of

closing the gap between real and virtual worlds. Researchers and companies alike have looked at improving the underlying display technologies [Lanman and Luebke, 2013; Maimone and Fuchs, 2013; Occulus Rift, 2014; Vuzix, 2014], capturing the real-world at a higher fidelity (in order to create virtual objects that are aware of the physical space), including work on Simultaneous Localization and Mapping (SLAM) [Davison et al., 2007; Thrun et al., 2005], as well as new sensor technologies such as depth or range cameras, and realistic rendering techniques [Gruber et al., 2012].

One significant area of commercial impact for spatial AR is in the mobile devices sector. In particular, harnessing the rising processing power, and camera and display resolutions on tablets and phones to create a porthole into the virtual world that is registered with the real-world [Layar, 2014; Olwal, 2009; Qualcomm Vuforia, 2014; Wagner, 2007]. Another high profile product in this space is the Google Glass [Google, 2014]. However, interestingly, this device does not support any of the spatial qualities of AR systems described previously and is more synonymous with wearable computing [Billinghurst and Starner, 1999; Starner et al., 1997b].

In many ways AR is the antithesis of NUI, the focus is on *user instrumentation* rather than unencumbered user experiences. The majority of AR systems have also focused on some of the fundamental technical challenges such as display hardware, mapping and pose tracking, or rendering rather than user interaction per se. The focus on input in early AR/VR systems was on 3D trackers, such as the Polhemus or 'Flock of Birds' [Ascension Technology Corporation, 2014; Polhemus, 2014] or visual trackers [Vicon, 2014], which required the use of markers or dedicated input devices. The Nintendo Wii [Nintendo, 2014] is an example of a commercial success based on these types of tracking technologies, but the necessity of a hand-held device limits interactive scenarios.

An early attempt to harness the dexterity of our hands for AR and VR was led by Zimmerman and Lanier in 1982, with the invention of the first hand tracking glove. Zimmerman invented an optical flex sensor mounted in a glove to measure finger bending, and worked with Lanier to incorporate ultrasonic and magnetic hand position tracking technology [Zimmerman et al., 1987] to create a fully articulated 3D hand tracker [Dipietro et al., 2008]. This created an even richer range of 3D freehand input that could be leveraged in computer interfaces. However, as shown later, the

instrumentation of the hand proved a barrier for unencumbered interactions, and limited scenarios of use [Brashear et al., 2003].

Despite these user centric limitations of AR, the promise of bringing real and virtual worlds closer together is compelling and clearly has captured the imagination of consumers and researchers alike.

1.1.3 The Camera: the Ultimate Sensor

Whilst a wide range of sensing technologies, from 3D magnetic motion trackers [Ascension Technology Corporation, 2014; Polhemus, 2014] through to capacitive and resistive touch sensors [Bill Buxton, 2013] have been employed as key components of AR and NUI systems, we have seen a convergence towards one main prototyping technology – the *camera*.

Camera technology is becoming truly ubiquitous and commodity, becoming widely available in computing devices such as laptops, tablets, smart phones and TVs. Low-resolution optical sensors are already widely used as an essential component of barcode scanners, fingerprint scanners, optical mice and projection keyboards. Higher resolution RGB and infrared cameras are widely used in eye and face trackers and are currently the key enabler of many NUI technologies e.g. Han [2005] and Wilson [2005]. Section 1.1.1 and 1.1.2 already described other input devices, such as the Xbox Kinect [Microsoft, 2014], Nintendo Wii [Nintendo, 2014] or Vicon [Vicon, 2014] that harness the power of cameras and computer vision. Cameras are also the key sensor for tracking and mapping in AR systems [Davison et al., 2007; Thrun et al., 2005].

In terms of flexibility, cameras are in many ways the *ultimate sensor* and can be used without specific hardware development. This presents an interesting shift in the development of new input devices and methods, from the development of new hardware prototypes and sensors to software algorithms and systems.

1.1.4 Depth Sensing

Depth cameras capture *multiple* range measurements of the scene in *real-time*. Since the advent of Xbox Kinect [Microsoft, 2014], depth cameras have become commodity and enabled considerable new research in the AR and NUI space. However, consumer depth cameras do not solve the challenges of AR or NUI in themselves. In many ways,

the problem of building new AR or NUI systems becomes even more *challenging* as these sensors now provide even richer data to work with.

However, real-time depth data also allows developers and researchers alike to move beyond some of the low-level computer vision problems that can be barriers for the development of NUI and AR systems. In particular, with regular RGB cameras it is difficult to *segment* the scene into foreground and background, and this is considerably simpler with depth data. Additionally, depth cameras remove the problem of computing depth using lower quality and computationally expensive *stereo* techniques (see Scharstein and Szeliski [2002]). These problems are a difficult hurdle for regular RGB cameras. Depth cameras have therefore helped researchers in the AR or NUI space move onto other computer vision problems, for instance human pose recognition [Shotton et al., 2011] or real-time scene reconstruction [Izadi et al., 2011b].

However, working with depth data is challenging, given the richness of the data, and the necessity to use or design more computationally complex 3D computer vision techniques [Hartley and Zisserman, 2004]. Furthermore, the current range of depth cameras suffers from a high degree of noise. For example, the first generation Kinect uses a method akin to stereo matching, where each small 2D patch of the reference dot pattern is matched by sweeping a window along the associated epipolar line in the observed IR image [Freedman et al., 2008b]. This leads to ambiguities causing outliers, holes, and fattening effects particularly at depth discontinuous. Time-of-flight (ToF) depth sensors, which resolve distance based on the measured time-of-flight of a signal emitted by the camera and reflected off a target, also suffer high degrees of noise. ToF sensors suffer from issues such as *multi-path* or *mixed pixels* [Remondino and Stoppa, 2013], where light returns from secondary surfaces to the same pixel during a single exposure, as well as '*flying pixels*' where depth is estimated across multiple captures and scene motion results in foreground and background contributions being averaged together [Remondino and Stoppa, 2013].

1.2 Thesis goals

The goal of this thesis is specifically to bring the worlds of NUI and AR together. These areas can be seen as very complimentary: while NUI focuses on sensing new forms of input, AR traditionally focuses on new forms of output that are registered to

the real world. Bringing these two fields together can lead to new user experiences that radically change the way we interact and even think about computing systems, and further increase the communication channel between user and computer. Like Engelbart and researchers that followed, our aims are to create more accessible or "user-friendly" paradigms for HCI, where computing interfaces could be operated by novice and experts, young and old alike. The combination of AR and NUI into cohesive user experiences and underlying systems is motivated by the vast research and consumer interest in these spaces.

Specifically, this thesis presents a number of interactive systems that combine the affordances of AR and NUI in novel ways. The aim is to provide the best of both worlds, to allow for wider adoption of these types of new technologies. The focus of this thesis will predominately be technical: it will outline new systems that combine AR and NUI in interesting new ways as a means to further HCI research in these fields. Each system will uncover the technical and user challenges in combining NUI and AR.

Each system will provide strong technical, systems, and engineering contributions to HCI pushing forward research on post-WIMP interfaces. Depth cameras will be used as the main prototyping tool for developing the systems. In so doing, a number of novel algorithms for depth cameras will be presented, which focus on sensing the user, other physical objects and environments in richer ways. This will constitute the main technical contribution of this thesis, beyond the systems that have been developed as a whole. In working with depth cameras, we will address some of the limitations of existing sensors, as well as existing AR and NUI systems in this field.

1.3 Thesis overview

As highlighted, this thesis is technically motivated and I aim to explore and demonstrate working systems that combine AR and NUI in novel ways. However, as will be demonstrated in this thesis, sensing the user and the real world, and spatially locating the virtual and the real is technically challenging.

The main aims of this thesis is the exploration and development of novel algorithms, techniques and working systems that enable 3D natural interactions in physical spaces. The contributions of this work span many layers of a mixed NUI and AR system (see Figure 1.2), ranging from low-level input sensing with custom hardware to raw data

processing pipelines and user input models for applications. Techniques and methods from the computer vision, wearable computing, natural user interaction and augmented reality community are utilized and combined in novels ways to create new forms of interaction experiences, in previously under-explored scenarios.

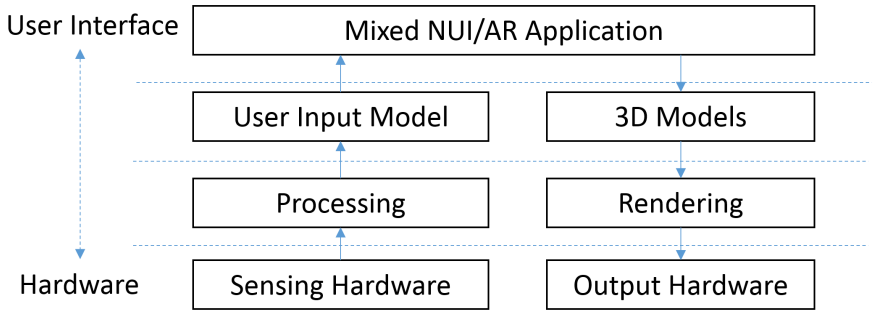

Figure 1.2: The software and hardware stack of mixed NUI/AR applications.

In Chapter 2, I will provide a detailed review of literature in the NUI and AR areas, in particular focusing on the fields of hand tracking, gestural interactions and spatial AR. In the core chapters of the thesis I will introduce a number of new systems AR/NUI systems.

In Chapter 3, I present a system called RetroDepth. The system combines multiple NUI modalities, including real-time freehand touch and 3D sensing and object sensing at high precision. It demonstrates how a variety of NUI and AR capabilities can be combined in a single compelling system.

In Chapter 4, I move one step beyond RetroDepth with a system called HoloDesk which also allows for real-time freehand interactions between real and virtual, but in a more 'direct' way. It takes the NUI concepts presented earlier in Chapter 3 further by combining additional concepts from AR, in particular spatial registration of real and virtual worlds. HoloDesk is a situated display and removes the need for direct user instrumentation. It supports mid-air and physics-based interactions above a desktop.

In Chapter 5, I move beyond the fixed (or situated) NUI/AR display and explore the use of a real-time SLAM system called KinectFusion. The specific contributions I make to this work is allowing for NUI affordances to be incorporated in this AR/SLAM system.

In Chapter 6, I take this work in Chapter 5 one step further by coupling input and output using a handheld projector unit. This removes the need for direct user instrumentation but allows for direct user feedback. In this chapter I also explore the use of a hybrid infrastructure and infrastructure-less system based on fixed and mobile sensors, as a means to compare with the infrastructure-free system in Chapter 5.

In Chapter 7, I introduce a body-worn hand tracker that I developed that allows users to take 3D natural interactions outside the living room to outdoor spaces. The sensor is completely self-contained and recovers the full 3D pose of the user's hand without requiring an external sensing infrastructure or covering the hand itself.

Finally, in Chapter 8, I conclude this work with a discussion of the approaches and systems I have presented. I summarize the contributions this thesis makes to the field of HCI, NUI and AR systems, and highlight limitations and potential future work.

Chapter 2

Related Work

2.1 Introduction

The previous chapter outlined some of the early work in the fields of HCI, NUI and AR, charting important early disruptions that helped shape and define these subfields, and more recent trends. I identified a key opportunity to bring the fields of NUI and AR closer together by creating novel technical systems and user experiences. In this chapter, I present and discuss the current state of the art in terms of NUI and AR, which informs the design and implementation of the systems presented in this thesis. The focus of this literature review will again be technical, focusing on new technologies and systems that have directly impacted the fields of AR and NUI.

2.2 Imaging Sensors

As mentioned in the previous chapter, imaging sensors have been at the heart of many NUI and AR systems. For example, Microsoft's first interactive tabletop utilizes cameras behind the touch surface for tracking fingers with diffuse illumination (DI) (see Figure 2.1 left). In 2006, Han [2005] presented another cheap and easily reproducible camera-based touch sensing technology (see Figure 2.1 right) that helped spark a large community comprising researchers, designers and hobbyists working on tabletop NUIs. RGB and depth cameras are also the key component for 3D pose estimation and mapping in SLAM AR systems [Davison et al., 2007; Thrun et al., 2005]. In Section

1.1.1 and 1.1.2 we already touched on other input devices, such as the Xbox Kinect [Microsoft, 2014], Nintendo Wii [Nintendo, 2014] or Vicon [Vicon, 2014] that harness the power of cameras and computer vision.

As mentioned, cameras are in many ways the *ultimate sensor* giving the richness of data that can be captured, but require the design of novel software systems and algorithms.

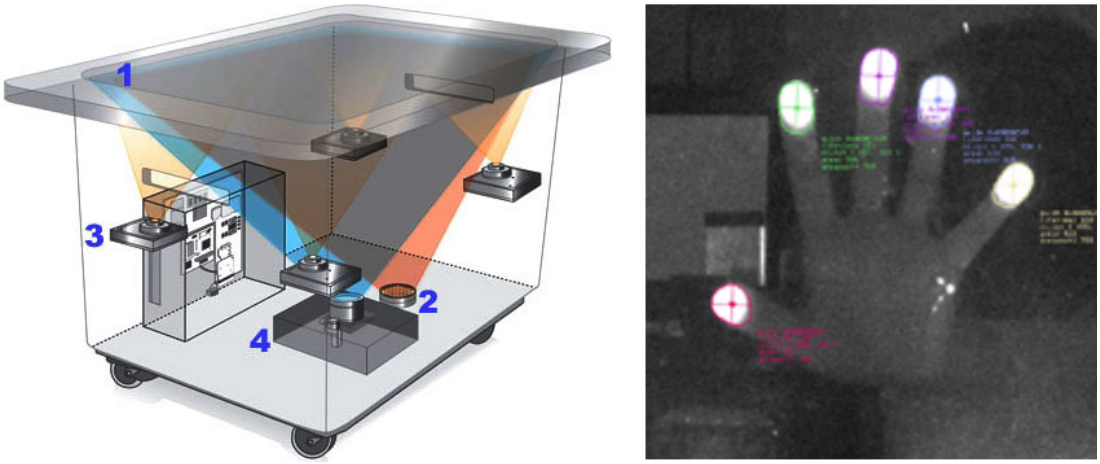

Figure 2.1: Left: Projector and cameras inside the original Microsoft Surface (source: [Microsoft, 2014]) . Right: Raw infrared image of touch points on a transparent sheet of acrylic (source: [Han, 2005]).

2.2.1 Depth Sensing

Depth cameras encode metric distance in each pixel and simplify the recognition and segmentation of individual objects in a scene. Since the advent of Xbox Kinect [Microsoft, 2014], depth cameras have become commodity and allowed developers and researchers to focus on higher level computer vision problems, in particular human pose recognition [Shotton et al., 2011] and real-time scene reconstruction [Izadi et al., 2011b]. Depth cameras are particularly interesting in NUI and AR, as they make the use of artificial markers in the scene obsolete [Fiala, 2005; Vicon, 2014; Wagner et al., 2008].

However, one of the main drawbacks with consumer depth cameras is that they suffer from high noise at object boundaries, which makes jitter-free tracking of objects,

fingers and the user or reconstruction of the exact geometry of the environment a hard problem. Related work demonstrates (see Section 2.3) that these boundaries (or silhouettes) are critical for interaction. For example, the current generation Kinect uses a method akin to stereo matching, where each small 2D patch of the reference dot pattern is matched by sweeping a window along the associated epipolar line in the observed IR image [Freedman et al., 2008b]. This leads to ambiguities when matching at object boundaries, where large depth discontinuities lead to outliers, holes, and edge fattening. Time-of-flight (ToF) depth sensors, which resolve distance based on the measured time-of-flight of a signal emitted by the camera and reflected off a target, also suffer higher degrees of noise at object boundaries. ToF sensors suffer from issues such as *multi-path* or *mixed pixels* [Remondino and Stoppa, 2013], where light returns from secondary surfaces to the same pixel during a single exposure, as well as '*flying pixels*' where depth is estimated across multiple captures and scene motion results in foreground and background contributions being averaged together [Remondino and Stoppa, 2013].

This issue with consumer depth cameras motivates the design of new high-precision depth estimation techniques especially at silhouette and object boundaries (see Chapter 3). Whilst consumer depth cameras lack precision, researchers have demonstrated much higher quality sensors, in particular by using dynamic structured light patterns (see Lanman and Taubin [2009] for a survey of relevant technologies). There are however limitations in such systems for our interactive scenarios. Most of these systems estimate depth by projecting multiple dynamic patterns into the scene and capturing multiple images with a camera. To deal with a moving scene (which is necessary in many interactive scenarios) and therefore limit motion artifacts, these patterns need to be projected and imaged at very high frame-rates. This necessitates costly high speed camera and projector hardware that can add considerably high cost to the system [Weise et al., 2007; Zhang, 2010]. Beyond procurement cost, there is also the computational cost associated with computing dense depth maps.

2.3 Sensing Touch

Early uses of cameras in the context of interaction coincide with pioneering work on NUI, indicating their importance for enabling such experiences. Beginning in 1979,

Krueger et al. [1985] developed Videoplace in addition to many other "'responsive environments"' that react to the shape and motion of the user. Videoplace was an artistic installation that allowed users to create and manipulate shapes and play with virtual characters onscreen using their own silhouette, which is captured with a video camera in front of a neutral background (see Figure 2.2 left). Krueger's work on "'artificial environments"', originally presented in the context of interactive art, is seen as a key influencer in the early innovation of VR and NUI.

HoloWall [Matsushita and Rekimoto, 1997] allows for similar types of interactions, however, users interact more directly with the projected content (see Figure 2.2 middle). In contrast to Videoplace, input is only sensed when the user is touching or in close proximity to a diffuse glass wall. DI and a camera placed behind the screen allow hand interactions to be captured. The original Microsoft Surface 1.0 tabletop [Wikipedia, 2014] and other *bottom-up* configurations extended this idea of HoloWall to create interactive tabletops. Jefferson Han and colleagues exploited the notion of frustrated total internal reflection (FTIR) to sense fingertips [Han, 2005; White, 1965].

Top-down configurations place a 2D camera above a physical surface, and have proved popular as they provide more flexibility in sensing, and allow any regular desk to be transformed into an interactive one [Koike et al., 2001; Matsushita and Rekimoto, 1997; Wellner, 1993; Wilson, 2005]. Because these systems rely on 2D cameras, segmentation of hands and objects from the (typically cluttered) desktop is non-trivial, and touch and hover is difficult to disambiguate. These challenges have led researchers to use stereo cameras for higher precision touch sensing, combined with simple "hardware" techniques for segmentation.

Aiming to bring these types of interactions to any environment, Wilson [2005] developed a compact and self-contained projection-camera system that turns ordinary tables into multi-touch surfaces (see Figure 2.2 right). The system consists of a short-throw projector pointed down onto the surface, and an infrared emitter and camera. The emitter and camera are horizontally displaced enabling the camera to capture hand shadows. Upon touch, the shadow gets partially occluded by the finger itself, which is detected by the system.

To address some of these limitations of top-down configurations, researchers first turned to *stereo vision*. Visual Trackpad [Malik and Laszlo, 2004] is one example that turned two ordinary cameras into a low-cost input device. The system uses a rigid

Figure 2.2: Left: Silhouette interaction in VIDEOPLACE (source: [Krueger et al., 1985]). Middle: User sensed via infrared reflection in HoloWall (source: [Matsushita and Rekimoto, 1997]). Right: Touch detection through infrared shadow shape analysis in PlayAnywhere (source: [Wilson, 2005]).

panel with rectangular interaction area above which 3D fingertips and hand gestures are captured with a pair of web-cameras (see Figure 2.3 left and middle). However, input is indirect and interaction is limited to hands only.

C-Slate [Agarwal et al., 2007; Izadi et al., 2007a] combines stereo-based touch estimation with stylus input, object recognition and simple depth sensing. Here, a commercially available horizontal tablet display is used for stylus input, whereas polarizing filters on the camera block light coming from the screen (see Figure 2.3 right).

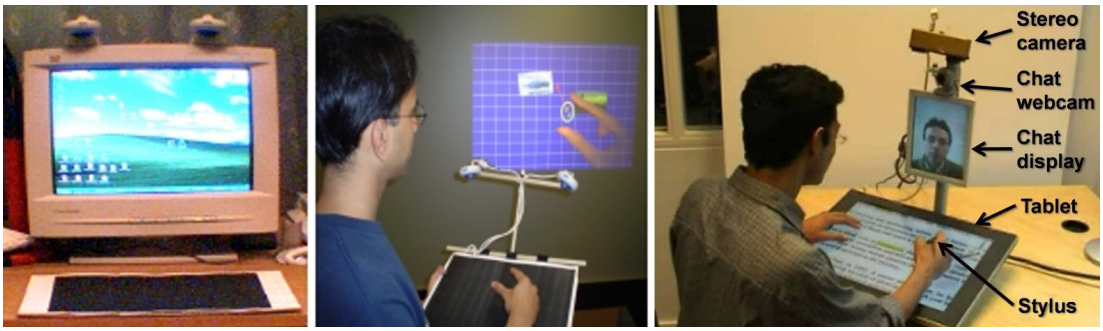

Figure 2.3: Left: Stereo camera setup for optical touch and hover detection in Visual Trackpad (source: [Malik and Laszlo, 2004]). Middle: Visual Trackpad in hand-held mode (source: [Malik and Laszlo, 2004]). Right: C-Slate, stereo-based touch estimation and stylus input (source: [Izadi et al., 2007a]).

2.4 Sensing Hands Above the Tabletop

A potential next step in exploring the design of interactive surfaces is to look at in-air interactions, even in combination with traditional 2D touch. However, going beyond the surface breaks the coupling between input and output. One way to adapt the dimensions of the output to the 3D input is to combine stereo projection onto a horizontal surface [Agrawala et al., 1997; Czernuszenko et al., 1997] with tracked input devices, or to combine back-projected tabletop displays with head mounted displays (HMDs) [Nakashima et al., 2007]. These systems require head-worn hardware and potentially use tracked input devices. These types of systems are at odds with the vision of NUIs: to build systems that are lightweight and 'walk-up and use', require no user instrumentation, and to support freehand interactions.

Other notable 2D camera systems for combined on-surface and in-air interactions include high-end (and costly) marker-based motion tracking systems [Marquardt et al., 2011] and imaging through the display. Izadi et al. [2008] developed an interactive tabletop that can project and sense through the display with a switchable diffuser (see Figure 2.4 left and middle-left). The diffuser is based on liquid crystal and can be rapidly switched between transparent and diffuse states. This system is extended by Hilliges et al. [2009b] to enable physics-based freehand interactions above the display (see Figure 2.4 middle-right and right). In this system, depth is estimated from diffuse IR light falloff that is emitted from underneath the surface and reflected off the hand. In this system, the user can conceptually pick up objects with a pinch gesture. The shadow of the user's hand is rendered in the 3D scene and is used as a feedback mechanism to indicate depth to the user. However, these systems require a large space in front or behind the display, hence allowing interactions only in a tabletop form factor. Researchers also explored ToF sensors placed behind [Hilliges et al., 2009b] and above [Wilson, 2007b] projection surfaces. However, these sensors suffered from low resolution and high noise making interactions coarse and limited.

Hirsch et al. [2009] present a more compact design with the BiDi screen that utilizes the liquid crystal panel itself to capture 3D gestures (see Figure 2.5 left). The display functions conventionally when the backlight is turned on, but allows a camera behind the screen to capture hundreds of tiny images of the scene through a mask created by the liquid crystal panel. The images are captured from slightly different perspectives

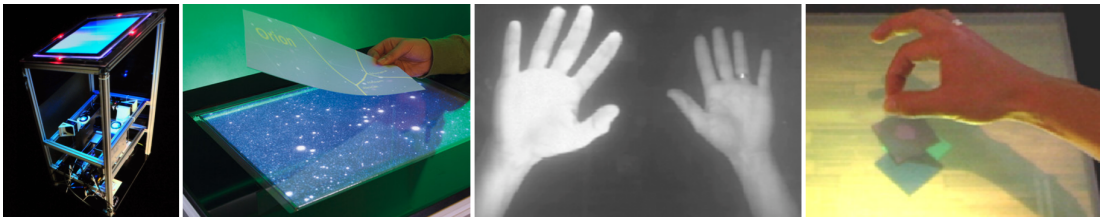

Figure 2.4: Left and middle-left: Projection and infrared sensing through the touch surface in SecondLight (source: [Izadi et al., 2007a]). Middle-right and right: DepthShadows extends SecondLight and allows for physics-based gestural interactions above the display (source: [Hilliges et al., 2009b]).

and are used to reconstruct the full light field in front of the display when the backlight is turned off. However, depth sensing is extremely coarse, and environment lighting needs to be highly controlled.

2.4.1 Other Non-Contact Sensing

Beyond the use of cameras, systems have demonstrated 2D multi-touch sensing using resistive and capacitive technologies [Bill Buxton, 2013]. However, these systems typically only sense touch on a 2D surface instead of 3D input or rich sensing of hands and other tangible objects. In terms of non-optical 3D sensing, stylus-based input using magnetic or inductive technologies have been presented (e.g. Subramanian et al. [2006]; Wesche and Seidel [2001]). Other non-camera-based systems include electric-field sensing techniques [Lee et al., 2006; Paradiso, 2002], use of scanning lasers [Paradiso, 2002] and laser-line generators [Takeoka et al., 2010], or IR proximity sensors [Annett et al., 2011; Izadi et al., 2009; Moeller et al., 2012] either placed around the bezel or embedded behind a display. These techniques either coarsely sense 3D input, or have a limited interaction volume, or are limited to 3D input only.

2.4.2 The Rise of Consumer Depth Cameras

With the advent of consumer depth cameras, many new systems for in-air interactions coupled with surface-based interactions have appeared. Examples include situated AR displays such as SpaceTop [Lee et al., 2013] and MirageTable [Benko et al., 2012], as well as augmented desktop systems [Haubner et al., 2013; Liu et al., 2013]. 3Gear

Systems Inc [2013] demonstrate markerless tracking of specific hand gestures above a desktop using a top-down Kinect camera, which is based on Wang et al. [2011].

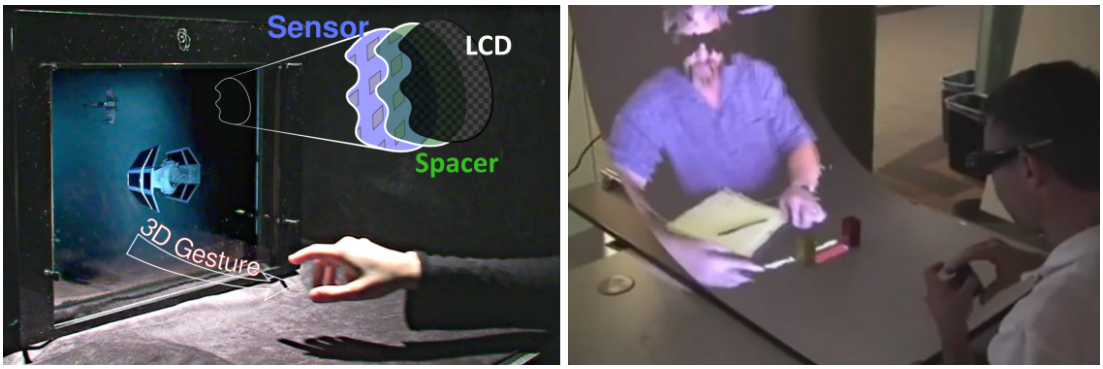

Figure 2.5: Left: BiDi, depth-sensing LCD for 3D interactions using light fields (source: [Hirsch et al., 2009]). Right: MirageTable, situated augmented reality system (source: [Benko et al., 2012]).

MirageTable [Benko et al., 2012] is a situated augmented reality system that consists of a single depth camera, a stereoscopic projector and a curved screen (see Figure 2.5 right). The system enables perspective corrected stereoscopic visualizations through head tracking, and enables physics-based freehand interactions, both using a depth camera. Proposed applications include virtual 3D model creation, interactive gaming with real and virtual objects, and a 3D teleconferencing experiences in a shared 3D space.

This and most of the other systems use a Kinect camera both for in-air interactions and touch sensing [Wilson, 2010] and inherently lack precision due to the quantization noise of the Kinect (see below) making such systems imprecise compared to other input devices [Dippon and Klinker, 2011].

Two recent notable products explore high-precision in-air and touch sensing using stereo cameras. The commercial Leap Motion sensor [Leap Motion Inc, 2013] is a small, self-contained unit containing a pair of wide field-of-view (FoV) IR cameras and IR LEDs pointing upwards on a desk (see Figure 2.6 left). The device provides tracking of fingertips and other strong peaks in the input signal, such as a pen, with high frame-rate and precision sensing. The device does not expose a full contour, only tracked 3D points mapping to high-level features such as fingertips. Further, segmentation is based on assuming that only free-space exists behind interacting objects. This limits the device to

in-air input only, which can result in arm fatigue during prolonged interactions. Haptix [Haptix, 2013] uses a similar small desktop unit with two IR cameras but instead points across the physical surface to detect multi-touch gestures only (see Figure 2.6 right).

Figure 2.6: Left: Leap Motion 3D controller (source: [Leap Motion Inc, 2013]). Right: Haptix, stereo-based multi-touch controller (source: [Haptix, 2013]).

2.5 Hand and Gesture Sensing

The literature discussed so far utilizes various parts of our hands for natural interactions on surfaces and in mid-air. While capturing shapes, sensing touch and extracting fingertips enable high degrees-of-freedom input, we are still not leveraging the full dexterity and expressiveness of the hand itself. The full 3D pose of our hands, provides more flexibility in terms of sensing dynamic and static hand gestures for high DoF input.

Given their dexterity, sensing the full 3D pose of a user's hand is a technically challenging and active area of research. One approach is to instrument the hand directly by wearing a glove with embedded sensors or markers (see Dipietro et al. [2008] for an overview). Originally designed for AR/VR, such sensors have since been leveraged for mobile scenarios. However, gloves can be cumbersome and uncomfortable to wear, can degrade tactile sensation and limit interaction with capacitive touch sensors, which are now standard on mobile phones.

2.5.1 Hand Tracking with Cameras

An approach that avoids user instrumentation is to place a camera in the environment pointed directly at the user's hand. As highlighted earlier in this chapter, many systems focus purely on fingertip detection for on-surface or in-air input. Other systems have recognized a small set of discrete 3D hand poses (e.g. Izadi et al. [2007b]).

The vision community has explored higher fidelity 3D hand tracking methods (see Erol et al. [2007] for a detailed survey). Real-time 3D pose recovery is becoming feasible, using either non-parametric methods that typically use nearest neighbor lookup into a database of RGB [Romero et al., 2010] or depth images [Wang et al., 2011], or attempt to fit a parametrized model of the hand to observed images [Oikonomidis et al., 2011].

Wang et al. [2011] demonstrate a marker-less tracking method that applies a data-driven pose estimation algorithm inspired by the marker-based technique of Wang and Popović [2009], and uses a precomputed database to map image features to a limited set of hand poses relevant to specific application gestures (see Figure 2.7 left). In contrast to this, Oikonomidis et al. [2011] present a model-based hand tracking approach that recovers the full pose of a continuously moving hand. Their method treats this as an optimization problem, which is solved using a variant of Particle Swarm Optimization (PSO), which estimates the hand model parameters that minimize the discrepancy between the appearance and 3D structure of hypothesized instances of a hand model and actual hand observations (see Figure 2.7 right). These systems offer some of the most sophisticated mechanisms for 3D hand tracking, without the need for direct user instrumentation. However, these approaches clearly lack mobility, requiring sensors embedded in the environment and high computational cost.

2.5.2 Depth Cameras On the Move

The rise of consumer depth cameras has also led to interest in 3D interaction on the move. Handheld systems can restrict freeform hand interactions that require both hands, however body-worn systems alleviate this issue. Harrison et al. [2011] developed OmniTouch, a wearable depth-sensing and projection system that enables interactive multi-touch applications on everyday surfaces. Beyond the shoulder-worn system, there is no instrumentation of the user or environment. ShoeSense [Bailly et al., 2012] is a

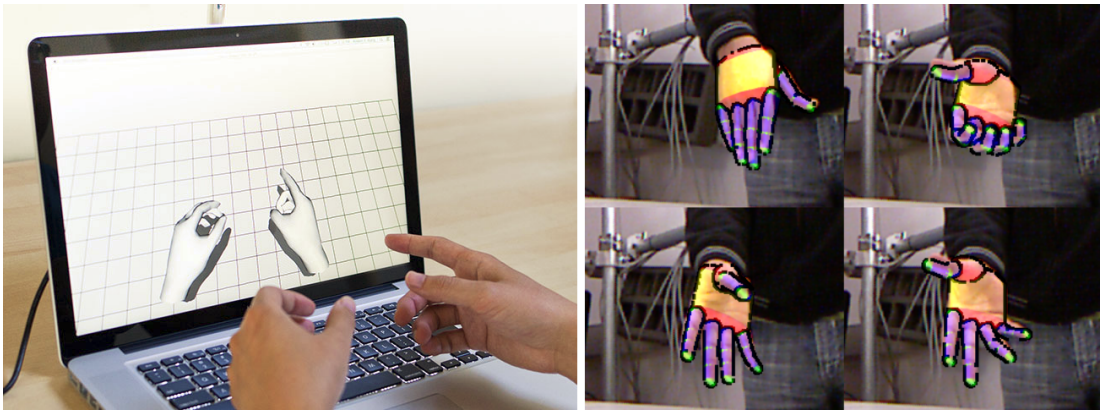

Figure 2.7: Left: 3Gear, depth camera based hand tracking (source: [Wang and Popović, 2009]). Right: Hand tracking with particle swarm optimization (source: [Oikonomidis et al., 2011]).

shoe-worn depth camera that is pointed upwards and senses a set of basic hand gestures. Both systems can suffer from occlusions of the hands from other parts of the body, and do not support full 3D hand pose recovery, but instead focus on sensing touch interaction with planar surfaces [Harrison et al., 2011] or detect simple pinch gestures for interaction [Bailly et al., 2012]. There are also practical barriers in making depth cameras mobile, in particular power consumption and form-factor.

2.5.3 Ultra-mobile Wearable Gesture Systems

The wearables literature has proposed many lightweight systems for mobile gestural interaction. Instead of supporting high-fidelity sensing, they use IR proximity sensors to detect coarse motion of fingers [Howard and Howard, 2001; Kim et al., 2007; Lee et al., 2011; Nakatsuma and Shinoda, 2011], sense muscle or tendon activity to recognize a small set of discrete hand gestures [Rekimoto, 2001; Saponas et al., 2009], or leverage acoustics to coarsely localize touch on the body [Harrison et al., 2010].

The Gesture Watch, developed by Kim et al. [2007], utilizes an array of infrared proximity sensors to sense very coarse hand gestures above a watch and interprets the gestures using a Hidden Markov Model (HMM). Researchers have also explored non-optical sensing (see Figure 2.8 left). Rekimoto [2001] developed GestureWrist, a wrist-worn device that recognizes simple hand gestures through capacitance sensing. The shape of the wrist is used for gesture recognition, which is recovered by measuring

distances between transmitter and receiver electrodes around the wrist (see Figure 2.8 middle and right).

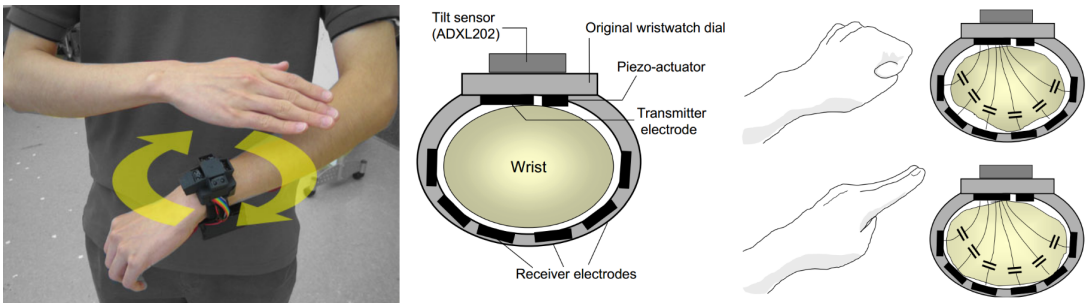

Figure 2.8: Left: The Gesture Watch (source: [Kim et al., 2007]). Middle and right: GestureWrist, hand gesture recognition based on capacitance sensing (source: [Rekimoto, 2001]).

Saponas et al. [2009] present a muscle-computer interface that senses muscular activity by measuring the electrical potential between pairs of electrodes around the forearm (see Figure 2.9 left and middle-left). The presented system is able to classify a number of static grip postures, which can be used to trigger simple actions. Skinput [Harrison et al., 2010] is another sensor that is worn around the forearm. This sensor resolves the location of finger taps on the arm and hand by analyzing vibrations that propagate through the body. A wearable pico-projector is used to turn the skin into a direct input surface (see Figure 2.9 middle-right and right).

Figure 2.9: Left and middle-left: Gesture classification with electromygraphy for muscle computer interfaces (source: [Saponas et al., 2009]). Middle-right and right: Skinput, finger tap detection by analyzing mechanical vibrations that propagate through the body (source: [Harrison et al., 2010]).

These system are all based on lightweight sensing components and machine learning for gesture classification. The focus, however, lies on coarse touch localization, motion gestures, or discrete gesture detection, rather than continuous high DoF interactions.

2.5.4 Body-Worn 2D Camera Systems

One final class of wearable systems uses 2D cameras to add higher fidelity sensing without greatly compromising form-factor. These systems use either monochrome cameras and diffuse IR illumination [Gustafson et al., 2010; Starner et al., 2000] or RGB cameras [Lee and Hollerer, 2007; Mistry and Maes, 2009; Pamplona, 2008; Starner et al., 1997a; Vardy, 1999].

Starner et al. [1997a] demonstrates an RGB camera mounted top down on a baseball cap that recognizes sentence-level American Sign Language with HMMs (see Figure 2.10 first and second). The Gesture Pendant [Starner et al., 2000] uses a similar technique to detect discrete hand gestures, but is extended to also enable continuous input using hand gestures (see Figure 2.10). This system consists of an infrared camera and a ring of infrared LEDs worn around the neck to allow for more robust segmentation of hands. Gustafson et al. [2010] present an almost identical neck-worn device, but focus on studying the feasibility of spatial interaction without a display or any other active output (see Figure 2.11 left).

Figure 2.10: First and second: RGB camera on a baseball cap for American Sign Language recognition (source: [Starner et al., 1997a]). Rest: The Gesture Pendant. Uses an infrared camera worn on a hat or around the neck and a ring of infrared for gesture detection (source: [Starner et al., 2000]).

Similar setups have been investigated in the context of AR. Mistry and Maes [2009] developed Sixth Sense, a neck-worn device that combines a pico-projector and an RGB camera to visually augment surfaces, walls and other physical objects. Input is enabled through tracked colored markers attached to the user's fingertips (see Figure 2.11 middle). Lee and Hollerer [2007] presented an augmented reality system that tracks the 6DoF pose of a rigid hand by directly extracting fingertips from an RGB image (see Figure 2.11 right). However, the hand is essentially reduced to a single plane that can only be used to rigidly move a 3D object or to control a virtual camera.

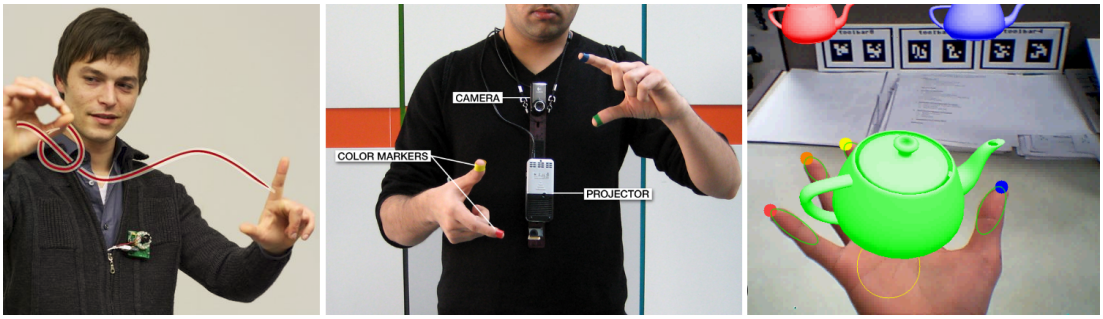

Figure 2.11: Left: Imaginary Interfaces utilizes a chest-worn infrared camera (source: [Gustafson et al., 2010]). Middle: Sixth Sense enables gestural interactions with a neck-worn camera-projector unit (source: [Mistry and Maes, 2009]). Right: HandAR tracks the 6DoF of a rigid hand for virtual camera control (source: [Lee and Hollerer, 2007]).

Inferring the full 3D pose of the hand is clearly challenging from such 2D input. Currently systems have only demonstrated simple 2D pinch gestures [Gustafson et al., 2010] or detecting fingers using marker [Mistry and Maes, 2009] or markerless approaches [Lee and Hollerer, 2007; Vardy, 1999] for simple pointing, open and closed hand gestures. Starner et al. [1997a, 2000] classify a wider set of discrete hand postures e.g. for sign language. The form-factor of these systems allows for various body-worn placements (see Mayol-Cuevas et al. [2009]). These include placement on shoulder or head [Lee and Hollerer, 2007; Starner et al., 1997a] and around the neck [Gustafson et al., 2010; Starner et al., 2000], which have the benefit of capturing both hands for bimanual input, but restrict the interaction space to a fixed region directly in front of the user's upper body. Interactions often cannot be subtle and are publicly visible (a potential barrier for adoption in public [Bailly et al., 2012]). Arm fatigue can be an issue for prolonged use.

A viable option to closer couple hand and sensor are cameras directly looking across the hand of the user. Vardy [1999] uses a watch-like RGB camera attached on the inner side of the wrist. The presented systems detects up to seven discrete gestures by counting fingers that are visible to the camera (see Figure 2.12 left). Pamplona [2008] presents a prototype of a wrist-worn camera that can recover the full pose of a continuously articulated hand. The camera looks across the hand and images fingertips with markers placed on each finger, inferring hand pose from the 3D fingertip estimate using inverse kinematics (IK). The system is fairly large in terms of form-factor and

the requirement of wearing markers on each finger is a clear limitation (see Figure 2.12 right).

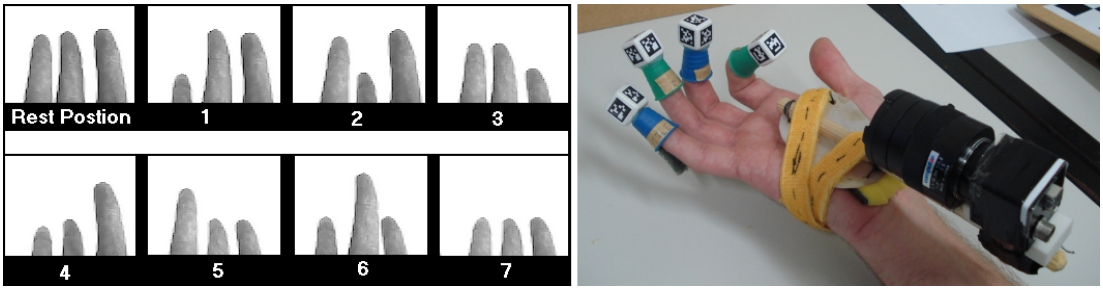

Figure 2.12: Left: Wrist-worn camera for finger gesture interaction (source: [Vardy, 1999]). Right: Finger tracking with visual markers attached to each fingertip (source: [Pamplona, 2008]).

2.6 Sensing the Environment

Beyond this work on NUI and wearable computing, we have seen many AR systems that exploit the geometry of the physical environment for virtual interactions. A system that is aware of the exact geometry and the changes made to the environment could allow us to leverage the space around us to extend the UI into the physical environment.

Reconstructing geometry using active sensors [Levoy et al., 2000], passive cameras [Hartley and Zisserman, 2004; Merrell et al., 2007], large database of online images [Frahm et al., 2010], or from unordered 3D points [Kazhdan et al., 2006; Zhou et al., 2011] are well-studied areas of research in computer graphics and vision. There is also extensive literature within the AR and robotics community on Simultaneous Localization and Mapping (SLAM), aimed at tracking a user or robot while creating a map of the surrounding physical environment (see Thrun [2003]). Given this broad topic, and our desire to build systems for *interaction*, this section is structured around specific design goals for exploiting the physical world in the context of NUI.

Interactive Rates. One of the primary goals of an interactive system is to achieve real-time rates for *both* camera tracking *and* 3D reconstruction. This speed is critical for permitting direct feedback and user interaction and differentiates the goals of this thesis from many existing reconstruction approaches that support only offline

reconstructions [Frahm et al., 2010], real-time but non-interactive rates (e.g. the Kinect-based system of Henry et al. [2010] reconstructs at ∼2Hz), or support real-time camera tracking but non real-time reconstruction or mapping phases [Klein and Murray, 2007; Newcombe and Davison, 2010; Newcombe et al., 2011].

Figure 2.13: Parallel tracking and mapping with a structure from motion system (source: [Klein and Murray, 2007]). Left: Tracked sparse image features. Right: Virtual object superimposed on a real scene.

No Explicit Feature Detection. Structure from motion (SfM) systems (e.g. Klein and Murray [2007]) or RGB plus depth (RGBD) techniques (e.g. Henry et al. [2010]; Huhle et al. [2010]) need to robustly and continuously detect sparse scene features. However, an approach to camera tracking that is less prone to failure caused by sensor noise and rapid camera motions is desired. Systems that directly work on the dense depth maps, e.g. acquired from the Kinect sensor, could operate more reliably and would also avoid the reliance on RGB (used in recent Kinect RGBD systems e.g. Henry et al. [2010]) allowing use in indoor spaces with variable lighting conditions.

High-Quality Reconstruction of Geometry. A core goal is to capture the geometry of the real scene with as much detail as possible (or *densely*). Many SLAM systems (e.g. Klein and Murray [2007]) focus on real-time tracking, using *sparse* maps mainly for localization rather than reconstruction (see Figure 2.13). Others have used simple point-based representations (such as surfels [Henry et al., 2010] or aligned point-clouds [Huhle et al., 2010]) for reconstruction. A system that reconstructs *dense* surfaces,

which more accurately approximate real-world geometry, is preferred to these point-based representations.

Dynamic Interaction Assumed. As this thesis explores tracking and reconstruction in the context of *user interaction*, it is critical that the representation we use can deal with a dynamically changing scene, where users directly interact in front of the camera. While there has been work on using mesh-based representations for live reconstruction from passive RGB [Merrell et al., 2007; Newcombe and Davison, 2010; Newcombe et al., 2011] or active ToF cameras [Cui et al., 2010; Weise et al., 2009], these mesh representations do not deal with changing, dynamic scenes.

Infrastructure. One other important question is the requirement for pre-installed *sensing* infrastructures – whether this is fiducial markers (e.g. Wagner et al. [2008]), augmentation of the user (e.g. the use of gloves in Rusinkiewicz et al. [2002]), fixed or large sensors [Levoy et al., 2000; Rusinkiewicz et al., 2002], or systems fully embedded in the environment (e.g. Cao and Balakrishnan [2006]; Vlasic et al. [2009]). Whilst mobile and infrastructure-less systems allows the user to readily explore and scan the environment, capturing different views of the larger scene, it is technically challenging and can impact robustness of user sensing. I explore both arrangements in this thesis.

2.7 Combining Input with Output

In the previous section I presented ways of capturing users and real physical spaces for interaction. In this section, we focus on work combining the virtual with the real.

Much research exists on spatially coupling real and virtual graphics for 3D interaction within the AR community (see Azuma et al. [2001, 1997]; Van Krevelen and Poelman [2010]; Zhou [2008] for a detailed survey). Early systems relied on head-worn video or optical see-through displays. Whilst enabling mobility and ubiquitous use, head-worn displays have drawbacks including small field-of-view, incorrect focus cues, inherent latency, and tracking inaccuracies, resulting in discomfort and fatigue during use [Hoffman et al., 2008].

However, AR is also moving towards more lightweight uses. For example, removing the need for head-mounted displays by leveraging mobile phones [Papagiannakis et al.,

2008] or tablets [Kim et al., 2011], and focusing more on hand-based interactions instead of specialized input devices (e.g. Lee and Hollerer [2007]).

2.7.1 Situated Augmented Reality

Whilst HMDs provide immersion and mobility, researchers have also explored more situated uses for AR to move away from their inherent issues [Bimber and Raskar, 2005]. In some senses, this work becomes the convergence of AR and tabletops, and is one of the closest relating to the goals of this thesis. Situated displays have been developed by mounting large optical combiners such as beam splitters [Bimber et al., 2001] or holographic screens [Olwal et al., 2005] in front of the real scene. ASTOR [Olwal et al., 2005] is an autostereoscopic optical see-through system that uses a transparent holographic optical element to separate the views produced by two, or more, projectors (see Figure 2.14 left). It is a minimally intrusive AR system that doesn't require the user to wear special glasses or any other equipment, since the user will see different images depending on the viewpoint. However, systems like these often prevent the user from directly interacting with the scene because the optical elements act as a physical barrier.

However, exceptions do exist where users can reach into the display and directly interact with virtual objects in 3D [Mulder and Liere, 2002; Poston and Serra, 1994]. Input is provided through tracked objects [Mulder and Liere, 2002], styli [Poston and Serra, 1994], sometimes augmented with haptic feedback (for example by a PHANTOM device [Johnson et al., 2004]). The Virtual Workbench [Poston and Serra, 1994] spatially combines input and output for direct interactions in a near-field desktop virtual reality system for examination of medical images (see Figure 2.14 middle-left). In this system, the output of a 3D stereoscopic display is optically co-located with a tracked 3D stylus using a mirror. The Personal Space Station [Mulder and Liere, 2002] uses a similar hardware setup but extends this with a pair of cameras, allowing users to interact with a variety of optically tracked objects (see Figure 2.14 middle).

The work by Prachyabrued and Borst [2011] provides dexterous interactions with physics-enabled 3D objects but requires to wear stereo glasses and a data glove (see Figure 2.14 middle-right). Hachet et al. [2011] use a capacitive touchscreen underneath a stereo display and beam-splitter setup to manipulate 3D objects floating between the

input plane and the user with bare hands (see Figure 2.14 right). However, interaction is bound to the surface and no in-air manipulations are possible.

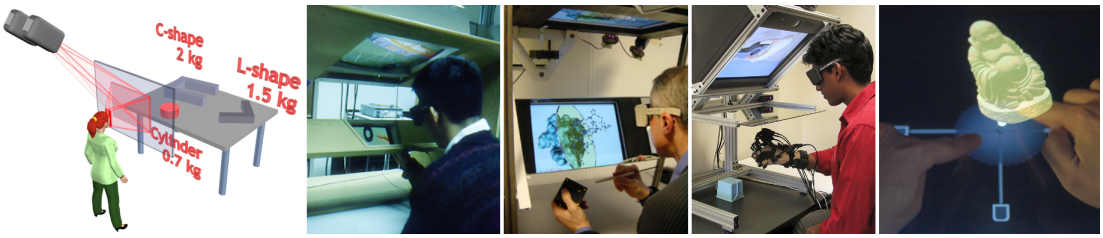

Figure 2.14: See-through displays. Left: ASTOR, autostereoscopic see-through display (source: [Olwal et al., 2005]). Middle-left: The Virtual Workbench (source: [Poston and Serra, 1994]). Middle: The Personal Space Station (source: [Mulder and Liere, 2002]). Middle-right: Virtual grasping in a situated AR setup (source: [Prachyabrued and Borst, 2011]). Right: Toucheo, 2D multi-touch see-through display (source: [Hachet et al., 2011]).

Many of the above systems use stereoscopic graphics to improve depth-perception, usually requiring some form of head-worn glasses, which can impact ergonomics and do not provide a walk-up-and-use experience without any user instrumentation. However, the lack of stereo depth-cues can be somewhat compensated by enabling others, including a technique commonly referred to as fish-tank VR [McKenna, 1992]. Many studies from the literature (e.g., Stavness et al. [2010]; Ware et al. [1993]) have suggested that view-point corrected graphics maybe more important than stereo in yielding a strong sense of 3D.

Glasses-free situated AR and 3D displays have been proposed. Yoshida et al. [2010] explore an optical configuration where a beam-splitter is combined with an array of projectors pointed at a retro-reflective surface to provide spatially coupled 3D interaction. Vermeer [Butler et al., 2011] leverages parabolic mirrors to optically levitate a 360° volumetric display in an interactive space. However, the focus of these systems is on novel 2D and 3D display technologies, rather than exploring rich physically realistic interactions. Their displays are often limited in terms of resolution and physical size, which limits the level of immersion.

2.7.2 Augmenting Physical Surfaces with Projections

Projections directly augment any existing physical surface, and are a common form of output in smart spaces, where the space itself becomes the input device [Cao and

Balakrishnan, 2006; Krueger et al., 1985; Wilson and Benko, 2010]. As pico-projector technology matures and begins to appear within phones and digital cameras, an interesting and under-explored scenario is the use of handheld projection to augment such spaces.

There is a great deal of research in the area of smart spaces including systems based on cameras and other situated sensors [Brumitt et al., 2000; Kidd et al., 1999]. To help scope the related work we focus on infrastructure and infrastructure-free projector-camera systems.

Infrastructure-based Sensing

Cao and Balakrishnan [2006] and Cao et al. [2007] used a high-end commercial motion capture system to enable full 6DoF tracking of a handheld projector and stylus (see Figure 2.15). This infrastructure-based approach provides interaction techniques for both a single projector and multiple projectors. In particular, the system includes full 6DoF spatial awareness of the projector. However, scene geometry is not sensed directly. Instead the system enables the user to interactively define multiple planar surfaces in the environment. This allows the projection to be pre-distorted so that the image appears corrected on a planar surface, even when the projector is held at an oblique angle. A flashlight metaphor [Teller et al., 2003] allows virtual content to appear anchored in the real world, with content being revealed when illuminated by the projector.

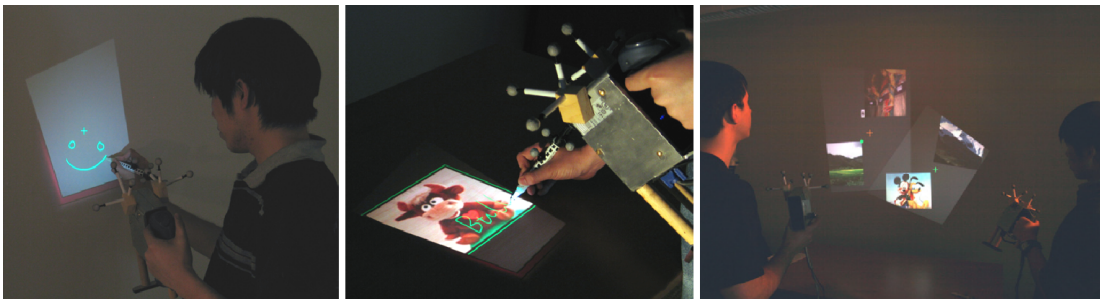

Figure 2.15: Cao and Balakrishnan [2006] and Cao et al. [2007]'s motion capture based 6DoF tracked handheld projector system and stylus-based interactions.

Intrastructure-free Sensing

Other researchers have taken a different perspective and explored infrastructure-free sensing. One common approach is to couple a camera with the mobile projector to support on-board sensing. For example Beardsley et al. [2005], Raskar et al. [2003] and Raskar et al. [2004] used calibrated projector-camera systems to detect fiducial markers attached to walls. Such markers can be used to recover the camera pose with respect to that marker, allowing the projection of a perspective-corrected graphics.

Willis et al. [2011a] took this paradigm one step further, creating a projector-camera system that simultaneously projected an invisible IR fiducial alongside the regular visible light image. This provided the ability for two devices to sense when their projections were in proximity, thus enabling new multi-projector interactions (see Figure 2.16 left).

Other infrastructure-free systems use an on-board camera to detect gesture and touch input from the user (rather than localize the device). For example, SixthSense [Mistry et al., 2009] detects finger gestures in the air by using colored markers worn on the fingers, and Hotaru [Sugimoto et al., 2005] allows touch input on the projection surface by attaching an LED marker to the finger (see Figure 2.16 middle-left).

Other systems use other on-board sensors (rather than a camera) to detect the orientation and linear and/or rotational acceleration of the handheld projector and thereby support limited spatially-aware interaction. For example MotionBeam [Willis et al., 2011b] uses an IMU to control the behavior and perspective of a projected game character (see Figure 2.16 middle-right).

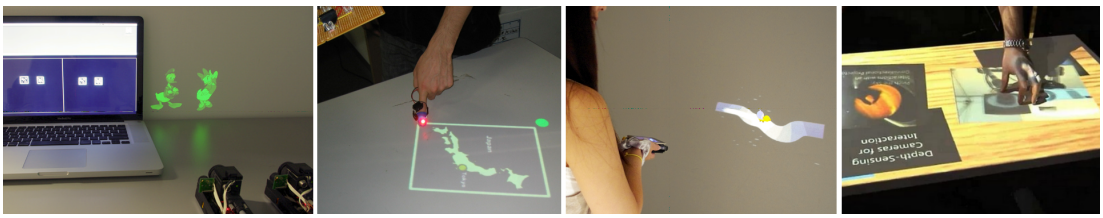

Figure 2.16: Left: Sidebyside [Willis et al., 2011a]. Middle-left: Hotaru [Sugimoto et al., 2005]. Middle-right: MotionBeam [Willis et al., 2011b]. Right: LightSpace [Wilson and Benko, 2010]

Geometry-aware

The work presented so far focuses on projector systems that are spatially aware, i.e. detect and respond to the position and orientation of a mobile projector. Ultimately, we are interested in increasing the sensing fidelity of such projectors, by enabling geometry awareness i.e. detecting the geometric structure of physical objects around the projector, including walls, tabletops as well as the user (e.g. their hands and body).

One category of system focus on geometry awareness within a single instrumented space. Examples have explored the use of steerable projectors and cameras to track and project content anywhere within the space [Ehnes et al., 2004; Molyneaux et al., 2007; Pinhanez, 2001]. More recently, LightSpace [Wilson and Benko, 2010] looks at static projection but more fine-grained sensing permitted by Kinect, to explore both on surface and in-air interactions across a wall and tabletop surface (see Figure 2.16 right). A logical progression of this type of infrastructure is to support 360-degree whole room sensing.

Other systems have explored geometry awareness in the context of mobile projection. The RFIG system [Raskar et al., 2004] can be used to detect objects in the environment in addition to walls. This allows the system to be used to augment objects with planar surfaces with overlaid digital content. Omnitouch [Harrison et al., 2011] specifically utilizes a short-throw Kinect camera coupled with a pico-projector, worn by the user, and enables touch interactions on planar objects in the field-of-view of the camera. The system does not provide spatial awareness and the geometry awareness is limited to the raw Kinect data and constrained to planar surfaces only.

Prior work has explored either spatially or geometry aware projection in the context of infrastructure-based or infrastructure-free systems. However, the combination of the two has been underexplored, and what the tradeoffs of taking an infrastructure or infrastructure-free approach are have yet to be investigated.

2.8 Discussion

In this chapter, I have covered many of the newer systems in NUI and AR. I have identified many properties of these systems that are compelling, but also the limitations

of this existing work. At a high level, whilst some work has begun to explore the intersection of AR and NUI, the current range of experiences and systems is limited.

The chapter identified the importance of the camera, as the ultimate sensor in providing the ability of both sensing the user and the environment. The work in this thesis will exploit this flexibility of sensing provided by the camera, in particular depth sensors, for making the technical challenges of combining NUI and AR more tractable. In particular, the ability to sense the geometry associated with the user and environment in real-time. However, one challenge is how to *interpret* this rich but low-level signal from the depth camera, to create UIs that are aware of the user and the space around them. I also identified a series of limitations of current depth cameras, in particular form-factor, power, and precision considerations which will be addressed in subsequent chapters.

I also illustrated the main limitations of work in this space in terms of interactive scenarios. For example, many systems do not allow for real-time processing limiting interactive use [Hartley and Zisserman, 2004; Henry et al., 2010; Kazhdan et al., 2006; Levoy et al., 2000; Merrell et al., 2007; Zhou et al., 2011]. Many NUI systems solely focus on 2D sensing, and are limited therefore when considering 3D interactions and input [Harrison et al., 2010; Kim et al., 2007; Mistry and Maes, 2009; Rekimoto, 2001; Starner et al., 1997a, 2000]. Systems that have demonstrated 3D sensing have either done this in isolation, removing the ability for 2D touch sensing. We would like to combine 3D and 2D sensing (Chapter 3), as touch still remains a powerful and ergonomic natural interaction, particularly during prolonged system interaction where mid-air interactions can lead to user discomfort and fatigue. The thesis also explores a higher fidelity of sensing than some of the coarse gestural systems currently allow [Harrison et al., 2010; Kim et al., 2007; Rekimoto, 2001; Saponas et al., 2009; Starner et al., 1997a, 2000; Vardy, 1999].

Another potential barrier to overcome is around the mobility of the user, in this thesis we explore systems that are either fixed (Chapter 3 and 4), but also ones that can support more mobile scenarios of use (Chapter 5 and 7). Finally, the coupling of environment sensing with user sensing is limited in existing work – systems predominately perform one or the other. In this thesis we explore a closer coupling (Chapter 5 and 6). I also explore the ability to more closely combine input and output spaces (Chapter 3), to bring AR affordances to the NUI arena.

Based on this deep examination of this related work. The remainder of this chapter identifies a number of requirements and design decisions when building the interactive systems that will be described.

2.8.1 Requirements and Design Decisions

User versus Space Centric

Both AR and NUI focus on leveraging the physical world in digital interaction. However, AR has a more *space centric* view, where the main focus is on sensing the physical environment around the user rather than the user themselves. A classic example are SLAM systems [Davison et al., 2007; Henry et al., 2010; Klein and Murray, 2007; Merrell et al., 2007; Newcombe and Davison, 2010; Newcombe et al., 2011] where mapping the world and estimating the pose of the device is the primary goal, rather than user sensing per se.

On the other hand, NUI is more *user centric* where the main focus is sensing the user in as much fidelity as possible [Gustafson et al., 2010; Harrison et al., 2010, 2011; Lee and Hollerer, 2007; Oikonomidis et al., 2011; Shotton et al., 2011; Starner et al., 1997a; Wang et al., 2011]. This thesis explores the interrelationship between user and space centric systems. Specifically, I will explore systems that exhibit both space and user centric properties.

Fixed versus Mobile

Another difference between AR and NUI is the *physical constraints* on interaction. AR tends to focus on *mobile* (or *infrastructure-free*) deployments where the user is moving freely within the space [Klein and Murray, 2007; Newcombe and Davison, 2010; Newcombe et al., 2011]. On the other hand, new NUI scenarios, in particular focusing on depth cameras, predominately focus on fixed *infrastructure-based* configurations, where user mobility is limited [Cao and Balakrishnan, 2006; Cao et al., 2007; Wilson and Benko, 2010]. This thesis will explore the trade-offs between these configurations. Specifically it will present NUI technologies that can be used on the move (Chapter 7), and the notion of situated spatial AR systems (Chapter 4).

Unencumbered Interactions

One of the key properties of NUI that we wish to retain is the ability to avoid heavy instrumentation of the user. This user instrumentation is potentially one of the key barriers for the adoption of AR, and arguably one of the strengths of NUI. Our systems therefore limit user instrumentation to a bare minimum, and prohibit the use of technologies such as HMDs or fully instrumented gloves.

Coupling of Input and Output

In contrast, one of the qualities of AR we aim to retain is the close relationship between the physical and virtual environment. We therefore wish to design systems that closely map these two worlds, and address technical challenges in fusing the two as seamlessly as possible.

A key design consideration when developing these types of cross NUI/AR systems is the coupling of input/output (IO). For AR systems a close coupling is crucial. For NUI, the coupling is tight in certain multi-touch systems [Bill Buxton, 2013], but IO are often decoupled in newer NUI scenarios such as leveraging depth cameras in the living room [Shotton et al., 2011]. In this thesis, I explore both tight and loose coupling.

Real-time Interaction

Given the focus on interaction, I aim to develop real-time systems, algorithms and techniques. High speed and low latency interaction are critical for achieving immersion and a suspension of disbelief [Ng et al., 2012]. This is a key requirement and adds a further technical challenge to the work described.

Fidelity of Input

Another key design aspect to explore is the fidelity of input. NUI systems have shown the benefits of touch-only interfaces right the way through to mid-air 3D input, or even physical or tangible objects [Bill Buxton, 2013; Ishii and Ullmer, 1997]. In this thesis I explore the use of many modes of interaction focusing on hands and objects, ranging from touch to full 3D tracking.

2.8.2 Overview of Technical Chapters

In the next chapters, I describe systems that address these limitations of existing work. In Chapter 3, I present a system called RetroDepth. The system combines multiple NUI modalities, including freehand touch and 3D sensing and object sensing at high precision. The system is real-time, user and space centric, and situated. It provides the lowest fidelity of IO coupling, supporting indirect interactions in a desktop setting. However, it demonstrates how a variety of NUI and AR capabilities can be combined in a single compelling system.

In Chapter 4, I move one step beyond RetroDepth with a system called HoloDesk which also allows for real-time freehand interactions between real and virtual, but in a more 'direct' way. It takes the NUI concepts presented earlier in Chapter 3 further by combining additional concepts from AR, in particular spatial registration of real and virtual worlds. HoloDesk is situated and removes the need for direct user instrumentation. It supports mid-air and physics-based interactions above a desktop.

In Chapter 5, I move beyond the fixed (or situated) NUI/AR display and explore the use of a real-time user centric and space centric SLAM system called KinectFusion. The system is mobile but with low IO coupling. The specific contributions I make to this work is allowing for NUI affordances to be incorporated in this AR/SLAM system.

In Chapter 6, I take this work in Chapter 5 one step further by coupling input and output using a handheld projector unit. This removes the need for direct user instrumentation but allows for direct user feedback. In this chapter I also explore the use of a hybrid infrastructure and infrastructure-less system based on fixed and mobile sensors, as a means to compare with the infrastructure-free system in Chapter 5.

In Chapter 7, I introduce a body-worn hand tracker that I developed that allows users to take 3D natural interactions outside the living room to outdoor spaces. The sensor is completely self-contained and recovers the full 3D pose of the user's hand without requiring an external sensing infrastructure or covering the hand itself.

Chapter 3

Natural Interactions with High-Precision Input

3.1 Introduction

As shown in previous chapters, NUI has captured the imagination of many researchers and end users alike, as they shift the paradigm of HCI away from the traditional mouse and keyboard, towards more expressive input modalities. Whilst the term is broad and can encompass touch, gesture, gaze, voice, and tangible input, as shown in the related work, NUI often implies leveraging the dexterity that the DoFs of our hands allow for interaction. Given the technical and user challenges in terms of enabling NUI through hands, this thesis will focus on this aspect alone, rather than extending the space to speech or gaze.

However, as shown in the related work, even when focusing on hands, NUI has vastly different interpretations. A decade ago, the term predominately referred to emerging multi-touch interfaces. More recently, it has become more synonymous with 3D touchless input using fingers, hands or whole body interactions, popularized by products such as Kinect and Leap Motion.

This has resulted in the emergence of many underlying technologies designed with these different scenarios in mind, each with strengths and weaknesses. For example, Leap Motion is only able to estimate fingertips of a hand in-air, but at a high precision,

where Kinect provides more flexibility in sensing dense depth maps of arbitrary scenes and tracking the human skeleton, but at a relatively low precision.

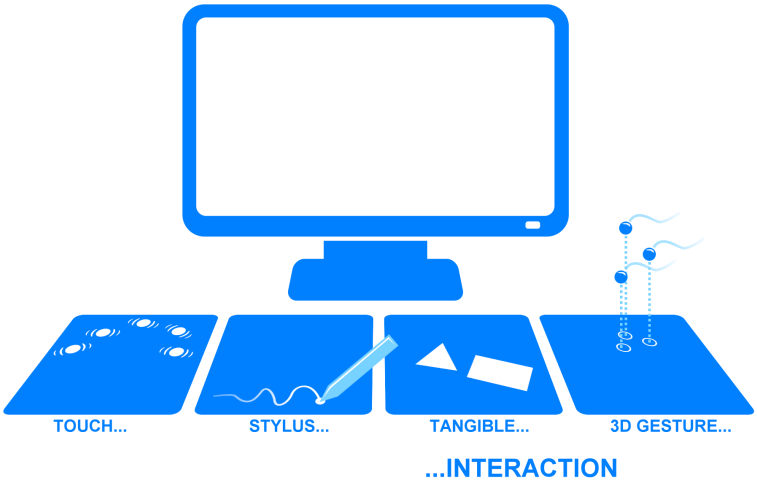

Figure 3.1: Various NUI modalities. Current systems sense touch, stylus input, tangible objects and 3D gestures via different mechanisms and a smooth transition between these modes of input cannot be accomplished.

In this chapter, I explore a new interaction scenario, where the focus lies on seamless switching between a variety of NUI modalities for high precision and expressive input, such as freehand touch, pressure, in-air interactions, as well as stylus and other tangible object input (see Figure 3.1 and Figure 3.2). In this chapter, I describe RetroDepth, a system specifically designed for interaction on and above physical surfaces of different shapes and sizes. This system is intended for prolonged and situated use in front of a display, complementing traditional desktop computing.

The RetroDepth system provides a middle ground between sensors that extract high-level features such as fingertips (e.g. Leap Motion) and more flexible depth cameras (e.g. Kinect) that expose raw depth maps but at lower precision. Our system is designed specifically to sense only the *3D silhouette* of interacting objects. The use of silhouettes greatly lowers the computation required for depth sensing as compared to computing a full dense depth map, and removes issues with performing stereo in textureless regions. Furthermore, the clean silhouettes obtained using our setup allow for extremely high precision contours.

In this chapter, I show numerous examples of how the resulting 3D silhouette information can be exploited for interactive scenarios, and is sufficiently accurate

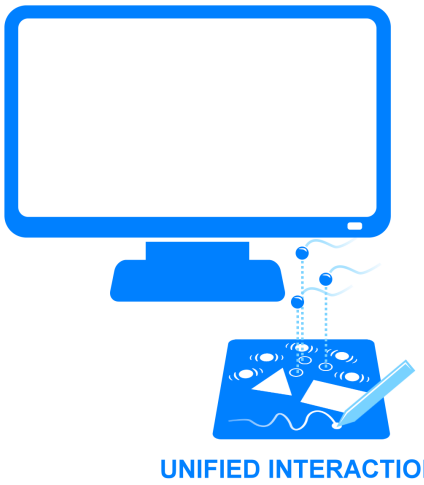

Figure 3.2: Unified NUI modalities. RetroDepth is a single sensor that allows for a wider range of NUI modalities to be combined, allowing seamless switching between these range of input capabilities.

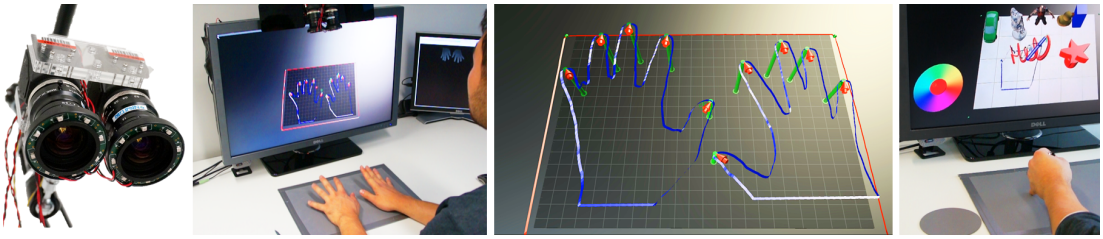

Figure 3.3: RetroDepth is a new 3D silhouette sensing system for high-precision interaction on and above physical surfaces.

enough to, e.g. distinguish between touch and hover. Interestingly, most consumer depth cameras, such as stereo cameras, Kinect or even ToF sensors, struggle most at the edges of silhouettes, where large depth discontinuities lead to noise and imprecise depth estimation. More specifically, I present a novel pipeline that tracks 3D fingers and objects, and infers hand poses and gestures, using 3D contours only. I compare our system with Kinect and Leap Motion, and conclude with a discussion of application scenarios, other physical configurations, limitations and future work.

3.2 System Overview

In the following sections I present the physical RetroDepth setup and demonstrate the interactive capabilities of the system. The physical configuration is shown in Figure 3.3. Two off-the-shelf monochrome cameras are positioned on a metal rail at a fixed baseline (6cm apart). Currently we use Lumenera PLT-425-NIR (running at 1024x512 and 60Hz) due to their IR response and global shutter capabilities, but there is no hard restriction on manufacturer or imaging sensor. A ring of 8 diffuse IR LEDs (OSRAM CHIPLED SFH 4053) operating at 850nm are attached around the lens of each camera. This placement and number of LEDs ensures that illumination is uniform and shadows are minimized. IR bandpass filters are placed on each camera lens.

This hardware setup is easily reproducible with only off-the-shelf components. The setup is powered entirely over USB (peak power for each LED is 260mW, and average power is 35mW). The LEDs are pulsed from each camera's General Purpose Input/Output (GPIO) pins, and only active during the camera exposure (2ms). A synchronization board is used to generate a 60Hz trigger to shutter the cameras simultaneously.

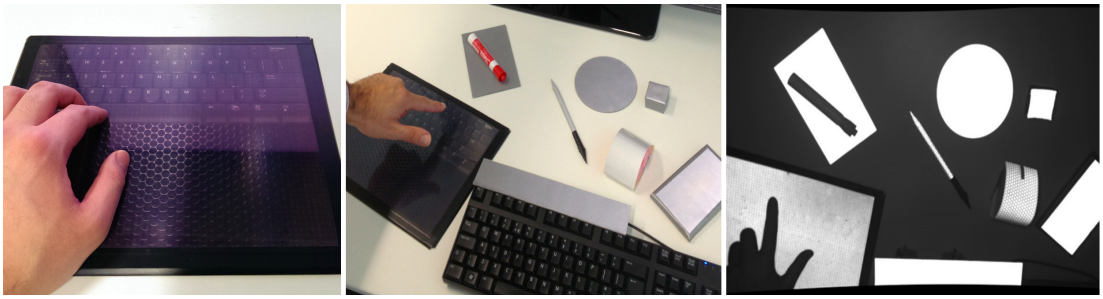

Figure 3.4: Left: passive retro-reflective keyboard, and series of other "physical widgets". Right: IR image of scene.

3.2.1 Retro-reflective Segmentation

Figure 3.4 also shows retro-reflective surfaces (such as the off-the-shelf 3M Scotchlite-Part 8910), which are cut in various shapes and sizes, and placed on a table. Upon shuttering, the cameras both simultaneously capture an image of the scene. All IR LEDs are switched on during this exposure time, producing a bright uniform response from

all the retro-reflective materials on the table, as shown in Figure 3.4 (right). This makes these reflective surfaces readily distinguishable as bright silhouettes in the captured stereo images. When objects interact on or above these surfaces, a sharp contrast is created between the background and foreground, e.g. the hand or marker pen shown in Figure 3.4 (right).

In a first pass, we extract contours of any bright silhouettes observed from each camera. We then perform stereo matching across these contour images. In a second pass, we *invert* the image within the contour boundaries, creating bright silhouettes within each retro-reflective region, which correspond to objects (e.g. hands or styluses), interacting on or above the surface (Figure 3.4). We again extract contours and perform stereo matching in these regions. This creates a clean separation between sensing the retro-reflective objects, and the 3D interactions occurring on or above them. Optionally, we interpolate to fill in a dense 3D silhouette of foreground objects.

3.2.2 Creating and Tracking Physical Widgets

Retro-reflective material is cheap and readily procurable. It is also easy to cut into various shapes and sizes. Figure 3.4 shows a variety of simple "*physical widgets*", made by cutting sheets of acrylic or cardboard into the desired shape and gluing the retro-reflective material on the upper-side. Additionally, a clear acetate sheet is used to print labels (most printer ink is invisible to IR) for the widget, such as keys for a 3D sensing touch keyboard (Figure 3.4, left).

Our stereo matching algorithm estimates the precise metric depth of the outlines of these physical widgets. After depth estimation, a machine learning algorithm classifies the widget based on silhouette shape. This classification occurs per frame in real-time and is robust to large parts of the widget being occluded by interacting hands and objects. Once classified the widget can be tracked in 6DoF. Given that new physical widgets are easy to fabricate, our machine learning algorithm is extensible in order to add new training classes at runtime.

3.2.3 Freehand 3D Interactions and Gestures

As the user's hand or physical tool interacts above a retro-reflective widget, a clear internal silhouette is visible (Figure 3.4, right). By inverting the image, our stereo

matching algorithm can precisely estimate the depth of these internal silhouettes. Even a small occluding region of the finger can be sensed accurately. This creates an *interaction volume* above each widget that allows freehand 3D input. Depending on the size of widget this can be bimanual input, single hand multi- or single finger as shown in Figure 3.5.

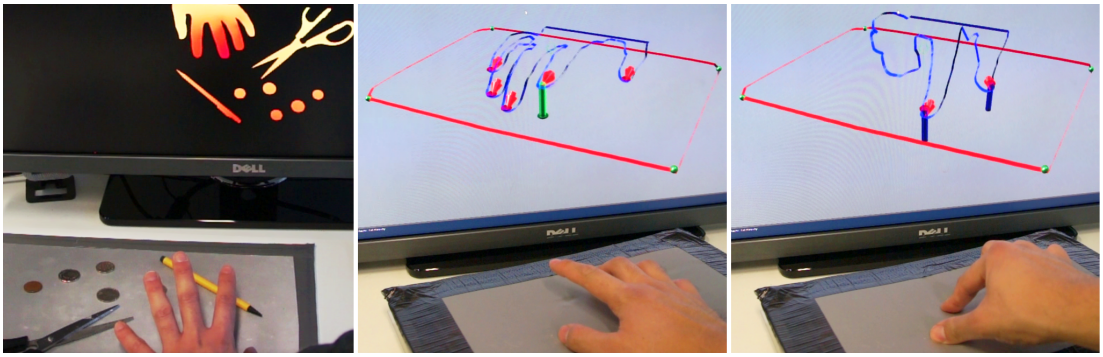

Figure 3.5: Left: Recovering the precise 3D contour of objects placed on the retro-reflector, including hands and other physical objects. Middle: Distinguishing between the widget (red contour) and the interacting objects (blue contour). Middle & Right: Sensing in-air (green), touch (red) and pressure (blue) input.

We repurpose the same machine learning pipeline to detect distinct hand shapes (i.e. when performing different gestures). Our system currently identifies a number of shapes including pointing, pinching, and whole hand interactions. Our approach can robustly handle variations across poses, and also estimate whether the left or right hand is observed.

After classifying the hand into distinct shapes, a machine learning classifier robustly identifies whether contour points belong to fingertips or the thumb, based on a curvature metric. These classified contour points can be clustered, and the individual fingertip peak identified. Simple line-fitting can be used to determine orientation of the fingertip in 3D. As shown in Figure 3.3 and Figure 3.5 this allows for robust real-time identification of the digits of the hand.

3.2.4 Stylus Input and Tangible Tools

As shown in Figure 3.5 and Figure 3.6 a variety of objects placed on top of a retroreflective surface can be clearly identified by their 3D contours. These objects could either

invoke specific UI commands, or can be used for continuous direct manipulation. One example of the latter is a stylus, which can be used for fine grained manipulations in 3D as shown in Figure 3.3 and accompanying video. During contour extraction, we look for very sharp peaks in the signal to identify the pen tip, and estimate the pen location and orientation in 3D (using line fitting).

3.2.5 Touch & Pressure

Beyond the 3D and tangible interactions outlined above, each retroreflective widget has the capability of supporting touch and pressure input purely with our depth estimation method. Given our two-pass approach, contours of each widget can be identified (and optionally classified). To determine when finger or stylus touches a flat surface, a plane is estimated based on the widget's contour and intersections with any classified tips identified. Later we evaluate the touch accuracy of our system. One interesting possibility as shown in the accompanying video and Figure 3.5 where a malleable material is used for the retroreflective widget. Here we show the precision of our contour-based depth estimation method, allowing very small changes in the 3D location of the fingertip to be identified when the user presses the surface, and using this as an approximation of *pressure*.

3.2.6 Example Physical Widgets

Figure 3.4 shows a variety of different widgets that take minutes to fabricate. As highlighted just by virtue of covering an object with retroreflective material, an interactive volume is created above the object, where touch, pressure, in-air and stylus input can be sensed easily without any computation on the widget itself. Widgets that allow for 3D touch pads, mice, and keyboards can be easily created. Sheets of retroreflector can be rolled out to enable ad-hoc 3D input devices. Widgets can be mapped to specific 2D user interface functions, such as color selection, and toolbars, as well as 3D objects which can be manipulated using touch and 3D input.

3.3 Processing Pipeline

So far we have provided a high level overview of the RetroDepth system and its sensing and interactive capabilities. In the next sections, we describe the software processing pipeline that takes pairs of synchronized images and estimates depth, and classifies widgets, hand poses and salient hand features. The pipeline is composed of fairly standard image processing tasks, followed by a new contour-based stereo and classification algorithm. We briefly discuss these initial steps before focusing on the novel components.

An example raw image from the stereo cameras is shown in Figure 3.4 (right). Here, the outline of the widgets are clearly visible. The silhouette of the user's hand is also clearly distinguishable as they touch and hover above. To detect the type of interface widget from its contour, localize it in 3D, and also determine when objects are touching or interacting directly above, we have built a new vision pipeline.

First an offline calibration process is performed which comprises of: 1) *intrinsic calibration* to compute the geometric parameters of each IR camera lens (focal length, principal point, radial and tangential distortion); 2) *stereo calibration* to compute the geometric relationship between the two cameras, expressed as a rotation matrix and translation vector; 3) *stereo rectification* to correct the camera image planes to ensure they are scanline-aligned to simplify disparity computation.

At runtime, the synchronized input IR images are captured from both cameras simultaneously. Each image is *undistorted* given intrinsic lens parameters, rectified to ensure that stereo matching can occur directly across scan lines of left and right images, and finally cropped to ignore non-overlapping parts.

The remainder of the online pipeline works in two steps. In the first step, the depth for the background image is computed. This is essentially the images where any visible widgets are brightly lit. The second step computes depth across the foreground image, where objects (within the silhouette) are segmented. Because of the sparse nature of the contour-based algorithm, we can efficiently do this two-pass approach.

We first work with the background images. We threshold both undistorted input images to remove low intensity pixels. We trace the contour of each binary image, which efficiently computes a set of 2D pixel coordinates corresponding to contour points

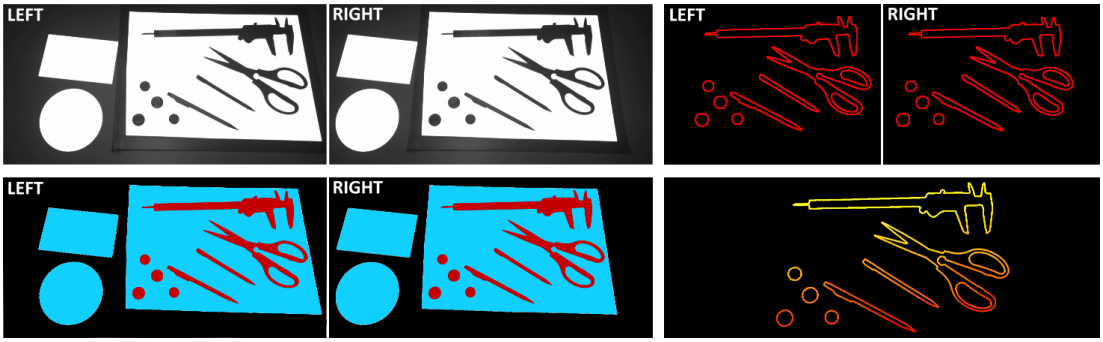

Figure 3.6: Early stages of processing pipeline. Top left: scanline-rectified stereo images. Bottom left: Foreground segmentation (in red) of internal objects within convex hull (blue). Note convex hull maps to each physical widget. Top right: extracted foreground contours. Bottom right: estimated depth of foreground silhouette (shown in false color).

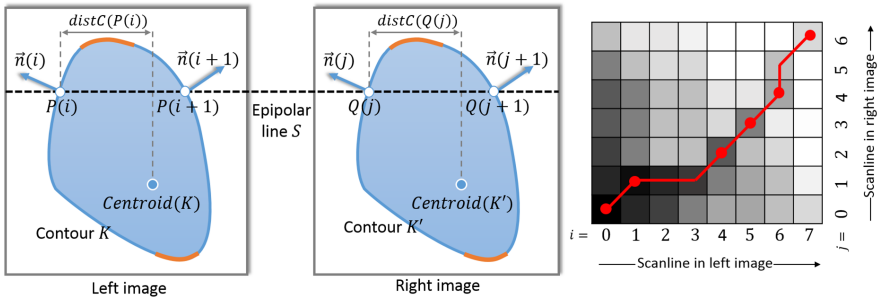

Figure 3.7: Left: contour-based depth estimation for rectified images. For notation please see text. Right: Cost Matrix $\mathcal{C}$. The brightness of each cell indicates the accumulated costs at $\mathcal{C}(i, j)$ (dark cells have low costs and bright ones high costs). The path of lowest costs is indicated in red. Matching pixels are marked with a red circle. All other pixels along the red path are occluded in the left or right image.

for each connected component. At this point, stereo is computed on the background contours to estimate the 3D silhouettes of the widgets.

What then remains is to compute the depth for internal foreground physical objects such as hands (the dark silhouettes inside the widget). We compute the convex hull of the widget's contour points for each background component, and generate a binary mask (1 within the convex hull, and 0 outside) for both camera images. For each valid pixel in the mask, we invert the binary image, and repeat contour extraction on these points. This leaves us with two pairs of contour images; one corresponding to bright regions in the input images, and the other the silhouettes (of foreground objects) within these bright regions. Finally, our efficient stereo matching algorithm is computed for the foreground. Figure Figure 3.6 shows an example of this processing pipeline.

3.3.1 Efficient Contour-based Stereo

Stereo algorithms identify corresponding points in both images that are projections from the same scene point. We refer to Scharstein and Szeliski [2002] and Brown et al. [2003] for an in-depth discussion of stereo matching approaches. Because our input images are rectified, corresponding points are known to lie on the same horizontal scanline in the left and right image (see Figure 3.7), which reduces the depth estimation to a 1D search task. The horizontal displacement is known as disparity d and is inversely proportional to depth. In the following, we focus on the key problem of determining the correspondences between contour points that lie on a particular scanline S.

A core contribution of our work is extending the standard dynamic programming dense stereo algorithms to sparse contours, which greatly reduces the amount of computation, whilst increasing precision of disparity estimation.

Let p and q be contour points in the left and right image, respectively. P and Q are lists of length $|P|$ and $|Q|$ that store the image x-coordinate of those contour points lying on scanline S in the left and right image, respectively. $P(i)$ denotes the i^{th} element in the list and we assume that the lists P and Q are sorted. For better understanding, Figure 3.7 illustrates rectified stereo images that show a contour as well as lists P and Q for scanline S.

To find correspondences between the points in P and Q, we compute the minimum-cost path through a matrix $\mathcal{C}$, illustrated in Figure 3.7 (right). $\mathcal{C}$ is of size $|P| \times |Q|$ and $\mathcal{C}(i, j)$ stores the minimum accumulated cost of an optimal path from $(0, 0)$ to (i, j). The path of minimum cost is indicated in red in Figure 3.7. Every path is restricted to start at cell $(0, 0)$ and to end at cell $(|P|, |Q|)$, because we require a mapping for all contour points. Furthermore, only three moves are possible to construct a path: a diagonal $45°$ move that indicates a match, as well as horizontal and vertical moves that represent points that are only visible in the left or right image, respectively. The restrictions imposed on the path imply the following properties of the solution:

Uniqueness property: Every (contour) point in P can only match to one point in Q and vice versa. **Ordering property:** If point $P(i)$ matches $Q(j)$ than $P(i + 1)$ can only match to $Q(j + \triangle)$ where $\triangle > 0$.

We enforce a third constraint, which we empirically found beneficial to improve robustness: Let K be the contour that $P(i)$ belongs to and K' the contour that $Q(j)$

belongs to. Then if $P(i)$ matches $Q(j)$ the height of the bounding boxes of K and K' must not differ by more than a threshold. We experimentally derived a value of 50 pixels, which worked robustly with our hardware setup.

3.3.2 Energy Function

The (accumulated) cost of a path is defined recursively (for $i > 0$ and $j > 0$) as:

$$\mathcal{C}(i,j) = \min \begin{cases} \mathcal{C}(i-1,j-1) + C_{\text{match}}(i,j) + C_{\text{smooth}}(i,j) \\ \mathcal{C}(i-1,j) + \lambda_{\text{occlusion}} \\ \mathcal{C}(i,j-1) + \lambda_{\text{occlusion}} \end{cases} \tag{3.1}$$

where the three different cases correspond to the three permitted moves discussed above. The boundary conditions are

$$\begin{aligned} \mathcal{C}(0,0) &= C_{\text{match}}(0,0) \\ \mathcal{C}(i,0) &= i \cdot \lambda_{\text{occlusion}} && i > 0 \\ \mathcal{C}(0,j) &= j \cdot \lambda_{\text{occlusion}} && j > 0 \end{aligned}$$

In Eq. 3.1, $\lambda_{\text{occlusion}}$ is a constant occlusion penalty and the remaining terms are defined as follows.

The data term C_{match} measures the compatibility of putatively matching contour points:

$$\begin{aligned} C_{\text{match}}(i,j) = &\|\vec{n(i)} - \vec{n(j)}\| \\ &+ |\text{distC}(P(i)) - \text{distC}(Q(j))| \,. \end{aligned} \tag{3.2}$$

Here, the first part measures the Euclidean distance of the normal vectors $\vec{n(i)}$ and $\vec{n(j)}$ at contour points indexed by $P(i)$ and $Q(j)$, respectively. The second part compares the horizontal distance of point $P(i)$ and $Q(j)$ to the centroid of their corresponding contours K and K', respectively: $\text{distC}(P(i)) = P(i) - \text{Centroid}(K)$.

The pairwise term E_{smooth} encourages solutions where the horizontal distance between two neighboring *matching* points $P(i)$ and $P(\varphi(i))$ is similar to the distance

of their matching points $Q(j)$ and $Q(l)$:

$$C_{\text{smooth}}(i,j) = \|P(i) - P(\varphi(i)) - (Q(j) - Q(\varphi(j)))\|, \tag{3.3}$$

where function $\varphi()$ returns the closest previous matching point for the current path.

3.3.3 Optimization via Dynamic Programming

Dynamic-programming is a well established technique to find the path of minimal cost through $\mathcal{C}$. The optimization consists of two consecutive steps. First, the cumulative costs $\mathcal{C}(i,j)$ are computed recursively for every pair (i,j) as per Eq. 3.1. At each recursion we also store the best "move" (i.e. the $\arg\min$ of Eq. 3.1) at (i,j) in a matrix $\mathcal{M}$ that has the same dimensions as $\mathcal{C}$. In the second step we reconstruct the best path by tracing back the best moves stored in $\mathcal{M}$ starting from $(|P|,|Q|)$ until we reach the origin $(0,0)$ of $\mathcal{M}$. Once the best path is computed the disparity $d(i)$ (with respect to the left image) can be derived as $d(i) = P(i) - Q(j)$ for matching points $P(i)$ and $P(j)$.

3.3.4 Post Processing

Although the estimated disparity map is usually of high quality, there might be outliers and contour points where no depth could be estimated (e.g. due to occlusion of this point in the other image). In the post-processing stage we aim to filter out wrong matches and assign *all* points along the contour to a depth value. In particular we apply the following steps:

Invalidation of horizontal lines. The depth estimation can be unreliable at contour points whose normal is almost perpendicular to the scanlines (such regions are marked in orange in Figure 3.7). This is because the term C_{match} is ambiguous in those regions. Therefore, we invalidate contour points whose normal vector has an angle of $\geq 85°$.

Outlier invalidation. The depth map obtained with dynamic-programming is computed independently for each scanline. As a consequence, the depth at neighboring contour points that lie across scanlines might be inconsistent. Thus we invalidate points

whose depth differs from those of the closest neighboring points along the contour by more than 3mm.

Contour Smoothing. We further smooth the depth values along the contour with a 1D mean filter of size 25 pixels. Note this filter can be implemented very efficiently using a sliding window technique with two operations per contour pixel.

Filling. Finally, we assign occluded and invalidated contour points to a depth value. This depth value is computed as a linear combination of the depth assigned to the closest valid contour points.

3.3.5 Discussion

One important feature of our algorithm is the underlying computational cost. In general, the complexity of a pixel-wise stereo matching algorithm can be characterized with $O(WHD)$, where W and H are the width and height of the images, respectively, and D is the number of tested depth hypotheses. In many scenarios $D = O(W)$, i.e. higher resolution images imply a larger set of depth hypothesis to be considered. Coarse-to-fine frameworks for stereo (e.g. Faugeras and Keriven [1998]; Strecha and Van Gool [2002]) and randomized stereo correspondence algorithms [Bleyer et al., 2011] (largely) remove the dependency of the runtime complexity on D. Since in our hardware setup the most discriminative regions in the image are the silhouettes, we restrict the computation of depth to pixels lying on the silhouette contours. Further, we obtain a substantial reduction in the search range D, since only a small set of extracted contour pixels in both images can be potential matches ($\sim$2-10 points).

3.4 1D Hand Pose and Fingertip Recognition

Once we have efficiently estimated the 3D silhouette, this high-quality contour contains valuable information about the scene objects, and in particular on the state and pose of any interacting hands – a key to enable truly interactive scenarios.

There are many options for detecting gestures and salient features of the hand. One simple approach detects points of high curvature as fingertip candidates (during contour

tracing). Whilst a heuristic, these types of rules can produce convincing results [Malik and Laszlo, 2004]. However, there are often failure cases, such as complex poses where knuckles, styluses or other high curvature features confuse the peak detector. To address some of these issues, we present a novel contour (1D) based hand state and part classification algorithm, which is complimentary to more regular peak detection in specific scenarios.

For hand state classification we use a random decision forest (RDF) [Breiman, 2001], which has been shown to be effective for both body [Shotton et al., 2011] and hand [Keskin et al., 2012] pose estimation using depth images. These approaches are trained to be invariant to pose and shape variations, use scale-invariant features, and achieve a robust result by averaging over multiple pixels. For RetroDepth, we adapt these approaches to silhouettes, keeping their benefits, while further reducing the complexity from 2D to 1D.

3.4.1 RDF based Contour Classification

As a key contribution of this work, our classifiers operate on 1D contours instead of 2D images. These contours are essentially 1D sequences of 3D points in world space, sampled with a rate determined by the camera resolution and setup. For notational convenience, and without loss of generality, we will assume that the contours can be parameterized by a variable s, such that $X(s)$ gives the world coordinates of the point s on the curve, and a unit step in s on the 1D sequence corresponds to a unit step along the contour in 3D world space.

Each point on the contour is associated with two class labels y^{s} and y^{f}, where y^{s} is a hand shape label (such as pointing, pinching, grasping or open hand) and y^{f} is a fingertip label (such as index, thumb, or non-fingertip). The task of the RDF is to classify the hand shape for the entire contour, and localize and identify the fingertips.

Decision trees use the internal split nodes to test and guide the input to one of the leaves, where class distributions of the output labels are stored. Test functions at split nodes are of the form: $f(F) < T$, where the function f maps the features F onto a line, and T acts as a threshold. We use the following test function:

$$f(s, u_1, u_2, \vec{p}) = [X(s + u_1) - X(s + u_2)]_{\vec{p}} \tag{3.4}$$

Figure 3.8: Left: For a given point s on the contour, we move by the offsets u_1 and u_2 to reach points $X(s+u_1)$ and $X(s+u_2)$. Middle: Hand shape class label y^{s} (one shared label for all pixels). Right: Fingertip class labels y^{f}.

where $[.]_{\vec{p}}$ is a projection onto the vector $\vec{p}$, and $\vec{p}$ is one of the primary axes $\vec{x}$, $\vec{y}$ or $\vec{z}$. This test probes two offsets on the contour, gets their world distance in the direction $\vec{p}$, and this distance is compared against the threshold T. See Figure 3.8 for an example. Because the parameterization of s is normalized, the offsets u_1 and u_2 are scale invariant. The test function splits the data into two sets and sends them each to a child node. The quality of a split is determined by the information gain defined as follows:

$$G(u_1, u_2, T) = H(C) - \sum_{s \in \{\mathrm{L,R}\}} \frac{|C_s|}{|C|} H(C_s) \tag{3.5}$$

where $H(C)$ is the Shannon entropy of the class label distribution of the labels y in the sample set C, and C_{L} and C_{R} are the two sets of examples formed by the split. At training time, multiple features and thresholds are uniformly sampled from a large range for each split node, and the one with the highest information gain is selected. Once leaves are reached, the class distribution of the remaining pixels in the node are saved. As each pixel has two labels from distinct sets in our case, we keep two histograms at each leaf: one for hand shape and one for fingertips.

At run-time, every pixel is evaluated independently with each tree in the forest and they descend into one of the leaves. For hand shape classification, the associated distributions read from the leaves are pooled across the contour to form a final set of state probabilities. The mode of this distribution is selected as the hand shape for the contour X. The fingertip localization algorithm is instead more akin to the part classification method of Shotton et al. [2011]. After classification, we apply a 1D running mode filter to the labeled contour to filter out the noisy labels. We finally apply

connected components to give labels to the fingers. The system selects the point with the largest curvature as the fingertip.

To train a single forest that jointly handles shape classification and fingertip localization, we first disregard the y^{f} labels and calculate the information gain only using y^{s} until we reach a certain depth m. From then on, we switch to using y^{f} labels instead, and do not use y^{s} to select features. This has the effect conditioning each subtree that starts at depth m to the shape class distributions at their roots. This is similar to the idea presented in Keskin et al. [2012], in that the low level features are conditioned on the high level feature distribution. However, instead of using a multi-layered approach, we handle the classification jointly in a single forest.

We use a realistic synthetic hand model to automatically generate high-quality hand contours with labels. The hand model has 32 DoF, which can be used to automatically pose the object in a range of parameters, with added random noise on every joint angle. We generate a total of around 8000 left-hand images for each hand state class. These are then mirrored and given right hand labels. Fingertips are labeled by mapping the model with a texture that signifies different regions with separate colors. An example of the hand shape and fingertip labels for a certain pose can be seen in Figure 3.8.

3.5 Classifying other Object Contours

Our method for hand posture classification can also be leveraged to classify retro-reflective widgets. Even though the shapes are mostly primitive shapes e.g. rectangles and circles, applying template matching is not optimal. This is mainly because the contour of the object changes while the user is interacting with it, and also because for every type of object, a new set of heuristics or rules need to be generated. An off-the-shelf RDF on the other hand can learn all the variation of the contours for every shape. It is accurate and fast, and can learn to recognize new objects at run-time.

To reduce the variation due to occlusion, we first estimate the convex hull of the objects and evaluate this simplified contour. We use real data for training, since labeling the contours is straightforward in this case. The same features and objective function are used as in Equation 3.4 and 3.5. As the contours are much simpler in this case, much shallower forests are sufficient for online training.

3.5.1 Recognizing a Stylus

Pens are typically much sharper than fingers and therefore easier to recognize with heuristics. On both small objects where we have partial hand contours, as well as on larger surfaces where we have the full hand contour, finding the sharpest point on the contour (and thresholding it) is a good estimator for the pen tip location. However, the pen changes the contour of the hand shape considerably, which reduces the accuracy of the hand state classifier and causes the fingertip localizer to fail. Therefore, when a pen is detected on a large surface, the pen contour points are first removed from the contour. This can be done by following the pen contour in both directions until the gradient of the contour changes abruptly. The corresponding points are then connected to form the hand contour.

3.5.2 Discussion and Results

When users interact with smaller widgets, where only a few fingers or the pen is visible, hand state recognition is not desired as we have only a partial hand contour. In these cases, the pen and fingers can be found by simple peak detection. When the hand is on the large rectangular surface however, simple heuristics fail to identify or localize fingers robustly (e.g. finding peaks at knuckles). Examples that would very likely be failure cases for standard heuristic-based approaches such as detecting pinch [Wilson, 2006] or fingertips [Malik and Laszlo, 2004] are shown in Figure 3.9. Whilst these reinforce the need for a machine learning approach, our method is also complimentary to more heuristic-driven approaches.

We conducted experiments with eight hand states, corresponding to left and right handed versions of four hand poses, namely “pointing”, “pinching”, “open hand looking down”, and “open hand looking up”. For each hand state we have 4000 synthetic left hand images, which are mirrored to retrieve data for the right hand. We train on half of the samples and test on the remaining ones.

We train an RDF with a single tree until depth 18 with the state labels only, and calculate the confusion matrix over all hand states. The RDF reaches perfect accuracy for both left and right handed versions of three of the command modes, and only shows confusion between the left– and right–handed pointing gestures. The success rate of left–handed pointing is 93% and right–handed pointing is 92%, and the corresponding

Class	Thumb	Index	Middle	Ring	Pinky	Reject
Accuracy	0.848	0.754	0.615	0.594	0.796	0.882

Table 3.1: Per-pixel classification accuracy for the individual fingertip classes, before filtering.

Figure 3.9: Examples of results based on complex poses and visually ambiguous poses. Ground truth data shown left and test results shown right. Poses shown from left to right: two pinch gestures, a splayed hand with an occluded finger, and fused fingers on a splayed hand.

left-right confusion is 7% for the former and 6% for the latter case. This gives us an overall command mode classification accuracy of 100% and left-right hand classification accuracy of 98% when all states are considered.

The synthetic images also have labels for the five fingertips and the reject class, corresponding to the non-fingertip areas. After the state classification training concludes at depth 18, we continue training with the fingertip labels until depth 25. The leaves are assigned the distributions of both state and fingertip labels. The success rates are given in Table 3.1. Even though we are operating on 1D contours, with much lower computation, the per-pixel classification rates are similar to the reported results in Shotton et al. [2011] and Keskin et al. [2012].

3.6 Contour Inpainting

So far we have purposefully pushed the limits of what can be achieved purely based on silhouettes. However, there are certain scenarios and legacy applications that require fully dense depth maps. In this section, we show a robust and fast method for inpainting the contour of objects. The results of our contour inpainting algorithm can be found in Figure 3.10.

In order to fill the interior Ω of our segmented contour $\partial\Omega$, we formulate the interpolation problem as a Laplace equation with Dirichlet boundary condition. Hence, contour depth values define a known scalar-valued boundary function $\hat{f}(x,y)|_{\partial\Omega}$ and we need to solve for the interior depth values $f(x,y)$ over Ω with:

Figure 3.10: Contour inpainting results in false color.

$$\begin{aligned} \Delta f(x,y) &= 0 \qquad && (x,y) \in \Omega - \delta\Omega \\ f(x,y)|_{\partial\Omega} &= \hat{f}(x,y)|_{\partial\Omega} && otherwise \end{aligned}$$

Then, we use a parallel Jaccobi/Gauss-Seidel solver on the GPU to solve the Laplace equation. In order to speed up convergence, we initialize the interior using a hierarchical pull-push interpolation with bi-cubic interpolation weights [Grossman and Dally, 1998; Marroquim et al., 2007]. Note that we mark pixel values of $\partial\Omega$ with a negative sign, which allows us (except for the pull-push hierarchy) to solve the system in place. The contour inpainting results can be seen in the supplemental video.

3.7 Experiments

We now evaluate the performance of our system compared to three baseline measures. As a first baseline, we choose Leap Motion (**Leap**) given the widely publicized precision of the system, and its availability as a commodity sensor. The other baselines are based on another, arguably more ubiquitous depth sensor: the Kinect. With this sensor, we chose to run two conditions. The first (**KinectLight**) uses the RetroDepth hand posture and fingertip tracking pipeline described previously, and uses contour-traced Kinect depth maps as input (instead of RetroDepth stereo data). This in part evaluates our software pipeline with a commodity sensor, but also gives us an indication of the precision of the stereo matching algorithm and hardware setup that the full RetroDepth system supports. The second Kinect baseline (**3Gears**) evaluates the hand tracker from 3Gear systems [3Gear Systems Inc, 2013]. This can be considered the current widely-available state-of-the-art hand tracker for Kinect with scenarios close to RetroDepth.

Figure 3.11: Study setup. Left: Kinect, Leap and RetroDepth setups for 3D acquisition task. Middle: Kinect touch evaluation. Right: Leap touch evaluation.

3.7.1 3D Target Acquisition

In our first task, we wanted to evaluate one of the core features of RetroDepth, its 3D input precision. For all conditions, the same physical setup was used as shown in Figure 3.11. This provided a physical input space of 30x30x30cm. Leap was placed on the table, Kinect and RetroDepth mounted top down. All cameras were configured to share the same center of origin. A large table was used with retro-reflective material visible for the RetroDepth condition. A large display was used for all tasks. This material was removed for the other conditions to avoid IR interference. A PC with a 6-core 2.8GHz processor and Nvidia GTX480 GPU was used for all conditions.

Participants Twelve participants (9 male, 3 female) between the ages of 21 and 39 were recruited to participate in the study. Participants were daily computer users, 2 were left handed. All had normal vision. Participants had no experience of using 3D hand gesture-systems, but some had experiences playing Kinect games.

Task We conducted a 3D targeting task where the physical interaction volume was subdivided into 27 equally sized cells (3x3x3 grid). The user begins by holding their finger in a 3D transparent box appearing centered and at the front of the 3D scene. After a fixed time, a billboarded target shown in Figure 3.11 (left screen) appeared centered in one cell for users to select. 3 trials were performed per condition. Starting and target positions were coupled a priori to reduce in-condition variance. This set was

used across all conditions, with the order randomized, ensuring that the total distance traveled was exactly equal across all users, and each condition. Measurement of task completion was triggered once the fingertip left the starting position and ended once the user entered the target triggering area. The targets were 5x5cm high.

Procedure The experiment employed a within-subject repeated measure design. The independent variable was the input type: RetroDepth, Leap, KinectLight, 3Gears. The dependent measure was the targeting accuracy (measured as a distance of the fingertip to the target center upon selection). The presentation order of the conditions was counterbalanced using a Latin Square design across users. A short training phase was first performed, after which each user was asked to perform the task as accurately as possible. The entire experiment lasted about 60 minutes. Participants filled out a post-experiment questionnaire upon completion. Users were observed at all times as the task were performed, and notes on subjective experiences taken.

3.7.2 Results

Our 12 participants produced a total of 3888 selections in this task. No effect was found between the four blocks of trails (e.g. from fatigue or learning effects). Additionally there was no significant differences between participants. This allows us to combine data to cluster per condition results as shown in Figure 3.12 (right). The plot shows the average of the target hits across all participants, clustered by condition. Note we ignore the Z contribution as the targets were planar, and depth is measured along the Y axis in our 3D scene. The more compact the cluster the more accurate the result. 95% confidence ellipses are shown for each condition.

We compute the minimum target sizes required to achieve 95% accuracy for each condition. RetroDepth achieves a minimum target size of 1.95x1.55cm with error X=7.89mm and Y=10.90mm. 3Gears achieves a minimum target size of 5.30x3.15cm with error X = 20.86mm and Y=29.74mm. KinectLight achieves a size of 2.89x2.08cm with error X= 15.24mm and Y=24.87mm. Leap performs similar achieving a size of 3.18x2.00cm with error X=17.53mm and Y=22.67mm.

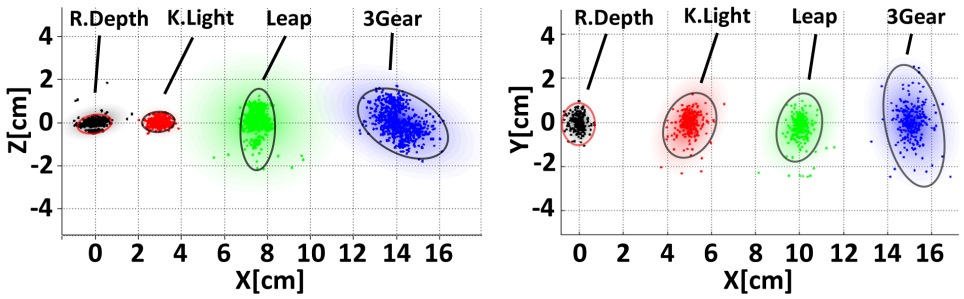

Figure 3.12: Left: touch targeting results, and right: 3D acquisition targeting results. Black=RetroDepth, red=KinectLight, green=Leap, blue=3Gear.

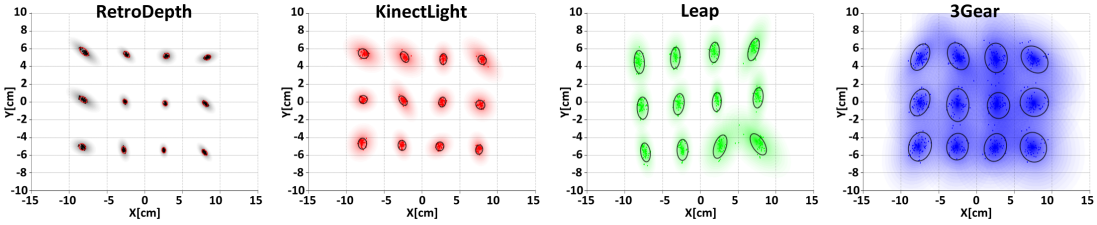

Figure 3.13: 2D touch results. Black=RetroDepth, red=KinectLight, green=Leap, blue=3Gear.

3.7.3 Physical Touch Performance

Another important feature of RetroDepth is the ability to support touch sensing. To evaluate the performance of the system against our baseline, we ran a further experiment, where each camera (Kinect, Leap and RetroDepth) was setup in top down arrangement Figure 3.11 (middle and right). For Leap this meant adding a sheet of IR absorbing film to the table. A 20cm×25cm acrylic sheet was used as a touch surface, with small laser etched targets (spaced 5cm apart), and sized for the user's fingertips. 12 participants, 3 women and 9 men, were asked to perform a physical touch task. The aim was to measure system performance in terms of touch accuracy, and overall sensor precision. Participants were asked to touch all 12 targets in turn over 5 trials. The 3D point was recorded once the user's self reported. Figure 3.13 shows the X-Y 2D plots for each of the sensors. Figure 3.12 (left) shows the Z-Y (in this case the Z axis corresponds to depth precision). Based on these statistics it is possible to define the minimum 3D target bounds to capture 95% of the touches. For RetroDepth this is 1.69x0.98x0.75cm, KinectLight is 1.49x1.29x0.85cm, Leap is 3.88x2.91x 1.63cm, 3Gear is 4.41x3.71x2.78cm.

3.7.4 Discussion

In general we found the results extremely encouraging given the level of precision of RetroDepth compared to these baselines in particular. It is however important to note that we are not comparing like-for-like necessary. In terms of fidelity, for example, Leap uses lower cost cameras, and we use more costly research cameras. Nor was the Leap designed for touch-like scenarios (i.e. pointing downwards). Although we feel that the touch accuracy is good enough to warrant future work on this scenario, based on our IR absorbing idea. On the other hand, RetroDepth is a research prototype whilst others are engineered products. Our system also offers different data (silhouettes) than the Leap and the Kinect, and none of these elements are being measured. However, all this in mind our results are encouraging in terms of precision and performance.

An interesting finding is how well Kinect performed in the KinectLight condition, particularly for touch sensing. This illustrates some benefits of the software pipeline we have implemented for improving touch and 3D acquisition. Another interesting finding is that user's perceptions and performance can be very different. Leap surprisingly did not perform as well as expected, but yet it was ranked a close second to RetroDepth in terms of preference. Many users described it as fluid, which is also an observation that frame-rate has a strong impact in terms of NUI experience, potentially even distorting views of sensor accuracy. The 3Gear performance also gives an interesting insight. Whilst the system appeared to be performing well, showing compelling 3D models fitting the Kinect data, this model fitting approach ultimately led to severe inaccuracies for touch and selection, as there will always be discrepancies between a generic model and the user.

Overall we feel that RetroDepth offers an exciting new research platform for future work on NUI interfaces, leading to new applications that incorporate touch, pressure, 3D input, and even tangible input. There are of course limitations with our approach. Ultimately, 3D silhouettes are a lower dimensional representation of the real world, and in that regard data is lost. The interesting finding from our experiments is just how much information is encoded in the silhouette allowing high accuracy rates for hand pose and part classification. Another clear finding is around frame rate, where user feedback suggests that trading some precision for speed may be desirable. At the moment, the main limiting factor is the camera frame rate rather the software pipeline.

3.8 Summary

In this chapter I have presented RetroDepth a system that allows for high-precision 3D sensing on and above physical surfaces. This has moved beyond many of the limitations of existing depth cameras (in particular the lack of precision and object boundaries) and NUI systems, which currently lack the combination of touch, 3D, and object sensing.

I have provided several key contributions: 1) A fully working real-time 3D silhouette sensing technology which combines (for the first time) the common NUI modalities of touch, pressure, stylus, objects, and 3D, into a single system. 2) a new stereo matching algorithm for computing fast and precise 3D contours. 3) a new 1D contour classification algorithm, for robustly identifying physical tools, hand poses and salient features. 4) A software pipeline with practical impact even for regular depth sensors like Kinect.

Whilst RetroDepth presents a powerful range of interactions, the biggest limitation of our system is that it is focused predominantly on NUI interactions with IO that is decoupled. In the next section, we bring in AR capabilities to these types of NUI systems. Specifically, I describe a system that more closely couples IO, and overlays the virtual directly on the real world, but carries some of the important features of RetroDepth in that the user interaction is unencumbered and situated.

Chapter 4

Direct Interactions in Mid-Air

4.1 Introduction

Chapter 3 presented a novel approach for combining various input modalities with a single new depth sensor. However, the focus was less on drawing upon qualities from AR in this system. In this chapter I investigate a way for more directly coupling 3D input with output that allows for a single 'walk up and use' situated AR/NUI display. A further contribution of this chapter lies in moving away from the pre-programmed gestures and interactions, as used in the previous chapter (see Chapter 3). Instead, in this chapter, I explore a physics-based interaction model that has been only applied to 2D or 2.5D tabletops previously, allowing for non pre-programmed physical interactions with digital content.

As highlighted so far, interactive touch surfaces allow for direct coupling of 2D input and output that more convincingly simulate the manipulation of real objects when compared to conventional input devices. However, this coupling is lost when input is extended beyond the flat surface. Situated AR [Bimber and Raskar, 2005] brings together research on AR and interactive tabletops. These systems enable interactions with physically situated displays that render 3D graphics aligned with the real-world. Whilst these systems couple input and output in interesting ways, they have yet to demonstrate realistic freehand interactions with 3D graphics *without* user instrumentation; requiring either head-worn or handheld input and output hardware.

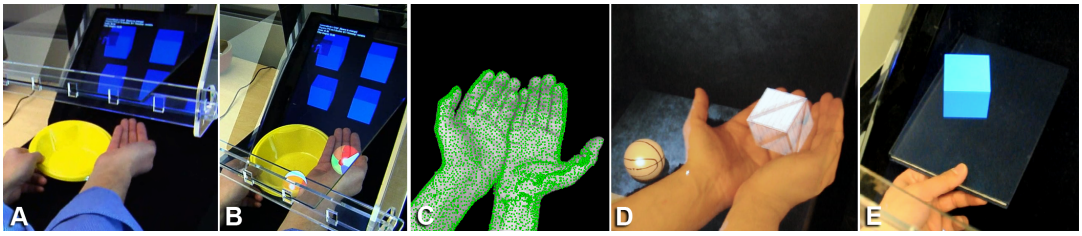

Figure 4.1: HoloDesk allows direct freeform interactions with 3D graphics, without any body-worn hardware. A + B) User sees a virtual image of a 3D scene through a half silvered mirror. Scene is corrected for the viewer's perspective. User can freely and directly reach into the 3D scene to interact with it. C + D) A novel algorithm is presented that allows diverse and unscripted whole-hand 3D interactions e.g scooping and grasping. E) Other real objects beyond hands can be used for interaction.

In this chapter, I present HoloDesk , an interactive situated AR system that combines an optical see-through display and Kinect camera to create the illusion that users are directly interacting with 3D content. Here, we extend dexterous interactions with our hands above the tabletop to provide a new technique for interpreting raw Kinect data, demonstrating fine-grained tracking of rigid (books, cups) and non-rigid (hands, paper) objects over time and in 3D. This allows users to get their hands literally 'inside' the display and directly touch 3D graphics *without* any body-worn hardware. The proposed technique correctly approximates collision and friction forces enabling a large set of emergent hand-based gestures including sweeping, scooping, lifting and throwing virtual objects. In particular the input representation simulates natural human grasping of virtual objects with more fidelity than previously demonstrated.

I describe the HoloDesk implementation in full, including the hardware configuration, system calibration, and GPU computer vision and rendering pipeline. I illustrate usage scenarios where HoloDesk could be applied including gaming, rapid prototyping and remote conferencing. Finally, I empirically evaluate HoloDesk in two further user studies, comparing direct and indirect instantiations of the system, and contrasting the 3D depth cues of our system with the additional use of stereo glasses. This draws out the importance of IO coupling, and the ability of moving away from user instrumentation for interaction.

4.2 System Overview

Our current physical configuration for HoloDesk is illustrated in Figure 4.2. A desktop sized interaction volume is viewed by the user through an optical see through mirror (Figure 4.2, A). This mirror (referred from now on as a beamsplitter) reflects light towards the user from an LCD display mounted above the mirror. This forms a virtual image that the user views overlaid onto the interaction volume. This volume is within easy reach of the user's hands and is also not physically blocked by the displayed image. This allows the user to literally place their hands 'inside' the display and see spatially coupled output. The LCD is angled away from the user to ensure that the viewable area of the virtual image is maximized to the user.

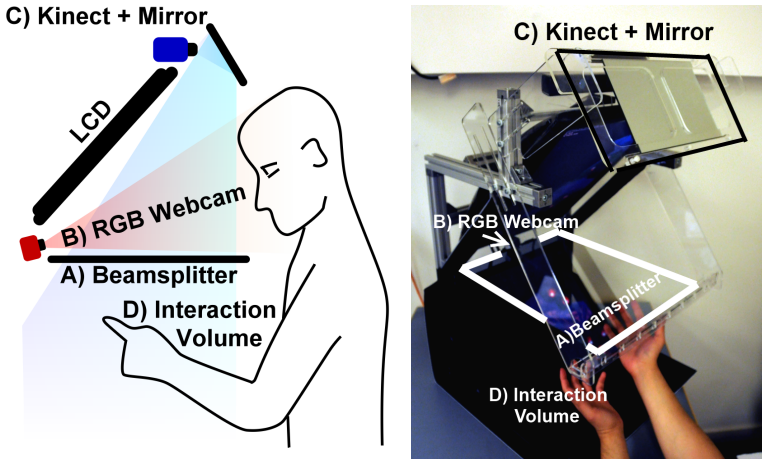

Figure 4.2: Physical setup of our current HoloDesk prototype with main components labeled.

When looking through the beamsplitter our system ensures that the user views virtual graphics correctly registered on top of real objects in the interaction space (Figure 4.2, D). To achieve this an RGB camera (Figure 4.2, B) is used to track the 6DoF pose of the user's head. By continuously estimating the 3D position and constantly updating the rendering of the 3D scene, the correct perspective is displayed to the user. This also creates motion parallax effects allowing users to look behind 3D objects to reveal occluded parts of the scene.

Our setup creates an interaction volume where users can view a 3D scene, spatially aligned to the real-world. Users can also freely move their hands or any physical object within this volume, without the display causing an obstruction, allowing the user to get their hands into the rendered scene. A Kinect camera (Figure 4.2, C) is mounted

above the LCD. A mirror is used to fold the Kinect's optics allowing it to sense the full interaction volume while limiting the overall height of the setup.

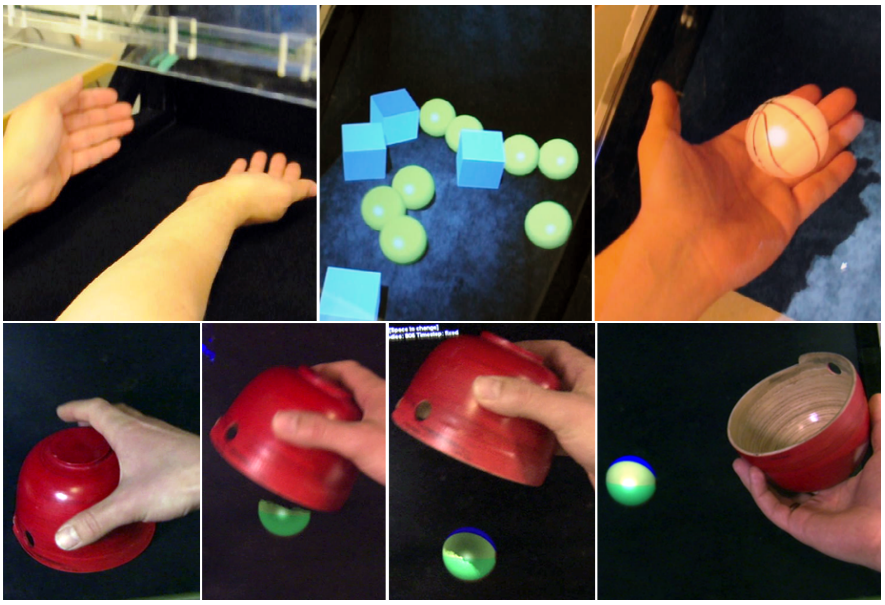

Figure 4.3: Top: User's hands in the interaction area. 3D scene as seen by the user. Virtual sphere resting on hand. Bottom: Real object occludes virtual, casts shadows.

We leverage the real-time depth data from Kinect to add further realism to the rendering of the virtual scene. The depth data allows the modeling of *occlusions* of virtual objects by the users hands, as well as correct *inter-shadowing* behavior (Figure 4.3, bottom-right), further enhancing the coupling between real and virtual worlds. Beyond creating an immersive effect these rendering techniques provide the user with strong depth cues aiding depth perception.

The Kinect is also leveraged to make the spatially aligned graphics *interactive*. As described later, techniques are presented for both real-time tracking of the user's hands and other physical objects within the interaction volume, without any markers or bodyworn sensors. A physics-based representation enables hands and other objects to realistically interact in 3D within the virtual scene. Users can scoop objects from underneath and balance them on their palms, use their full hands to push objects around, juggle objects or perform more advanced interactions such as grasping (see Figure 4.3). Dexterous and bimanual interactions allow users to combine coarse interactions, such as sweeping, with more accurate manipulation, such as grasping and rotation.

4.2.1 Application Scenarios

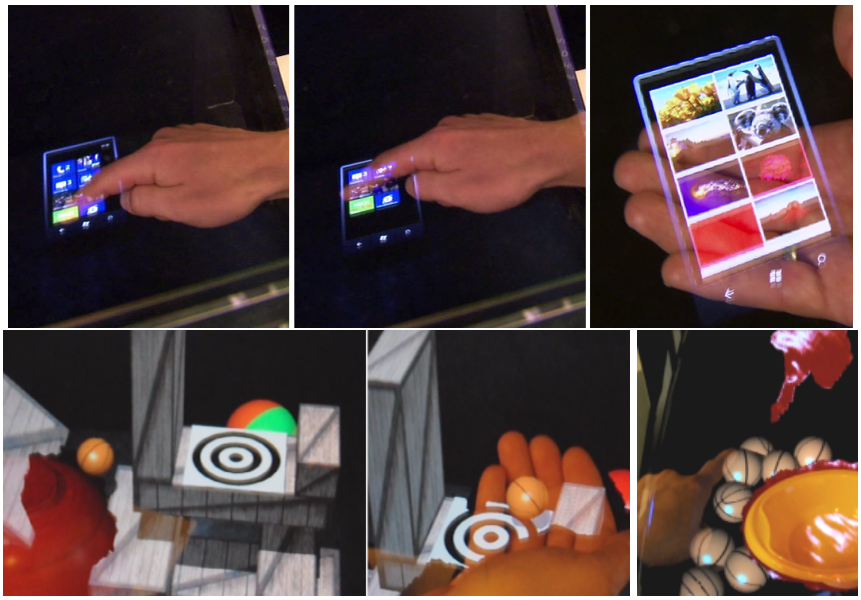

Figure 4.4: Application scenarios. Top: Interacting with a virtual prototype. The UI changes as user touches screen. Prototype can be picked-up. Bottom-left: Virtual and real objects are fused in a physics-enabled game. Bottom-Right: Remote users point to virtual object.

HoloDesk supports many application and interaction possibilities. One of the unique strengths of HoloDesk is the ability for users to rapidly experience a seamless mix of real and virtual content. This interplay leads to interesting tangible gaming possibilities. For example, Figure 4.4 (bottom) shows how physical and virtual objects are fused in a physics-based game, a user guides a virtual ball through obstacles containing both virtual and real slopes, bridges and holes. Other games such as chess and boardgames can be augmented with digital content to enhance gameplay.

The system can also be utilized by designers to rapidly experiment with 3D models and physical prototypes they are creating. Although our system lacks haptic feedback it can still give the user a sense of a 'virtual' prototype's shape and size in their hands (Figure 4.4, top). These virtual prototypes can also overcome certain physical constraints, to rapidly explore new form factor designs e.g. a smartphone that can be stretched to different sizes, including a tablet form factor. Our system tracks the user's hands allowing such virtual prototypes to become 'touch'-enabled (Figure 4.4, top).

Another application scenario for HoloDesk is telepresence (Figure 4.4, bottom) Here interactions within the volume of one HoloDesk are captured and relayed in real-time

to a remote user at another unit. Users of both systems share a single virtual 3D scene, viewed from different perspectives. The hands of the remote participant provide a visual awareness cue, augmenting traditional A/V conferencing features.

4.2.2 Physics-Enabled Interactions

Many of the application and interactions described so far leverage HoloDesk's ability to spatially couple virtual onto the real. However, this interplay between physical and digital can be taken further than just rendering – allowing the real to *interact* with the virtual in realistic ways. We present a GPU-based algorithm that processes the Kinect depth data in real-time, tracks hands and other objects in a fine-grained manner, and represents these in a 3D physics simulation. This algorithm extends the 2D physics representations of Wilson et al. [2008] enabling physically realistic dexterous *3D input* within a virtual scene. This approximates the range of motion and dexterity our hands and real-world interactions exhibit. As shown later this enables many different emergent interactions between real and virtual (Figure 4.5, middle).

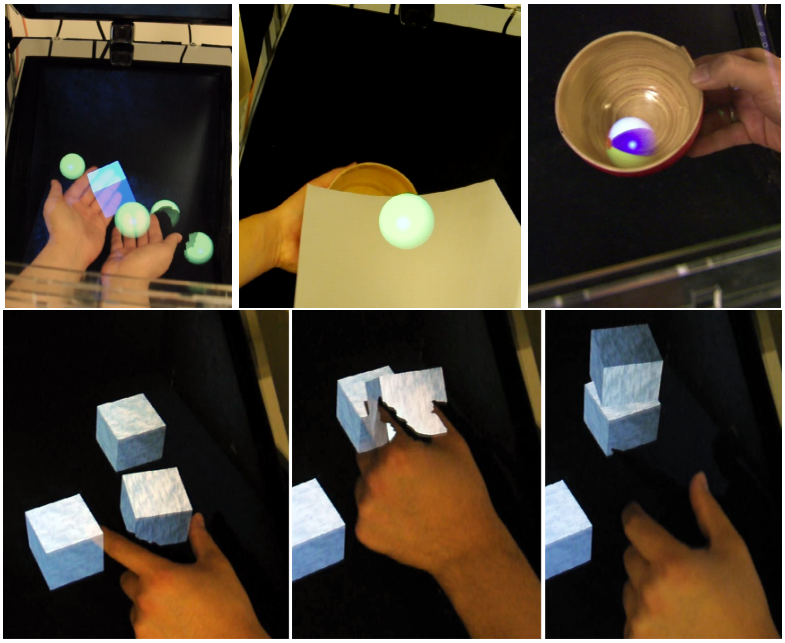

Figure 4.5: HoloDesk interactions. Top-Left: two-handed scooping of virtual spheres. Top-Middle: Sphere rolling on deforming sheet of paper. Top-Right: Virtual sphere moves with physical bowl. Bottom: Grasping virtual objects.

The proposed model allows users to perform rich free-form 3D interactions such as juggling (Figure 4.5, left). The system approximates the shape of objects in the interactive volume but also the deformation and motion over time of these objects, based on a depth-aware optical flow algorithm. This technique allows real objects to exert friction forces onto virtual objects allowing for interactions such as natural grasping (Figure 4.5, middle), something not previously demonstrated.

4.3 System Implementation

4.3.1 Calibration

Given our aim to support walk-up-and-use scenarios we have decided against the use of stereoscopic imagery. Instead head-tracking is used to create viewpoint corrected renderings of the virtual scene, making motion parallax one of the primary depth cues in our system. To guarantee tight spatial coupling between input and output we need to calibrate the Kinect, the head-tracking camera, and the virtual image plane relative to each other and a fixed real-world origin. Whilst non-trivial, this calibration is a one-off procedure as the setup has no moving parts, and aspects of the process can be automated.

The Kinect camera contains two separate cameras, an IR camera to capture depth data and a regular RGB camera. We use a standard checkerboard method [Zhang, 2000] to retrieve the intrinsic calibration matrix $\mathbf{K}$, as well as the 6DoF transform $\mathbf{T}$ (containing a 3x3 rotation and 3D translation vector) for each camera. The latter specifies the extrinsic pose of the cameras relative to a single fixed real-world origin within the interaction volume. A virtual checkerboard is displayed on the LCD, and imaged by the Kinect RGB camera to determine the pose of the screen relative to this camera. Since the pose of the Kinect RGB camera is also known relative to the world origin, the pose of the LCD screen relative to the origin can be computed. The RGB webcam (used for head tracking) is calibrated by placing a physical checkerboard pattern orthogonal to the XZ plane of the earlier defined world origin.

We use the OpenCV face tracker to determine the 6DoF pose of the user's head relative to the RGB webcam. We can now define a correct perspective off-center

projection to produce viewpoint corrected renderings of the 3D scene (Fig. 4.3). As the head pose changes, rendered objects at different depths exhibit correct motion parallax.

4.3.2 Core GPU-based Pipeline

For blending of virtual and real with minimal latency and interactive framerates, a GPU-based processing pipeline (Figure 4.6) is implemented in HLSL and CUDA.

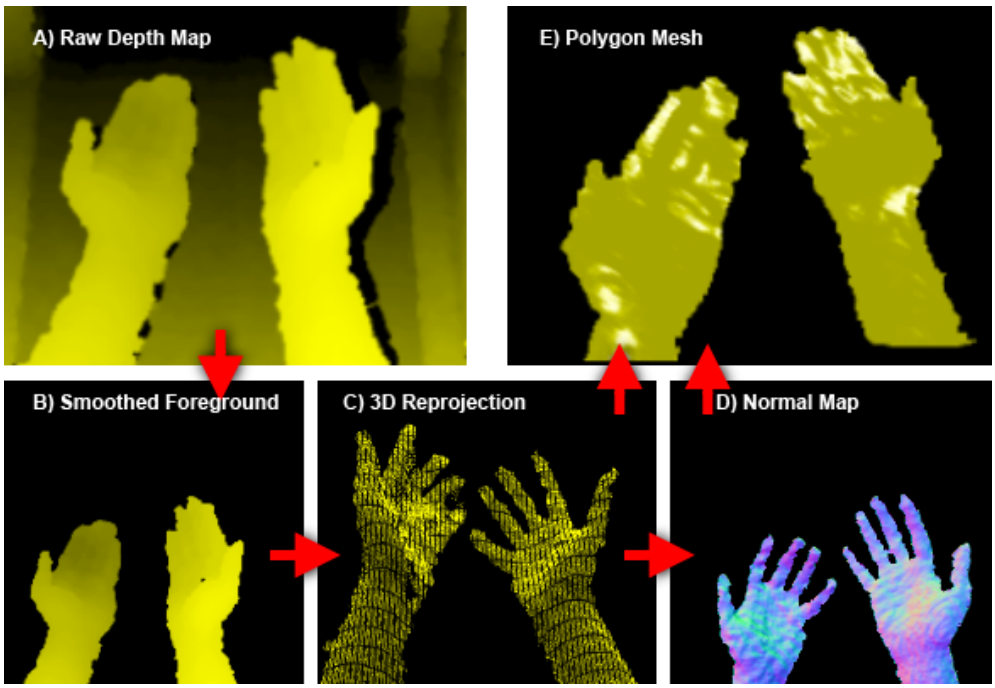

Figure 4.6: GPU pipeline overview, from raw depth-map to smoothed mesh.

First a reference background depth map, used for foreground segmentation, is computed by averaging multiple depth map samples from the Kinect camera (without any physical objects present in the interaction volume). Next a depth edge preserving bilateral filter [Paris and Durand, 2007] is applied to the foreground depth map to reduce noise.

From smoothed depth data a polygon mesh is computed on the GPU as follows: First, a flat base mesh is stored in GPU memory. Second, a pixel shader in parallel reprojects all points within the depth map as 3D points in camera coordinate space, and stores these in a texture map. Using this texture map, the base mesh is extruded

in a vertex shader, displacing Z values. Normals are calculated using a cross product between neighboring reprojected 3D points in the texture map. This shader also culls vertices (and associated triangle) based on any invalid depth measurements or a simple distance test across depth discontinuities, in the depth map.

This approach allows physical objects to be meshed in real-time, using the full depth map. From this mesh shadows can be cast from real objects onto the 3D virtual scene. Similarly the mesh data can model occlusions of virtual objects by hands or other physical objects (see Figure 4.3, bottom).

4.3.3 Modeling Physics-based Interactions in 3D

So far we described the system calibration and our approach for extracting real-time meshes using a Kinect. These are critical in real-time rendering of 3D scenes for the user's current head position with the ability for hands and other objects to correctly shadow and occlude the virtual.

However, we have yet to describe how the Kinect can be leveraged for *input* between the physical and the 3D scene. To enable user interaction in the virtual scene we need to model 3D shape, motion and deformation of physical objects in the interaction volume.

Simulating human grasping In designing our technique we were especially interested in simulating natural human grasping. While humans employ many different strategies to manipulate physical objects in the real world, grasping is arguably the most common mode of manipulating objects in 3D [Prachyabrued and Borst, 2011].

When using natural whole-hand input in the context of 3D physics simulations one needs to accurately model collision and friction forces exerted onto virtual objects. Collision forces can be modelled by introducing geometry that approximates the shape of physical objects into the simulation – for example using a number of small spherical rigid bodies per depth measurement. Modelling friction forces however, requires persistent geometry (e.g., a hand representation that deforms over time) and an accurate and fine-grained motion estimate in all three dimensions.

Whilst models for physically inspired interactions have been proposed for interactive tabletops [Wilson et al., 2008], and even extended to supporting grasping of objects in 2D [Wilson, 2007a], we present a novel technique for representing the Kinect depth data

within a physics simulation that extends these methods and enables true 3D interactions. It requires no need for data glove to be worn [Prachyabrued and Borst, 2011], and works with *any* physical object.

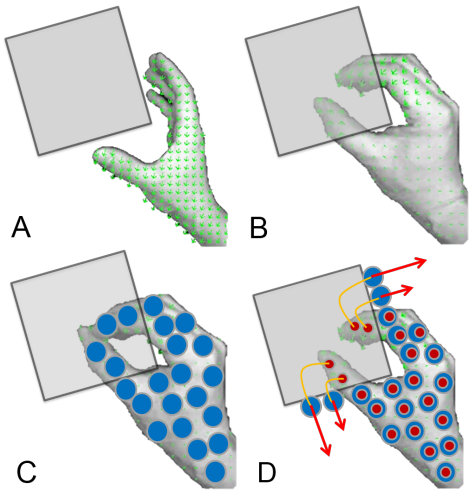

Figure 4.7: Simulating natural human grasping. A+B: Lack of haptic feedback causes virtual objects to be penetrated during grasping. C: Rigid bodies intersecting causes unstable physics simulation states. D: Kinematic particles (red) connected to dynamic proxies (blue) via joints (yellow).

Physics-based representation Conceptually our representation works as follows: The data from the Kinect camera is approximated by numerous, small spherical rigid bodies within a physics simulation (currently Nvidia's PhysX). These approximate the shape of any 3D physical object sensed by the Kinect. A motion estimate per particle is obtained from a *depth-aware optical flow* computation. In their entirety the particles closely approximate the shape of 3D objects placed within the interaction volume, but also deformation and motion of these objects in 3D. These particles are part of the physics simulation and exert collision and friction forces onto virtual objects.

One main issue that occurs when users interact with virtual objects using these rigid particles is *interpenetration* (Figure 4.7, A + B). For example, particles can easily enter another rigid virtual object that a user is attempting to grasp (Figure 4.7, C). Because the physics engine is simulating rigid body interactions, interpenetration causes extreme forces to be exerted onto virtual objects, leading to unstable simulation results. We solve this issue by reenforcing each dynamic particle by connecting it to another Kinematic particle, via a spring and damper (see Figure 4.7, D). The kinematically controlled

particle can be moved freely within the physics simulation based on the computed flow field. However, it will not take part in the rigid body simulation – meaning it will not cause any interpenetrations. The associated dynamic particles only exert forces against other objects without interpenetration, as they are not directly moved programmatically, but move as a by-product of the Kinematic object moving.

Our model can handle any physical object, either rigid – such as books or bowls (Figure 4.1, B+E) – or deforming objects as they interact with virtual objects (Figure 4.1, D). The technique also simulates human grasping as shown in Figure 4.5, bottom due to the system modeling 3D motion and the resulting friction forces between real and virtual objects.

In the next section we describe the main parts of our GPU input pipeline which allows raw Kinect data to be represented as rich interactions within a physics engine.

4.3.4 Depth-aware Flow

Our input pipeline is shown in Figure 4.8. The first step involves *tracking* the physical objects in the interaction volume. We perform tracking at an *atomic level*, using a GPU implementation of the optical flow algorithm by Brox et al. [2004], which produces smooth flow fields while avoiding over-smoothing at depth discontinuities. This is especially important when objects overlap, e.g. if hands are placed above each other. We have chosen this form of tracking as opposed to others as optical flow makes *no* assumptions about the actual object that is being tracked, it supports deforming or dynamically changing objects as well as rigid objects.

Our algorithm computes a 3D motion vector for each re-projected vertex from the depth map. Rather than first computing the flow directly on the Kinect depth map, we have found that a more robust method is to use the more textured Kinect RGB image. A dense 3D flow field can still be obtained by first computing the 2D flow field and using the depth map as a lookup to calculate 3D displacements.

In preparation for optical flow computation the smoothed depth map is used for foreground segmentation of the RGB image, which is assumed to be rectified to the Kinect IR camera using intrinsic and extrinsic camera calibration matrices.

Let $\mathbf{I}_i(\mathbf{u})$ be the intensity of a pixel $\mathbf{u} = (x, y)$ in the previous foreground RGB frame. $\mathbf{D}_i(\mathbf{u})$ is the corresponding depth value in the smoothed depth map. $\mathbf{v}_i$ is

the associated vertex that is reprojected from the depth map using the inverse of the depth camera's intrinsic calibration matrix $\mathbf{K}$. Our optical flow algorithm computes a displacement vector $\mathbf{p}_i(\mathbf{u}) = \mathbf{f}$ for a pixel $\mathbf{u}$ at frame i to its position at $i+1$. $\mathbf{p}_i : \mathbb{N}^2 \rightarrow \mathbb{R}^2$ is the flow field. We assume constant frame-to-frame pixel brightness and a smooth flow field. The energy function is:

$$\arg\min \int_{\mathbf{u}} \left(\mathbf{I}_i(\mathbf{u}) - \mathbf{I}_{i+1}(\mathbf{u} + \mathbf{p}_i(\mathbf{u}))\right)^2 + \alpha \cdot \nabla \mathbf{p}_i(\mathbf{u})^2$$

where α is weighting the brightness constancy and smoothness constraints.

The algorithm produces stable results even if the object only has few texture features, like a blank sheet of paper. In this case, the flow field is mostly driven by its contour. The smoothing constraint of the optical flow algorithm then propagates flow vectors from the contour into the object, again yielding a smooth flow field. Listing 1 details how we compute 3D displacements per pixel, using a lookup into the two depth maps after solving the point-to-point correspondence problem in the flow computation.

Listing 1 Computation of 3D offset between two frames

```
rectify RGB images to depth maps at frames i and i + 1
extract intensity images from RGB
perform background subtraction on intensity images
compute optical flow
for each image pixel u do
    read flow vector for this position
    f ← p_i(u)
    look up 3D positions in both frames
    v_i = D_i(u) · K^-1 · [u, 1]
    v_{i+1} = D_{i+1}(u + f) · K^-1 · [u + f, 1]
    subtraction yields 3D offset
    Δ(u) = v_{i+1} − v_i
end for
```

4.3.5 Updating Particles

We generate and update physics particles per frame to approximate the 3D shape of objects in the interaction volume, based on the flow field computed previously. For

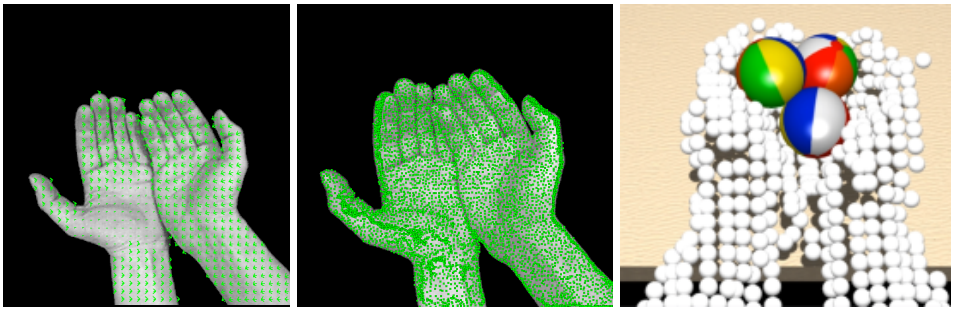

Figure 4.8: Depth aware optical flow. Left: Flow field computed from RGB image. Middle: Tracked particle positions. Right: Physics particles interacting with virtual spheres.

each foreground pixel in the RGB image with a valid measurement in the depth map, a particle position is calculated: $\mathbf{v} = \mathbf{D}_i(\mathbf{u}) \cdot \mathbf{K}^{-1} \cdot [\mathbf{u}, 1]$.

After the flow computation every particle's 3D position $\mathbf{v}$ is projected onto a pixel coordinate $\mathbf{u}$, retrieving the displacement vector $\boldsymbol{\Delta}(\mathbf{u})$. We employ bilinear interpolation when reading displacements from the flow field to smoothly set the new 3D position of the particle: $\mathbf{v}' = \mathbf{v} + \boldsymbol{\Delta}(\mathbf{u})$.

Particles are deleted if their projected 3D position $\mathbf{v}$ falls onto background pixels in the depth map, indicating that no correspondences where found. Particles are also removed after a maximum life time to balance the overall particle density and inherent tracking errors. We experimentally derived a maximum life time of 150 frames for our system.

By computing a per particle motion vector we can model a variety of motion within our physics simulation. Particles can exert lateral and friction forces onto other virtual objects, as shown in Figure 4.5 and the accompanying video.

4.4 Using HoloDesk

4.4.1 Mixed Reality Physics: Informal Observations

In enabling such physics-based interactions on HoloDesk our aim was to offer rich 3D interactions that move away from scripted and pre-defined gestures. Our aim is to allow users to manipulate objects in open-ended ways using collision and friction forces. Hoping that users would design their own 'interaction techniques' as they learn how to use HoloDesk.

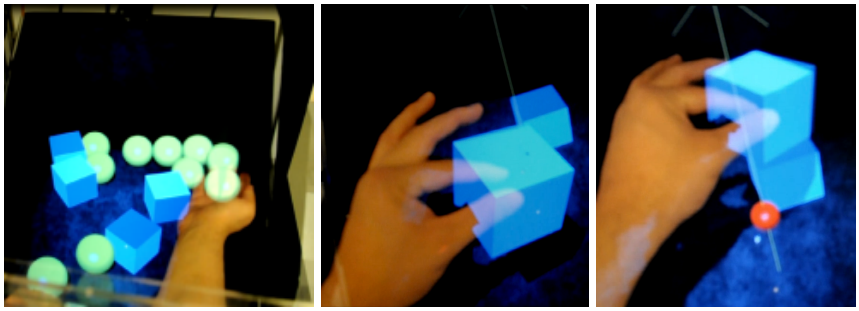

Figure 4.9: Left: Physics playground. Right: Emergent physics-based interactions. A user creating a stack of virtual cubes.

To verify this design goal we conducted two separate informal sessions, where we observed several hundred users perform open-ended tasks during which they received no instructions and were left to explore the system capabilities on their own. We were interested in emergent use patterns of our physics techniques, which would not be evidenced in a controlled experiment. We report our observations made during these two informal sessions and illustrate a number of emergent interactions which we observed frequently.

On both occasions users were given the opportunity to explore the systems capabilities in a "physics playground" – a 3D scene containing a number of dynamic and static physics enabled objects (see Figure 4.9, left). Furthermore the users were given physical objects such as a notebook or bowls.

Figure 4.10: Emergent physics-based interactions. Bi-manual juggling with two virtual spheres, at times both are in mid-air.

We observed many users initially only touching and pushing virtual objects on the ground plane of the 3D scene, not even realizing they had 3D control over these virtual objects. When they realized that they could lift objects up and interact in 3D, users were

positively surprised (Figure 4.9, right). One participant commented *"this is so weird – I can really lift this"*. We observed that grasping became the preferred interaction when participants tried to quickly reposition objects.

We also observed interactions that suggest modelling the entire shape of physical objects is in fact important, as opposed to modelling only fingertips. We often observed users scooping objects from underneath using both hands. In other instances, users had a virtual sphere rolling about on their palm and fingers while exerting fine-grained control over it using individual fingers and adjusting the overall posture of the hand (see Figure 4.10). Users, after some practice, even managed to juggle with both hands and several virtual spheres, as well as to move the sphere from their palm to the back of the hand and back again by a quick flip of the wrist.

Figure 4.11: Mixing physical and virtual. Left: A user fluently transferring virtual spheres from hands into plastic bowl and back.

We also observed how participants leveraged physical objects for interaction. For example, bowls were used to balance virtual objects (Figure 4.11, left) and a wooden bowl was used to create support for a virtual ball to cross a bridge (Figure 4.4, bottom left). Users leveraged the ability of our system to model both rigid and deforming objects in more exploratory ways. For example, virtual objects sitting on top of real cloth – users managed to move virtual blocks with the cloth as they pulled it away.

A final interesting observation was that users often adjusted the position and posture of their hands to conform to the 3D shape of virtual objects almost as if they could feel them. For example, using the palm and fingers to form a cup like shape when interacting with smaller round objects. In contrast users often utilized a fully flat hand to lift larger geometry such as the teapot in Figure 4.12.

These observations and user feedback suggests that a strong immersive experience was created by HoloDesk. The lack of user instrumentation allowed bystanders to

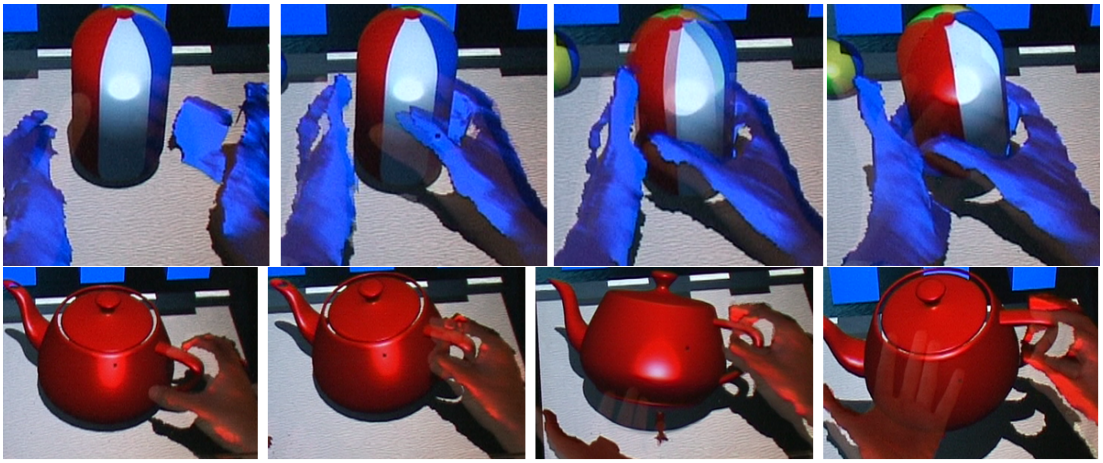

Figure 4.12: Hand postures conforming to different geometry. Top: Grasping a virtual capsule. Bottom: Lifting a virtual teapot.

experience the system (even if their viewpoint was distorted), and allowed them to quickly take turns interacting.

4.4.2 Limitations

While the technique enables a number of rich interaction possibilities, including natural grasping, it has a number of limitations which we want to discuss here.

Obviously the technique does not provide a full simulation of object motion in the interaction volume. In particular, the technique only models the parts of objects that are visible to the Kinect camera. For example, a virtual object that sits inside a real bowl will fall through the bottom of that container if another object overlaps and occludes the physical bowl even if only momentarily.

This is especially interesting in the context of grasping as this limits grasping strategies to hand configurations where fingertips are always visible to the camera. Otherwise virtual objects will slide out of grasping control. We observed that some users tried to lift objects by "clawing" them where the back of the hand occludes the fingers. But users were quick to realize that simple thumb and forefinger methods work best for grasp, and many appropriated these over time.

It would seem that a deformable 3D mesh representing real objects in the interaction volume would achieve the highest degree of fidelity and alleviate many of these issues. Finally, constructing and updating such an animated mesh in realtime is difficult and

computationally expensive, requiring robust tracking of features and accurate deformation of the 3D object. This remains an active research problem and will be part of our future research efforts.

Interestingly, this limitation highlights the possibility to utilize our physics representation in circumstances where only sparse tracking is available, for example a system that is capable of tracking fingertips in 3D but does not provide accurate shape data. Here fingertips could be represented by individual proxy objects in order to enable grasping simulation with the limitation that other interactions such as scooping, lifting and pushing would not be possible. The key difference with our new interaction model is that users have the choice to use fingertips to grasp for maximum efficiency or to use their entire hand when expressiveness is more important.

4.4.3 Formal User Study

There are also aspects of the HoloDesk design more suited to formal evaluation. The first is our assumption that HoloDesk provides benefits in the *directness* of 3D interaction it supports. We formally evaluate the impact of directness on user performance in a 3D target acquisition task. Our hypothesis is:

(**H1**) *Direct spatial coupling of input and output in HoloDesk improves 3D target acquisition in terms of selection speed and accuracy, when compared to an indirect variant.*

Our second assumption is that HoloDesk provides an experience close to glasses-based 3D interaction, but without user instrumentation. One feature of glasses-based 3D, lacking within HoloDesk, is *stereoscopic* depth cues. To evaluate this we performed an experiment hypothesizing:

(**H2**) *Monoscopic depth-cues of HoloDesk provide similar 3D target acquisition in terms of selection speed and accuracy, when compared to a variant of our system that provides stereoscopic cues using 3D shutter glasses.*

In terms of related studies, Knödel and Hachet [2011] evaluate direct multi-touch versus indirect touch for 2D and 3D interaction, showing that direct-touch shortens completion times, but indirect interaction improves efficiency and precision, particularly for 3D visualizations. Graham and MacKenzie [1996] explored physical versus virtual pointing, using a half silvered mirror and IR-based motion capture system, showing

similar performance for initial ballistic movement, but problems with the final part of target acquisition in the virtual condition. Ware et al. [1993] conducted an experiment on the use of motion parallax – or fishtank VR – compared with stereoscopic depth for a 3D path navigation task, showing statistically significant improvements when head-tracking was compared with stereo alone, but the fastest performance being stereo with motion parallax coupled.

While we do not anticipate that 3D pointing is going to be a prominent use of the system, we want to highlight that the studies were chosen to evaluate the spatial coupling of the 3D graphics and input for HoloDesk, and the monoscopic depth cues of a "walk up and use" system versus stereoscopic imagery. We evaluate the benefits of directness and stereo in the context of HoloDesk, which has a different physical setup to the studies described previously.

4.4.4 Experiment

To evaluate our two hypotheses, we conducted a formal user evaluation with three physical variants of HoloDesk: 1) The standard setup (**DHD**). 2) The same setup with the addition of stereo output (**SHD**) and Nvidia 3D Vision LCD shutter glasses. 3) An indirect variant (**IHD**) as shown in Figure 4.13, where the user interacted indirectly in front of the screen.

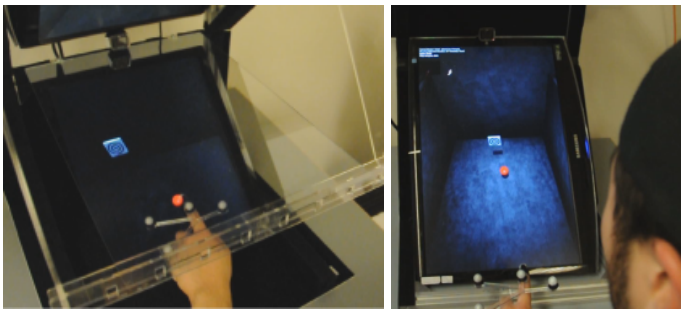

Figure 4.13: User study setup. User performing 3D target acquisition task. Left: Direct setup. Right: Indirect Setup.

To assure the best possible stereo effect in the SHD condition we measured interocular distance using a head-rest and callipers. At runtime we corrected eye-convergence values, per user, to keep the zero-parallax plane steady and coincident with the actual display plane.

To isolate the effects of *directness* and *depth cues* from tracking issues and to support head tracking whilst wearing stereo glasses, we make use of a Vicon motion capture system for tracking the location of the user's fingertip and head position across all conditions. As shown in Figure 4.13, markers were attached to a finger-worn thimble at an offset to avoid visual occlusion of the fingertip. The stereo glasses were tracked via markers to predict the user's eye position.

In all conditions, the screen resolution was 1680x1050 in portrait mode. All conditions provided viewpoint-corrected perspective rendering of the same 3D scene (see Figure 4.13), with a physical input space of $50x50x50cm^3$. A 6-core 2.8GHz PC with Nvidia GTX480 GPU was used in all conditions.

Participants Twelve participants (9 male, 3 female) between the ages of 21 and 40 were recruited to participate in the study. Participants were daily computer users, 2 were left handed. All had normal vision. Participants were screened for stereopsis in a simple *two-alternative forced choice* task.

Task To evaluate the performance of the HoloDesk system we performed a 3D target acquisition task. The interaction volume was subdivided into 27 equally sized cells (3x3x3 grid). Each trial was based on a 3D sphere appearing centered in one of these cells, upon selection a billboarded target (Figure 4.13) appeared centered in another cell for users to select. 20 trials were performed per block. Each user performed 5 blocks per condition (to avoid effects due to arm fatigue). Starting and target positions were coupled a priori to reduce in-condition variance. This set was used across all blocks, with the order randomized – ensuring that the total distance travelled was exactly equal across all users, and each block.

The user's fingertip was rendered as a blue sphere in the 3D scene. Each trial required the user to first select the starting red sphere, at which point the target would appear. Measurement of task completion time (**tct**) was triggered once the fingertip left the starting position and ended once the user indicates target selection using a foot pedal. User self-reporting was used to be able to measure both **tct** and accuracy without introducing experimenter bias.

Procedure We used a 3x1 within-subject repeated measure design. The independent variable was display type: IHD, DHD, and SHD with the dependent measures: 1) average **tct**; and 2) accuracy, measured as the 3D euclidean distance upon user selection from the fingertip to the target's center.

Presentation order of the conditions was counterbalanced using a Latin Square design. Users performed a training phase consisting of a single block per condition. A 30 second rest period occurred after completion of each block, and an additional minute after each condition. Users were asked to perform the task as quickly and accurately as possible. The entire experiment lasted about 60 minutes. Participants filled out a post-experiment questionnaire upon completion. Users were observed at all times as the task were performed and notes on subjective experiences were taken.

4.4.5 Results

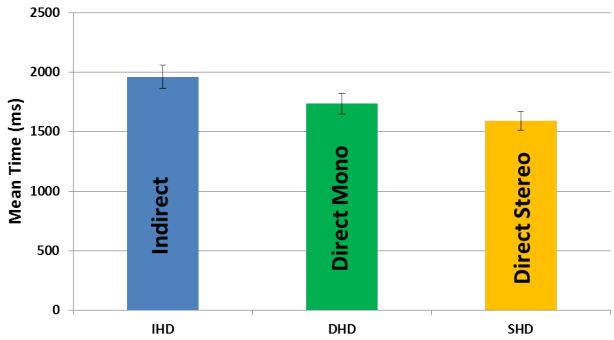

Figure 4.14: task completion time across standard direct (DHD), indirect (IHD), and stereo (SHD) variants of HoloDesk. Error bars represent +/-Standard Error of the Mean (SEM).

The average overall **tct** was 1693ms. A repeated measures ANOVA yielded a significant difference between the three Display Type conditions ($F_{2,22} = 48.57, p < 0.01$). Figure 4.14 shows the mean task completion time across all three conditions (DHD, IHD, and SHD). The stereo SHD condition was the fastest 1590ms, followed by direct DHD 1737ms and finally the indirect IHD condition 1962ms. The Post-hoc pair-wise comparisons (Bonferroni corrected) showed a significant difference between all conditions ($p < 0.01$). Analyzing the timings across blocks, a linear improvement in mean completion time for all three conditions was observed up to the third block

(all $p < 0.01$). There was however no significant learning after the third block (all $p > 0.31$).

The overall average distance from the target center was only $5.4mm$. A repeated measures ANOVA yielded no significant differences between the three Display Type conditions ($F_{2,22} = 1.75, p > 0.1544$).

4.4.6 Contrasting Direct versus Indirect Spatial Coupling

The results are promising in terms of supporting hypothesis H1. The direct DHD condition is significantly faster than the IHD condition without trading 3D targeting accuracy. This was observed quantitatively as well qualitatively throughout the trial. In the indirect IHD condition there was typically a ballistic movement to get the cursor into approximately the correct position of the target. However, a considerable fine adjustment in X, Y, Z directions could be observed. In the direct DHD users were able to more quickly move towards the target, by directly touching the target on the HoloDesk screen. The fine adjustment tended to be on the Z axis only (see below).

Our questionnaire showed that 9 of the 12 users had a preference for the direct DHD condition over the indirect when asked which of the two they preferred. In interviews overall participants commented that they found the DHD condition easier (e.g. *"much easier to press the targets"* P3 and *"you can reach out to touch them"* P7). Three users indicated a frustration that they could not directly reach out and touch the screen in the IHD condition.

4.4.7 Monoscopic Depth Cues versus Stereo

However, our results also indicate that stereo is statistically faster than the standard direct DHD condition, which does not support our initial hypothesis H2. This is backed up in related literature [Ware et al., 1993] and our qualitative observations. This can be attributed to a number of reasons. First, we observed that minimal head motion was used across the two direct conditions, and so the parallax depth cue were under utilized (no instructions about motion parallax were given). Second, we found a repeating pattern in the DHD condition of fast ballistic motions followed by fine adjustments along the Z axis. In the stereo condition users appeared to more readily 'touch' the target without any Z adjustment.

However, we observed that the Z adjustment varied in the DHD condition based on *where* the target was relative to the display plane. It seemed to be more prevalent for targets rendered in front of the display plane while users seemed to need less fine adjustment for targets on or inside the image plane.

This led us to further analyze the different spatial target locations (fully randomized presentation). Figure 4.15 shows promising results. Mean time for **tct** in the DHD condition was $1601ms$ on the image plane (O-DHD), $1612ms$ behind (B-DHD) and $1998ms$ in front (F-DHD). For stereo these were $1579ms$, $1582ms$, and $1609ms$ for on (O-SHD), behind (B-SHD) and in front (F-SHD) of the display plane respectively. As reported earlier a statistical difference between the display types ($F_{4,44} = 64.81, p < 0.01$) exists. Post-hoc pairwise tests show no significant difference between O-DHD and B-DHD nor when each of these are compared to any of -SHD conditions ($p > 0.3$). F-DHD is statistically different from all other conditions ($p < 0.01$).

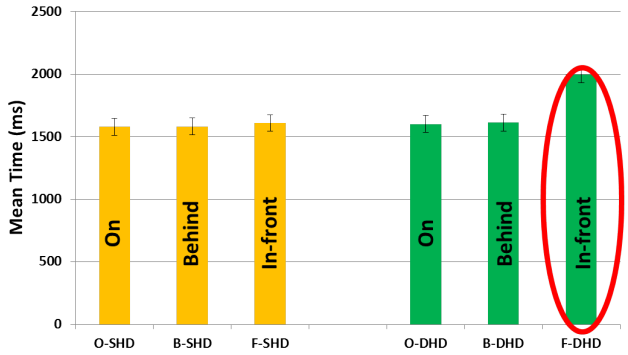

Figure 4.15: Task completion across standard direct (DHD) and stereo (SHD) variants of HoloDesk , but looking at target location on (O-), behind (B-), in-front (F-) of display plane. Error bars represent +/-SEM.

This further analysis shows that the placement of 3D targets is *critical* in enabling improved depth perception in the standard non-stereo DHD condition. It is clearly more difficult to directly 'touch' targets within DHD if these are placed in-front of the display plane, while targets placed behind or on the display plane show no significant differences between the DHD and SHD conditions. This was also evidenced in our observations where users would be able to directly touch the target when it was placed on or behind the display plane, but seemed to 'overshoot' when attempting to touch targets in-front of the display plane. This finding leads us to consider future designs of

HoloDesk where the image plane is mapped as close as possible to the location of the beamsplitter.

Interestingly, in terms of the post-study questionnaire and interviews, 7 users preferred DHD over SHD. Three users actually complained of feeling disoriented and dizzy after using the stereo condition. Whilst this might be due to issues in the stereo parameters or prolonged use, this observation has been seen in other studies [Ware et al., 1993]. Furthermore, some users did not like wearing the stereo glasses (e.g. *"it just felt uncomfortable"* P5, *"they weren't cool"* P8). However, some others had a clear preference for stereo (e.g. *"I felt like I was really touching things"* P1 and *"Everything looked more 3D"* P3).

4.5 Summary

In this chapter I presented an interactive system called HoloDesk that allows users to directly interact with 3D objects using rich physically inspired interactions. I have described the system implementation in full, focusing on the novel algorithm for supporting 3D physics-based interactions. I have demonstrated many interactions that HoloDesk supports, described application scenarios, and evaluated the system through formal and informal evaluations. The studies have highlighted the importance for IO coupling, as well as the ability to compete with user instrumented setups (based on stereo glasses).

I summarize the contributions as follows: 1) a novel system allowing for dexterous freeform interactions without any user instrumentation. 2) a new physics representation based on depth-aware optical flow, which extends existing techniques [Hilliges et al., 2009a; Wilson et al., 2008] by supporting full 3D interaction without hand-worn sensors. 3) A user study quantitatively evaluating the relative impact of directness and stereoscopic depth cues on HoloDesk. The results highlight that monoscopic depth cues combined with direct IO coupling compares with stereo, particularly for virtual objects behind the image plane. This adds to the existing literature studying 3D interactions, and carries implications for future designs.

This chapter also demonstrates the interactive opportunities for coupling AR/NUI. However, I also highlight limitations of our setup in being a situated (or fixed) display. Mobility is another key factor in AR systems. Furthermore, the focus of both RetroDepth and HoloDesk so far have been on user centric interactions, although some level of

object sensing is offered. In the next chapter we explore more mobile scenarios, with a system that focuses on higher fidelity environment sensing in combination with NUI-based freehand interactions.

Chapter 5

Freeform Interactions on Any Surface

5.1 Introduction

Chapter 3 and 4 demonstrated freehand interactions above a tabletop with a high-precision sensor and a system for direct manipulation of virtual objects in mid-air. Going further towards the vision of closely coupling AR and NUI, and in particular supporting natural interactions anywhere, in this chapter, I explore mechanisms for extending input and output beyond the tabletop into the physical space around us.

Smart spaces have been the focus of many ubiquitous computing scenarios Brumitt et al. [2000]; Kidd et al. [1999]; Wilson and Benko [2010] and are particularly appealing as users are not required to engage with a fixed system, allowing them to stay completely mobile. While many ubiquitous computing applications have demonstrated context-awareness of such spaces, fine grained interactions with the environment itself remains an under-explored area.

5.2 Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera

While depth cameras are not conceptually new, Kinect has made such sensors accessible to all. The quality of the depth sensing, given the low-cost and real-time nature of the device, is compelling, and has made the sensor instantly popular with researchers and enthusiasts alike.

Figure 5.1: KinectFusion enables real-time detailed 3D reconstructions of indoor scenes using only the depth data from a standard Kinect camera. A) user points Kinect at coffee table scene. B) Phong shaded reconstructed 3D model (the wireframe frustum shows current tracked 3D pose of Kinect). C) 3D model texture mapped using Kinect RGB data with real-time particles simulated on the 3D model as reconstruction occurs. D) Multi-touch interactions performed on any reconstructed surface. E) Real-time segmentation and 6DOF tracking of a physical object.

The Kinect camera uses a structured light technique [Freedman et al., 2008a] to generate real-time *depth maps* containing discrete range measurements of the physical scene. This data can be reprojected as a set of discrete 3D points (or *point cloud*). Even though the Kinect depth data is compelling, particularly compared to other commercially available depth cameras, it is still inherently noisy (see Figures 5.2B and 5.3 left). Depth measurements often fluctuate and depth maps contain numerous 'holes' where no readings were obtained.

To generate 3D models for use in applications such as gaming, physics, or CAD, higher-level *surface* geometry needs to be inferred from this noisy point-based data. One simple approach makes strong assumptions about the connectivity of neighboring points within the Kinect depth map to generate a *mesh* representation. This, however, leads to noisy and low-quality meshes as shown in Figure 5.2C. As importantly, this approach creates an incomplete mesh, from only a single, fixed viewpoint. To create a complete (or even watertight) 3D model, different viewpoints of the physical scene must be captured and *fused* into a single representation.

In this chapter, I present a novel interactive reconstruction system called KinectFusion (see Figure 5.1). The system takes live depth data from a moving Kinect camera and, in real-time, creates a single high-quality, geometrically accurate, 3D model. A user holding a standard Kinect camera can move within any indoor space, and reconstruct a 3D model of the physical scene within seconds. The system continuously tracks the 6 degrees-of-freedom (DOF) pose of the camera and fuses new viewpoints of the scene into a global surface-based representation. A novel GPU pipeline allows for accurate

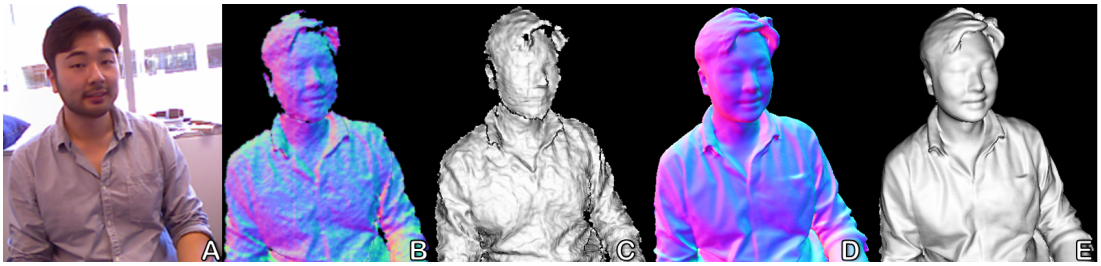

Figure 5.2: A) RGB image of scene. B) Normals extracted from raw Kinect depth map. C) 3D Mesh created from a single depth map. D and E) 3D model generated from KinectFusion showing surface normals (D) and rendered with Phong shading (E).

camera tracking and surface reconstruction at interactive real-time rates. This chapter details the capabilities of our novel system, as well as the implementation of the GPU pipeline in full.

We demonstrate core uses of KinectFusion as a low-cost handheld scanner, and present novel interactive methods for *segmenting* physical objects of interest from the reconstructed scene. We show how a real-time 3D model can be leveraged for *geometry-aware* augmented reality (AR) and physics-based interactions, where virtual worlds more realistically merge and interact with the real.

Placing such systems into an interaction context, where users need to dynamically interact in front of the sensor, reveals a fundamental challenge – no longer can we assume a static scene for camera tracking or reconstruction. I illustrate failure cases caused by a user moving in front of the sensor. I describe new methods to overcome these limitations, allowing camera tracking and reconstruction of a static background scene, while simultaneously segmenting, reconstructing and tracking foreground objects, including the user. We use this approach to demonstrate real-time multi-touch interactions *anywhere*, allowing a user to appropriate any physical surface, be it planar or non-planar, for touch.

The remainder of this section is structured into two parts: The first provides a high-level description of the capabilities of KinectFusion. The second describes the technical aspects of the system, focusing on our novel GPU pipeline.

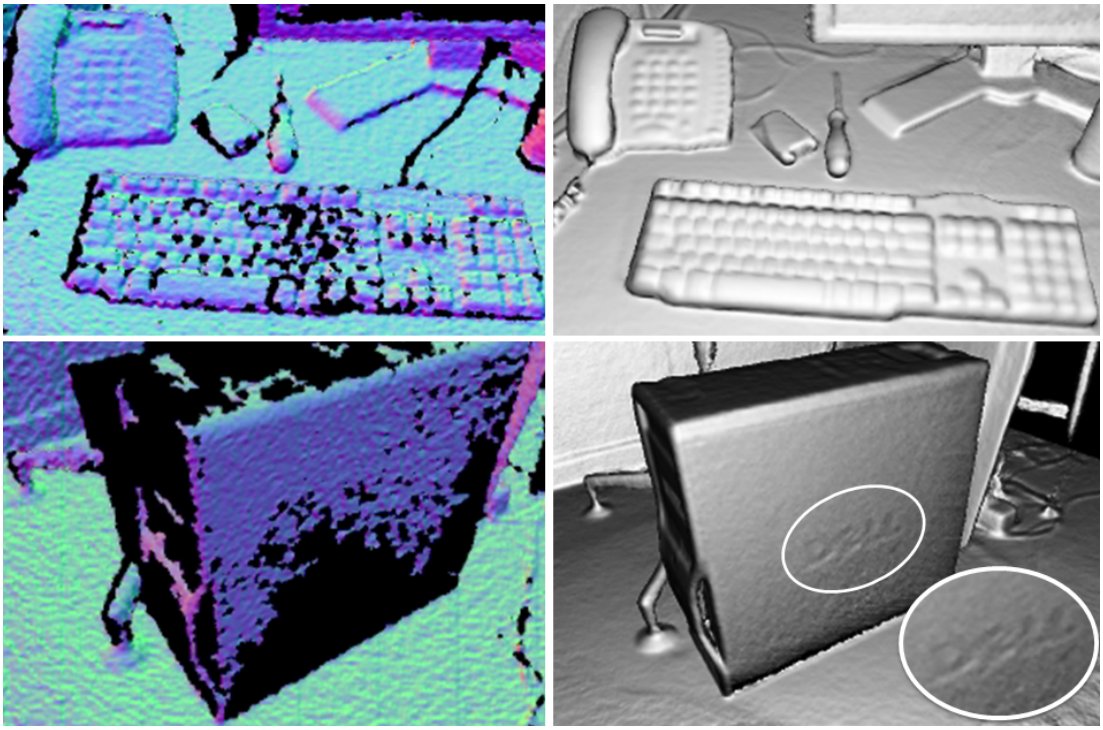

Figure 5.3: Left: Raw Kinect data (shown as surface normals). Right: Reconstruction shows hole filling and high-quality details such as keys on keyboard, phone number pad, wires, and even a DELL logo on the side of a PC (an engraving less than 1mm deep).

5.2.1 Utilizing the Physical Environment

Our system allows a user to pick up a standard Kinect camera and move rapidly within a room to reconstruct a high-quality, geometrically precise 3D model of the scene. To achieve this, our system continually tracks the 6DOF pose of the camera and fuses live depth data from the camera into a single global 3D model in real-time. As the user explores the space, new views of the physical scene are revealed and these are fused into the same model. The reconstruction therefore grows in detail as new depth measurements are added. Holes are filled, and the model becomes more complete and refined over time (see Figure 5.3).

Even small motions, caused for example by camera shake, result in new viewpoints of the scene and hence refinements to the model. This creates an effect similar to (multi-frame) image super-resolution [Farsiu et al., 2004] – adding greater detail than appears visible in the raw signal (see Figure 5.3). As illustrated in Figures 5.2 and 5.3, the reconstructions are high-quality, particularly given the noisy input data and speed of

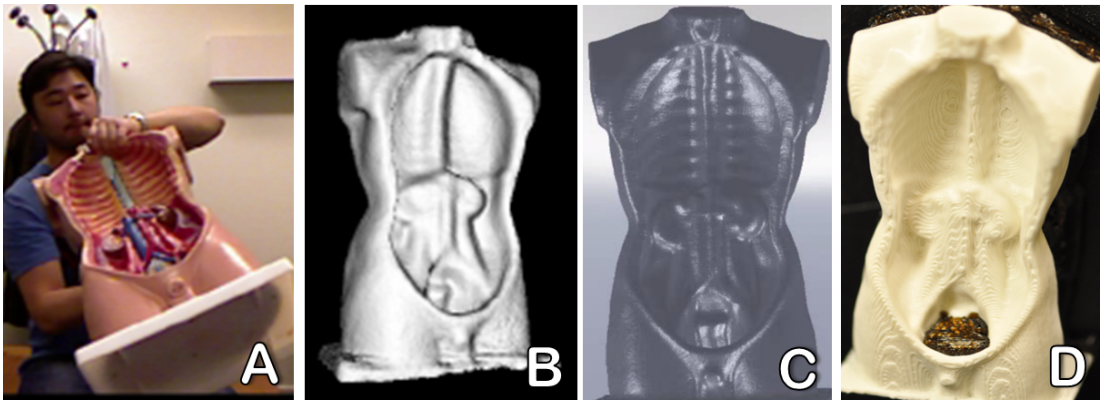

Figure 5.4: A) User rotating object in front of fixed Kinect. B) 360° 3D reconstruction. C) 3D model imported into SolidWorks. D) 3D printout from reconstruction.

reconstruction. The reconstructed model can also be texture mapped using the Kinect RGB camera (see Figures 5.1C, 5.5B and 5.6A).

Low-cost Handheld Scanning

A basic and yet compelling use for KinectFusion is as a low-cost object scanner. Although there is a body of research focusing on object scanning using passive and active cameras [Cui et al., 2010; Weise et al., 2009], the speed, quality, and scale of reconstructions have not been demonstrated previously with such low-cost commodity hardware. The mobile and real-time nature of our system allows users to rapidly capture an object from different viewpoints, and see onscreen feedback immediately. Reconstructed 3D models can be imported into CAD or other 3D modeling applications, or even 3D printed (see Figure 5.4).

Our system can also be used in 'reverse' – without any code changes – whereby the tracking algorithm predicts the 6DOF pose of a handheld object that is rotated in front of a fixed Kinect camera (as long as the object moves rigidly and occupies the majority of the depth map). While fingers and hands may initially form part of the 3D reconstruction, they are gradually integrated out of the 3D model, because they naturally move as a process of rotating the object.

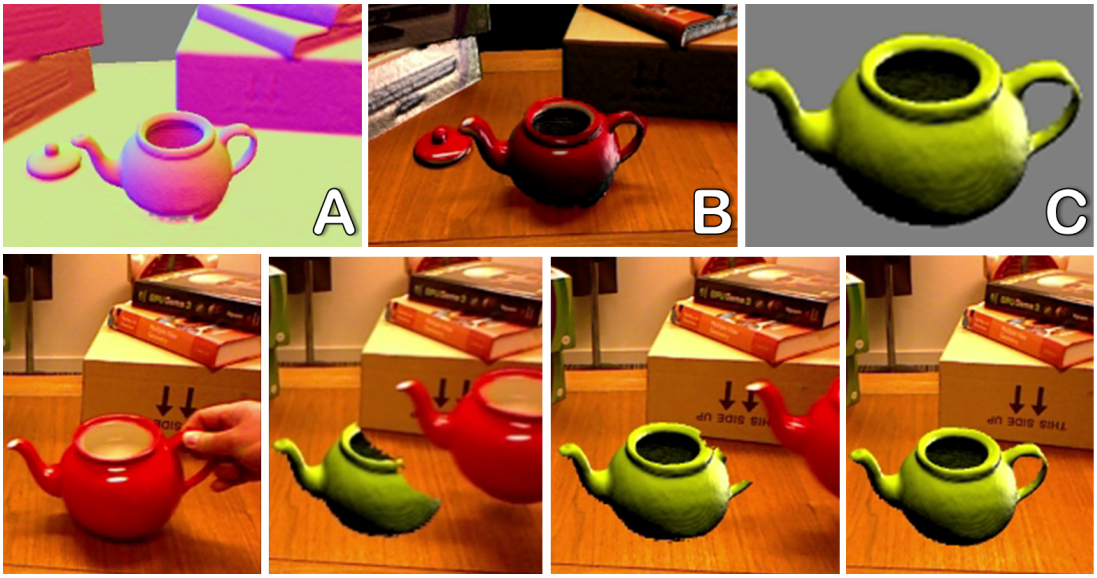

Figure 5.5: Fast and direct object segmentation. First entire scene is scanned including object of interest – the teapot. 3D reconstruction shows surface normals (A) and texture mapped model (B). Bottom left to right: Teapot physically removed. System monitors real-time changes in reconstruction and colors large changes yellow. C) This achieves accurate segmentation of teapot 3D model from initial scan.

Object Segmentation through Direct Interaction

Users may also wish to scan a specific smaller physical object rather than the entire scene. To support this, KinectFusion allows a user to first reconstruct the entire scene, and then accurately *segment* the desired object by moving it physically. The system continuously monitors the 3D reconstruction and observes changes over time. If an object is physically removed from view or moved within the scene by the user, rapid and large changes in the 3D model are observed. Such changes are detected in real-time, allowing the repositioned object to be cleanly segmented from the background model. This approach allows a user to perform segmentation rapidly and without any explicit GUI input, simply by moving the object directly (see Figure 5.5).

Geometry-Aware Augmented Reality

Beyond scanning, KinectFusion enables more realistic forms of AR, where a 3D virtual world is overlaid onto and interacts with the real-world representation. Figure 5.6 (top row) shows a virtual metallic sphere composited directly onto the 3D model, as

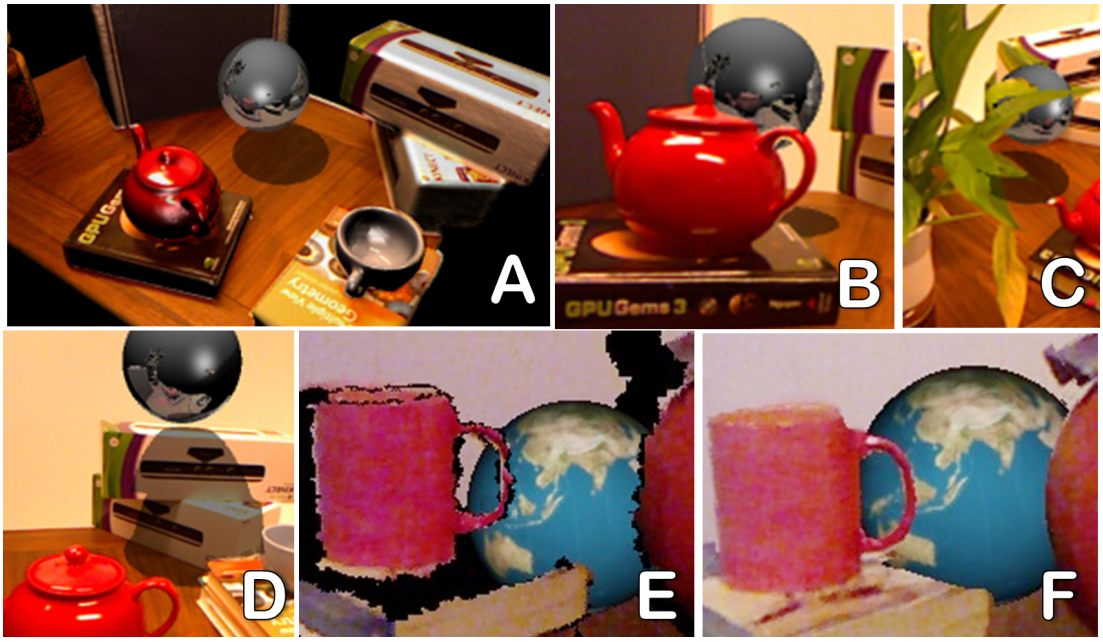

Figure 5.6: Virtual sphere composited onto texture mapped 3D model (A) and calibrated live Kinect RGB (B, C and D). Real-time 3D model used to handle precise occlusions of the virtual by complex physical geometries (B and C). Comparing occlusion handling using live depth map (E) versus 3D reconstruction (F). Note noise at depth edges, shadows and incomplete data (e.g. book) in live depth map. Virtual sphere casts shadows on physical (D) and reflects parts of the real scene (B and D).

well as the registered live RGB data from Kinect. The virtual sphere can be rendered from the same perspective as the tracked physical camera, enabling it to be spatially registered as the Kinect moves. As shown in Figure 5.6 (B, C and D), the live 3D model allows composited virtual graphics to be precisely occluded by the real-world, including geometrically complex objects. This quality of occlusion handling is not possible with the raw depth map (Figure 5.6E), especially around the edges of objects due to significant noise along depth discontinuities. Precise occlusions are critical for truly immersive AR experiences, and have not been achieved in sparsely mapped real-time AR systems (e.g. Klein and Murray [2007]).

Raytraced rendering effects can be calculated in real-time to create realistic shadows, lighting and reflections that consider both virtual and real geometries. For example, Figure 5.6 (B and D) shows how the virtual can cast shadows onto the real geometry, as

Figure 5.7: Interactive simulation of physics directly on 3D model even during reconstruction. Thousands of particles interact with reconstructed scene. Reconstruction, camera tracking, and physics simulation all performed in real-time.

well as reflect parts of the real scene onto the virtual. These can even be parts that are occluded from the perspective of the physical camera.

Taking Physics Beyond the 'Surface'

Taking this ability of combining and spatially aligning real and virtual worlds one step further, the virtual can also begin to interact dynamically with the reconstructed scene by simulating aspects of real-world physics. Many types of applications such as gaming and robotics can benefit from such physically precise real-time simulation. Rigid body collisions can be simulated live even as the 3D model is being reconstructed. Figure 5.7 shows thousands of particles interacting with the 3D model as it is reconstructed, all in

Figure 5.8: A user moves freely in front of a fixed Kinect. Live raw data (top) and shaded reconstruction (bottom). Left: Scene without user. Middle: User enters scene, but is only partially reconstructed due to motion. Right: Continued scene motions cause tracking failure.

real-time. This ability to both reconstruct a scene and simultaneously perform physics computations on the model is *unique*, and opens up a potential for more physically realistic AR applications.

5.2.2 Reaching into the Scene

It is important to note that, like most of the related literature on SLAM and camera-based reconstruction, our core system described so far makes a fundamental assumption – that camera tracking will be performed on a *static* scene. Once we switch focus from *reconstructing* the scene towards *interacting* within it, these assumptions no longer hold true. Physical objects such as the user's hands, will inevitably appear in the scene, move dynamically and impact tracking and reconstruction. Our camera tracking is robust to transient and rapid scene motions (such as the earlier example in Figure 5.5). However, prolonged interactions with the scene are problematic as illustrated in Figure 5.8.

While this is clearly a challenging problem within computer vision, our GPU-based pipeline is extended to approximate *camera motion* from *scene motion* for certain user

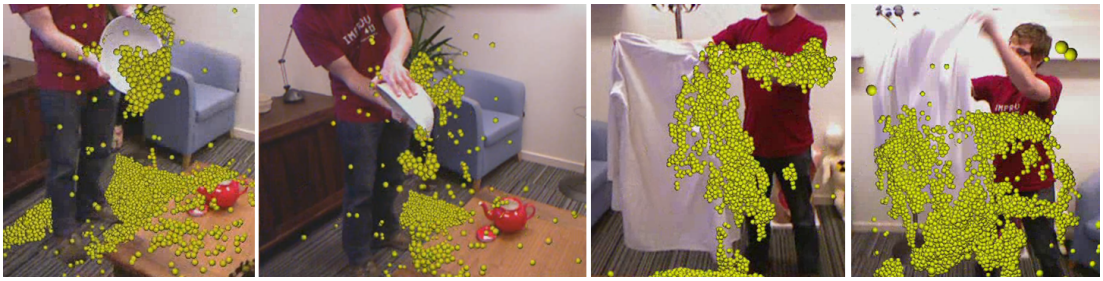

Figure 5.9: Interactive simulation of particle physics on dynamic scenes. Particles interact with dynamically changing foreground user, whilst physical camera and user move. User can collide with the physics-enabled virtual objects.

interaction scenarios. When a user is interacting in the scene, the camera tracking 'locks' onto the background and ignores the foreground user for camera pose prediction (shown later in Figure 5.15). This foreground data can be tracked (in 6DOF) and reconstructed independently of camera tracking and reconstruction of the static background.

This ability to reconstruct and track the user in the scene can enable novel extensions to our physics-based simulation shown earlier. Rigid particles can now collide with the rapidly changing dynamic foreground. Figure 5.9 demonstrates particles interacting with a dynamically updated reconstruction of a moving user. This enables direct interaction between the user and the physics-enabled virtual objects.

Furthermore, as we have captured geometry of both the background scene and foreground user, e.g. hands or potentially the full body, we can determine where the two *intersect*. These points of intersection can be leveraged (potentially with other methods for fingertip detection) to robustly detect touch on arbitrarily shaped surfaces – including non-planar geometries. This allows direct touch applications, such as those demonstrated in the interactive tabletop community, to reach *any* surface in the user's environment (see Figure 5.10).

5.2.3 GPU Implementation

Our approach for real-time camera tracking and surface reconstruction is based on two well-studied algorithms [Besl and McKay, 1992; Curless and Levoy, 1996; Rusinkiewicz and Levoy, 2001], which have been designed from the ground-up for parallel execution on the GPU. A full formalization of our method is provided in Newcombe et al. [2011], as well as quantitative evaluation of reconstruction performance. The focus of this

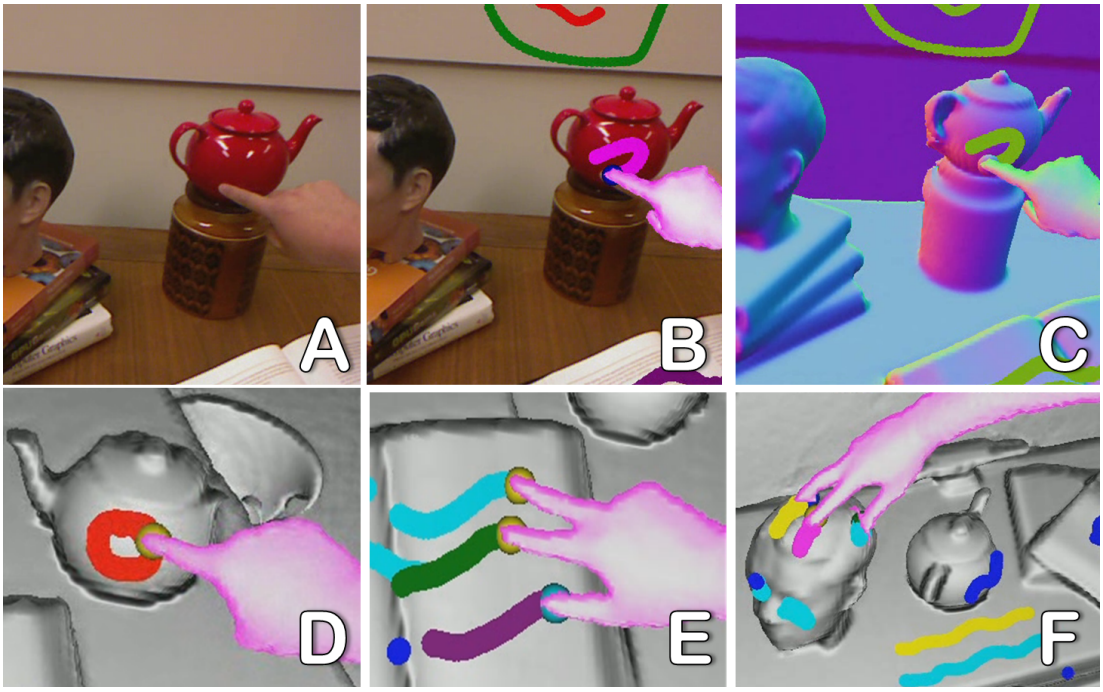

Figure 5.10: Enabling touch input on arbitrary surfaces with a moving camera. A) Live RGB. B) Composited view with segmented hand and single finger touching curved surface. C) rendered as surface normals. D) Single finger drawing on a curved surface. E) Multi-touch on regular planar book surface. F) Multi-touch on an arbitrarily shaped surface.

section is purely on the novel core and extended GPU pipeline, which is critical in enabling interactive rates.

The main system pipeline consists of four main stages (labeled appropriately in Figure 5.11):

a) Depth Map Conversion The live depth map is converted from image coordinates into 3D points, referred to here as vertices, and normals in the coordinate space of the camera.

b) Camera Tracking In the tracking phase, a rigid 6DOF transform is computed to closely align the current oriented points with the previous frames, using a GPU implementation of the Iterative Closest Point (ICP) algorithm [Besl and McKay, 1992]. Relative transforms are incrementally applied to a single transform that defines the global pose of the Kinect.

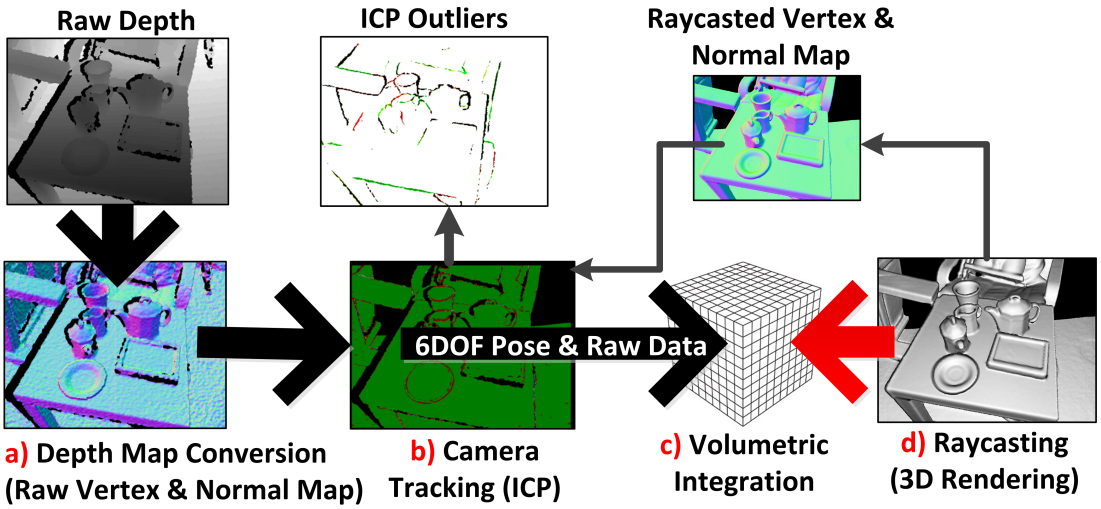

Figure 5.11: Overview of tracking and reconstruction pipeline from depth map to rendered view of 3D scene.

c) Volumetric Integration Instead of fusing point clouds or creating a mesh, we use a volumetric surface representation based on Curless and Levoy [1996]. Given the global pose of the camera, oriented points are converted into global coordinates, and a single volumetric 3D model is updated. Each voxel stores a running average of its distance to the assumed position of a physical surface.

d) Raycasting Finally, we raycast the volume to extract the stored implicit surface, and create 3D renderings of the reconstructed scene. When using the global pose of the camera, this raycasted view of the volume equates to a synthetic depth map, which can be used as a less noisy more globally consistent reference frame for the next iteration of ICP. This allows tracking by aligning the current live depth map with our less noisy raycasted view of the model, as opposed to using only the live depth maps frame-to-frame.

Each of these steps is executed in parallel on the GPU using the CUDA language. We describe each of these steps of the pipeline in the following sections.

Depth Map Conversion

At time i, each CUDA thread operates in parallel on a separate pixel $\mathbf{u} = (x, y)$ in the incoming depth map $\mathbf{D}_i(\mathbf{u})$. Given the intrinsic calibration matrix $\mathbf{K}$ (of the Kinect

infrared camera), each GPU thread reprojects a specific depth measurement as a 3D vertex in the camera's coordinate space as follows: $\mathbf{v}_i(\mathbf{u}) = \mathbf{D}_i(\mathbf{u})\ \mathbf{K}^{-1}[\mathbf{u}, \mathbf{1}]$. This results in a single vertex map $\mathbf{V}_i$ computed in parallel.

Corresponding normal vectors for each vertex are computed by each GPU thread using neighboring reprojected points: $\mathbf{n}_i(\mathbf{u}) = (\mathbf{v}_i(x+1, y) - \mathbf{v}_i(x, y)) \times (\mathbf{v}_i(x, y+1) - \mathbf{v}_i(x, y))$ (normalized to unit length $\mathbf{n}_i / ||\mathbf{n}_i||$). This results in a single normal map $\mathbf{N}_i$ computed in parallel.

The 6DOF camera pose at time i is a rigid body transform matrix $\mathbf{T}_i = [\mathbf{R}_i|\mathbf{t}_i]$ containing a 3x3 rotation matrix ($\mathbf{R}_i$) and translation vector ($\mathbf{t}_i$). Given this transform, vertex and normal maps can be converted into global coordinates: $\mathbf{V}_i^{\mathrm{g}}(\mathbf{u}) = \mathbf{T}_i \mathbf{V}_i(\mathbf{u})$ and $\mathbf{N}_i^{\mathrm{g}}(\mathbf{u}) = \mathbf{R}_i \mathbf{N}_i(\mathbf{u})$ respectively.

Camera Tracking

ICP is a popular and well-studied algorithm for 3D shape alignment (see Rusinkiewicz and Levoy [2001] for a detailed study). In KinectFusion, ICP is instead leveraged to track the camera pose for each new depth frame, by estimating a single 6DOF transform that closely aligns the current oriented points with those of the previous frame. This gives a relative 6DOF transform $\mathbf{T}^{\mathrm{rel}}$ which can be incrementally applied together to give the single global camera pose $\mathbf{T}_i$.

The important first step of ICP is to find correspondences between the current oriented points at time i with the previous at $i-1$. In our system we use *projective data association* [Rusinkiewicz and Levoy, 2001] to find these correspondences. This part of the GPU-based algorithm is shown as pseudocode in Listing 2. Given the previous global camera pose $\mathbf{T}_{i-1}$, each GPU thread transforms point $\mathbf{v}_{i-1}$ into camera coordinate space, and perspective projects it into image coordinates. It then uses this 2D point as a lookup into the current vertex ($\mathbf{V}_i$) and normal maps ($\mathbf{N}_i$), finding corresponding points along the *ray* (i.e. projected onto the same image coordinates). Finally, each GPU thread tests the compatibility of corresponding points to reject *outliers*, by first converting both into global coordinates, and then testing that the Euclidean distance and angles between them are within a threshold.

Given these set of corresponding oriented points, the output of each ICP iteration is a single relative transformation matrix $\mathbf{T}^{\mathrm{rel}}$ that minimizes the point-to-plane error

Listing 2 Projective point-plane data association.

```
1:  for each image pixel u in depth map D_i in parallel do
2:    if D_i(u) > 0 then
3:      v_{i-1} ← T_{i-1}^{-1} v^g_{i-1}
4:      p ← perspective project vertex v_{i-1}
5:      if p in vertex map V_i then
6:        v ← T_{i-1} V_i(p)
7:        n ← R_{i-1} N_i(p)
8:        if ||v − v^g_{i-1}|| < distance threshold and
          n . n^g_{i-1} < normal threshold then
9:          point correspondence found
10:       end if
11:     end if
12:   end if
13: end for
```

metric [Chen and Medioni, 1992], defined as the sum of squared distances between each point in the current frame and the tangent plane at its corresponding point in the previous frame:

$$\arg\min \sum_{\mathbf{u}} ||(\mathbf{T}^{\text{rel}}\mathbf{v}_i(\mathbf{u}) - \mathbf{v}^{\text{g}}_{i-1}(\mathbf{u})) \cdot \mathbf{n}^{\text{g}}_{i-1}(\mathbf{u})||^{\mathbf{2}} \tag{5.1}$$

We make a linear approximation to solve this system, by assuming only an incremental transformation occurs between frames [Chen and Medioni, 1992; Low, 2004]. The linear system is computed and summed in parallel on the GPU using a tree reduction. The solution to this 6x6 linear system is then solved on the CPU using a Cholesky decomposition.

One of the key novel contributions of our GPU-based camera tracking implementation is that ICP is performed on *all* the measurements provided in each 640×480 Kinect depth map. There is no sparse sampling of points or need to explicitly extract features (although of course ICP does implicitly require depth features to converge). This type of *dense* tracking is only feasible due to our novel GPU implementation, and plays a central role in enabling segmentation and user interaction in KinectFusion, as described later.

Volumetric Representation

By predicting the global pose of the camera using ICP, any depth measurement can be converted from image coordinates into a single consistent global coordinate space. We integrate this data using a *volumetric* representation based on Curless and Levoy [1996]. A 3D volume of fixed resolution is predefined, which maps to specific dimensions of a 3D physical space. This volume is subdivided uniformly into a grid of voxels. 3D points are then integrated into voxels using a variant of Signed Distance Functions (SDFs) [Osher and Fedkiw, 2002], specifying a relative distance to the actual surface. These values are positive in-front of the surface, negative behind, with the surface interface defined by the *zero-crossing* where the values change sign.

In practice, we only store a truncated region around the actual surface [Curless and Levoy, 1996] – here referred to as Truncated Signed Distance Functions (TSDFs). Whilst this approach has been studied in the context of laser range finders, this representation carries many advantages for the Kinect sensor data, particularly when compared to other representations such as meshes. It implicitly encodes uncertainty in the range data, efficiently deals with multiple measurements, fills holes as new measurements are added, accommodates sensor motion, and implicitly stores surface geometry.

Volumetric Integration

To achieve real-time rates, we describe a novel GPU implementation of volumetric TSDFs. The full 3D voxel grid is allocated on the GPU as aligned linear memory. This results in empty space, which constitutes the majority of voxels in a typical scene, occupying memory. Whilst clearly not memory efficient (a 512^3 volume containing 32-bit voxels requires 512MB of memory), our approach is *speed* efficient. Given the memory is aligned, access from parallel threads can be *coalesced* to increase memory throughput. This allows a full sweep of a volume (reading and writing to every voxel) to be performed incredibly quickly on commodity graphics hardware (e.g. a 512^3 sweep, accessing over 130 gigavoxels, takes $\sim 2ms$ on a NVIDIA GTX470).

Our algorithm provides three novel contributions. First, it ensures real-time coalesced access to the voxel grid, whilst integrating depth data *projectively*. Second, it generates TSDF values for voxels within the current camera frustum that do not contain a direct measurement in the current depth map. The latter allows continuous surface estimates to be discretized into the voxel grid, from the point-based Kinect depth maps.

Listing 3 Projective TSDF integration leveraging coalesced memory access.

```
1:  for each voxel g in x,y volume slice in parallel do
2:    while sweeping from front slice to back do
3:      v^g ← convert g from grid to global 3D position
4:      v ← T_i^{-1} v^g
5:      p ← perspective project vertex v
6:      if v in camera view frustum then
7:        sdf_i ← ||t_i − v^g|| − D_i(p)
8:        if (sdf_i > 0) then
9:          tsdf_i ← min(1, sdf_i / max truncation)
10:       else
11:         tsdf_i ← max(−1, sdf_i / min truncation)
12:       end if
13:       w_i ←min(max weight, w + 1)
14:       tsdf^avg ← (tsdf_{i−1} w_{i−1} + tsdf_i w_i)/(w_{i−1} + w_i)
15:       store w_i and tsdf^avg at voxel g
16:     end if
17:   end while
18: end for
```

Third, it is *much* simpler to implement than hierarchical techniques (e.g. Zhou et al. [2011]).

The pseudocode Listing 3 illustrates the main steps of our algorithm. Due to the large number of voxels typically within a volume, it is not feasible to launch a GPU thread per voxel. To ensure coalesced memory access, a GPU thread is assigned to each x and y position on the front slice of the volume. In parallel, GPU threads then *sweep* through the volume, moving along each slice on the Z-axis. Given the resolution of the volume, and the physical dimensions this maps to, each discrete 3D grid location can be converted into a vertex in global coordinates. A metric distance from the camera center (the translation vector of the global camera transform) to this vertex can be calculated. This 3D vertex can also be perspective projected back into image coordinates to lookup the actual depth measurement along the ray. The difference between measured and calculated distances gives a new SDF value for the voxel (line 7). This is normalized to a TSDF (lines 9 and 11) and averaged with the previous stored value using a simple running weighted average (line 13) [Curless and Levoy, 1996]. Both the new weight and averaged TSDF are stored at the voxel.

Raycasting for Rendering and Tracking

A GPU-based raycaster is implemented to extract the implicit surface from the volume for rendering (see pseudocode Listing 4). In parallel, each GPU thread walks a single ray and renders a single pixel in the output image. Given a starting position and direction for the ray, each GPU thread traverses voxels along the ray, and extracts the position of the implicit surface by observing a zero-crossing (a change in the sign of TSDF values stored along the ray). The final surface intersection point is computed using a simple linear interpolation given the trilinearly sampled points either side of the zero-crossing. Assuming the gradient is orthogonal to the surface interface, the surface normal is computed directly as the derivative of the TSDF at the zero-crossing [Osher and Fedkiw, 2002]. Therefore each GPU thread that finds a ray/surface intersection can calculate a single interpolated vertex and normal, which can used as parameters for lighting calculations on the output pixel.

Our rendering pipeline shown in Figure 5.12 also allows conventional polygon-based graphics to be *composited* on the raycasted view, enabling blending of virtual and real scenes with correct occlusion handling (see Figure 5.6). In the first step (labeled a), a mesh-based scene is rendered using *graphics camera* parameters identical to the physical global camera pose ($\mathbf{T}_i$) and intrinsics ($\mathbf{K}$). Instead of rendering to the framebuffer, the content of the vertex buffer, surface normals and unshaded color data are stored into off-screen vertex, normal and color maps respectively (labeled b), and used as input during raycasting (labeled c). For each GPU thread, a distance from the associated mesh vertex to the camera center is calculated in grid coordinates (Listing 4 lines 7 and 8). This distance acts as an additional termination condition while stepping along each ray (line 20). This allows accurate occlusion testing between volumetric and mesh surface geometries.

Ambient, diffuse and specular lighting contributions can be calculated *across* reconstructed and virtual geometries (see Figure 5.6). More advanced shading calculations can be performed by walking along the second (and possibly further) bounce of each ray. Shadows are calculated after the first ray hits a voxel or mesh surface (Listing 4 line 18 and 21), by walking a secondary ray from the surface to light position (using grid coordinates). If a surface is hit before ray termination then the vertex is shadowed.

Listing 4 Raycasting to extract the implicit surface, composite virtual 3D graphics, and perform lighting operations.

```
1: for each pixel u ∈ output image in parallel do
2:     ray^start ← back project [u, 0]; convert to grid pos
3:     ray^next ← back project [u, 1]; convert to grid pos
4:     ray^dir ← normalize (ray^next − ray^start)
5:     ray^len ← 0
6:     g ← first voxel along ray^dir
7:     m ← convert global mesh vertex to grid pos
8:     m^dist ← ||ray^start − m||
9:     while voxel g within volume bounds do
10:        ray^len ← ray^len + 1
11:        g^prev ← g
12:        g ← traverse next voxel along ray^dir
13:        if zero crossing from g to g^prev then
14:            p ← extract trilinear interpolated grid position
15:            v ← convert p from grid to global 3D position
16:            n ← extract surface gradient as ∇tsdf(p)
17:            shade pixel for oriented point (v, n) or
18:            follow secondary ray (shadows, reflections, etc)
19:        end if
20:        if ray^len > m^dist then
21:            shade pixel using mesh maps or
22:            follow secondary ray (shadows, reflections, etc)
23:        end if
24:    end while
25: end for
```

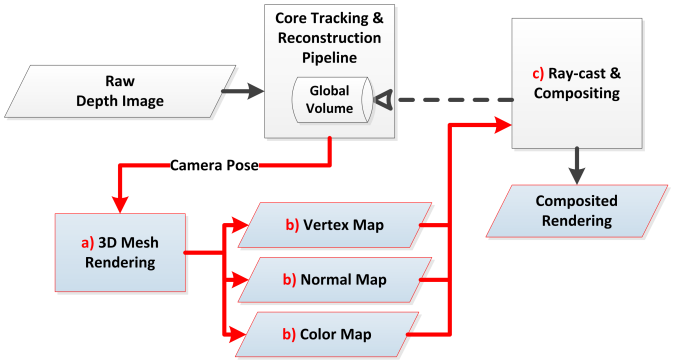

Figure 5.12: Rendering pipeline combining raycasting of volume with compositing of virtual polygon-based graphics.

For reflections, once the first ray hits a surface, a new ray direction is calculated, based on the surface normal and initial ray direction.

A novel contribution of our raycaster is the ability to view the implicit surface of the reconstructed 3D model, composite polygon geometry with correct occlusion handling, and provide advanced shading requiring raytraced operations, all in real-time, through a single algorithm. Any 6DOF graphics camera transform can be used to raycast the volume, including arbitrary third-person views allowing user navigation of the 3D model. However, another *key* contribution of our raycaster, is in generating higher-quality data for ICP camera tracking. When the raycast camera transform equates to the physical camera pose, the extracted vertices and normals equate to depth and normal maps (from the same perspective as the physical camera) but with considerably less noise, shadows and holes than the raw Kinect data. As shown in Newcombe et al. [2011], this allows us to mitigate issues of drift and reduce ICP errors, by tracking directly from the raycasted model as opposed to frame-to-frame ICP tracking.

Simulating Real-World Physics

Taking the merging of real and virtual geometries further, the GPU pipeline is extended to support simulation of physically realistic collisions between virtual objects and the reconstructed scene. A particle simulation is implemented on the GPU, based on Grand [2007] and Harada [2007]. Scene geometry is represented within the simulation by a set of static particles (see Figure 5.13). These are spheres of identical size, which remain stationary but can collide with other dynamically simulated particles. Whilst an approximation, this technique models every discrete surface voxel within the volume

in real-time, achieving compelling results even for very small and arbitrarily shaped objects such as a book's edges or a teapot's handle in Figure 5.7, bottom right.

Static particles are created during volume integration. As the volume is swept, TSDF values within a threshold close to zero (defining the surface interface or *zero level set*) are extracted. For each surface voxel, a static particle is instantiated. Each particle contains a 3D vertex in global (metric) space, a velocity (always 0 for static particles), and an ID. One key challenge then becomes detecting collisions. We use a spatially subdivided uniform grid to identify neighboring particles [Grand, 2007]. Each cell in the grid has a unique ID. Each dynamic or static particle is assigned a grid cell ID by converting the particle's global vertex to grid coordinates. Our system then maintains two lists – one containing static particles; the other dynamic. In both, particles are binned into the grid cells by sorting them by their current grid ID (using a GPU-based radix sort). During each simulation step, a GPU thread is launched per dynamic particle. Each thread processes collisions by examining (3^3) neighborhood of cells (first for other dynamic-dynamic collisions and then dynamic-static). The Discrete Element Method (DEM) [Harada, 2007] is used to calculate a velocity vector when pairs of particles collide. The particle's global velocity is incremented based on all neighboring collisions, gravity, and interactions with the bounding volume. Each particle is then repositioned based on the accumulated velocity per simulation step.

Figure 5.7 shows thousands of particles interacting with the reconstructed scene. A major contribution is that the system maintains interactive rates despite the overhead of physics simulation, whilst maintaining real-time camera tracking and reconstruction. By default only dynamic particles are composited during raycasting and again can be correctly occluded by the reconstructed geometry (see Figures 5.7 and 5.13).

5.2.4 Interacting in the Scene

The core system described so far makes assumptions that the scene will remain reasonably static. Clearly in an interaction context, users want to move freely in front of the sensor. This opens up two main challenges.

First, ICP tracking assumes a single rigid transform occurred per frame due to camera motion. User interaction in front of the sensor will cause scene motion *independent* of camera motion, which breaks this assumption. Because our ICP tracking is dense (i.e.

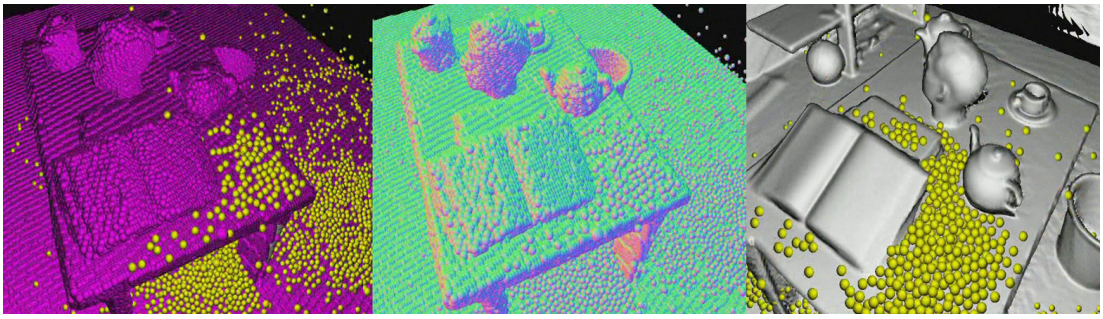

Figure 5.13: Simulating physics on the real-time reconstruction. Left: Surface is approximated as series of static particles (updated per integration sweep) which interact with dynamic particles. Every surface voxel is represented by a static particle. Middle: Surface normals of static and dynamic particles. Right: Shaded scene with only dynamic particles composited.

uses all available points) our system is resilient to transient scene motions. For example, in Figure 5.5, even when the user moves the object, enough *background* points will remain for ICP to converge. However, large or longer-term scene motions will cause tracking failure.

Second, whilst our system supports real-time reconstruction, surface predictions are refined over time using a running weighted average of distance values. By adapting the weighting, higher precedence can be given to new TSDF values, allowing for faster model updates, but the trade-off is additional noise being introduced to the reconstruction. In practice, a weight is chosen to balance quality of reconstruction with regular updates to the reconstruction based on scene changes. However, this does not support a continuously moving scene. Typically a user freely moving in the scene leads to associated depth data being partially integrated into the volume (Figure 5.8 middle). As camera tracking relies directly on this model, which is now inconsistent with the live data, failures will occur (Figure 5.8 right).

ICP outliers for segmentation

To begin to explore dynamic user interaction with the reconstructed scene, a novel extension to the core GPU pipeline is provided (shown in Figure 5.14). The technique leverages a unique property of dense ICP tracking. When *all* depth measurements are used, *outliers* from projective data association can form a strong initial predication as to parts of the scene moving independent of camera motion – if enough rigid background points are present for ICP to converge. Our solution robustly *segments* a moving

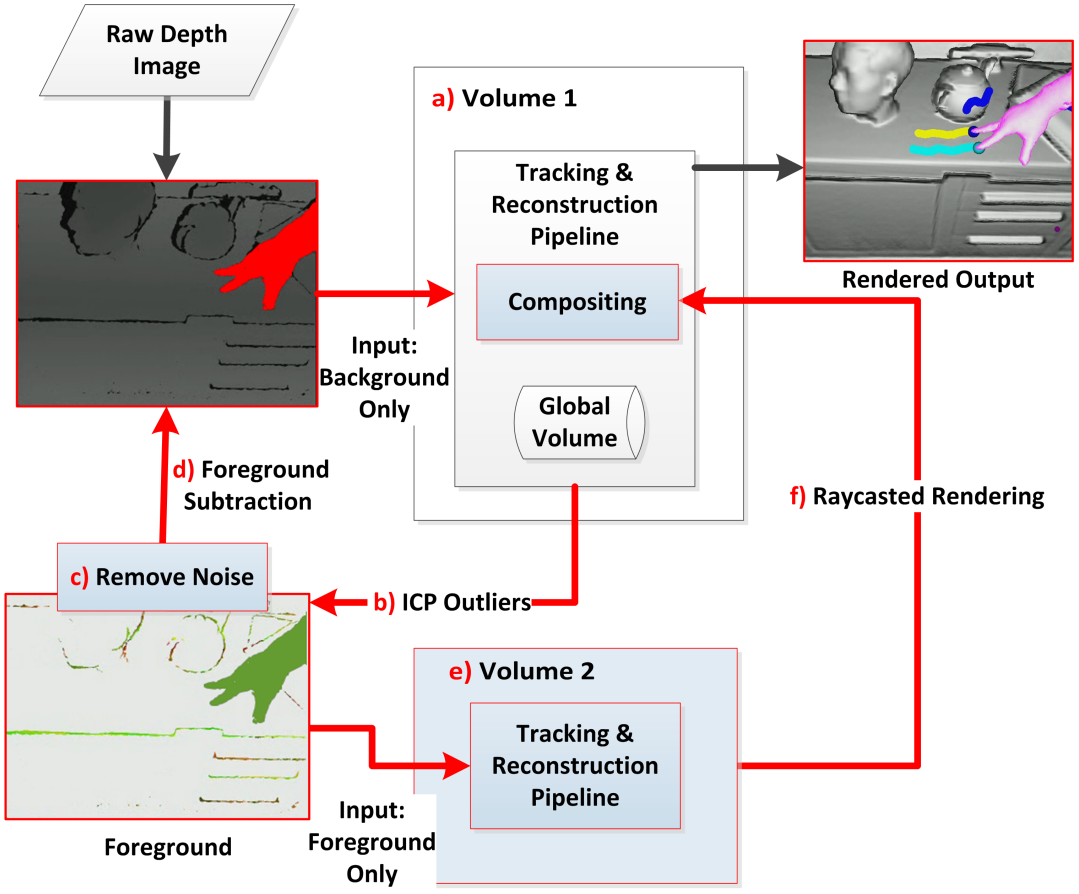

Figure 5.14: Extended GPU pipeline for real-time foreground and background segmentation, tracking and reconstruction.

foreground object from the background, allowing tracking failures to be reduced, and enabling users to interact directly in the scene.

This pipeline assumes that at least parts of a rigid scene have been initially reconstructed using the core reconstruction pipeline (labeled a). After this initial scan, a moving object entering the scene contains oriented points with significant disparity to already reconstructed surfaces. These fail ICP projective data association and are copied into an *outlier map* (labeled b). Next, a depth-aware connected component analysis is performed on the outlier map to cluster large connected patches and remove smaller outliers due to sensor noise (labeled c). In consecutive frames large connected patches are used to mask the incoming depth map where foreground scene motion has been detected (labeled d). This stops associated foreground depth measurements being used for reconstruction or tracking in the core pipeline. Large patches of outliers can be

Figure 5.15: Moving user is segmented and reconstructed, independent of background. Left to right: 1) Live RGB. 2) ICP outliers (for initial segmentation prediction). 3) final composited scene showing foreground shaded differently to background. 4) Composited normal maps.

additionally reconstructed using a second volume (labeled e) – potentially running on a separate GPU with different reconstruction settings. A final step raycasts the two separate volumes and composites the output (labeled f), using the same method as Figure 5.12.

Overall our technique yields compelling results in stabilizing tracking and therefore improving reconstruction quality for a static background, even when parts of the scene continually move in front of the camera. Furthermore, it allows a foreground object to be robustly segmented, and potentially reconstructed separately of the background (see Figure 5.15).

Detecting touch on arbitrary surfaces

The pipeline can be further extended to support multi-touch input by observing when foreground and background intersect. We extend the default raycasting of the back-

Listing 5 Create touch map – testing if foreground and background vertices overlap.

```
1:  V_fg^g ← raycasted vertex map from foreground volume
2:  for each pixel u ∈ O (touch map) in parallel do
3:     cast single ray for u (as Listing 4)
4:     if zero crossing when walking ray then
5:        v_bg^g ← interpolated global zero crossing position
6:        if ||v_bg^g − V_fg^g(u)|| < threshold then
7:           O(u) ← V_fg^g(u)
8:        end if
9:     end if
10: end for
```

ground volume to output a touch map, as shown in pseudocode Listing 5. Using the raycasted foreground vertex map as input, each GPU thread again walks a ray through the background volume. If a zero crossing is located, the corresponding foreground vertex (along the same ray) is tested (line 6). If foreground and background are within range, the foreground position is output in the touch map. A depth-aware connected component analysis of the touch map suppresses noise and labels fingertip candidates, which are tracked over time. Examples of enabling multi-touch on both planar and non-planar surfaces are shown in Figures 5.10 and 5.16.

Towards Modeling of Dynamic Scenes

The ability of now distinguishing moving foreground robustly from background raises interesting questions regarding how best to reconstruct such moving surfaces. The key challenge becomes how to integrate foreground data into a second volume so that correspondence between surface measurements can be ensured over time. As an initial exploration, we have experimented with independently predicting the pose of the foreground object using another instance of ICP. Again *dense* ICP is performed but only using the *foreground* oriented points (from the live depth map and raycasted second volume). In practice we have found that dense ICP converges even if small parts of the foreground are moving non-rigidly. A compelling example is a user's arm (Figure 5.15) where ICP converges on the rigid parts even if fingers are moving non-rigidly. This offers a coarse method for predicting the pose of the foreground object, relative to the global camera transform.

Figure 5.16: Segmentation, tracking and reconstruction of user's arm with moving Kinect. Top: Arm is first introduced and reconstruction contains a great deal of noise (left), surface is rapidly refined based on separate ICP-based pose prediction (right). Bottom left: the moving surface is reconstructed to a much higher quality than the raw Kinect signal. Bottom right: The intersection between foreground and background surfaces are used for multi-touch detection.

Using this predicted pose, depth measurements can be aligned and fused into the second volume. A surface prediction of the foreground, which becomes more refined and complete, can be built up over time. Because the foreground surface will likely be moving, we give more weight to new measurements being integrated. One simple extension uses a per-voxel weighting, adapted based on a running average of the derivative of the TSDF (prior to integration). This allows us to adapt the weight of individual surface voxels, giving higher priority to new measurements when the rate of change is high (e.g. fingers or hands), and lower if the TSDF measurements are stable (e.g. the forearm). Figures 5.16 and 5.15 shows our initial results based on foreground ICP tracking and per-voxel adaptive weighting. Note there is considerably less noise than the raw Kinect data – the user's arms, hand and fingers are clearly identifiable – and that this foreground reconstruction occurs alongside camera tracking and refinement of the background reconstruction.

For our physics simulation, we can now represent the entire foreground reconstruction as static particles, allowing us to model collisions between the user, and the dynamic particles per frame (as shown in Figure 5.9). This approach can also be used purely to continuously track rigid objects held in the user's hand – allowing tracking

independent of camera motion and without any marker or prior knowledge of the object. One example is shown in Figure 5.1 (far right) where an already reconstructed teapot (from Figure 5.5) is tracked in 6DOF and re-registered with the real physical object.

5.3 Summary

In this chapter I have presented the interactive capabilities of KinectFusion, a real-time 3D reconstruction and interaction system using a moving standard Kinect depth sensor. The contributions are threefold. First, I detailed in full a novel GPU pipeline for achieving dense tracking, reconstruction, segmentation, rendering, and interaction, all in real-time using only a commodity camera and graphics hardware. Second, I have demonstrated core novel uses for our system: as a low-cost object scanner and for advanced AR and physics-based interactions. Third, we demonstrate new methods for segmenting, tracking and reconstructing dynamic users and the background scene simultaneously. Creating the ability to enable multi-touch on any indoor scene even with complex, non-planar, geometries.

KinectFusion demonstrates one of the first non-situated AR/NUI systems that brings together properties of both fields in complementary ways, into a single self contained system. Whilst the real and virtual worlds are closely coupled from the perspective of geometry, in the examples presented, I have yet to demonstrate how *output* can be incorporated in this setup. As highlighted in this thesis, I wish to move away from user instrumentation, so HMDs are not a viable option. In the next chapter we explore the ability to bring 3D reconstruction, SLAM, and hand-based NUI interactions together with output from a less intrusive display technology – mobile pico-projection. I will also explore further the benefits of infrastructure-based versus infrastructure-free systems, and explore the potential for combining these two approaches.

Chapter 6

Augmenting Physical Surfaces Anywhere

6.1 Introduction

The previous chapters demonstrated how by giving computers a better sense of the space around them, we can design new NUI/AR capabilities, especially in combination with rich ways of segmenting and sensing the user. In the previous chapter I presented KinectFusion, a system that arguably exhibits the deepest coupling of real and virtual demonstrated so far. However, in this system I did not touch upon the issue of output or display. In this chapter, I explore this very question.

Projection systems offer a lightweight and compelling way to directly augment physical spaces without instrumenting the user. Prior work has explored either spatially or geometry aware projection in the context of infrastructure-based or infrastructure-free systems. However, the combination of the two has been under-explored, and it is unclear what the trade-offs of taking an infrastructure or infrastructure-free approach are. In contrast, this chapter aims to tackle the fundamental technical challenge of supporting both spatial and geometry sensing to augment physical spaces with interactive projection.

This section presents two novel systems that enable handheld projectors to be used for interaction in everyday spaces. Unlike prior work described in Section 2.7.2, these systems provide both a high degree of spatial awareness, where the device can sense its location and orientation in 3D space, and geometry awareness, where the system can

construct the 3D structure of the world around it, which can encompass the user as well as other physical objects, such as furniture and walls. Previous work in this area has predominantly focused on infrastructure-based spatial-aware handheld projection and interaction.

The prototypes take two opposing approaches in realizing both spatial and geometry awareness. The first system embeds some of the sensing into the environment. A novel infrastructure-based system uses four ceiling-mounted Kinect cameras to both track the 3D location of the handheld projector but also reconstruct the geometry of an entire room. The projector is coupled with an onboard infrared (IR) camera and Inertial Measurement Unit (IMU), which additionally enables finer sensing of users hands and the orientation of the device. This creates a system with both spatial and geometry-awareness, which allows novel types of interaction, including shadow and physics-enabled virtual interactions. Existing mobile projection systems often use an off-the-shelf tracking solution such as a Vicon motion capture system [Cao and Balakrishnan, 2006; Cao et al., 2007], which only provides 3D pose, but no geometry sensing. Tracking based systems allow for simple manipulation of virtual objects, such as rotation, translation, scale, selection and drawing, however, lack the directness of mapping between physical and virtual objects and manipulation that shape or geometry-aware systems can provide.

The second system takes an infrastructure-less sensing approach providing whole-room geometry and spatial awareness through a handheld device that combines a pico-projector with a Kinect depth camera. The KinectFusion system described in the previous chapter, is used to both estimate the 6DoF pose of the device, while at the same time creating a detailed reconstruction of the scene. Although the systems share similar goals they have their unique trade-offs. The following sections describe each system in detail, as each informs the design of handheld projection for augmented spaces. Also novel interactive scenarios are demonstrated that each system can enable.

6.2 System Overview

The first system presented, called RoomProjector, uses multiple fixed Kinect cameras to generate a coarse representation of the surfaces in an environment, and track objects and people that inhabit the space. This provides spatial-awareness similar to the Vicon

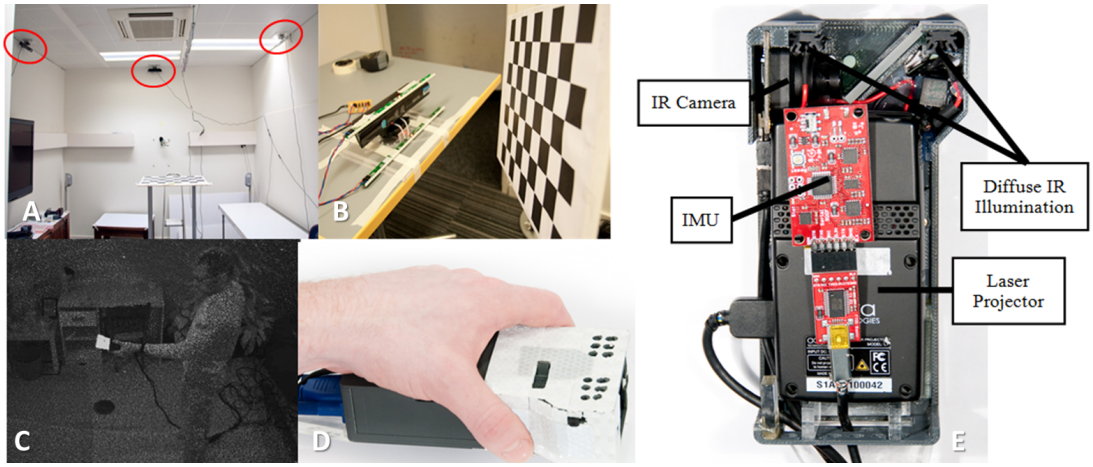

Figure 6.1: A) Room infrastructure setup shows three of four ceiling-mounted Kinect cameras (red circles). Note the calibration pattern in the center is used for estimating the extrinsic pose of each camera. B) Intrinsic calibration of the IR Kinect camera using a checkerboard pattern illuminated using diffuse IR. D) For location sensing the projector is covered with IR reflective tape. C) This allows the projector to be easily identified in the 2D Kinect IR image. When visible to multiple Kinect cameras, the 3D location of the projector can be determined by triangulation. E) Projector hardware main components.

motion capture systems used in Cao and Balakrishnan [2006]; Cao et al. [2007], but goes beyond previous handheld projector systems in terms of the geometry-awareness it affords.

6.2.1 Infrastructure-based Sensing

Instead of using traditional diffuse IR illumination coupled with high-speed IR camera tracking of retro-reflective markers (such as in the Vicon motion capture system used in Cao and Balakrishnan [2006]), the infrastructure-based approach relies on multiple fixed Kinect depth-sensing cameras.

The RoomProjector prototype uses four regular Kinect cameras mounted on the ceiling (2.75m high) at the mid-point of each wall in a rectangular room (4x3m) and angled down at 45 (see Figure 6.1A). To sense the whole room simultaneously, the depth maps from each independent camera are registered with respect to each other by performing standard camera calibration [Hartley and Zisserman, 2004]. To compute the intrinsic projection parameters (focal length and principal point) and radial and tangential lens distortion, multiple views of a black-and-white checkerboard pattern are

captured at different positions and angles in view of the Kinect IR camera. A linear array of wide-angle IR LEDs is used as a source of diffuse IR to illuminate the target, whereas the Kinect structured IR source is covered during this process to ensure even illumination (see Figure 6.1A and B).

The system is calibrated for depth errors in the sensor signal (particularly evident at large distances from the camera) by recording the measured distance from the Kinect to a large flat surface at a number of different distances, and comparing to ground truth using a laser range finder. The extrinsic 6DoF pose of each Kinect is determined using a large printed checkerboard visible to all cameras at once, and defines a shared real-world origin (see Figure 6.1A).

6.2.2 Geometry Reconstruction

GPU-based computer vision processing pipeline (shown in Figure 6.2) is used to transform the raw depth data from the Kinect sensors into surface meshes representing the coarse geometrical structure of the room as follows: A reference background frame is captured per camera, by averaging a number of depth map samples when the room is empty. This data is re-projected using the camera calibration matrices as a single fused point cloud. A background mesh is generated without users in the room, using an offline Poisson surface reconstruction [Kazhdan et al., 2006]. When users enter the scene the reference background frame is used to extract the foreground. This data is smoothed using an edge preserving bilateral filter. Normals and a polygon mesh are computed. This technique processes 640x480 pixels from each camera in real-time (at 30FPS the framerate of the Kinect).

6.2.3 Projector Tracking

In order to track the 3D location of the handheld projector within the space, it is covered with retro-reflective tape and leverages the fact that the Kinects structured light pattern will appear much brighter in the 2D Kinect IR image when reflected off the projector. This allows the projector to be located within the 2D IR image, as pixels will have very high intensity values (see Figure 6.1C). Depth measurements are still reported in these locations of the depth map.

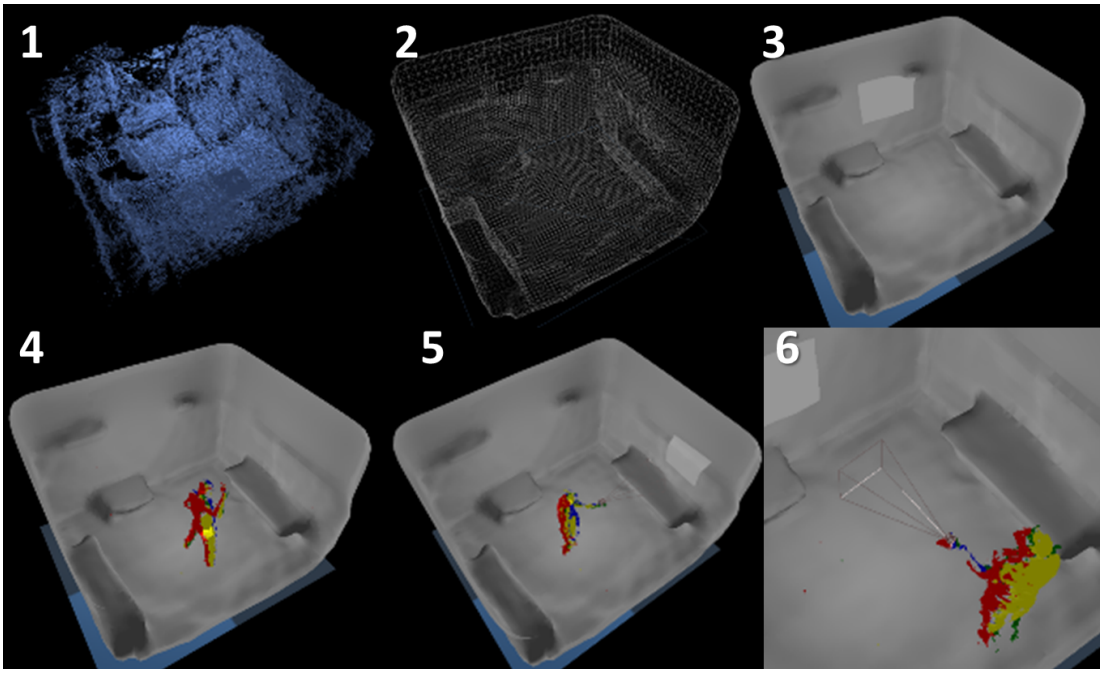

Figure 6.2: The vision pipeline: 1) Aligned point-clouds from each camera. 2) Meshed representation (shown in wireframe) using Poisson Surface Reconstruction. 3) Fully shaded mesh. 4) Foreground segmentation (Note: color of foreground object is composed of different colors representing each camera). 5 and 6) Tracked projector with frustum and projection rendered.

Triangulation of the objects position in 3D space is performed by first binarising each image from the four cameras with a threshold, performing closing and dilation morphology operations to first remove noise then join regions, then extracting the largest connected component. As the 3D location of each of the pixels associated with the largest connected components can be obtained from the depth map, rays from the camera center can be projected through each of these 3D points for each camera and calculate intersections. Then all ray intersection locations are stored and the center of mass of this 3D point cloud used for the location of the projector. A Kalman filter is subsequently used to increase robustness to multi-camera occlusions (which can otherwise cause brief loss of tracking) and reduce jitter.

This Kinect-based infrastructure provides the projector system with a coarse surface representation of the entire room geometry, as well as its absolute location (3DoF) within the space. To sense the orientation of the device, and of course provide projection output, a prototype handheld device was designed (shown Figure 6.1E) that is coupled with our Kinect-based sensing infrastructure. This uses an Aaxatech L1 Laser pico-

projector with 800x600 pixel resolution. A Microstrain 3DM-GX3 IMU is mounted above the projector and generates device orientation estimates at a rate of 500Hz.

6.2.4 Environment-Aware Projected Content

With the 3D orientation data from the IMU and the 3D position tracked from the Kinect-based infrastructure, the 6DoF pose of the projector can now be determined within the virtual reconstruction (see Figures 6.2 and 6.3).

The addition of coarse surface geometry of the room, allows the system to determine which prominent physical surface the user is pointing at with the projector (e.g. table, floor or wall). Virtual digital content, such as 2D images, can now be associated with any surface of the reconstructed room mesh by simply projective texturing onto the 3D model. Once the 3D model is textured with this virtual content, the projector can be used to reveal the content, when the device points directly at that region of the reconstructed 3D model.

This allows a flashlight-like metaphor as in Cao and Balakrishnan [2006]; Cao et al. [2007]; Teller et al. [2003] to be implemented very easily with the system as shown in Figures 6.3. However, this carries the additional benefit that the given that the surface geometry is known, the projected content can be automatically corrected to account for off-axis projection (in contrast to existing systems which require a specific manual calibration step e.g. Cao and Balakrishnan [2006]; Cao et al. [2007]).

6.2.5 Freehand Shadow Interactions

Beyond associating content within the environment using a geometry-aware flashlight metaphor, the RoomProjector also allows for novel freehand user interactions. It does this with a novel fusion of on-board and infrastructure-based sensing. Two 950nm IR LEDs (which do not interfere with the 830nm wavelength of the Kinect) are mounted in the projector case either side of the projection aperture (see Figure 6.1E). A monochrome IDS UI- 1226-LE camera with 752x480 pixel resolution, 60Hz frame rate and 60 wide-angle lens is used for sensing objects in front of the device. The optical axes of the projector and camera are coaxial an IR hot-mirror, mounted directly in front of the projector lens and angled at 45 redirects IR light from the scene sideways into the camera. Mounting the projector and IR camera coaxially allows the projector-camera

relationship to be represented with a one- off projective homography calibration to account for small offsets and scaling without the need for full 3D pose recovery of the projector relative to the camera.

This IR camera senses the presence of hands of the user interacting in front of the device. This allows the system to support novel forms of interaction for a handheld projector by reacting to different gestures made in front of it. Of course, when a user places their hand in front of the device to gesture, a real shadow is naturally cast onto the projection. The coaxial nature of the optics means that 2D camera image (which shows nearby IR reflective objects such as hands) exactly maps the shadow that will be cast onto the 2D projected image.

Using this technique it is possible to create shadow-based interactions which effectively enable indirect interaction with projected content at a distance. This concept is illustrated in Figure 6.3A and B where it is shown how the virtual shadow can be used to perform physics- based interactions such as controlling virtual balls, which respond to collisions with the shadows as if they were real.

This technique first segments the hand from the background using a Gaussian blur, binary threshold and closing morphological operations to close holes. The resulting binary mask image is then down-sampled, and for each foreground pixel a static rigid body is created in a 2D physics simulation. These rigid bodies interact with the other dynamic objects, such as the virtual spheres in the physics simulation.

While the use of shadows have been discussed in other related work [Cowan et al., 2011; Krueger, 1991; Naemura et al., 2002; Xu et al., 2006] by leveraging physics, projection onto the hands and fingertip tracking, the following sections demonstrate a number of simple yet compelling techniques that further capture the natural affordances of real-world shadows for interaction.

6.2.6 Shadow Menus and Fingertip Gestures

A natural extension to the physics enabled shadow interaction metaphor is to combine simple finger-based gestures. One compelling example is shown in Figure 6.4E. When the user holds their hand directly in front of the projector menu items are associated with each finger and displayed onto the wall above the fingertips. The user then touches their palm with the fingertip, in a manner akin to placing the associated menu item

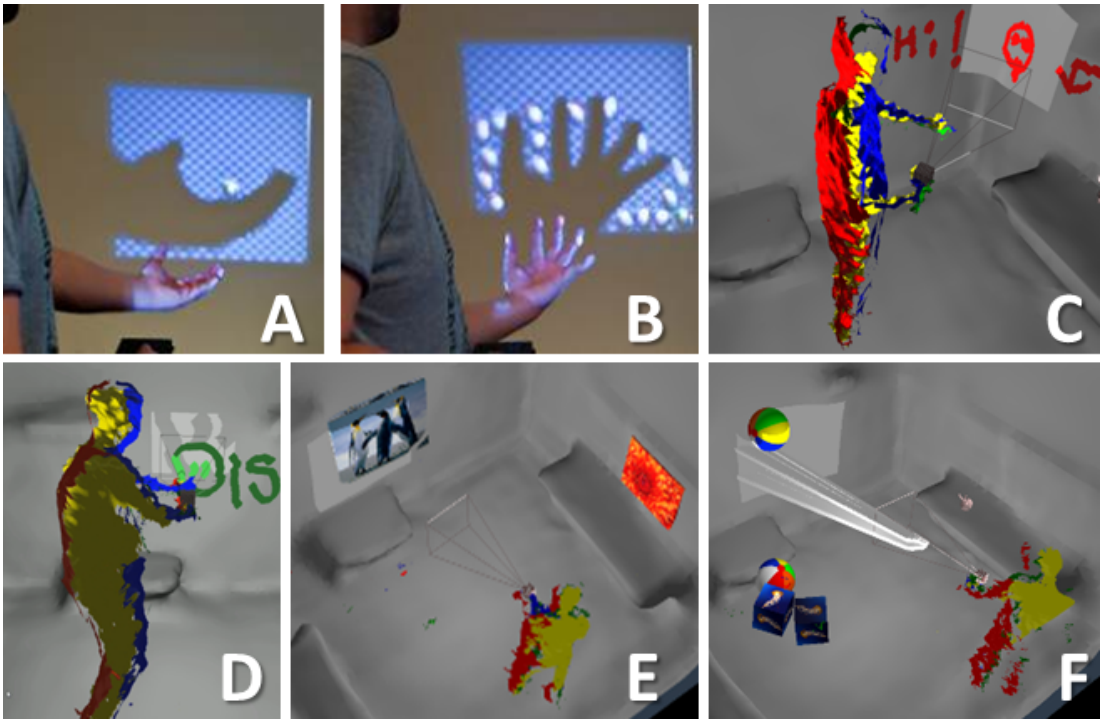

Figure 6.3: Spatially and geometry-based shadow interactions: A and B) User interacts with virtual physics-enabled objects using a real shadow. C) Sensing projector pose, users hands (rendered green), and implementing a flashlight metaphor to enable writing and painting using the real shadow. D) Painting in midair. E) Flashlight metaphor implemented by texturing the 3D model and reveal data through the projector. Note: projection is automatically corrected for the textured surface. F) Debug output showing how physics interactions are enabled within the space through rods raycast onto the 3D model.

in the palm of the hand, to activate the particular menu item. Active menu items are rendered on the palm to provide visible feedback of selection.

Figure 6.4F and G shows one final example of a shadow technique for interacting with a large document using the projector. Here a thumb and forefinger gesture activates and deactivates fingertip-based annotation.

To implement these techniques, once the hand is segmented, fingertips are detected by first tracing the contour around the hand, and then using a peak-and-valley algorithm to label candidate pixels [Sato et al., 2000] as shown in Figure 6.4A-D. Connected component analysis provides 2D coordinates for each fingertip, and these are tracked over time using a Kalman filter. A simple gesture recognizer allows the motion of and distance and angles between fingertips or other state (such as a finger disappearing from view) to trigger commands.

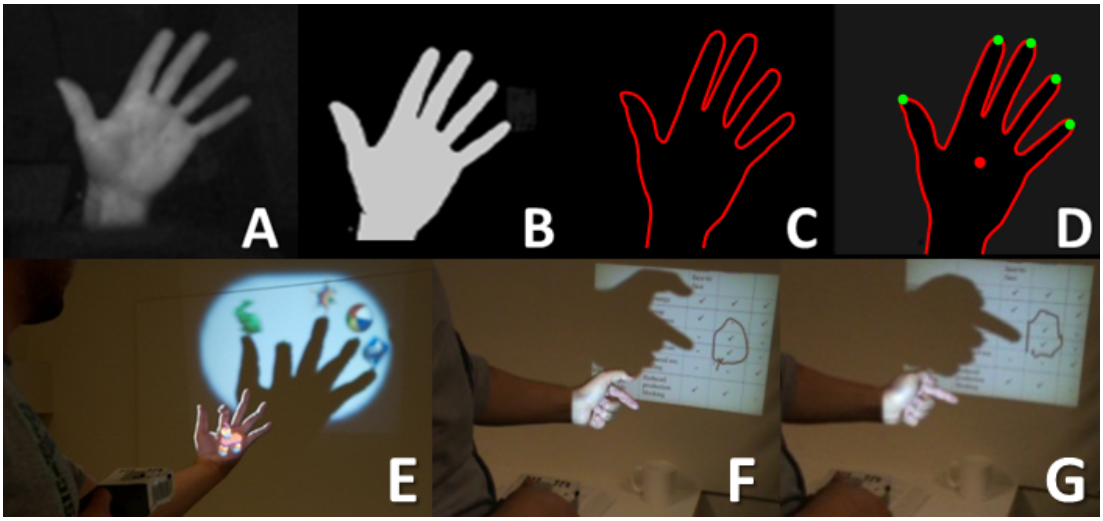

Figure 6.4: Fingertip sensing using onboard IR camera: A) raw diffuse IR image captured by onboard camera. B) image is corrected and binarized. C) the contour of the hand is traced. D) fingertips are sensed using peak-and-valley algorithm. These fingertip locations can be used for enable a shadow menu (E) or finger-based shadow gestures for document interaction (F and G).

6.2.7 Spatially-Aware Shadows

So far these shadow interactions are conducted in the coordinate space of the camera, rather than the global (or room) coordinate space. To enable these interactions to coexist with environment-aware features, which require the Kinect-based infrastructure, we need to fuse the data from the on-board camera and the infrastructure. For certain 2D interactions, for example using the shadow to draw on the projection screen as shown in Figure 6.4F and G, as the 3D pose of the projector is known, the 2D fingertip location sensed using the IR camera can be raycast onto the surface mesh of the room. This allows the system to sense exactly within the room where a shadow is being projected. For example, a continuous stream of point sprite ink can be created and rendered at the raycast 3D location as shown in Figure 6.3C. These annotations remain fixed in the real world, allowing the flashlight metaphor to extend the interaction space beyond the frustum of the projector.

The shadow-based physics interactions can therefore be extended to support more detailed 3D interactions. Instead of using a 2D physics simulation and creating rigid bodies to interact with 2D objects, the technique highlighted in Wilson et al. [2008] is used. Here a Sobel filter is run on the IR handheld camera image. Any valid pixels on the contour of the hand will have a rigid, ray-like box object created from the projector

center to a 3D location (which is determined by raycasting the 2D pixel coordinate into the 3D scene and testing for a 3D intersection with the room mesh). This enables the user to perform basic interactions with 3D virtual objects, as these rigid boxes exert a collision force whenever they intersect another virtual object. Hence, the user can pick virtual objects up, hold them, or push them around merely using this shadow, as shown in Figure 6.3F. For other interactions, the true 3D location of the hand is required. To achieve this the users hands must be localized. This is done by taking the segmented foreground and using a machine learning-based classifier for identifying the users hands [Shotton et al., 2011]. The system makes use of the on-board camera image, but now combines this with the hand classifier, which uses the depth data from the room cameras to coarsely provide an estimated location of the hand in front of the projector, as well as a bounding box or sphere around the hand position. This allows the system to either map fingertip estimates from the on-board camera onto the bounding region of the sensed hand. Or alternatively map recognized gestures in the camera image, such as a pinch gesture, with the 3D location of the hand, as shown in Figure 6.3C and D.

6.2.8 Complementing shadow interaction with shadow projection

A problem that exists when rendering 3D objects in the room is that we can only render these using a 2D projection onto available surfaces. Sometimes this projection looks reasonable, whereas other times it is unclear how the 2D projection should appear. This is particularly the case when the 3D object is in mid-air, away from the surface being projected onto but still in the field of view of the projector.

Here we begin to explore the possibilities for better user feedback of arbitrary virtual 3D objects, not by rendering them, but instead rendering the shadows of the object [Naemura et al., 2002]. This effect is demonstrated in Figure 6.5, where the projector is pointed towards a virtual 3D object, a Piñata dangling in mid-air in the room. Moving the projector around this object casts different shadows onto the surface behind, and therefore gives the user a sense of the objects geometry and 3D location without needing to render the full, shaded object which given that it is in the middle of the room would appear incorrect when projected on a wall.

The concepts for revealing 3D objects through their shadows nicely compliment the idea of real interactive shadows. Indeed, it is possible to combine the two concepts. In

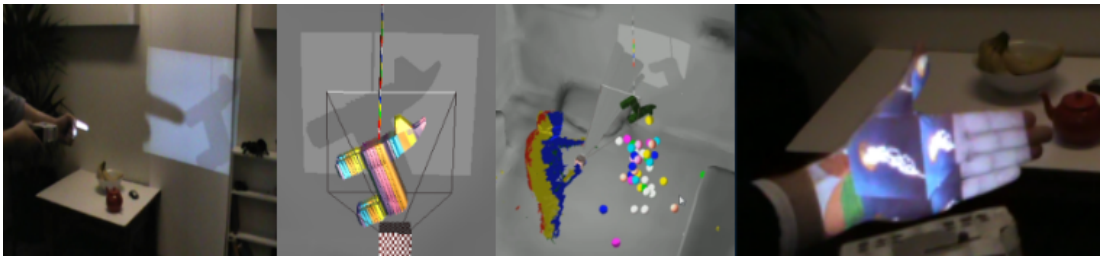

Figure 6.5: From left: Revealing a virtual 3D object by viewing the corresponding virtual shadow rendered onto the projected display. This shadow can be made to interact with the real shadow generated by the users hands. Revealing the hidden scene by projecting an augmented reality view onto the hand.

Figure 6.5 left, we show how the virtual shadow of the object can be knocked side-to-side by the real shadow being cast by the users hands onto the projection. A simple yet effective extension to this technique, inherently supported with the RoomProjector, is for the user to place their hands in front of the projector to reveal a full rendering of a 3D object which was being rendered as a shadow. The hand therefore becomes a kind of viewport or window into the virtual world, as shown in Figure 6.5 (far right).

6.2.9 Beyond infrastructure

In addition to interactions based on the IR image from the handheld unit, the room-based infrastructure delivers some unique ways of enabling wider-scale, room-based interaction. In this sense, the small sensing window of projector is overcome, and user interaction is enabled throughout the entire room. The mesh representation of surfaces in the room may be used to control interactions with virtual objects. The use of multiple Kinect cameras minimizes the sensing (and hence interaction) dead space that would otherwise occur due to occlusions with a single mobile camera. Sensing the environment and distinguishing foreground objects such as the user in the room enables a wide variety of scenarios which are not possible by previous systems such as Cao and Balakrishnan [2006]; Cao et al. [2007]; Wilson and Benko [2010].

The main limitation to the room infrastructure is coarseness. Due to the distance between the cameras and objects and surfaces, only relatively prominent surfaces can be recovered from the scene. The hybrid tracking of the projector can occasionally be noisy and error-prone due to issues of camera occlusions or ferrous objects interfering with the

IMU. This coarseness is formally evaluated later, but led to our second infrastructure-free prototype.

6.3 SLAMProjector System Overview

The RoomProjector system introduced a variety of interaction possibilities. Our second prototype embeds a projector with a mobile Kinect camera, and uses a system capable of building a model of the environment in real-time and simultaneously tracking the location of the device within this reconstructed model [Izadi et al., 2011a]. SLAM systems which support this functionality are common in the AR and robotics communities [Du et al., 2011; Izadi et al., 2011a] and typically use a single camera to reconstruct small scenes and augment them with virtual content in real time.

The SLAMProjector system combines a pico-projector with a single Kinect depth-sensing camera. We leverage a SLAM system to recover the pose of the projector in real-time while simultaneously building a dense reconstruction of the environment [Izadi et al., 2011a]. This system uses depth data only as opposed to RGB, which is useful when working with pico-projectors with low brightness, enabling room lighting to be significantly dimmed.

6.3.1 Infrastructure-free flashlight and enhanced geometry-awareness

The SLAMProjector prototype measures 140x65x53mm and contains the same laser projector as our previous prototype. Also housed are the core elements of an off-the-shelf Kinect camera: the RGB and IR cameras and the IR emitter, which all connect to a single control PCB, as seen in Figure 6.6 (top).

The SLAM system tracks the 6DoF pose of the Kinect device, and simultaneously creates a high-quality 3D model of the scene (for details see Izadi et al. [2011a]). Unlike the RoomProjector, which used a shared optical axis for the projector and camera, we have to calibrate the full intrinsic and extrinsic parameters of the projector and cameras in this system using a known checkerboard calibration pattern.

The flashlight metaphor can be used to accurately re-render content in exactly the same location without the need for any infrastructure, unlike our previous system and the work of others [Brumitt et al., 2000; Cao and Balakrishnan, 2006; Mistry et al.,

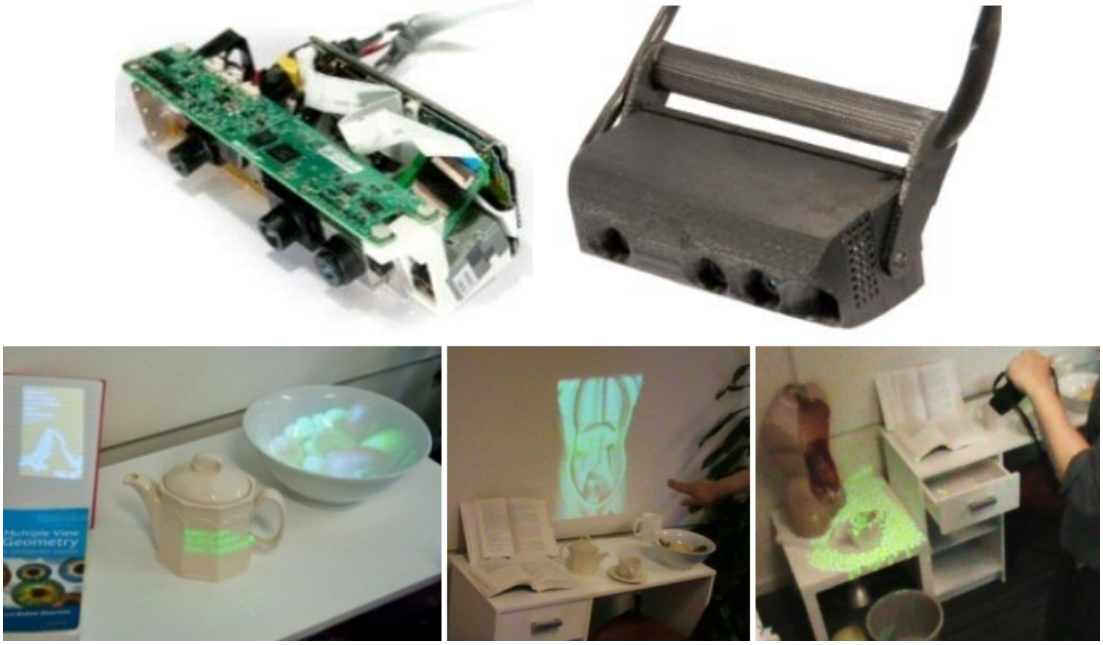

Figure 6.6: Top: SLAMProjector hardware. The main components from left-to-right are the IR emitter, RGB camera, IR camera and projector. Bottom left: Warping projected content onto arbitrary shaped surfaces in real-time. Bottom center: A real-world object is scanned using the SLAMProjector and the 3D segmented object pasted onto a nearby wall. Bottom right: particles interact with the scene and are rendered in real-time using a flashlight metaphor.

2009; Molyneaux et al., 2007]. These digital textures can be warped correctly onto planar objects, but given the 6DoF pose of the device, it is possible to render 2D textures onto a surface with any arbitrary geometry using projective texturing, as highlighted in Figure 6.6 (bottom left).

Perhaps the most unique feature of our system is the ability to quickly acquire and update an accurate 3D model of a dynamic scene in real-time, and in this sense the SLAMProjector is far the more geometry-aware than our earlier prototype.

This 3D model can be re-rendered back using the projector to both act as feedback to the user of the underlying SLAM system (e.g. to show the extent and quality of the reconstruction tracking and inaccuracies) but also as a mechanism for coupling interaction with output. One example is shown in Figure 6.6 (bottom center). Here the user can touch any object to automatically segment a high quality model from the plane. Once these segmented objects are acquired multiple virtual copies can be generated in any arbitrary location the user wishes. In this example, the user pastes a 3D scan of a model human torso onto the wall.

Figure 6.7: Left and Center Left: Painting on any surfaces as in Izadi et al. [2011a] but with coupled output. Center Right: Virtual buttons are triggered by touch if the projector is close to the surface (implying the user can reach the surface) or (Right) by shadow if the user is at a distance.

6.3.2 SLAMProjector Interactions

We believe that the ability to couple projector output to user interactions such as those in Izadi et al. [2011a] can be compelling. For example, in Figure 6.7 we show how a user can paint on any surface, using mechanisms described in Izadi et al. [2011a]. Adding the direct output of the projector makes this more natural and gives the user direct feedback. Similarly, in Figure 6.6 where the particles fall onto and into arbitrary surfaces, the projection gives them a more fluid feeling.

However, this scenario is different to the painting example, in that we typically paint onto surfaces, whereas 3D particles can exist anywhere within free space. As highlighted in our previous RoomProjector prototype, such 3D renderings cannot be handled using a 2D projection effectively. So while the SLAMProjector may have the greatest 3D capabilities, at times the fidelity of the input cannot be mapped naturally to the fidelity of the output - the projection itself.

One of the interesting possibilities that can be explored with the SLAMProjector is the transition from direct multi-touch interaction to the indirect interactions outlined in the previous sections. Multi-touch is an intuitive way of interacting with objects up close and within easy reach, but in a larger room indirect shadow interactions may be preferred. In the example shown in Figure 6.7 (right), the 3D model is used as a means of calculating how far the device (and hence the user) is to the projection. If the projection distance is below a threshold (typically around 1-1.2m), the user can interact directly using multi-touch gestures. As the device is moved further away from the projection surface, the interaction automatically switches to indirect interaction. In this example, we demonstrate how this implicit mode switch can be used to change

between shadow touch for activating a virtual button, to direct touch, simply by moving the projector closer to or further from the surface. Note how the labels on the virtual buttons change to reflect this.

6.4 System Evaluation

We have shown two very different prototypes for augmenting indoor pervasive computing spaces using interactive, environment-aware handheld projectors, each enabling different interaction possibilities. So far we have qualitatively introduced and discussed the two systems based on their sensing capabilities. Furthermore, we have highlighted the interaction techniques enabled by each systems unique capabilities.

Both systems rely on information about their 3D pose and geometry of the environment in order to correctly display spatially registered graphics. Both systems also depend on sensing human input for interaction. In this section we detail initial findings from a number of experiments we conducted to evaluate tracking and input accuracy. Tracking accuracy is of great importance in terms of user experience, especially for the kind of spatial interactions described even small error in pose noticeably reduces visual quality when renderings are not aligned with the real world. Likewise, input accuracy directly impacts user experience a system that does not respond correctly to input will clearly be frustrating to use.

6.4.1 Tracking accuracy

The first experiment compares tracking accuracy of each system. We use an 8 camera Vicon motion capture system as ground truth. A test scene is constructed on a circular tabletop in the center of the room and a gantry fitted to the table with a tiltable projector mount attached. A controlled circular motion around the tabletop can be performed repeatedly for both systems. We alter the mounting height and angle of the projector relative to horizontal to simulate realistic positions and orientation in respect to the tabletop ‘projection screen.

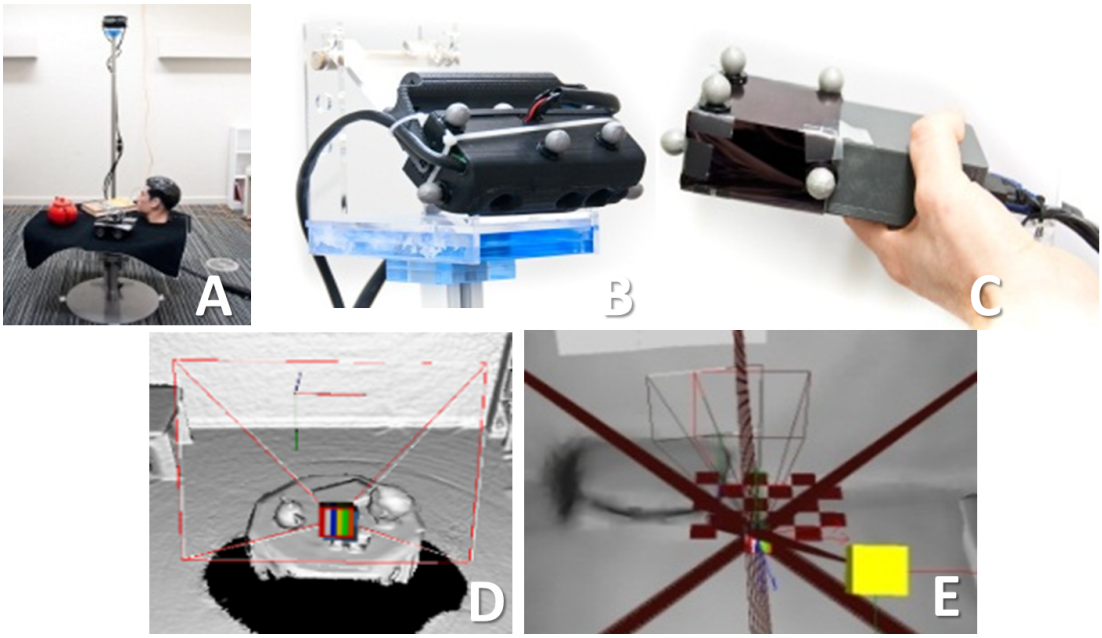

Figure 6.8: A) Experimental setup showing test scene and gantry. B) SLAMProjector with Vicon markers. C) RoomProjector with Vicon markers. E) RoomProjector exhibits good positional accuracy but rotation is erroneous (grey frustum is rotation as reported by our prototype, red is ground truth). D) SLAM Projector provides better rotation estimates but is prone to drift over time.

6.4.2 Procedure

Each projector was moved around the tabletop at walking speed once at each of the three mounting heights above the ground 0.6m, 1.1m and 1.6m and at each of the tilt angles of 0, 22.5, 40 and 45, resulting in 12 total revolutions around the tabletop. The table contains various small objects (e.g. a book, a toy car) in arbitrary locations on the table. Once placed, these objects remain static for all the experiments. Figure 6.8 shows the experimental setup and projectors with Vicon markers attached. For both prototypes we compute a calibration matrix to align the respective world origins with the Vicon coordinate system. We synchronize pose and orientation streams from all three systems, which are sampled at 30Hz (the Kinect camera update rate).

6.4.3 Results

Figure 6.9(a-c) summarize result from this initial experiment. For the RoomProjector we can see that the error in position is relatively low along all three axes with a

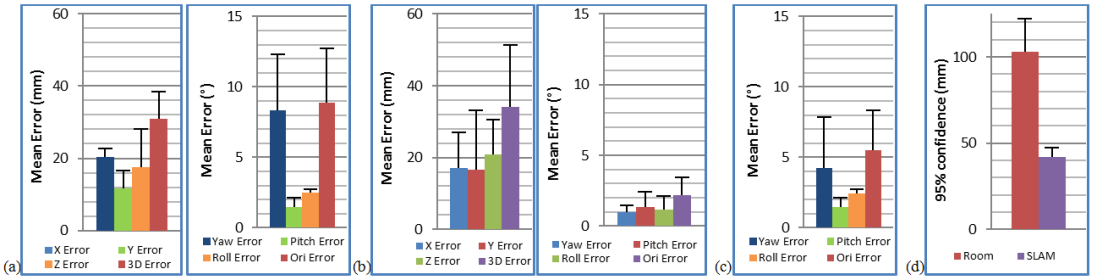

Figure 6.9: Mean error, relative to ground truth, split into location and rotation error in each dimension and combined error for (a) RoomProjector and (b) SLAMProjector system. (c) Mean orientation error of RoomProjector, 3D magnetic calibration applied, (d) Button diameter required to encompass 95% of touches for Room Projector and SLAM Projector over all participants and targets. Error bars show SD.

Mean error of $30.9mm, SD = 7.6$ in 3D location. In contrast the orientation error is relatively high with a combined Mean error of $8.9°, SD = 3.8$. However, when this error is decomposed into the rotations about individual axes it becomes apparent that the yaw error dominates ($M = 8.3°, SD = 3.9$). This can be explained by magnetic distortions in the room. Yaw is measured by the IMUs magnetometer while pitch and roll are from the 3-axis accelerometer. To quantify how significant this effect was, we performed a one-off full magnetic 3D calibration by measuring the yaw error when the projector was lying horizontally and aligned with the room yaw origin using a 3D grid at 0.5m intervals throughout the whole room, then re-performed the experiment. As can be seen in Figure 6.9 (c), applying the nearest neighbor calibration to the projector orientation during the experiment almost halves the yaw error ($M = 4.2°, SD = 3.6$), however, unless continually updated, this method of calibration only works well for static environments as any ferrous objects moved inside the room will introduce new error.

In contrast, the SLAM system provides comparable but slightly worse positional accuracy ($M = 34.0mm, SD = 17.2 vs. M = 30.9mm$). A paired t-test reveals that the difference is not statistically significant ($p > 0.05$). However, the SLAM system does provide better rotational accuracy ($M = 2.2°, SD = 1.3 vs. M = 8.9°$ Mean orientation error), and here a paired t-test shows statistical significance ($p < 0.05$).

When compared qualitatively, despite higher latency, the SLAMProjector provides much tighter visual integration of real-world and projected graphics. Due to the projective nature of the projector systems, the large Mean orientation error in the

RoomProjector is one explanation, as a few degrees error results in visually significant offsets the further away the projection surface is. This makes orientation error very noticeable when the graphics need to be tightly coupled with a physical surface. In contrast, 3D location error is visually much less apparent, with small lateral shifts due to 3D location error appearing as a minor offset from the true projection position, but still in the right area of all but the smallest projection targets.

It is interesting that the location accuracy of the RoomProjector is very good making this system an interesting candidate in scenarios that only require 3D position, or using this tracking method in combination with a better source of rotation data than an IMU. However, in practice the room is currently limited to only track one object as there is no easy way to distinguish between multiple retro-reflective objects e.g. fiducial markers would have to be significantly larger than the projector to be detected over typical working distances of 2-6m. One possible solution would be to correlate movement seen in the IR cameras with motion sensed by the IMU, as shown in Molyneaux et al. [2007].

The tracking in the RoomProjector system is also stateless, i.e., each frame a new location is computed without considering the previous frames. However, the SLAM system tracks off the model it is building, hence pose error incorporated in the model is cumulative and 3D location in particular can potentially begin to drift over time. Typically this occurs when the majority of the frame is taken up by only one or two planar surfaces (e.g. if it sees only one or two walls). More experiments are necessary to fully quantify this issue.

6.4.4 Touch Accuracy

One of the compelling possibilities of geometry-aware projector systems is to enable multi-touch interaction on arbitrary surfaces. Both our systems provide touch input but at very different levels of fidelity. To quantify these capabilities and differences we conducted a second experiment investigating touch accuracy across our two systems.

For the SLAM system we use the touch detection technique in Izadi et al. [2011a]. This technique robustly detects and tracks multiple fingertips based on the distance to reconstructed surfaces. For the RoomProjector we detect touch by initially looking at a 3D estimate of the users hand using the technique outlined previously (as in Shotton et al. [2011]). This position is noisy and centered on the hand rather than individual fingers,

which are not resolvable. We attempt to refine this 3D position by combining it with the onboard IR camera as described previously. This projects fingertip positions sensed in the 2D IR camera into the 3D scene by ray-casting through the projector center until the finger disappears (when it is close to the surface it has the same illumination brightness as the surface and hence cannot be segmented). We decide whether a finger touches a target based on the intersection of the last detected ray with reconstructed surfaces. We compare the distance along the ray to the intersection point with the distance of the users hand as measured by the fixed Kinect cameras.

Procedure. Ten virtual targets are positioned on physical surfaces in the room; 7 targets were planar (3 vertical, 4 horizontal) and 3 non-planar in various orientations (e.g on a large beach ball, or life-size head model as shown in Figure 6.8A). These positions are fixed and defined in world coordinates. The actual target is projected onto the real world, similar to buttons or other UI elements in a real world application. We place physical markers at target locations, enabling participants to easily find the sparsely distributed targets in the room. To isolate touch accuracy from tracking error we use the 6DoF pose from the Vicon system for both systems, while still detecting touch with the projector.

8 users (5 male, 3 female) between the ages of 24 and 43 years were recruited to perform this target acquisition task. Each participant performed 3 blocks of 10 target acquisition rounds, where each button was shown until a click was recorded. The presentation order of conditions was counter-balanced. Presentation order of targets was sequential as we measure accuracy, not task completion time, hence motor memory and other learning effects play less of a role. Participants were asked to be as accurate as possible with their touches.

Results and Discussion. The participants produced 240 clicks, on the 10 surfaces. Our results represent the real-world performance of our system and hence the cumulative error of system and user. This error can be decomposed into 3 sources error in the calibration between the projector and camera system, error in finger detection location (the dominant source of error), and user error when clicking targets. Over all users and targets the SLAMProjector performed significantly better than the RoomProjector ($Mean3DError = 33.8mm, SD = 5.5, versusM = 75.9mm, SD = 19.0$).

Figure 6.9(d) shows the button diameter which consistently encompasses 95% of user clicks.

When comparing the results for SLAMProjector with Du et al. [2011], at first glance, our 95% results appear worse than even their far distance condition. However, the far results reported in Du et al. [2011] are for shoulder to arms length distances (average 40-60cm), which is closer than typical SLAMProjector sensing distances, which average 70-100cm due both to the minimum sensing distance for Kinect and also as the SLAMProjector is held in the opposite hand. Hence we believe our results for SLAMProjector are consistent with an extrapolation of those from Du et al. [2011], but with the additional benefit of fully non-planar touch detection.

6.5 Discussion

To further explore the sensing fidelity of each system, and in consequence the interaction fidelity, we have performed two experiments quantifying tracking and input accuracy. As shown in Table 6.1, when comparing our systems with the state of the art we achieve better fidelity of sensing for both spatial and geometry awareness and touch sensing with SLAMProjector comparable to the current state of the art [Du et al., 2011] for touch, but also offering the addition of non-planar multi-touch and spatial awareness.

The RoomProjector prototype is an interesting system to compare to the others as it has similarities to Cao and Balakrishnan [2006]; Cao et al. [2007]; Raskar et al. [2003, 2004], as they support 6DoF tracking. The Vicon system used by Cao offers more accurate tracking than our Kinect-plus-IMU system, but the study showed our system provides relatively accurate location data, hence can be useful in scenarios where a more expensive and intrusive tracking system cannot be used or where orientation is less important. Furthermore, the use of Kinect allows the coarse geometry of the room to be sensed for free which is one step beyond, Cao's system, which requires users to define planar spaces interactively. The added fidelity allows for richer interactions such as automatic geometry distortion correction, whole body interactions with physics objects, shadow interaction with 3D objects, and more accurate registration between the virtual and the physical.

The final SLAMProjector prototype is the most advanced in terms of sensing fidelity, and also removes the need for infrastructure. It provides stand-alone tracking

Table 6.1: Comparison of the portable projector systems in related work.

	Spatial-aware	Geometry-aware	User Input
RoomProjector (infrastructure, high computation, non-mobile)	-6DoF, 31mm,9° mean 3D pose accuracy from infrastructure	-planar and non-planar -background capture step -coarse surfaces	- whole body -hand gestures and fingertip recognition (@50cm dist. max)
SLAMProjector (infrastructure-less, high computation, mobile)	-6DoF, 34mm,2° mean 3D pose accuracy, stand-alone	-planar and non-planar -no user in the loop -fine surfaces	-hand gestures at arms length -finger touch 41.7mm button dia. for 95% detection at 0.7-1m -3D sensing
Cao and Balakrishnan [2006]; Cao et al. [2007] (infrastructure, medium computation, non-mobile)	6DoF -high accuracy from infrastructure	-multiple planar surfaces -requires user in the loop for surface definition	-physical button -device motion
Harrison et al. [2011] (infrastructure-less, low computation, mobile)	None	- planar surfaces at $< 1m$ distance	-finger touch detection, 31-38.5mm button dia. for 95% detection at arms length (far)
Raskar et al. [2003, 2004] (infrastructure, medium computation, mobile)	-6DoF from fiducial markers or active tags and tilt sensing	-single planar object surface from active tag or fiducial tagged surface	-physical button -device motion
Mistry et al. [2009] (infrastructure-less, low computation, mobile)	None	None	-hand gestures and fingers with coloured caps only

comparable with RoomProjector, yet provides far higher geometry awareness than the RoomProjector or any of the related work. In terms of gestural input, the users hands can be segmented and used to implement both multi-touch and indirect shadow interaction.

However, this does not simply mean that infrastructure-based systems should be replaced by the SLAMProjector. One unique capability that RoomProjector offers (which is not exploited by other infrastructure based approaches) is the ability for this system to sense beyond the frustum of the projector-camera system. Here, the system can coarsely sense humans within the whole space and allow for whole body interactions. For mobile systems, it is rare to be able to capture the entire user while

using the system to track and map. Another issue with the SLAM system when used for shadow interaction is that if the user is occluding a large part of the Kinect depth image this can degrade the tracking quality of the projector. Drift can also occur on rapid movement or when pointed at a mostly planar scenes with little varying depth.

There is, however, another important issue that surfaced, and that is whether the fidelity of 2D projected output falls significantly behind the 3D sensing. The painting examples work perfectly for both systems when painting directly onto a surface, with the ink appearing immediately in place. However, when painting in 3D the output fidelity clearly cannot match the sensing. Hence there are two interesting avenues that emerge in terms of future interaction research. The first is exploring how to leverage other feedback mechanisms to reveal 3D scenes using inherently 2D output. Shadows and revealing the world through the users hands are two, but there are certainly others. The other is technical, exploring the use of technologies to overcome the limitations of 2D projection, coupling input fidelity more closely with output. Here mobile stereo projection, or video see-through phones coupled with projection are interesting routes for exploration, as are more traditional forms of augmented reality, such as optical see-through HMD or head mounted projection.

6.6 Summary

In this chapter I presented two prototype environment-aware handheld projector systems for augmenting indoor computing spaces and providing users with interactive interfaces anywhere within a room. Both systems have interesting design characteristics and interaction possibilities. RoomProjector is more infrastructure heavy than SLAMProjector. However, both offer unique possibilities and are novel systems in their own right. We introduced novel gesture-based interactions with mobile projectors through a variety of hand gestures in front of the projector. These include shadow-based interactions that utilize both real and virtual shadows as well as the use of hands for an AR view of content in 3D space.

In many extents these systems offered the tightest coupling of AR and NUI. However, environment sensing was arguably at a higher fidelity than the user interaction. Much of the NUI interactions were still limited to either 2D sensing or coarse rigid body tracking of hands. As highlighted in the related work chapter and RetroDepth and

HoloDesk systems, hands are extremely dexterous, but we have yet to exploit this dexterity in a truly mobile AR scenario. In the next chapter we tackle this fundamental challenge to enable full articulated hand tracking for mobile AR scenarios.

Chapter 7

Freehand 3D Interactions Anywhere

7.1 Introduction

In this chapter I explore the far end of the mobility axis, and focus on leveraging the full expressiveness of our hands for input and output in fully mobile AR scenarios. Our hands are extremely dexterous, making them the primary mechanism to manipulate and interact with the physical world. Understandably as presented in Chapter 2, a considerable focus of NUI research has been in transferring such 'natural' hand manipulations into the digital domain. However, current user interfaces rarely leverage the full dexterity of our hands. This is largely due to the technical challenges in sensing the full 3D pose of the hand, with its many DoFs.

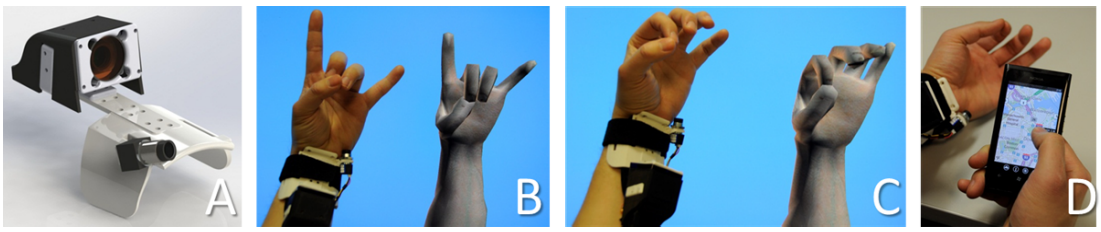

Figure 7.1: A) Digits is a wrist-worn sensor for freehand 3D interactions on the move. By instrumenting only the wrist, the user's entire hand is left to interact freely without wearing a data glove. B and C) Digits recovers the full 3D pose of a user's hand. D) Spatial interactions using a mobile phone and Digits.

Whilst sensing the full 3D pose of the hand is becoming more tractable, particularly with the advent of consumer depth cameras, there are certain domains where it still

remains a fundamental challenge. The mobile domain is one such example, and the focus of this chapter. In mobile settings, computational cost, power consumption, form-factor, everyday use, and self-containment are all key requirements. This makes hand tracking solutions impractical that require the sensor to be embedded in the environment, leverage currently bulky and power inefficient depth cameras, or that require the hands of the user to be fully covered.

This chapter explores this very challenge and enables dexterous 3D hand interactions on the move. Our system, called Digits, is a compact form-factor wrist-worn device, built using easily reproducible off-the-shelf hardware, which is orders of magnitude lower power than current depth cameras. The sensor is fully contained on the body, requiring no external sensing infrastructure in the environment, and allows diverse bare hand interactions on-the-move, liberating users from any fixed devices or locations and allowing them to be truly mobile.

This focus on mobile use does not come at the cost of sensing fidelity. We present efficient processing techniques for robustly detecting features of the hand using the wrist-worn device, and new algorithms, which leverage knowledge regarding the biomechanical constraints of the hand to recover a fully articulated 3D model of the hand.

We leverage the recovered 3D hand pose to recognize diverse hand gestures, including discrete and continuous 3D gestures. We demonstrate interactive scenarios, including: eyes-free spatial interactions on the move, in-air 3D interactions around the periphery of the mobile device, combinations of on surface and above the surface input on mobile phones, controlling large displays from a distance using freehand gestures, and immersive 3D gaming experiences.

Design Considerations As illustrated by the breadth of the related work on NUI, in Chapter 2, this is a rich and challenging design space. With Digits we aim to bring some of the high fidelity sensing found in data gloves and environment-based vision techniques to mobile settings. This requires the full recovery of 3D articulated hand poses in real-time without the requirement to wear a glove. We strive to build a system that is more practical than using today's depth camera technologies, which as outlined carry serious limitations for mobile use. We identify an interesting area in this design space and focus on a wrist-worn device to recover the full 3D hand pose, but do this with only a 2D camera.

We build upon learnings from the wearables literature that have shown the benefits of wrist-worn devices [Howard and Howard, 2001; Kim et al., 2007; Lee et al., 2011] for supporting eyes-free, always-available interactions [Ashbrook, 2010], as well as overcoming some of the limitations of other sensor placements, including occlusions of the hands by other body parts, imposing interactions that are overly performance-centric (instead of private or subtle [Bailly et al., 2012]), and physically constraining the interaction space due to line of sight requirements of the sensor.

7.2 System Overview

Digits is a small camera-based sensor attached to the wrist that optically images a large part of the user's bare hand. The camera is placed so that the upper part of the palm and fingers are imaged as they bend inwards towards the device. Two separate infrared (IR) illumination schemes are used to simplify signal processing. The use of IR allows the illumination to be invisible to the user, and offers a level of robustness to ambient visible light. Both illumination schemes use low cost, readily procurable, and low-power components.

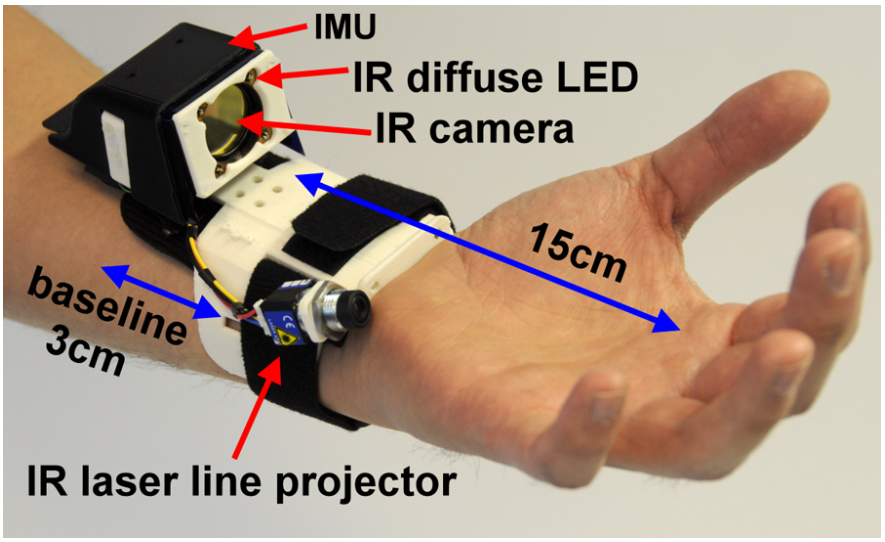

Figure 7.2: Digits main hardware components attached to a wrist brace.

First, an IR laser line generator projects a thin IR line across the user's hand which intersects with the fingers as they bend inwards [Villar et al., 2009]. This approach

can be used to robustly sample a single 3D point on each of the fingers and thumb. From these five sparse samples, and by exploiting biomechanical constraints, we derive a forward kinematics (FK) algorithm to reconstruct a fully articulated hand skeleton. This initial approximation allows us to support a variety of 3D hand poses as shown in Figure 7.9. However, each fingertip is essentially reduced to a single dimension of input, limiting its motion to curling either towards or away from the sensor (see Figure 7.7B).

To more faithfully replicate the pose of the hand, more features need to be sampled on the hand. A complimentary method uses a ring of modulated IR LEDs to uniformly illuminate the user's hand. We demonstrate how to robustly extract the 2D positions of fingertips from this IR image, with an associated coarse depth estimate, by modeling the light falloff from the IR LEDs. This fingertip sensing approach can be coupled with the laser line sensing method to derive a new inverse kinematics (IK) based algorithm for computing the full joint-angle configuration of the hand. This method allows for even more realistic reconstructions of the hand as shown in Figure 7.12, resulting in higher DoF input.

These methods work together to help constrain the otherwise ill-posed problem of recovering the full 3D hand pose from a 2D image. Because each method may be useful by itself, we keep the description of the two approaches separate. This also aids in understanding the underlying concepts, techniques and algorithms presented in this chapter. Finally, an inertial measurement unit (IMU) can be used to approximate wrist and forearm motions in 3D (see Figure 7.2).

Digits is primarily a new type of sensor and an enabling technology for interaction, however before describing the system in full, we demonstrate its interactive capabilities.

7.2.1 Interactive Scenarios

We have explored a number of interactive scenarios enabled by our system which we briefly illustrate here. Each scenario looks at a different configuration of output coupled with Digits. The first scenario looks at spatial interaction with a situated display (see Figure 7.3A+B). For example, interacting with a TV at home or a large public display. Here the user interacts at a distance using a Digits device. Application scenarios can include gaming or CAD, where the user can perform a variety of continuous or discrete hand gestures to support spatial navigation, pointing or selection in 3D (Figure 7.3A+B).

From this tracked 3D hand model, discrete gestures can be robustly recognized by looking at the joint angle configuration (Figure 7.3C).

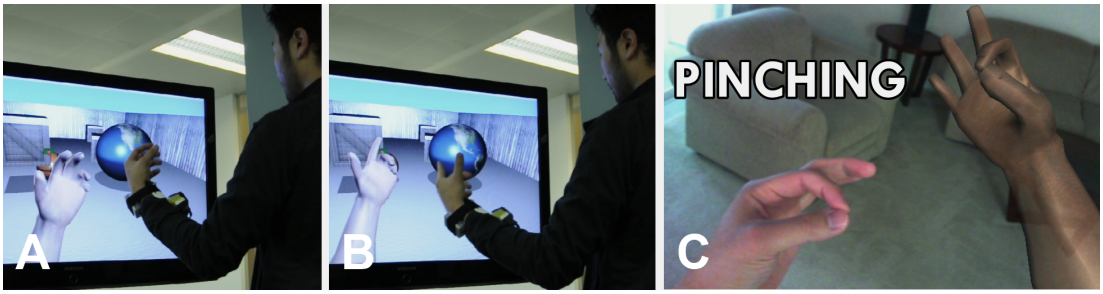

Figure 7.3: Illustration of potential Digits application scenarios. A+B) Continuous interaction with 3D content on a large display. C) Gesture recognition performed on reconstructed hand model.

Mobility was a strong motivation in our work. We envision Digits will expand the physical interaction area of mobile devices beyond the display. In one application scenario (see Figure 7.4A+B) the user holds and interacts with a tablet (or phone) using the dominant hand and uses the non-dominant hand to provide 3D input to the application. For example, semantic zooming is initiated with an in-air pinch gesture, and the zoom factor is controlled with the remaining digits.

An interesting possibility here is to support on-screen interactions and simultaneous freehand interactions. For example, dividing between fine-grained interactions on the touchscreen and coarser navigation tasks using the non-dominant hand and Digits. Our tracker is gloveless and enables the user to interact with the mobile device and operate Digits with the same hand. This allows standard 2D touch gestures to be coupled with above-the-surface interactions in 3D. For example, for quickly changing the Z-order of a selected object by first touching the target and then performing an in-air pinch-based zoom.

Finally, the display could be removed entirely in an eyes-free interaction scenario. The 3D input capabilities of Digits can allow spatial interactions with invisible UIs, such as dials, sliders, or buttons without visual output (see Figure 7.4C+D). For example, we can leverage both proprioceptive knowledge and spatial memory to allow the user to set the volume on a mobile phone by directly reaching out and interacting with a virtual dial; turning their hand to the right of the body and performing typing gestures on a virtual number pad to place a call; or moving the hand to the left of the body and

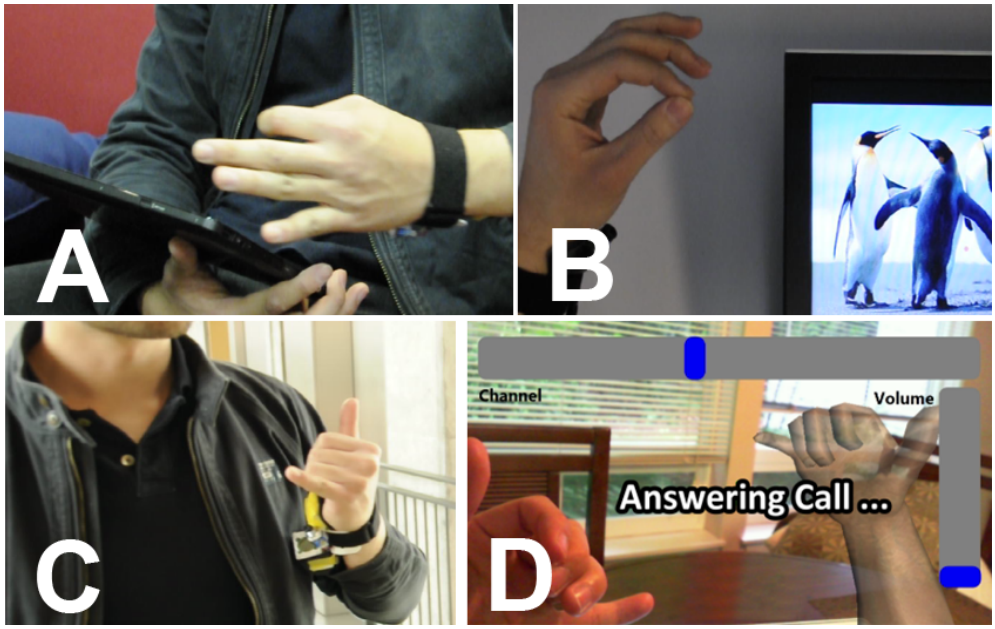

Figure 7.4: Digits application scenarios. A+B) Extending interaction space around a mobile device into 3D. C+D) Non-visual UIs allow users to manipulate application parameters without looking at or touching a physical device (GUI elements are for illustration only).

touching their thumb and individual fingers to activate other phone functions. One interesting possibility here is to detect the type of action by the initial 3D shape of the hand. For example, if the user requires to change the volume, they simply configure their hand as if they are holding a virtual dial, which can then be rotated to set the desired level.

These scenarios illustrate the utility of Digits as a general purpose platform for a variety of hand-based interactions. In the next sections, we describe the system implementation in full, focusing on how we sense features of the hand and from these build different articulated models of the hand, with varying levels of fidelity.

7.3 Sensing Hardware

The Digits hardware is shown in Figure 7.2. A PointGrey FireFly MV IR camera (640x480 resolution capturing frames at 90Hz) is attached to a wristband worn on the anterior (inner) side of the wrist. An IR laser line generator (Gated Cameo 1260 from Global Laser) operating at 850nm, with 105° angular spread is positioned at a fixed

baseline from the camera. 4 diffuse IR LEDs (OSRAM CHIPLED SFH 4053) again operating at 850nm are attached around the lens of the camera. Finally, an IMU (x-IMU from x-IO Technologies) provides absolute tri-axis orientation data of the forearm at 120Hz.

The hardware is designed to be simple, easily reproducible with off-the-shelf components, and low-power. The setup is powered entirely over USB, both laser and the 4 LEDs powered and driven by strobes from the camera General Purpose Input/Output (GPIO) pins. This results in a total power draw of less than 400mW (16mW for the laser, 60mW for the IR LEDs, and 300mW for remaining camera hardware). This compares favorably to the 3.4 to 5W consumption of the current generation Kinect cameras. A Digits device weighs around 75g, 124g with the wireless IMU, and is lighter than standard watches with a metal wristband (160-180g). The device is attached to the forearm with a 2cm wide Velcro band around the wrist, and the contact area at the inner wrist is covered with soft padding (5x5cm).

7.4 Signal Processing

The main processing pipeline is broken down into the following steps:

Background Subtraction We reduce the influence of ambient IR light by capturing three consecutive frames from the IR camera under different illumination conditions (giving an effective frame rate of 30Hz). The first frame turns all active illumination off, capturing only ambient IR. This is subtracted from the other two frames; the first with only the laser on, and the second with just LEDs on. As shown, ambient IR light is greatly reduced in our input images e.g. from room lights or sunlight (Figure 7.5).

Image Rectification Rectifies both actively lit images based on a previous intrinsic camera calibration step.

Finger Separation Splits the LED lit IR image into regions that correspond to different unique fingers or thumb.

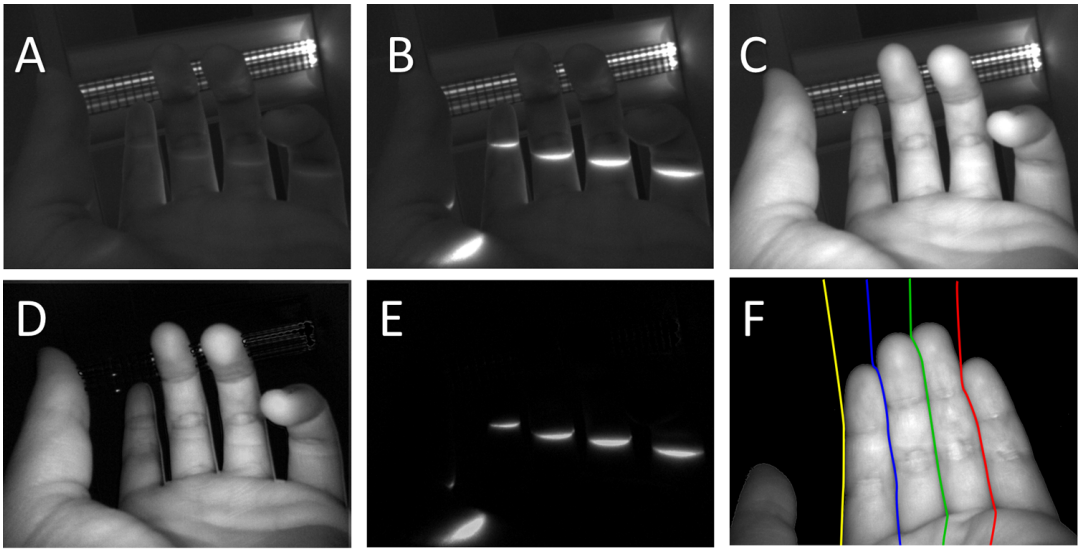

Figure 7.5: Background subtraction. A) Active illumination off. B) IR laser + background IR. C) IR LEDs + background IR. D) Background subtracted IR LEDs. E) Background subtracted IR laser. F) Finger separation via seam carving.

Laser Line Sensing Triangulates the 3D points where each finger or thumb intersects with the laser line generator.

Forward Kinematics These 3D points are passed to a new forward kinematics algorithm, which reconstructs the full 3D hand pose, based on assumptions regarding the biomechanical constraints of the hand.

Diffuse IR Fingertip Detection Additionally, depending on the hand pose, we can also use the LED illuminated image to robustly extract high-quality surface normals and coarse depth estimation for robust detection of fingertips.

Inverse Kinematics The 3D points sensed from the laser and 2D fingertip locations are passed onto a new IK model for higher DoF recovery of hand poses.

Finger separation One of the important initial pipeline steps associates regions of the image with each of the digits of the hand. A technique utilized in *seam carving* [Avidan and Shamir, 2007] is used to disambiguate the main vertical boundary between pairs of fingers in the IR illuminated image. A one dimensional sobel filter finds vertical edges

in the IR image. We detect *valleys* [Malik, 2003] between two fingers as concavities in the traced hand contour. At each valley location, we trace multiple vertical paths along the edges, and use dynamic programming to detect the path with the lowest overall energy by penalizing paths not following the edge (see Figure 7.5F). This method divides the image into five areas each mapped to a unique digit.

Laser Line Sensing The laser line generator projects a horizontal line above the palm that intersects with parts of each finger. These intersections appear as bright regions in the 2D IR camera image, and move towards the palm as the finger is bent, and in the opposite direction when the finger is straightened (see Figure 7.7). With a fixed known baseline between laser and the camera, it is possible to triangulate the exact 3D position of each laser line segment (our current implementation uses a 3cm baseline as a reasonable trade-off between depth accuracy and ergonomics, see Figure 7.2).

In a one-off process, the camera and laser are calibrated: The camera's intrinsic parameters are retrieved using a checkerboard calibration method [Zhang, 2000], and are used for image rectification. Next, the user moves the same target and intersects it with the laser line (see Figure 7.6A). The 6DoF extrinsic pose of the target is computed relative to the camera center [Zhang, 2000]. The user clicks on an intersection point, and the associated 3D point is recorded. The process is repeated until three non-coplanar points are selected to define the laser plane relative to the camera.

To triangulate the 3D intersections of the finger and laser, the background subtracted and rectified image is first binarized for connected component analysis. Intersections are clearly visible as elongated ellipsoids which are filtered based on size and shape. Merged connected components (when fingers are close to each other) are separated using the previous seam carving output. The centroid of each connected component is reprojected using the camera intrinsics. From the camera center a ray through the centroid is intersected with the derived laser plane (see Figure 7.6B). This defines a 3D point for each finger, relative to the camera.

7.5 A Simple Kinematic Hand Model

After retrieving the 3D intersections of laser with fingers, we use a new kinematic method to recover the hand pose. Figure 7.7 shows the main finger bones and joints

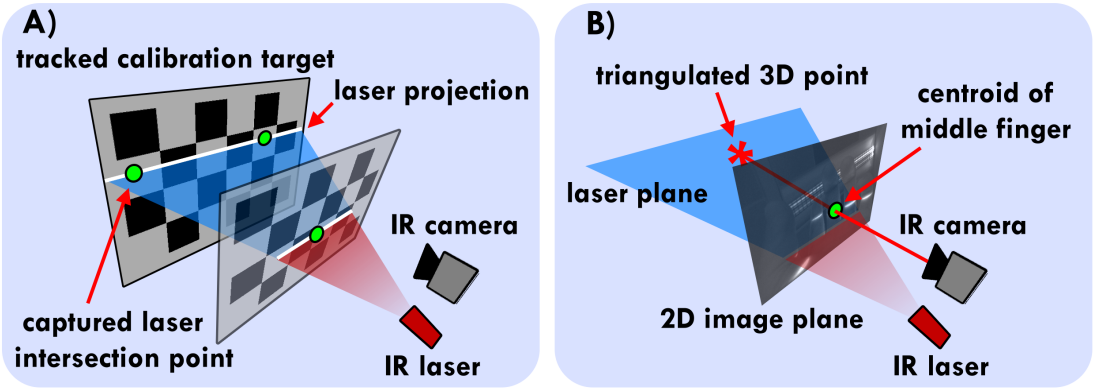

Figure 7.6: A) Laser plane calibration procedure. B) Reprojected ray intersecting with the laser plane.

in a hand. Each finger is comprised of three bones, namely proximal, middle, and distal phalanges. From fingertip to palm, these bones are interconnected by a 1DoF revolute joint called the distal interphalangeal (DIP) joint, a 1DoF revolute proximal interphalangeal (PIP) joint, and a 2DoF spherical joint called the metacarpophalangeal (MCP) joint [Becker and Thakor, 1988].

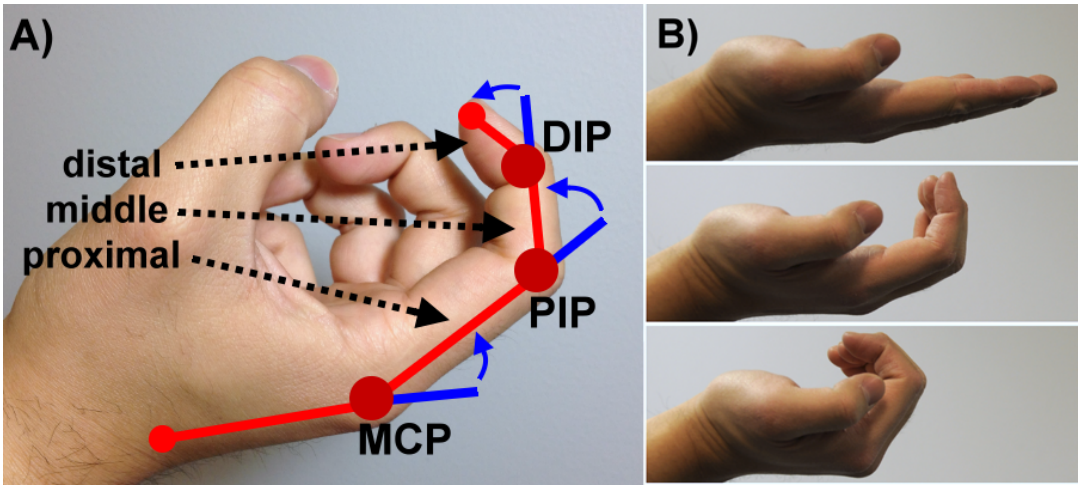

Figure 7.7: A) Illustration of main finger bones and joints. B) Natural flexing of fingers. Proposed forward kinematics model reconstructs this behavior.

These bones do not move in an entirely independent fashion. Specifically, it has been shown that the DIP joint angle depends on the PIP angle owing to interaction of tendons attached to middle and distal phalanges [Becker and Thakor, 1988]. Furthermore, as

shown experimentally by Kamper et al. [2002] during natural flex of the finger (i.e. when not explicitly controlling MCP independently of other joints) MCP joint angles depend on the PIP angle when bending the middle and distal phalanges. We leverage the interdependency of these bones and joints to map from sparsely sensed 3D locations on each finger to a 3D articulated model of the hand.

Specifically, during natural flex of each finger (see Figure 7.8 Right) a linear relationship exists between all three joints of the finger, such that both MCP and DIP can be derived if the PIP angle is known. Kamper et al. [2002] experimentally found the ratio between PIP and DIP is $\frac{1}{0.84}$ and $\frac{1}{0.54}$ for PIP to MCP respectively. Using these ratios, we can approximate a common finger motion, when an outstretched finger curls inwards until it touches the palm, only with a single parameter.

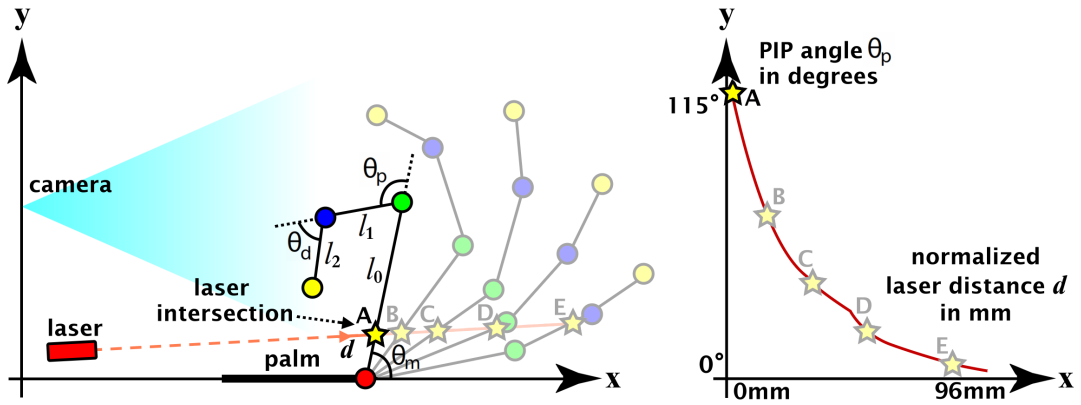

Figure 7.8: Left: Forward kinematics model for a single finger intersecting with laser line. Right: Graph mapping between laser distance and PIP joint angle.

7.5.1 Calculating Joint Angles

To determine the articulation of each finger during natural flex, we experimentally derived the mapping between 3D intersection points and the PIP joint angle using a simple forward kinematic model Figure 7.8. We simulate each of the bones with predefined lengths (l_0, l_1, l_2 respectively). Average bone lengths can be taken from the literature [Becker and Thakor, 1988; Kamper et al., 2002] or measured per user. We compute the PIP angle at simulation time as θ_p $[0°, 120°]$. MCP and DIP are calculated as $\theta_m = a\theta_p$ and $\theta_d = b\theta_p$ respectively where a and b are the joint ratios derived from Kamper et al. [2002].

During calibration, we also derive the 6DoF pose of the palm with respect to the camera (using a checkerboard placed on the palm), which allows us to determine the offset of the laser and its direction with respect to the palm. We simulate the forward kinematics model by changing the joint angle θ_p. At $0°$ the finger is outstretched and fully bent at $120°$. For each of the angles of θ_p we measure the intersection between the laser ray and each of the bones, and take the minimum distance. The sampled data is plotted in Figure 7.8 and used to map from laser distance to θ_p. The graph can be fitted using the following cubic function (where d is the distance to the intersection in mm):

$$\theta_p = -0.0003 * d^3 + 0.059 * d^2 - 4.07 * d + 119.75 \tag{7.1}$$

As fingers have similar anatomy, it is reasonable to assume that this function is valid for all fingers. We therefore provide a simple one-off online calibration process for each finger, where we plot the principal axis of motion for each finger. New intersections are normalized along this axis. Because we normalize along a 3D line, this approach also works for the thumb which moves more diagonally in the sensor image. While articulated thumb motion is reasonably tracked in practice, results could be further refined by explicitly building a similar model as in Figure 7.8 purely for the thumb. Our model can be extended to lateral motions of fingers (i.e. allowing fingers to move left and right), by mapping deviation from the calibrated principal axis to a 3D rotation, which is applied to each finger after articulating finger bend.

7.5.2 Strengths and Limitations

This approach provides a simple, but yet natural approximation of hand poses. Figure 7.1 and Figure 7.9 show a number of real-world hand poses and how these can be replicated using Digits. Whilst the combination of laser line sensing and forward kinematics model is powerful in its own right, there are limitations. In particular, it assumes a strong relationship between all of the joints of each finger. However, there are also common cases where for example the MCP joint moves independently of the PIP. To deal with these wider range of hand poses, we need to sense other parts of the hand and provide an extended kinematics model. Through experimentation we have found that illumination from IR LEDs can be used to robustly detect fingertips, which allows us to define a more complete kinematic model of the hand.

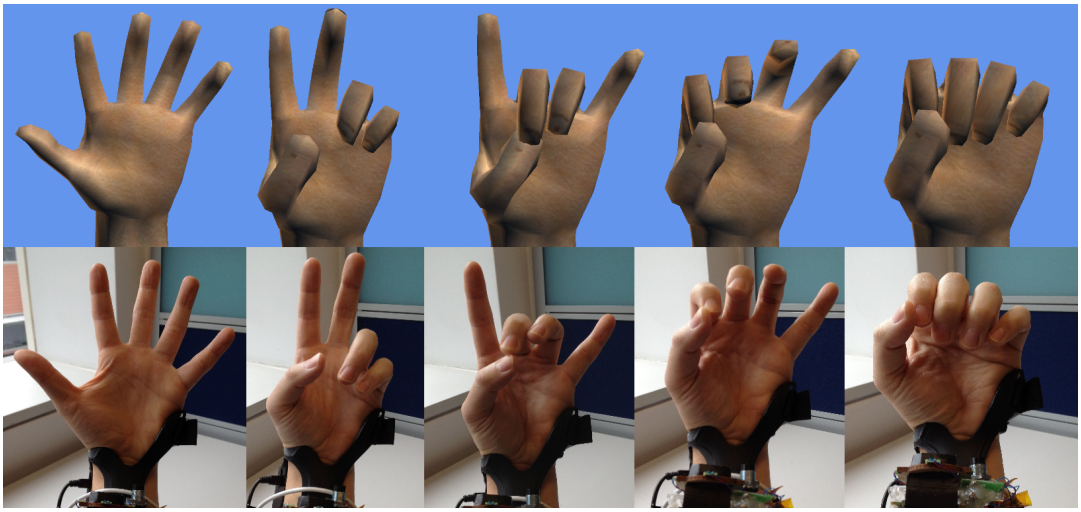

Figure 7.9: Top: Various hand poses supported by our forward kinematics model. Bottom: User's real hand poses.

7.6 Fingertip Detection

Despite it being unfeasible to place a full depth camera on the wrist (due to form-factor and power), we demonstrate how to generate high-resolution normal maps and coarse depth estimates, by modeling the light falloff from the LEDs. This provides a robust method for identifying fingertips in the 2D image, even when fingers are directly facing the camera.

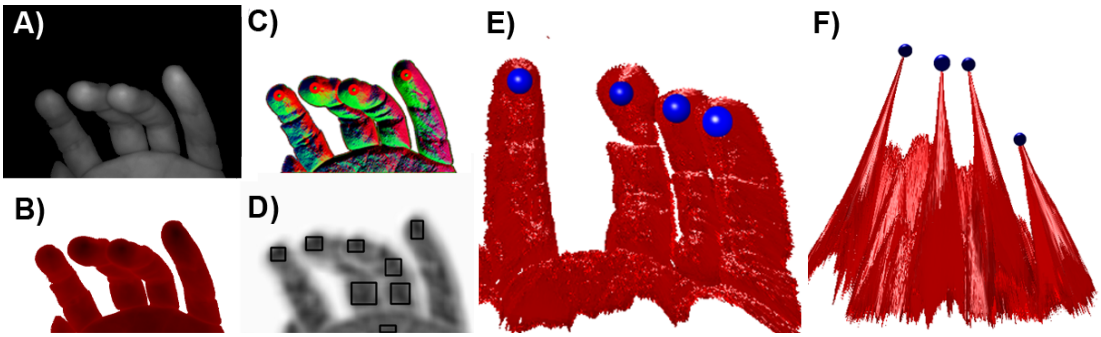

Figure 7.10: Our fingertip detection pipeline. A) Background subtracted imaged fingers. B) Estimated depth encoded in red color channel. C) Normal map computed from depth map. D) Response map from template matching (darker is closer match), black rectangles mark fingertip candidates. E) Recovered mesh viewed from the physical camera's view. F) Depth distortion visible when viewed from off-center viewpoint. Note the distinct peaks caused by fingertips.

Depth Approximation Our approach adapts work on shape-from-shading (SFS) (see overview in Prados and Faugeras [2006]). Our scenario makes SFS more tractable, given partially known parameters for (LED) light position, light power and radial intensity falloff, as well as the ability to approximate the skin reflectance model as purely Lambertian (given the typically small angle of incidence between surface and light source [Marschner et al., 1999]).

We first estimate distance measurements for each pixel under the *inverse-square law*. The intensity at distance d is $I = 1/d^2$ solving for d this gives us an initial distance estimate for each pixel u as: $d(u) = \sqrt{I(u)}$. This distance estimate is then attenuated according to the radial falloff in light intensity for pixels further away from the light's central ray. The final distance value is computed as:

$$D(u) = \sqrt{I(u)} * \frac{1}{\cos(\arctan(\frac{(u-pp)}{fl}))}$$

with known principal point pp and focal length fl from camera calibration. This computes a depth map, where each pixel can be reprojected as a 3D point in camera coordinate space. Finally, we compute surface normals for each pixel from adjacent pixels in the depth map (see Figure 7.10).

Strengths and Limitations This technique provides only approximated depth values, as pixel intensity depends on many factors beyond distance to the light source and light falloff. In particular, we do not model the shape and material of the imaged 3D surface nor do we take the surface orientation into account. While the resulting depth map looks plausible when viewed from the cameras perspective, distortions and non-linearity of the signal become clearly visible when viewed off-center (Figure 7.10F).

However, this approach can be powerful in detecting fingertips and computing a relative depth estimate. Because fingertips are fairly spherical in shape (in particular when pointing towards the camera) they produce a distinct signal as it can be seen clearly in the normal map (Figure 7.10C) and the mesh rendering (Figure 7.10F). Each fingertip produces a very distinct peak in depth very similar in shape to a Gaussian sphere centered around the finger's most protruding part.

To detect and track these peaks many methods are viable. We have obtained robust results with just simple template matching (based on matching scores as squared

distances between a sliding synthetic fingertip template and the live normal map). Figure 7.10D-F shows the final result of our technique, reliably detecting fingertip locations. In particular, our technique works for fingertips pointing towards the camera and multiple fingers touching each other, situations in which simple 2D techniques such as peak-and-valley algorithms, or connected component analysis alone would have difficulties.

7.7 A New Kinematics Model

Fingertip estimates can be combined with the laser line intersections to recover a more accurate hand model using inverse kinematics (IK). IK typically derives joint angles from the 3D position of the end effector – the fingertip. We do not have an accurate 3D measurement for fingertips, and the 3D point sampled with the laser is not directly associated with the end effector. However, the two sensing modalities can be combined to derive a new IK model enabling separate articulation of the MCP joint and the PIP/DIP joints.

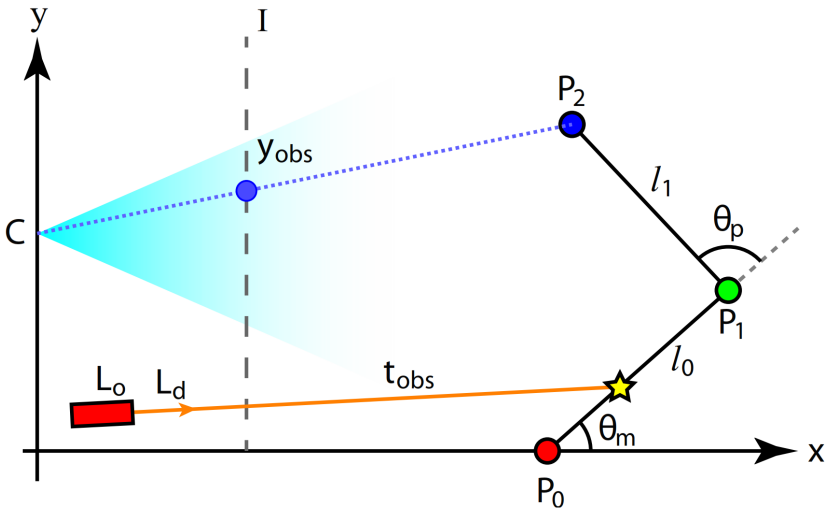

Figure 7.11: IK model that articulates both the MCP joint and the PIP/DIP joints based on an sensor fusion approach

Figure 7.11 illustrates a simplified parametrization of our IK model. Note for illustrative purposes we have reduced the problem to 2D and combined PIP and DIP joints, which cannot be moved independently unless the finger is pressed against a

surface. In this parametrization, the palm is again assumed to be resting directly on the X axis. The position of the MCP joint is given by P_0, the position of the PIP joint is P_1 and the end effector is at P_2. Whilst the 3D location of the end effector is not known, we can observe the projection of the point (y_{obs}) on the image plane (I) (as this equates to the centroid of the detected fingertip in the IR image). Given the calibration matrix of the camera, we can project a ray from the camera center (C) through the image plane. We know that P_2 exists somewhere along this ray.

The length of each of the bones (l_0, l_1), of the finger are again assumed to be known, either by measurement or from assuming predefined values. We are solving for the MCP and PIP joint angles given as θ_m and θ_p respectively. We can also parameterize the laser as an offset from the origin (L_o) and direction (L_d). We also have an observed 3D point with distance t_{obs}, sensed from the laser (L_{obs}) which we know intersects one of the bones (the specific bone is unknown).

P_0 is known ahead of time, using the laser line to calculate the minimum 3D extent of each finger (as described previously). This allows us to calculate P_1 by applying a local transform (translation by bone length l_0 and rotation around the joint angle θ_m) to P_0. So that $P_1 = \mathbf{R}(\theta_m) \cdot [l_0, 0]^T + P_0$ and $P_2 = \mathbf{R}(\theta_p) \cdot [l_1, 0]^T + P_1$.

Our aim now is to find the optimal combination of θ_m and θ_p to best describe the observed data (the location of the 2D fingertip sensed using the LEDs and 3D point measured with the laser). For our fingertip location, we define the following energy function:

$$E_{led} = |proj(P_2) - y_{obs}|^2 \tag{7.2}$$

This function generates estimated positions for P_2 given variations of θ_m and θ_p, and projects these onto the image plane I (using the intrinsic camera calibration parameters). It has a low energy for points that are close to the observed point on the image plane y_{obs}.

Our second energy function first calculates intersections between the laser line and each bone in the finger, based on variations of θ_m and θ_p and takes the minimum:

$$\begin{aligned} t = &\min\{\text{isect}(\vec{L_0 L_d}, \overline{P_0 P_1}), \\ &\text{isect}(\vec{L_0 L_d}, \overline{P_1 P_2})\} \end{aligned} \tag{7.3}$$

It then minimizes the distance between the observed 3D laser point L_{obs} and this estimated intersection:

$$E_{las} = |tL_d + L_o - L_{obs}|^2 \tag{7.4}$$

Our full energy function is specified as:

$$\arg\min_{\theta_m,\theta_p} E = E_{led}\lambda_{led} + E_{las}\lambda_{las} \tag{7.5}$$

This allows us to weight the contribution of either the LED or laser based sensing accordingly, using a scalar (λ). In our current implementation, we evaluate this energy function across θ_m $[0°, 100°]$ and θ_p $[0°, 90°]$ in a brute force manner and select the value with the minimum energy.

This new kinematic model give us even higher DoF input sensing. As shown in Figure 7.12 a wider range of hand poses can be more accurately predicted from the raw sensor data. This includes a wide range of poses that are difficult to predict using our simpler kinematics model. The combination of the two sensing modalities – both laser line and light falloff – allow us to solve the otherwise ill-posed IK problem.

Figure 7.12: Truer hand pose recovery with inverse kinematics. Note PIP and MCP joint angles recovered correctly.

7.8 Initial Evaluation

We performed a preliminary evaluation of Digits, to get a sense of the *accuracy* and *repeatability* of the system at reconstructing different hand poses. Whilst not a detailed user or system evaluation, this study was meant to provide an initial feasibility test for this approach to 3D hand tracking.

7.8.1 Gesture Feasibility Experiment

To provide a first step towards quantifying accuracy and repeatability, we asked participants to mimic hand postures rendered on screen in an interactive 3D application. We used ground truth data gathered using a Vicon motion tracking system as proposed in Metcalf et al. [2008]. Six static hand poses, as shown in Figure 7.13, were generated. In the experiment we tested the full IK model described in the previous section.

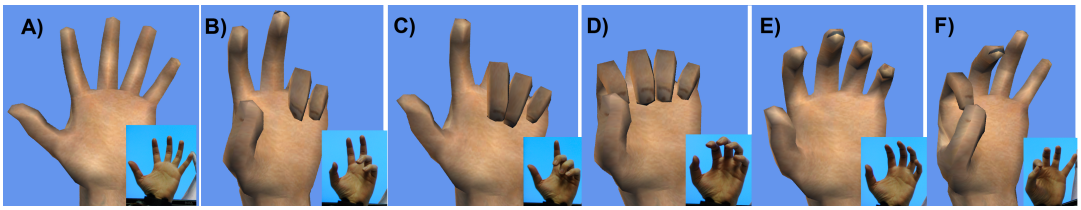

Figure 7.13: Hand postures in experiment. A) Open palm B) Counting two C) Pointing D) Grasping small object E) Grasping large object F) Pinching.

Participants We recruited 12 participants aged 24 to 39, with a mean age of 32. 10 participants were right handed. None of the participants had any physical disabilities or limited range of motion. Related literature has shown that anthropometric differences in the upper limbs can make comparisons across genders incorrect [Dipietro et al., 2003]. Therefore for this first experiment we only tested Digits with male participants.

Procedure and Task The system was mounted on each user's wrist, as in Figure 7.13. The system was calibrated for each user's hand (including bone lengths and finger motion range and trajectory). For left handed users, the camera and laser placement was adjusted and a full camera/laser calibration performed. During the study phase participants were presented with the six reference 3D hand configurations on screen (Figure 7.13). A second 'live' hand model, driven by the participant's input, was rendered on-top of the reference hand. We ignored wrist and forearm reconstruction in this study (and hence did not use the onboard IMU). Left and right-handed reference postures were selected based on the participant's handedness.

Participants started each trial in a neutral pose (fully closed fist). The experimenter triggered each task. Subsequently a reference pose was rendered on screen and users were asked to align their instrumented hands as closely as possible (labeled as the

'acquisition' phase). Users self-reported once they were satisfied with their pose (by pressing a button with their uninstrumented hand), at which point they were asked to hold the pose for 3 seconds (labeled as the 'static' phase). Each user completed 3 blocks each consisting of 6 trials, which included all the 6 hand postures in random order. Users had 10 minutes training time with the application.

Results and Discussion The mean *acquisition time* was 4074 ms. The first block averaged 4417 ms, second 3981 ms and third 3823 ms. A repeated measures ANOVA reveals that there is a significant main effect for *acquisition time* across blocks ($F_{2,28} = 48, p < 0.01$). Post-hoc analysis (Bonferroni corrected $\frac{\alpha}{n} = \frac{0.05}{3} = 0.0167$) reveals linear improvement between blocks 1 and 2 ($p < 0.01$) but no significant difference between blocks 2 and 3 ($p > 0.35$).

Repeatability As a measure of repeatability we compute intraclass correlation coefficients (ICC), the standard test for repeatability [Dipietro et al., 2003; Metcalf et al., 2008; Wise et al., 1990] for this kind of time series data. For each of the six poses we computed ICC scores per finger and per joint between the three blocks of repetition (static phase only). We repeatedly selected two blocks randomly. Then for each block a trial was selected randomly and a ICC score was computed between them. This procedure was repeated 40 times to ensure consistency and ICC scores were averaged together. For brevity we only report individual scores for the PIP angles but overall the ICC scores were in the range of 0.6 to 0.97 (moderate to strong correlation). Correlation between repetitions was found to be moderate for little finger ($ICC = 0.66$) and good for thumb ($ICC = 0.74$) to strong for index finger ($ICC = 0.88$), ring finger ($ICC = 0.81$) and middle finger ($ICC = 0.84$). Overall these results suggest that Digits produces moderate to strongly correlated joint-angle measurements when the same pose is repeated multiple times and by multiple participants.

Accuracy To determine accuracy, data was averaged across blocks and participants, keeping joints and gestures separate. Averaging across fingers would not be very meaningful as finger motion is not entirely independent (see discussion on biomechanical constraints earlier). Again for brevity we only report results for PIP bend angles but results for PIP bend and MCP bend and tilt angles are comparable.

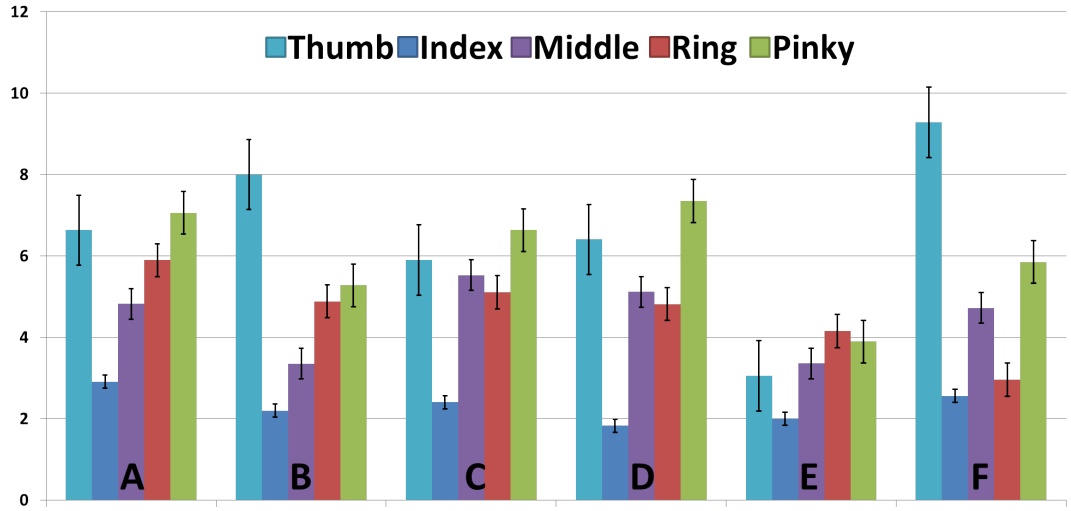

Figure 7.14: Mean error in PIP joint-angle (in degrees) per finger and for all six gestures. Error bars show std deviation.

Figure 7.14 summarizes results for the five fingers and all six gestures. Mean errors throughout the data remain relatively small for all fingers and gestures (all $< 9°$). The best accuracy is usually found with the index finger (min $1.8°$ pose "D", max $2.9°$ pose "A") this could be attributed to the importance of the index finger for many everyday activities (e.g., pointing, pinching, scratching). For most poses the error is largest for the thumb, this maybe explained by the fact that our IK model treats the thumb as a regular finger while in reality the thumb shows a more complex motion (cf Metcalf et al. [2008]). The mean error for the pinky is also comparatively high (min $3.8°$ pose "E", max $7.3°$ pose "D") this may be explained by the camera placement in our setup – to achieve a compact form factor the camera was moved very close to the palm so that the pinky may not be visible in certain circumstances.

Although preliminary, these overall results are promising. The achieved accuracy of the tracking and specifically the joint-angle error rates are comparable to those reported in studies on data gloves [Dipietro et al., 2003; Wise et al., 1990]. Furthermore, the average accuracy (between $2°$ and $9°$) is better than that defined in the literature for manual goniometry (between $7°$ and $10°$) which is considered *"in clinical practice to be the gold standard of joint angle measurement"* [Metcalf et al., 2008].

7.9 Discussion

Digits is a general purpose wearable 3D hand tracker. It avoids direct instrumentation of the user's hand, but instead is wrist-worn. We have shown how the full 3D pose of the user's hand can be inferred, without requiring high-fidelity and currently impractical hardware such as a depth camera to be worn on the body. The preliminary evaluation of our prototype system demonstrates tracking close to existing data gloves, but without direct instrumentation of the hand.

7.9.1 A Tale of Two Kinematic Models

In this chapter we have introduced two complementary ways to recover 3D pose from a small number of samples on the user's hand. While the different models (based on forward and inverse kinematics respectively) provide different levels of interactive expressiveness they should not be seen as one fully replacing the other. Both have their own strengths and weaknesses and utility in different contexts. Our first model has the advantage of simplicity. It allows reconstruction of rich 3D hand poses from a single measurement on each finger which already enables a number of compelling spatial continuous gestural interactions (Figure 7.1 and Figure 7.9) and can facilitate gesture recognition in mobile applications. The second model adds higher DoF sensing and more independent movement of each finger joint, and therefore provides more fidelity in recovering the user's hand pose. This model also allows a truer reconstruction of the hand as shown in Figure 7.15. This can be useful for example in a physics-enabled application where you wish to model grasping of an arbitrary virtual object or a medical application where accurate joint-angle measurements are needed. We enable this by adding only simple additional hardware, although there is more algorithmic complexity.

7.9.2 Limitations

With our current implementation being a vision based technique, occlusions resulting from crossed fingers, overly bent thumb and handheld objects are problematic for hand pose reconstruction. However, these are special cases and we anticipate they can be avoided by careful gesture design. Furthermore, more advanced techniques for finger separation and identification could be used to mitigate these issues. In the current

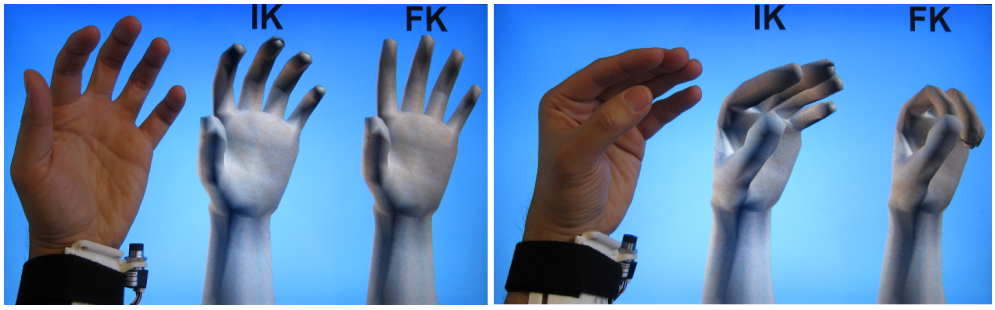

Figure 7.15: Failure cases for our forward kinematics (FK) model. Notice mismatch in PIP angle (left) and MCP angle (right). Using our inverse kinematics (IK) model we can mitigate such issues, and offer a truer reconstruction.

Digits prototype we do not model wrist bend and rotations about the forearm explicitly. In particular a fully flat or over-arching hand is problematic, while our image processing techniques can cope better with lateral motion of the hand relative to the camera.

Our proof-of-concept implementation already is wearable and not overly bulky but requires user instrumentation. Furthermore, the device is still tethered to a PC or laptop where computations are being performed. It is however conceivable to further miniaturize the device until it is either a standalone watch-like device or fully integrated into a regular watch. Future depth camera technologies such as time-of flight might enable such compact form factor devices. For the time being we argue that our approach is the best trade-off between practicality and availability concerns but also between computational complexity, power consumption and form-factor.

7.9.3 Always Available Input

It is worthwhile noticing that with Digits input is *not* restricted to a fixed space around the user, but instead moves with the user's hand. For example, gestures can be conducted in front of the body (much like other body-worn systems [Harrison et al., 2011; Mistry and Maes, 2009; Starner et al., 2000]). However, gestural input may also happen in a more subtle, effortless way. For example, performing hand gestures whilst the hand is lowered by the side of the body or resting on a physical surface, avoiding arm fatigue during long periods of use. Examples of this are shown in the accompanying video figure.

7.9.4 Emergent Interactions

There are other emergent interactive features of Digits which can be fruitful to explore in future work. For example, Digits may be used to track the spatial positions of fingers from the other (non-instrumented hand) when both hands interact. Using the palm of the reference hand as a track pad, or using the segments of fingers on the reference hand to control sliders in a GUI.

Whilst Digits has been designed to be a general purpose interaction platform, we have demonstrated both in this chapter and accompanying video interactive scenarios using this technology. We believe Digits is particularly useful for mobile scenarios, where the sensor can coexist with existing personal devices such as mobile phones and tablets. The combination of touch and 3D input in free space around the device is particularly interesting. Also of interest are eyes-free interfaces which allow for interactions without needing to remove mobile devices from our pockets. One final application area for Digits is in gaming, where technologies such as the Xbox Kinect, or Nintendo Wii do not currently support the level of fidelity of hand sensing. Again such a device could be complimentary to these existing sensing modalities. For example, combining the Kinect full body tracker with high fidelity freehand interactions of Digits.

7.10 Summary

This chapter presented Digits a wrist-worn system for sensing the full 3D pose of the user's hand without external sensing infrastructure or full instrumentation of the hand (unlike existing data gloves). The system targets mobile AR settings, and is specifically designed to be low-power and easily reproducible using only off-the-shelf hardware that is both smaller and more power efficient than current consumer depth cameras. We have shown two complimentary methods for robustly tracking features of the hand, and we demonstrated the evolution of a kinematics model for reconstructing the full articulated hand from these sparse samples. We demonstrated the utility of Digits for a variety of application scenarios, including 3D spatial interaction on mobile phones, eyes-free interaction on the move, and gaming. And we evaluated the feasibility of this our approach through preliminary qualitative and quantitative studies.

Chapter 8

Conclusion

8.1 Thesis Contributions

This thesis makes a number of contributions to the state of the art for NUI, AR, and HCI, and demonstrates how hardware, computer-vision algorithms and user interaction techniques can be combined to create novel user experiences and systems. The main aim of this work was to make the entire end-to-end interaction between the user and the virtual and the physical environment as seamless as possible. This required extending cutting edge hardware, such as a Kinect depth sensor, with real-time algorithms and the development of custom sensing and output hardware to bring the worlds of NUI and AR closer together.

The contributions of this work can be summarized as follows:

- *The design and development of a novel 3D silhouette sensor and processing pipeline for high precision input with various NUI modalities on and above physical surfaces.*

  In Chapter 3 we looked at a desktop NUI scenario, where the user would benefit most from high DoF, yet precise input. Beyond traditional interactions with mouse and keyboard, we were interested in allowing touch and mid-air interactions with hands, styluses and tangible objects. A pair of infrared cameras and retro-reflective optics were used to generate high contrast silhouette images that can be efficiently processed in real time with our contour-based stereo vision algorithm.

- *The design and development of a situated augmented reality display and a new physics model for dexterous direct 3D interactions in a mixed-reality space.*

  In Chapter 4 I primarily focused on moving away from separate input and output to a single combined space that allows a user to directly manipulate virtual objects with completely unscripted physics-based interactions. I presented a situated augmented reality setup that senses input with a Kinect depth sensor and optically combines the virtual and the physical with a semi-silvered mirror and simulated motion parallax. Physical surfaces are tracked with a GPU-accelerated optical flow algorithm and are approximated as collections of collision particles that model velocity, collision and friction forces. This alternative interaction paradigm frees designers from scripting a fixed set of interaction techniques and allows users for more expressive interactions. This system provided a rich AR/NUI experience, in a fixed form factor.

- *The development of algorithms for tracking objects in 3D space and direct touch interactions on any surface with a real-time dense surface reconstruction system.*

  In Chapter 5 I enabled input sensing beyond the desktop into the physical space around us. Using a Kinect depth camera and a novel real-time algorithm for volumetric surface reconstruction we can generate a dense model of the geometry of our environment that is more complete than the raw depth input. We track the absolute position and orientation and correctly model occlusions and physics interactions between virtual and physical objects. We segment and track objects in the real world and enable multi-touch interaction on any surface by comparing the input depth with the reconstructed geometry of the environment. This offered an extremely close coupling of real and virtual worlds, but lacked display capabilities.

- *The design and development of projection-based hand shadow interaction techniques.*

  In Chapter 6 we focused on visually augmenting physical surfaces with digital projection. Two approaches were explored: A portable projector that utilizes a sensing infrastructure that is always aware of the geometry of the environment and the user, and an infrastructure-free approach that uses a Kinect depth sensor

and allows the user to be completely mobile. Both approaches include a camera co-axially or closely attached to the projector allowing a user to exploit the shadow of their hands to directly interact with remote surfaces. We uncovered the pros and cons of infrastructure-based and infrastructure-free approaches, and provided display capabilities to the work presented in 5.

- *The design and development of a completely self-contained wrist-worn sensor and a processing pipeline for articulated 3D hand tracking on the move.*

  In Chapter 7 we explored a wearable 3D hand tracking device that is conceivably worn throughout the whole day like a watch. We moved from a portable system to a wearable device that gives the user the ultimate mobility and freedom and allows for dexterous gestural interactions outside the living room. Form-factor, computation cost and power consumption are big concerns and motivated us to build custom sensing hardware that only captures certain parts of the hand that can still help us to reconstruct the 3D pose of the full hand. We also created simplified models of a human hand in Chapter 7 that leverage the interdependency of certain joint movements. These allow us to recreate an approximation of the shape of the hand with five 1D measurements with a forward kinematics model, and a more realistic hand model with additional five 2D fingertip positions with an inverse kinematic model. This approach could be extended to other body parts, like the arm, legs and the entire human body.

  This work showed the power of combining high fidelity hand-based NUI interactions, in mobile AR contexts.

8.1.1 Concluding Remarks

In conclusion this thesis has provided a technical exploration of how AR and NUI can be combined. We have built a variety of novel systems to lead this technical exploration. I discuss these in light of the requirements identified in Section 2.8.1:

Unencumbered Interactions. In my work, I primarily sensed input using camera-based technologies that provide very rich input data, and high degree of flexibility. I designed a number of computer vision and machine learning algorithms to replace

approaches that previously required instrumenting the user. In Chapter 4 I presented a situated augmented reality display that does not require the user to wear glasses or gloves for direct 3D interactions in mid-air. I used a simple optical arrangement, viewpoint-dependent rendering and a physics simulation-based surface approximation for robust 3D interactions with depth data. In Chapter 5 I presented a method for capturing high-quality models only based on a series of noisy depth input images. I showed how touch could be detected without a touch sensor by capturing and comparing the geometry of the environment and the user. In Chapter 7 I presented a wearable hand tracker that estimates the full pose of the hand without covering the hand itself, by sampling distances at each finger with a laser line and processing these with a biomechanical kinematics model of the hand.

Coupling of Input and Output. Instead of limiting interactions to what we see on the flat 2D display, and input through the mouse and keyboard, I incorporated the rich structures of physical objects and the entire space around the user for interaction. For seamless interactions between the real and the physical, I reduced the number of steps required to engage with the system. In HoloDesk, I used optical and visual techniques and depth sensing to remove the need for user instrumentation. The interaction model in HoloDesk was explicitly chosen to foster easy transfer of real-world knowledge when manipulating virtual objects. Moreover, AR was used in KinectFusion, Augmented Projectors and HoloDesk to remove the perceived boundary between the real and the virtual.

Real-time Interaction. I demonstrated how graphics processors can accelerate many image processing steps described in each technical chapter. Another powerful way to achieve interactive rates is to approximate the user input. In RetroDepth (see Chapter 3), I focused only on the outline of objects instead of regarding all pixels that constitute an object in the input image. This was based on the observation that when we interact with physical objects, we mostly use the outer regions of our hands or tools to push, pull, grasp, pick or touch an object. In Digits (see Chapter 7) on the other hand, I leveraged the interdependency of finger joint movements and only use rough depth estimates to approximate the overall shape of the hand.

Fidelity of Input. The systems presented aimed to preserve and utilize higher DoFs and object shape for interaction (e.g. hand and physical tools). In KinectFusion I used algorithms that reconstruct the full geometry of the environment and user's hand for multi-touch and augmented reality interactions. In RetroDepth I designed and implemented algorithms that tracked the full 3D contour of the hand and other objects for dexterous input. In Digits I designed and implemented track the full 3D hand pose. In HoloDesk I designed and implemented algorithms that allow for unconstrained physics-based interactions using any object.

All the systems are real-time allowing for interactive scenarios. This is a crucial technical challenge which adds important requirements in algorithm and systems design. As identified, all the systems exhibit a high degree of user and space centric properties, which is one of the key motivations of linking NUI and AR. Furthermore, all the systems are unencumbered in terms of user input in a way that AR systems have not exhibited in the past. All systems also exhibit a high degree of input fidelity. In addition I identified two further design decisions that I varied across these systems. The first explores the coupling of IO, starting with a low coupling and then moving towards tighter coupling. The second explores the use of *fixed* versus *mobile* infrastructures. the thesis began with IO decoupled systems with fixed infrastructures, moving towards a closer coupling of IO and enabling truly mobile scenarios.

8.2 Future Work

So far this thesis has demonstrated the capabilities and advantages of camera-based real-time sensing and graphical output for creating new 3D interaction experiences. There are, however, limitations to our approach, which I will discuss in this section.

Vision-based methods can be prone to failure in difficult lighting situations or when the target of interest is occluded. Foreground objects, such as the user's hand, can be difficult to segment, if there is little contrast in intensity or texture between the foreground and the background. One relatively simple solution that could alleviate this issue is the use of a thermal (or far-infrared) camera that can pick up tiny differences in temperature. Their use for user interaction has been demonstrated [Larson et al., 2011] and are becoming increasingly affordable [FLIR One, 2014]. The image of a

thermal sensor can allow for robust segmentation of body parts in a visually cluttered or low-contrast scene.

Occlusions could be overcome through electric field sensing (EFS). EFS has previously been used for bare hand tracking and posture recognition [Lee et al., 2006]. Any conductive object near an EFS system perturbs the electric field and creates a unique signature, which could be used to support vision-based hand tracking in difficult ambient light situations even when there is no direct line-of-sight to the target. The signature of the electric field could be learned with machine learning techniques using the output of the vision-based tracker as ground truth only while the lighting is optimal. In situations where the lighting isn't optimal or when occlusions to the target exit, the system can fall back to directly regressing the tracking data from the EFS signature.

One of the most limiting factors in 3D interactions in NUI and AR is the lack of convincing haptic feedback, without the user instrumentation. Recently, there have been promising developments regarding haptic feedback technologies that go beyond mere vibrations without instrumenting the user. Sinclair et al. [2013] present an actuated 3D touch display on a robotic arm, Sodhi et al. [2013] present an actuator that can shoot a focused vortex over a large distance, Carter et al. [2013] present an array of ultra-sonic actuator that creates focused pressure points in mid-air, [Lopes and Baudisch, 2013] demonstrate a method for directly actuating the user's arms and fingers using electrical muscle stimulation and Rekimoto [2013] presents a mobile tactile actuator that creates the illusion of a pulling/pushing force.

On a more conceptual level, there is an interesting question about what it means to turn "any" surface interactive. In Chapter 5 we presented KinectFusion a real-time dense surface reconstruction system that acquires the full geometry of an environment around the user within seconds, segments individual objects in the scene and detects touch on any surface. What are the next steps in really utilizing the acquired geometry for user interaction beyond simple augmented reality visualizations? Work on *scene understanding* in computer vision [Cheng et al., 2013; Kohli et al., 2009; Li et al., 2009; Salas-Moreno et al., 2013] are very promising and demonstrate automatic segmentation and semantic labeling of objects within scenes. A real-time system that has access to these semantic labels within our environment would radically change the way perceive and interact with the space around us.

Appendix

Please see attached DVD for supplementary papers and videos.

References

3Gear Systems Inc. http://threegear.com/, 2013. 17, 55

Agarwal, A., Izadi, S., Chandraker, M., and Blake, A. High precision multi-touch sensing on surfaces using overhead cameras. In *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP'07. Second Annual IEEE International Workshop on*, pages 197–200. IEEE, 2007. 15

Agrawala, M. et al. The two-user responsive workbench: support for collaboration through individual views of a shared space. In *ACM Transactions on Graphics (SIGGRAPH)*, pages 327–332, 1997. ISBN 0-89791-896-7. 16

Annett, M., Grossman, T., Wigdor, D., and Fitzmaurice, G. Medusa: a proximity-aware multi-touch tabletop. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 337–346. ACM, 2011. 17

Apple. http://www.apple.com/iphone, 2014. 3

Ascension Technology Corporation. http://www.ascension-tech.com/, 2014. 5, 6

Ashbrook, D. L. *Enabling mobile microinteractions*. Phd thesis, Georgia Institute of Technology, 2010. URL `http://gradworks.umi.com/34/14/3414437.html`. 136

Avidan, S. and Shamir, A. Seam carving for content-aware image resizing. *SIGGRAPH Comput. Graph.*, 26(3):10, July 2007. ISSN 07300301. doi: 10.1145/1276377.1276390. URL `http://doi.acm.org/10.1145/1275808.1276390`. 141

Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., and MacIntyre, B. Recent advances in augmented reality. *IEEE Comput. Graph. Appl.*, 21(6):34–47, November 2001. ISSN 0272-1716. doi: 10.1109/38.963459. URL `http://dx.doi.org/10.1109/38.963459`. 27

Azuma, R. T. et al. A survey of augmented reality. *Presence*, 6(4):355–385, 1997. 27

Baecker, R. M. Picture-driven animation. In *Proceedings of the May 14-16, 1969, Spring Joint Computer Conference*, AFIPS '69 (Spring), pages 273–288, New York, NY, USA, 1969. ACM. doi: 10.1145/1476793.1476838. URL `http://doi.acm.org/10.1145/1476793.1476838`. 2

Bailly, G., Müller, J., Rohs, M., Wigdor, D., and Kratz, S. ShoeSense: A New Perspective on Hand Gestures and Wearable Applications. In *ACM SIGCHI*, pages 1239–1248, 2012. URL `http://www.gillesbailly.fr/publis/BAILLY_CHI12a.pdf`. 20, 21, 24, 136

Beardsley, P., van Baar, J., Raskar, R., and Forlines, C. Interaction using a handheld projector. *Computer Graphics and Applications, IEEE*, 25(1):39–43, 2005. ISSN 0272-1716. doi: 10.1109/MCG.2005.12. 31

Becker, J. C. and Thakor, N. V. A study of the range of motion of human fingers with application to anthropomorphic designs. *IEEE Trans. Biomed. Eng.*, 35(2):110–117, 1988. URL `http://www.ncbi.nlm.nih.gov/pubmed/3350537`. 143, 144

Benko, H., Jota, R., and Wilson, A. Miragetable: freehand interaction on a projected augmented reality tabletop. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 199–208. ACM, 2012. xii, 17, 18

Besl, P. J. and McKay, N. D. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14:239–256, February 1992. ISSN 0162-8828. doi: 10.1109/34.121791. URL `http://portal.acm.org/citation.cfm?id=132013.132022`. 94, 95

Bill Buxton. http://www.dgp.toronto.edu/OTP/papers/bill.buxton/IRG.html, 1994. 3

Bill Buxton. http://www.billbuxton.com/multitouchoverview.html, 2013. 3, 6, 17, 35

Billinghurst, M. and Starner, T. Wearable devices: new ways to manage information. *Computer*, 32(1):57–64, 1999. 5

Bimber, O. and Raskar, R. *Spatial Augmented Reality: Merging Real and Virtual Worlds*. A. K. Peters, Ltd., Natick, MA, USA, 2005. ISBN 1568812302. 28, 61

Bimber, O. et al. The extended virtual table: An optical extension for table-like projection systems. *Presence: Teleoper. Virtual Environ.*, 10:613–631, December 2001. ISSN 1054-7460. doi: 10.1162/105474601753272862. URL `http://dl.acm.org/citation.cfm?id=1246729.1246734`. 28

Bleyer, M., Rhemann, C., and Rother, C. Patchmatch stereo - stereo matching with slanted support windows. In *British Machine Vision Conference*, 2011. 49

Bolt, R. *The human interface: where people and computers meet*. Lifetime Learning Publications, 1984. URL `http://books.google.com/books?id=BMIZAQAAIAAJ`. 4

Bolt, R. A. “put-that-there”: Voice and gesture at the graphics interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '80, pages 262–270, New York, NY, USA, 1980. ACM. ISBN 0-89791-021-4. doi: 10.1145/800250.807503. URL `http://doi.acm.org/10.1145/800250.807503`. 4

Brashear, H., Starner, T., Lukowicz, P., and Junker, H. Using multiple sensors for mobile sign language recognition. In *Wearable Computers, 2003. Proceedings. Seventh IEEE International Symposium on*, pages 45–52, Oct 2003. doi: 10.1109/ISWC.2003.1241392. 6

Breiman, L. Random Forests. *Machine Learning*, 45(1):5–32, October 2001. 50

Brown, M. Z., Burschka, D., and Hager, G. D. Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):993–1008, 2003. 46

Brox, T. et al. High accuracy optical flow estimation based on a theory for warping. *Computer*, 4(May), 2004. URL http://www.springerlink.com/index/87a4ckjqm92lp3j9.pdf. 71

Brumitt, B., Meyers, B., Krumm, J., Kern, A., and Shafer, S. A. Easyliving: Technologies for intelligent environments. In *Proceedings of the 2Nd International Symposium on Handheld and Ubiquitous Computing*, HUC '00, pages 12–29, London, UK, UK, 2000. Springer-Verlag. ISBN 3-540-41093-7. URL http://dl.acm.org/citation.cfm?id=647986.743885. 30, 85, 122

Butler, A. et al. Vermeer: Direct Interaction with a 360°Viewable 3D Display. In *Proc. ACM UIST*, 2011. URL http://portal.acm.org/citation.cfm?id=1449715.1449729. 29

Buxton, W., Baecker, R., Clark, W., Richardson, F., Sutherland, I., Sutherland, W. B., and Henderson, A. Interaction at lincoln laboratory in the 1960's: Looking forward – looking back. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 1162–1167, New York, NY, USA, 2005. ACM. ISBN 1-59593-002-7. doi: 10.1145/1056808.1056864. URL http://doi.acm.org/10.1145/1056808.1056864. 1

Cao, X. and Balakrishnan, R. Interacting with dynamically defined information spaces using a handheld projector and a pen. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, pages 225–234, New York, NY, USA, 2006. ACM. ISBN 1-59593-313-1. doi: 10.1145/1166253.1166289. URL http://doi.acm.org/10.1145/1166253.1166289. xiv, 27, 29, 30, 34, 112, 113, 116, 121, 122, 130, 131

Cao, X., Forlines, C., and Balakrishnan, R. Multi-user interaction using handheld projectors. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 43–52, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-679-0. doi: 10.1145/1294211.1294220. URL http://doi.acm.org/10.1145/1294211.1294220. xiv, 30, 34, 112, 113, 116, 121, 130, 131

Carter, T., Seah, S. A., Long, B., Drinkwater, B., and Subramanian, S. Ultrahaptics: Multi-point mid-air haptic feedback for touch surfaces. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 505–514, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2268-3. doi: 10.1145/2501988.2502018. URL http://doi.acm.org/10.1145/2501988.2502018. 162

Chen, Y. and Medioni, G. Object modeling by registration of multiple range images. *Image and Vision Computing (IVC)*, 10(3):145–155, 1992. 98

Cheng, M.-M., Zheng, S., Lin, W.-Y., Warrell, J., Vineet, V., Sturgess, P., Crook, N., Mitra, N. J., and Torr, P. H. S. Imagespirit: Verbal guided image parsing. *CoRR*, abs/1310.4389, 2013. 162

Colwell, W. and Hurst, S. Discriminating contact sensor, October 7 1975. URL http://www.google.com/patents/US3911215. US Patent 3,911,215. 3

Cowan, L. G., Li, K. A., and Park, F. ShadowPuppets : Supporting Collocated Interaction with Mobile Projector Phones Using Hand Shadows. In *ACM SIGCHI*, pages 2707–2716. ACM Press, 2011. ISBN 9781450302678. doi: 10.1145/1978942.1979340. URL http://portal.acm.org/citation.cfm?doid=1978942.1979340. 117

Cui, Y., Schuon, S., Chan, D., Thrun, S., and Theobalt, C. 3d shape scanning with a time-of-flight camera. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conf. on*, pages 1173 –1180, june 2010. doi: 10.1109/CVPR.2010.5540082. 27, 89

Curless, B. and Levoy, M. A volumetric method for building complex models from range images. In *ACM Transactions on Graphics (SIGGRAPH)*, 1996. 94, 96, 99, 100

Czernuszenko, M. et al. The immersadesk and infinity wall projection-based virtual reality displays. *SIGGRAPH Comput. Graph.*, 31:46–49, May 1997. ISSN 0097-8930. doi: http://doi.acm.org/10.1145/271283.271303. URL http://doi.acm.org/10.1145/271283.271303. 16

Davison, A., Reid, I., Molton, N., and Stasse, O. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6): 1052–1067, June 2007. ISSN 0162-8828. 5, 6, 11, 34

Dipietro, L., Sabatini, A. M., and Dario, P. A Survey of Glove-Based Systems and Their Applications. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, 38(4):461–482, 2008. ISSN 10946977. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4539650. 5, 19

Dipietro, L., Sabatini, A. M., and Dario, P. Evaluation of an instrumented glove for hand-movement acquisition. *Int. J. Rehabil. Res.*, 40(2):179–189, 2003. URL http://www.ncbi.nlm.nih.gov/pubmed/15077642. 151, 152, 153

Dippon, A. and Klinker, G. Kinecttouch: accuracy test for a very low-cost 2.5 d multitouch tracking system. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 49–52. ACM, 2011. 18

Doug Engelbart Institute. http://www.dougengelbart.org/firsts/mouse.html, 2014. 1

Du, H., Henry, P., Ren, X., Cheng, M., Goldman, D. B., Seitz, S. M., and Fox, D. Interactive 3d modeling of indoor environments with a consumer depth camera. In *Proceedings of the 13th international conference on Ubiquitous computing*, UbiComp '11, pages 75–84, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0630-0. doi: 10.1145/2030112.2030123. URL http://doi.acm.org/10.1145/2030112.2030123. 122, 130

Ehnes, J., Hirota, K., and Hirose, M. Projected augmentation - augmented reality using rotatable video projectors. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '04, pages 26–35, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2191-6. doi: 10.1109/ISMAR.2004.47. URL http://dx.doi.org/10.1109/ISMAR.2004.47. 32

Engelbart, D. X-y position indicator for a display system, November 17 1970. URL http://www.google.com/patents/US3541541. US Patent 3,541,541. 1

Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D., and Twombly, X. Vision-based hand pose estimation: A review. *Comput. Vision Image Understanding*, 108(1-2): 52–73, October 2007. ISSN 10773142. doi: 10.1016/j.cviu.2006.10.012. URL http://dx.doi.org/10.1016/j.cviu.2006.10.012. 20

Farsiu et al., S. Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing*, 13(10):1327–1344, 2004. 88

Faugeras, O. D. and Keriven, R. Variational principles, surface evolution, pdes, level set methods, and the stereo problem. *IEEE Transactions on Image Processing*, 7(3): 336–344, 1998. 49

Fiala, M. Artag, a fiducial marker system using digital techniques. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 590–596. IEEE, 2005. 12

FLIR One. http://www.flir.com/flirone/, 2014. 161

Frahm, J.-M., Georgel, P. F., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.-H., Dunn, E., Clipp, B., and Lazebnik, S. Building Rome on a cloudless day. In *Proc. Europ. Conf. on Computer Vision (ECCV)*, 2010. 25, 26

Freedman, B., Shpunt, A., Machline, M., and Arieli, Y. Depth Mapping Using Projected Patterns. Patent Application, 10 2008a. URL http://www.patentlens.net/patentlens/patent/WO_2008_120217_A2/en/. WO 2008/120217 A2. 86

Freedman, B., Shpunt, A., Machline, M., and Arieli, Y. Depth mapping using projected patterns, October 2 2008b. URL http://www.google.com/patents/US20080240502. US Patent App. 11/899,542. 7, 13

Google. http://glass.google.com/, 2014. 5

Graham, E. D. and MacKenzie, C. L. Physical versus virtual pointing. In *Proc. ACM CHI*, pages 292–299, 1996. doi: http://doi.acm.org/10.1145/238386.238532. URL http://doi.acm.org/10.1145/238386.238532. 77

Grand, S. L. Broad-phase collision detection with CUDA. In *GPU Gems 3*. Addison-Wesley, 2007. 103, 104

Grossman, J. and Dally, W. J. *Point sample rendering*. Springer, 1998. 55

Gruber, L., Richter-Trummer, T., and Schmalstieg, D. Real-time photometric registration from arbitrary geometry. In *Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, ISMAR '12, pages 119–128, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-1-4673-4660-3. doi: 10.1109/ISMAR.2012.6402548. URL http://dx.doi.org/10.1109/ISMAR.2012.6402548. 5

Gustafson, S., Bierwirth, D., and Baudisch, P. Imaginary interfaces. In *ACM UIST*, pages 3–12, 2010. ISBN 9781450302715. doi: 10.1145/1866029.1866033. URL http://portal.acm.org/citation.cfm?id=1866029.1866033. xiii, 23, 24, 34

Hachet, M. et al. Toucheo: Multitouch and Stereo Combined in a Seamless Workspace. In *Proc. ACM UIST*, October 2011. xiv, 28, 29

Han, J. Y. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, UIST '05, pages 115–118, New York, NY, USA, 2005. ACM. ISBN 1-59593-271-2. doi: 10.1145/1095034.1095054. URL http://doi.acm.org/10.1145/1095034.1095054. xii, 6, 11, 12, 14

Haptix. http://www.haptixtouch.com/, 2013. xiii, 19

Harada, T. Real-time rigid body simulation on gpus. In *GPU Gems 3*. Addison-Wesley Professional, 2007. 103, 104

Harrison, C., Tan, D., and Morris, D. Skinput: Appropriating the Body as an Input Surface. *ACM SIGCHI*, 3(8):453–462, 2010. ISSN 21508097. URL http://portal.acm.org/citation.cfm?id=1753326.1753394&coll=GUIDE&dl=ACM&type=series&idx=SERIES260&part=series&WantType=Proceedings&title=CHI. xiii, 21, 22, 33, 34

Harrison, C., Benko, H., and Wilson, A. D. OmniTouch: Wearable Multitouch Interaction Everywhere. In *ACM UIST*, pages 441–450. ACM, 2011. ISBN 9781450307161. doi: 10.1145/2047196.2047255. URL `http://dl.acm.org/citation.cfm?id=2047255`. 20, 21, 32, 34, 131, 155

Hartley, R. and Zisserman, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. 7, 25, 33, 113

Haubner, N., Schwanecke, U., Dörner, R., Lehmann, S., and Luderschmidt, J. Detecting interaction above digital tabletops using a single depth camera. *Machine Vision and Applications*, pages 1–13, 2013. 17

Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Proc. of the Int. Symposium on Experimental Robotics (ISER)*, 2010. 26, 33, 34

Hilliges, O., Izadi, S., Wilson, A. D., Hodges, S., Garcia-Mendoza, A., and Butz, A. Interactions in the Air: Adding Further Depth to Interactive Tabletops. In *ACM UIST*, ACM UIST, pages 139–148, 2009a. ISBN 978-1-60558-745-5. 83

Hilliges, O., Izadi, S., Wilson, A. D., Hodges, S., Garcia-Mendoza, A., and Butz, A. Interactions in the air: adding further depth to interactive tabletops. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 139–148. ACM, 2009b. xii, 16, 17

Hinckley, K. *Haptic issues for virtual manipulation*. PhD thesis, University of Virginia, 1996. 2

Hirsch, M., Lanman, D., Holtzman, H., and Raskar, R. Bidi screen: a thin, depth-sensing lcd for 3d interaction using light fields. In *ACM Transactions on Graphics (TOG)*, volume 28, page 159. ACM, 2009. xii, 16, 18

Hoffman, D. et al. Vergenceaccommodation conflicts hinder visual performance and cause visual fatigue. *IJVR*, 8(3):33, 2008. 27

Howard, B. and Howard, S. Lightglove: Wrist-worn virtual typing and pointing. In *IEEE ISWC*, pages 172–173, 2001. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=962130`. 21, 136

Huhle, B., Schairer, T., Jenke, P., and Straßer, W. Fusion of range and color images for denoising and resolution enhancement with a non-local filter. *Computer Vision and Image Understanding*, 114(12):1336–1345, 2010. 26

Ishii, H. and Ullmer, B. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, pages 234–241. ACM, 1997. 35

Izadi, S., Agarwal, A., Criminisi, A., Winn, J., Blake, A., and Fitzgibbon, A. C-slate: a multi-touch and object recognition system for remote collaboration using horizontal surfaces. In *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP'07. Second Annual IEEE International Workshop on*, pages 3–10. IEEE, 2007a. xii, 15, 17

Izadi, S., Hodges, S., Taylor, S., Rosenfeld, D., Villar, N., Butler, A., and Westhues, J. Going beyond the display: a surface technology with an electronically switchable diffuser. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 269–278. ACM, 2008. 16

Izadi, S., Hodges, S., Butler, A., West, D., Rrustemi, A., Molloy, M., and Buxton, W. Thinsight: a thin form-factor interactive surface technology. *Communications of the ACM*, 52(12):90–98, 2009. 17

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., and Fitzgibbon, A. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *ACM UIST*, pages 559–568, 2011a. ISBN 9781450307161. URL `http://research.microsoft.com/en-us/um/people/pkohli/papers/uist_2011.pdf`. xxi, 122, 124, 128

Izadi, S. et al. C-Slate: A Multi-Touch and Object Recognition System for Remote Collaboration using Horizontal Surfaces. In *IEEE Tabletop*, volume 0, pages 3–10. IEEE, 2007b. ISBN 0769530133. doi: 10.1109/TABLETOP.2007.34. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4384105`. 20

Izadi, S. et al. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proc. ACM UIST*. ACM, 2011b. URL http://portal.acm.org/citation.cfm?id=1449715.1449760. 7, 12

Johnson, A., Scharver, C., and Leigh, J. Designing Cranial Implants in a Haptic Augmented Reality Environment. *Communications of the ACM*, 47(8):32–38, 2004. ISSN 00010782. URL http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=13944750&site=ehost-live&scope=site. 28

Kamper, D. G., George Hornby, T., and Rymer, W. Z. Extrinsic flexor muscles generate concurrent flexion of all three finger joints. *J. Biomech.*, 35(12):1581–1589, 2002. URL http://www.ncbi.nlm.nih.gov/pubmed/12445611. 144

Kazhdan, M., Bolitho, M., and Hoppe, H. Poisson surface reconstruction. In *Proc. of the Eurographics Symposium on Geometry Processing*, 2006. 25, 33, 114

Keskin, C., Kirac, F., Kara, Y. E., and Akarun, L. Hand Pose Estimation and Hand Shape Classification Using Multi-layered Randomized Decision Forests. In *Proc. ECCV*, 2012. URL http://link.springer.com/chapter/10.1007/978-3-642-33783-3_61?LI=true. 50, 52, 54

Kidd, C. D., Orr, R., Abowd, G. D., Atkeson, C. G., Essa, I. A., MacIntyre, B., Mynatt, E. D., Starner, T., and Newstetter, W. The aware home: A living laboratory for ubiquitous computing research. In *Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*, CoBuild '99, pages 191–198, London, UK, UK, 1999. Springer-Verlag. ISBN 3-540-66596-X. URL http://dl.acm.org/citation.cfm?id=645969.674887. 30, 85

Kim, J., He, J., Lyons, K., and Starner, T. The Gesture Watch: A Wireless Contact-free Gesture based Wrist Interface. In *IEEE ISWC*, pages 1–8, 2007. ISBN 9781424414529. doi: 10.1109/ISWC.2007.4373770. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4373770. xiii, 21, 22, 33, 136

Kim, S. W., Treskunov, A., and Marti, S. DRIVE: Directly Reaching Into Virtual Environment with bare hand manipulation behind mobile display. In *Proc. IEEE 3D UI*, pages 107–108, 2011. ISBN 978-1-4577-0063-7. 28

Klein, G. and Murray, D. W. Parallel tracking and mapping for small AR workspaces. In *Proc. (ISMAR)*, 2007. xiii, 26, 34, 91

Knödel, S. and Hachet, M. Multi-touch RST in 2D and 3D Spaces: Studying the Impact of Directness on User Performance. In *Proc. IEEE 3D UI*, 2011. URL `http://iparla.inria.fr/publications/2011/KH11`. 77

Kohli, P., Ladický, L., and Torr, P. H. Robust higher order potentials for enforcing label consistency. *Int. J. Comput. Vision*, 82(3):302–324, May 2009. ISSN 0920-5691. 162

Koike, H., Sato, Y., and Kobayashi, Y. Integrating paper and digital information on enhanceddesk: a method for realtime finger tracking on an augmented desk system. *ACM Transactions on Computer-Human Interaction*, 8(4):307–322, 2001. 14

Krueger, M. W. *Artificial reality II*, volume 10. Addison-Wesley Reading (Ma), 1991. 117

Krueger, M. W., Gionfriddo, T., and Hinrichsen, K. Videoplace: an artificial reality. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '85, pages 35–40, New York, NY, USA, 1985. ACM. ISBN 0-89791-149-0. doi: 10.1145/317456.317463. xii, 14, 15, 30

Lanman, D. and Luebke, D. Near-eye light field displays. In *ACM SIGGRAPH 2013 Emerging Technologies*, SIGGRAPH '13, pages 11:1–11:1, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2340-6. doi: 10.1145/2503368.2503379. URL `http://doi.acm.org/10.1145/2503368.2503379`. 5

Lanman, D. and Taubin, G. Build your own 3d scanner: 3d photography for beginners. In *ACM SIGGRAPH 2009 Courses*, page 8. ACM, 2009. 13

Larson, E., Cohn, G., Gupta, S., Ren, X., Harrison, B., Fox, D., and Patel, S. Heatwave: Thermal imaging for surface user interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2565–2574, New York,

NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1979317. URL http://doi.acm.org/10.1145/1978942.1979317. 161

Layar. http://www.layar.com/, 2014. 5

Leap Motion Inc. http://leapmotion.com/product, 2013. xiii, 18, 19

Lee, J., Park, K. S., and Hahn, M. The 3d sensor table for bare hand tracking and posture recognition. In *Advances in Multimedia Modeling*, pages 138–146. Springer, 2006. 17, 162

Lee, J., Olwal, A., Ishii, H., and Boulanger, C. Spacetop: integrating 2d and spatial 3d interactions in a see-through desktop environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 189–192. ACM, 2013. 17

Lee, S. C., Li, B., and Starner, T. AirTouch: Synchronizing In-air Hand Gesture and On-body Tactile Feedback. In *IEEE ISWC*, pages 3–10, 2011. ISBN 9781457707742. doi: 10.1109/ISWC.2011.27. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5959583. 21, 136

Lee, S., Buxton, W., and Smith, K. C. A multi-touch three dimensional touch-sensitive tablet. *SIGCHI Bull.*, 16(4):21–25, April 1985. ISSN 0736-6906. doi: 10.1145/1165385.317461. URL http://doi.acm.org/10.1145/1165385.317461. 3

Lee, T. and Hollerer, T. Handy AR: Markerless Inspection of Augmented Reality Objects Using Fingertip Tracking. In *Proc. ISWC*, pages 1–8. IEEE, 2007. ISBN 9781424414529. doi: 10.1109/ISWC.2007.4373785. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4373785. xiii, 23, 24, 28, 34

Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S. E., Davis, J., Ginsberg, J., Shade, J., and Fulk, D. The digital Michelangelo Project: 3D scanning of large statues. In *ACM Transactions on Graphics (SIGGRAPH)*, 2000. 25, 27, 33

Li, L.-J., Socher, R., and Li, F.-F. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *CVPR*, pages 2036–2043. IEEE, 2009. ISBN 978-1-4244-3992-8. 162

Liu, Y., Weibel, N., and Hollan, J. Interactive space: A framework for prototyping multi-touch interaction on and above the desktop. In *proc. CHI*, volume 13, 2013. 17

Lopes, P. and Baudisch, P. Muscle-propelled force feedback: Bringing force feedback to mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2577–2580, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2481355. URL `http://doi.acm.org/10.1145/2470654.2481355`. 162

Low, K. Linear least-squares optimization for point-to-plane icp surface registration. Technical report, TR04-004, University of North Carolina, 2004. 98

Maimone, A. and Fuchs, H. Computational augmented reality eyeglasses. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 29–38, Oct 2013. doi: 10.1109/ISMAR.2013.6671761. 5

Malik, S. Real-time hand tracking and finger tracking for interaction. Technical report, University of Toronto, 2003. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.135.1334`. 142

Malik, S. and Laszlo, J. Visual touchpad: a two-handed gestural input device. In *Proceedings of the 6th international conference on Multimodal interfaces*, pages 289–296. ACM, 2004. xii, 14, 15, 50, 53

Marquardt, N., Jota, R., Greenberg, S., and Jorge, J. A. The continuous interaction space: interaction techniques unifying touch and gesture on and above a digital surface. In *Human-Computer Interaction–INTERACT 2011*, pages 461–476. Springer, 2011. 16

Marroquim, R., Kraus, M., and Cavalcanti, P. R. Efficient point-based rendering using image reconstruction. In *SPBG*, pages 101–108, 2007. 55

Marschner, S. R., Westin, S. H., Lafortune, E. P. F., Torrance, K. E., and Greenberg, D. P. Image-Based BRDF Measurement Including Human Skin. In *Eurographics Render-*

ing Workshop, volume 5, pages 1–15, 1999. ISBN 9781412814560. URL http://www.cs.virginia.edu/~mjh7v/bib/Marschner99.pdf. 147

Matsushita, N. and Rekimoto, J. Holowall: designing a finger, hand, body, and object sensitive wall. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 209–210. ACM, 1997. xii, 14, 15

Mayol-Cuevas, W., Tordoff, B., and Murray, D. On the Choice and Placement of Wearable Vision Sensors. *IEEE Trans. Syst. Man Cybern. Part A Syst. Humans*, 39 (2):414–425, March 2009. ISSN 1083-4427. doi: 10.1109/TSMCA.2008.2010848. URL http://dl.acm.org/citation.cfm?id=1656621.1656633. 24

McKenna, M. Interactive viewpoint control and three-dimensional operations. In *Proc SI3D*, volume 25, pages 53–56. ACM, 1992. ISBN 0897914678. doi: 10.1145/147156.147163. URL http://portal.acm.org/citation.cfm?id=147156.147163. 29

Merrell, P., Akbarzadeh, A., Wang, L., Mordohai, P., Frahm, J.-M., Yang, R., Nistér, D., and Pollefeys, M. Real-time visibility-based fusion of depth maps. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2007. 25, 27, 33, 34

Metcalf, C., Notley, S., Chappell, P., Burridge, J., and Yule, V. Validation and Application of a Computational Model for Wrist and Hand Movements using Surface Markers. *IEEE Trans. Biomed. Eng.*, 55(3):1199–1210, 2008. URL http://eprints.soton.ac.uk/265184/. 151, 152, 153

Microsoft. http://www.xbox.com/kinect, 2014. xii, 4, 6, 12

Mistry, P. and Maes, P. SixthSense: a wearable gestural interface. In Anjyo, K., editor, *ACM SIGGRAPH ASIA Sketches*, page 60558. ACM, 2009. ISBN 9781605588582. doi: 10.1145/1667146.1667160. URL http://portal.acm.org/citation.cfm?id=1667160&dl=ACM. xiii, 23, 24, 33, 155

Mistry, P., Maes, P., and Chang, L. Wuw - wear ur world: a wearable gestural interface. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA

'09, pages 4111–4116, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-247-4. doi: 10.1145/1520340.1520626. URL http://doi.acm.org/10.1145/1520340.1520626. 31, 122, 131

Moeller, J., Kerne, A., and Moeller, J. Zerotouch: An optical multi-touch and free-air interaction architecture. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, pages 2165–2174. ACM, 2012. 17

Molyneaux, D., Gellersen, H., Kortuem, G., and Schiele, B. Cooperative augmentation of smart objects with projector-camera systems. In Krumm, J., Abowd, G., Seneviratne, A., and Strang, T., editors, *UbiComp 2007: Ubiquitous Computing*, volume 4717 of *Lecture Notes in Computer Science*, pages 501–518. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-74852-6. doi: 10.1007/978-3-540-74853-3_29. URL http://dx.doi.org/10.1007/978-3-540-74853-3_29. 32, 123, 128

Mulder, J. D. and Liere, R. V. The personal space station: Bringing interaction within reach. In *Proc. VRIC*, 2002. xiv, 28, 29

Naemura, T., Nitta, T., Mimura, A., and Harashima, H. Virtual shadows - enhanced interaction in mixed reality environment. In *Proceedings of the IEEE Virtual Reality Conference 2002*, VR '02, pages 293–, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1492-8. URL http://dl.acm.org/citation.cfm?id=580130.835870. 117, 120

Nakashima, B. K. et al. A 2D-3D Integrated Tabletop Environment for Multi-user Collaboration. *Computer Animation and Virtual Worlds*, 18(1):39–56, November 2006 2007. doi: 10.1002/cav. 16

Nakatsuma, K. and Shinoda, H. Touch interface on back of the hand. In *ACM SIGGRAPH Emerging Technologies*, page 4503, 2011. URL http://dl.acm.org/citation.cfm?id=2048278. 21

Newcombe, R. A. and Davison, A. J. Live dense reconstruction with a single moving camera. In *Proc. of the IEEE (CVPR)*, 2010. 26, 27, 34

Newcombe, R. A., Lovegrove, S., and Davison, A. J. Dense Tracking and Mapping in Real-Time. In *Proc. of the Int. Conf. on Computer Vision ({ICCV})*, 2011. 26, 27, 34

Newcombe et al., R. A. Real-Time Dense Surface Mapping and Tracking with Kinect. In *Submission to ISMAR '11*, 2011. 94, 103

Ng, A., Lepinski, J., Wigdor, D., Sanders, S., and Dietz, P. Designing for low-latency direct-touch input. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 453–464, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1580-7. doi: 10.1145/2380116.2380174. URL `http://doi.acm.org/10.1145/2380116.2380174`. 35

Nintendo. http://www.nintendo.com/wii, 2014. 5, 6, 12

Occulus Rift. http://www.oculusvr.com/, 2014. 5

Oikonomidis, I., Kyriazis, N., and Argyros, A. A. Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *IEEE ICCV*, pages 2088 – 2095, November 2011. ISBN 10.1109/ICCV.2011.6126483. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6126483&tag=1`. xiii, 20, 21, 34

Olwal, A. *Unobtrusive Augmentation of Physical Environments: Interaction Techniques, Spatial Displays and Ubiquitous Sensing*. PhD thesis, KTH Royal Institute of Technology, 2009. 5

Olwal, A. et al. ASTOR: An Autostereoscopic Optical See-through Augmented Reality System. In *Proc. (ISMAR)*, pages 24–27. IEEE, 2005. ISBN 0769524591. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1544658`. xiv, 28, 29

Osher, S. and Fedkiw, R. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002. 99, 101

Pamplona, V. The image-based data glove. In *Symposium on Virtual and Augmented Reality*, pages 204–211, 2008. URL `http://vitorpamplona.com/deps/papers/2008_SVR_IBDG.pdf`. xiii, 23, 24, 25

Papagiannakis, G. et al. A survey of mobile and wireless technologies for augmented reality systems. *Comput. Animat. Virtual Worlds*, 19:3–22, February 2008. ISSN

1546-4261. doi: 10.1002/cav.v19:1. URL http://dl.acm.org/citation.cfm?id=1348083.1348087. 27

Paradiso, J. A. Several sensor approaches that retrofit large surfaces for interactivity. In *UbiComp 2002 Workshop on Collaboration with Interactive Walls and Tables, Gothenburg, Sweden*, 2002. 17

Paris, S. and Durand, F. A Fast Approximation of the Bilateral Filter Using a Signal Processing Approach. *IJCV*, 81(1):24–52, December 2007. ISSN 0920-5691. doi: 10.1007/s11263-007-0110-8. URL http://www.springerlink.com/content/l2262j3814733q56/. 68

Pinhanez, C. The everywhere displays projector: A device to create ubiquitous graphical interfaces. In Abowd, G., Brumitt, B., and Shafer, S., editors, *Ubicomp 2001: Ubiquitous Computing*, volume 2201 of *Lecture Notes in Computer Science*, pages 315–331. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42614-1. doi: 10.1007/3-540-45427-6_27. URL http://dx.doi.org/10.1007/3-540-45427-6_27. 32

Polhemus. http://www.polhemus.com/, 2014. 5, 6

Poston, T. and Serra, L. The virtual workbench: dextrous VR. In *Proc. ACM VRST*, pages 111–121, 1994. ISBN 981-02-1867-2. URL http://dl.acm.org/citation.cfm?id=207072.207133. xiv, 28, 29

Prachyabrued, M. and Borst, C. W. Dropping the ball: Releasing a virtual grasp. In *Proc. ACM VRST*, pages 59–66, 2011. ISBN 978-1-4577-0063-7. URL http://dl.acm.org/citation.cfm?id=2013881.2014212. xiv, 28, 29, 69, 70

Prados, E. and Faugeras, O. Shape From Shading. In Paragios, N., Chen, Y., and Faugeras, O., editors, *Handbook of Mathematical Models in Computer Vision*, pages 375–388. Springer, 2006. URL http://hal.inria.fr/inria-00377417/PDF/chapter-prados-faugeras.pdf. 147

Qualcomm Vuforia. http://www.vuforia.com/, 2014. 5

Raskar, R., Baar, J., Beardsley, P., Willwacher, T., Rao, S., and Forlines, C. ilamps: Geometrically aware and self-configuring projectors. *ACM Trans. Graph*, 22:809–818, 2003. 31, 130, 131

Raskar, R., Beardsley, P., van Baar, J., Wang, Y., Dietz, P., Lee, J., Leigh, D., and Willwacher, T. Rfig lamps: interacting with a self-describing world via photosensing wireless tags and projectors. *ACM Trans. Graph.*, 23(3):406–415, August 2004. ISSN 0730-0301. doi: 10.1145/1015706.1015738. URL `http://doi.acm.org/10.1145/1015706.1015738`. 31, 32, 130, 131

Rekimoto, J. GestureWrist and GesturePad: unobtrusive wearable interaction devices. In *IEEE ISWC*, volume 5, pages 21–27, 2001. ISBN 0769513182. doi: 10.1109/ISWC.2001.962092. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=962092`. xiii, 21, 22, 33

Rekimoto, J. Traxion: A tactile interaction device with virtual force sensation. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 427–432, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2268-3. doi: 10.1145/2501988.2502044. URL `http://doi.acm.org/10.1145/2501988.2502044`. 162

Remondino, F. and Stoppa, D. *Tof range-imaging cameras*. Springer, 2013. 7, 13

Romero, J., Kjellstrom, H., and Kragic, D. Hands in action: real-time 3D reconstruction of hands in interaction with objects. In *IEEE ICRA*, pages 458–463, 2010. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5509753`. 20

Rusinkiewicz, S. and Levoy, M. Efficient variants of the icp algorithm. *3D Digital Imaging and Modeling, Int. Conf. on*, 0:145, 2001. doi: http://doi.ieeecomputersociety.org/10.1109/IM.2001.924423. 94, 97

Rusinkiewicz, S., Hall-Holt, O., and Levoy, M. Real-time 3D model acquisition. In *ACM Transactions on Graphics (SIGGRAPH)*, 2002. 27

Salas-Moreno, R., Newcombe, R., Strasdat, H., Kelly, P., and Davison, A. Slam++: Simultaneous localisation and mapping at the level of objects. In *Computer Vision*

and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 1352–1359, June 2013. 162

Saponas, T. S., Tan, D. S., Morris, D., Balakrishnan, R., Turner, J., and Landay, J. A. Enabling always-available input with muscle-computer interfaces. In *ACM UIST*, pages 167–176, 2009. ISBN 9781605587455. doi: 10.1145/1622176.1622208. URL `http://portal.acm.org/citation.cfm?doid=1622176.1622208`. xiii, 21, 22, 33

Sato, Y., Kobayashi, Y., and Koike, H. Fast tracking of hands and fingertips in infrared images for augmented desk interface. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, FG '00, pages 462–, Washington, DC, USA, 2000. IEEE Computer Society. ISBN 0-7695-0580-5. URL `http://dl.acm.org/citation.cfm?id=795661.796230`. 118

Scharstein, D. and Szeliski, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *International Journal of Computer Vision*, 2002. 7, 46

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. Real-time Human Pose Recognition in Parts from Single Depth Images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 7, 12, 34, 35, 50, 51, 54, 120, 128

Sinclair, M., Pahud, M., and Benko, H. Touchmover: Actuated 3d touchscreen with haptic feedback. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '13, pages 287–296, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2271-3. doi: 10.1145/2512349.2512805. URL `http://doi.acm.org/10.1145/2512349.2512805`. 162

Sodhi, R., Poupyrev, I., Glisson, M., and Israr, A. Aireal: Interactive tactile experiences in free air. *ACM Trans. Graph.*, 32(4):134:1–134:10, July 2013. ISSN 0730-0301. doi: 10.1145/2461912.2462007. URL `http://doi.acm.org/10.1145/2461912.2462007`. 162

Starner, T., Weaver, J., and Pentland, A. A wearable computer based American sign language recognizer. In *IEEE ISWC*, volume 1 of *ISWC*

'97, pages 130–137, 1997a. ISBN 0-8186-8192-6. doi: 10.1109/ISWC.1997.629929. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=629929. xiii, 23, 24, 33, 34

Starner, T., Auxier, J., Ashbrook, D., and Gandy, M. The gesture pendant: a self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. In *IEEE ISWC*, pages 87–94, 2000. ISBN 0769507956. doi: 10.1109/ISWC.2000.888469. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=888469. xiii, 23, 24, 33, 155

Starner, T., Mann, S., Rhodes, B., Levine, J., Healey, J., Kirsch, D., Picard, R. W., and Pentland, A. Augmented reality through wearable computing. *Presence: Teleoperators and Virtual Environments*, 6(4):386–398, 1997b. 3, 5

Stavness, I., Lam, B., and Fels, S. pCubee : A Perspective-Corrected Handheld Cubic Display. In *Proc. ACM CHI*, pages 1381–1390, 2010. ISBN 9781605589299. doi: 10.1145/1753326.1753535. URL http://doi.acm.org/10.1145/1753326.1753535. 29

Strecha, C. and Van Gool, L. PDE-based multi-view depth estimation. In *Proc. 3DIMPVT*, pages 416–425, 2002. 49

Subramanian, S., Aliakseyeu, D., and Lucero, A. Multi-layer interaction for digital tables. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 269–272. ACM, 2006. 17

Sugimoto, M., Miyahara, K., Inoue, H., and Tsunesada, Y. Hotaru: Intuitive manipulation techniques for projected displays of mobile devices. In Costabile, M. and Patern, F., editors, *Human-Computer Interaction - INTERACT 2005*, volume 3585 of *Lecture Notes in Computer Science*, pages 57–68. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-28943-2. doi: 10.1007/11555261_8. URL http://dx.doi.org/10.1007/11555261_8. xiv, 31

Sutherland, I. E. Sketchpad: A man-machine graphical communication system. In *Proceedings of the May 21-23, 1963, Spring Joint Computer Conference*, AFIPS '63 (Spring), pages 329–346, New York, NY, USA, 1963. ACM. doi: 10.1145/1461551.1461591. URL http://doi.acm.org/10.1145/1461551.1461591. 2

Sutherland, W. R. *The On-Line Graphical Specification of Computer Procedures*. PhD thesis, Massachusetts Institute of Technology, 1966. 2

Takeoka, Y., Miyaki, T., and Rekimoto, J. Z-touch: an infrastructure for 3d gesture interaction in the proximity of tabletop surfaces. In *ACM International Conference on Interactive Tabletops and Surfaces*, pages 91–94. ACM, 2010. 17

Teller, S., Chen, J., and Balakrishnan, H. Pervasive pose-aware applications and infrastructure. *IEEE Comput. Graph. Appl.*, 23(4):14–18, July 2003. ISSN 0272-1716. doi: 10.1109/MCG.2003.1210859. URL http://dx.doi.org/10.1109/MCG.2003.1210859. 30, 116

Thrun, S., Burgard, W., and Fox, D. *Probabilistic Robotics*. Intelligent robotics and autonomous agents series. Mit Press, 2005. ISBN 9780262201629. URL http://books.google.fr/books?id=k_yOQgAACAAJ. 5, 6, 11

Thrun, S. Robotic mapping: A survey. In Lakemeyer, G. and Nebel, B., editors, *Exploring Artificial Intelligence in the New Millennium*, chapter 1, pages 1–35. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. ISBN 1-55860-811-7. URL http://dl.acm.org/citation.cfm?id=779343.779345. 25

Van Krevelen, D. and Poelman, R. A survey of augmented reality technologies, applications and limitations. *International Journal of Virtual Reality*, 9(2):1, 2010. 27

Vardy, A. The wristcam as input device. In *IEEE ISWC*, pages 199 – 202, 1999. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=806928. xiii, 23, 24, 25, 33

Vicon. http://www.vicon.com/, 2014. 5, 6, 12

Villar, N., Izadi, S., Rosenfeld, D., Benko, H., Helmes, J., Westhues, J., Hodges, S., Ofek, E., and Butler, A. Mouse 2.0. In *ACM UIST*, pages 33–42, New York, New York, USA, October 2009. ISBN 9781605587455. doi: 10.1145/1622176.1622184. URL http://dl.acm.org/citation.cfm?id=1622176.1622184. 136

Vlasic et al., D. Dynamic shape capture using multi-view photometric stereo. *ACM Trans. Graph.*, 28(5), 2009. 27

Vuzix. http://www.vuzix.com/), 2014. 5

Wagner, D. *Handheld Augmented Reality*. PhD thesis, Graz University of Technology, 2007. 5

Wagner, D., Langlotz, T., and Schmalstieg, D. Robust and unobtrusive marker tracking on mobile phones. In *ISMAR*, pages 121–124, 2008. 12, 27

Wang, R., Paris, S., and Popović, J. 6d hands: markerless hand-tracking for computer aided design. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 549–558. ACM, 2011. 18, 20, 34

Wang, R. Y. and Popović, J. Real-time hand-tracking with a color glove. *SIGGRAPH Comput. Graph.*, 28(3):1, 2009. ISSN 07300301. doi: 10.1145/1531326.1531369. URL `http://portal.acm.org/citation.cfm?doid=1531326.1531369`. xiii, 20, 21

Ware, C., Arthur, K., and Booth, K. S. Fish Tank Virtual Reality. In *Proc. ACM CHI*, pages 37–42. ACM, 1993. ISBN 0897915755. doi: 10.1145/169059.169066. URL `http://portal.acm.org/citation.cfm?id=169059.169066`. 29, 78, 81, 83

Weise, T., Leibe, B., and Van Gool, L. Fast 3d scanning with automatic motion compensation. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007. 13

Weise, T., Wismer, T., Leibe, B., and Gool, L. V. In-hand scanning with online loop closure. In *IEEE Int. Workshop on 3-D Digital Imaging and Modeling*, 2009. 27, 89

Wellner, P. Interacting with paper on the digitaldesk. *Communications of the ACM*, 36 (7):87–96, 1993. 14

Wesche, G. and Seidel, H.-P. Freedrawer: a free-form sketching system on the responsive workbench. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 167–174. ACM, 2001. 17

White, W. Method for optical comparison of skin friction-ridge patterns, August 17 1965. URL `http://www.google.com/patents/US3200701`. US Patent 3,200,701. 14

Wikipedia. http://en.wikipedia.org/wiki/microsoft_pixelsense, 2014. 14

Willis, K. D., Poupyrev, I., Hudson, S. E., and Mahler, M. Sidebyside: ad-hoc multi-user interaction with handheld projectors. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 431–440, New York, NY, USA, 2011a. ACM. ISBN 978-1-4503-0716-1. doi: 10.1145/2047196.2047254. URL `http://doi.acm.org/10.1145/2047196.2047254`. xiv, 31

Willis, K. D., Poupyrev, I., and Shiratori, T. Motionbeam: a metaphor for character interaction with handheld projectors. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1031–1040, New York, NY, USA, 2011b. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1979096. URL `http://doi.acm.org/10.1145/1978942.1979096`. xiv, 31

Wilson, A. D. Playanywhere: a compact interactive tabletop projection-vision system. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 83–92. ACM, 2005. xii, 6, 14, 15

Wilson, A. D. Robust computer vision-based detection of pinching for one and two-handed gesture input. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 255–258. ACM, 2006. 53

Wilson, A. D. Simulating Grasping Behavior on an Imaging Interactive Surface. In *Proc. ACM ITS*, pages 125–132. ACM, 2007a. ISBN 978-1-60558-733-2. doi: http://doi.acm.org/10.1145/1731903.1731929. 69

Wilson, A. D. Depth-sensing video cameras for 3d tangible tabletop interaction. In *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP'07. Second Annual IEEE International Workshop on*, pages 201–204. IEEE, 2007b. 16

Wilson, A. D. Using a depth camera as a touch sensor. In *ACM international conference on interactive tabletops and surfaces*, pages 69–72. ACM, 2010. 18

Wilson, A. D. and Benko, H. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proc. ACM UIST*, pages 273–282, 2010. ISBN 978-1-4503-0271-5. doi: http://doi.acm.org/10.1145/1866029.1866073. URL `http://doi.acm.org/10.1145/1866029.1866073`. xiv, 30, 31, 32, 34, 85, 121

Wilson, A. D. et al. Bringing Physics to the Surface. In *Proc. ACM UIST*, pages 67–76, 2008. ISBN 978-1-59593-975-3. 66, 69, 83, 119

Wise, S., Gardner, W., Sabelman, E., Valainis, E., Wong, Y., Glass, K., Drace, J., and Rosen, J. M. Evaluation of a fiber optic glove for semi-automated goniometric measurements. *Int. J. Rehabil. Res.*, 27(4):411–424, 1990. URL `http://www.rehab.research.va.gov/jour/90/27/4/pdf/wise.pdf`. 152, 153

Xu, H., Iwai, D., Hiura, S., and Sato, K. User interface by virtual shadow projection. In *SICE-ICASE, 2006. International Joint Conference*, pages 4814–4817, 2006. doi: 10.1109/SICE.2006.314974. 117

Yoshida, T. et al. Repro3d: full-parallax 3d display using retro-reflective projection technology. In *ACM Transactions on Graphics (SIGGRAPH)*. ACM, 2010. ISBN 978-1-4503-0392-7. doi: http://doi.acm.org/10.1145/1836821.1836841. URL `http://doi.acm.org/10.1145/1836821.1836841`. 29

Zhang, S. Recent progresses on real-time 3d shape measurement using digital fringe projection techniques. *Optics and Lasers in Engineering*, 48(2):149 – 158, 2010. ISSN 0143-8166. 13

Zhang, Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000. ISSN 01628828. doi: 10.1109/34.888718. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=888718`. 67, 142

Zhou, F. Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. In *Proc. (ISMAR)*, pages 193–202, 2008. ISBN 978-1-4244-2840-3. doi: http://dx.doi.org/10.1109/ISMAR.2008.4637362. URL `http://dx.doi.org/10.1109/ISMAR.2008.4637362`. 27

Zhou, K., Gong, M., Huang, X., and Guo, B. Data-parallel octrees for surface reconstruction. *IEEE Trans. on Visualization and Computer Graphics*, 17, 2011. 25, 33, 100

Zimmerman, T. G., Lanier, J., Blanchard, C., Bryson, S., and Harvill, Y. A hand gesture interface device. In *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface*, CHI '87, pages 189–192, New York, NY, USA, 1987. ACM. ISBN 0-89791-213-6. doi: 10.1145/29933.275628. URL `http://doi.acm.org/10.1145/29933.275628`. 5