UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Thomas Vincent

Thèse dirigée par **Laurence Nigay**
et codirigée par **Takeshi Kurata**

préparée au sein **du Laboratoire d'Informatique de Grenoble (LIG) équipe Ingénierie de l'Interaction Homme-Machine (IIHM)**
et de **l'école doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique (ED-MSTII)**

Handheld Augmented Reality Interaction: Spatial Relations

Thèse soutenue publiquement le **2 octobre 2014**,
devant le jury composé de :

Mr, Emmanuel Dubois
Professeur, Université de Toulouse, Président
Mr, Stéphane Huot
Directeur de Recherche, INRIA Lille, Rapporteur
Mr, Matt Jones
Professeur, Swansea University, Rapporteur
Mr, Tewfik Meftah
Docteur, Schneider Electric, Examinateur
Mr, Hartmut Seichter
Professeur, Fachhochschule Schmalkalden, Examinateur
Mr, Takeshi Kurata
Professeur, AIST, University of Tsukuba, Co-Directeur de thèse
Mme, Laurence Nigay
Professeur, Université Joseph Fourier - Grenoble 1, Directeur de thèse

Acknowledgements

In the first place, thanks to the members of the jury. Thanks to Matt Jones and Stéphane Huot for accepting to review my work and providing insightful comments. Thanks to Emmanuel Dubois, Hartmut Seichter and Tewfik Meftah for accepting to take part in the jury.

Thanks to everyone who contributed to this thesis and collaborated during those three years. Thanks to Laurence Nigay and Takeshi Kurata my supervisors for taking me as a PhD student, guiding my work and working with me throughout those three years. Thanks to the AMIE project members and to the team in AIST, Koji Makita, Masakatsu Kourogi and Jun Nishida for the collaboration while preparing demonstrations. Thanks to the internship students I had the chance to follow during this thesis: Sébastien, Matthieu, Diego, Rémi. Thanks to George for kindly proof-reading the English of papers we submitted and to Benjamin Penin for proof-reading the English of most of this manuscript.

Thanks to all the members of the EHCI team for the work atmosphere, the discussions, the coffee breaks and the climbing sessions. Thanks to Joëlle Coutaz and Renaud Blanch for welcoming me in the team in 2008 and giving me the taste of HCI that motivated me to start this thesis. Thanks to Michael Ortega for the fruitful collaboration and all the rest.

Finally, thanks to Leïla my wife and my kids for everything.

Contents

Chapter 1

Introduction

Chapter Content

1.1 Field

This thesis belongs to the research field of Human-Computer Interaction (HCI). Its aim is to explore interaction with handheld Augmented Reality (AR) systems.

Augmented Reality (AR) systems aim to merge the user's perception of digital information with their perception of the physical environment. Caudell and Mizell first coined the term "Augmented Reality" in 1992 to denote a system that displays digital information registered in the physical world [Caudell and Mizell, 1992]. In 1997, Azuma proposed his initial definition of an AR system [Azuma, 1997] (which he confirmed and updated in 2001 [Azuma et al., 2001]). An AR system is defined as a system that:

- Combines real and virtual objects in a real environment
- Runs interactively and in real time
- Registers (i.e., aligns) real and virtual objects with each other

This definition gives AR a wide scope, rather than limiting the concept to specific technologies or interaction modalities. While displaying digital information registered in the physical surroundings has been explored for different human senses (e.g., audio [Betsworth et al., 2013], haptic [Robinson et al., 2009]), combining the perception of digital objects with that of physical objects has mostly been studied for the visual sense.

Our research has also embraced the mainstream concern that is **visual augmentation**.

This definition does not restrict AR to specific input/output devices. For instance, different types of display device can provide AR content: optical and video see-through Head-Mounted Displays, camera-equipped handheld devices or projection-based displays (also known as Spatial AR). Different set-ups offer different ways of controlling the viewpoint in the augmented scene and different interaction capabilities. Handheld AR is a specific case where a camera-equipped handheld device, such as a smart phone or tablet, is used to display AR content. With such set-ups, the handheld device's screen displays a view of the physical environment that is usually captured by the device's back-facing camera. The camera images are overlaid with computer-generated graphics that are aligned with the physical objects viewed by the camera (Figure 1). For example, *NaviCam* [Rekimoto, 1995], a seminal prototype handheld AR application, overlays text information on the images shot by the handheld device's back-facing camera and displayed on the screen. The overlaid information relates to the physical objects being viewed. The system actually uses a vision-based algorithm to detect fiducial markers in the camera images and then overlay the corresponding information.

In this thesis we focus on **handheld AR** systems.

Indeed, it is possible to run standalone AR applications with he handheld devices and tracking systems (e.g., Vuforia SDK[1]) that are currently available. Moreover, unlike Head-Mounted Displays, which the user must wear, handheld device set-ups allow the user to use AR systems more casually, thus allowing better integration of AR systems into other tasks.

To enable the digital content's spatial alignment with the perception of the physical environment, a handheld AR system relies on a tracking system that estimates camera position and orientation relative to the physical surroundings (e.g., a vision-based tracking algorithm). In addition, rendering digital content for seamless integration into the physical surroundings is challenging for display devices and in terms of computer-graphics (e.g., realistic lighting and depth-congruent display device). Thus, handheld AR systems (and AR systems in general) are challenging use cases for tracking systems, but also for rendering techniques and display devices.

However, beyond these challenges, **user interaction** with handheld AR systems also needs further research, as it is subject to an array of specific constraints that it is important to address.

First, handheld AR systems inherit the specific constraints of handheld devices. Screen real estate is limited and direct touch on the screen, the de facto standard input modality on such devices, suffers from the 'fat-finger' problem [Roudaut et al., 2008]. When interacting with the touch-screen, the finger occludes the screen. With direct-touch interaction on the screen, the active point used by the system is ambiguous as it lies under the finger and is not represented by a cursor. What is more, unlike with a mouse, touch interaction lacks a hovering state [Buxton, 1990].

Furthermore, in the specific case of handheld AR, because the camera images representing the physical surroundings and the digital augmentation are displayed on the screen simultaneously (as schematized in Figure 1), the competition for screen real estate is even more intense. For instance, let us consider a case where many annotations (e.g., 20 annotations linked to the tree, as opposed to the three annotations in Figure 1) must be displayed on top of the camera images: first, the screen is too small to be able to display the full title of every annotation; secondly, there

[1]https://www.vuforia.com/

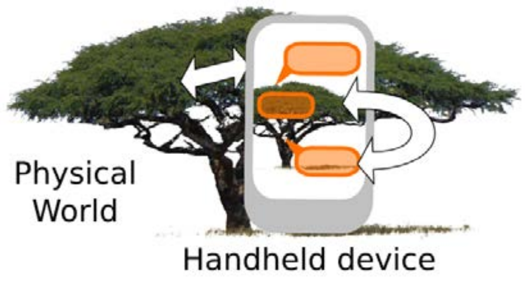


Figure 1.1: Handheld Augmented Reality: The camera images representing the Physical World and the digital Augmentation share the handheld device's screen space. The spatial relation between the physical surroundings and the on-screen content is a key characteristic of AR, but also a constraint for interaction.

are so many annotations displayed that the camera images representing the physical surroundings (e.g., the tree in Figure 1) is no longer visible.

In addition, the spatial coupling between the on-screen content and the physical surroundings makes touch interaction difficult: the viewpoint is controlled by the device's position and orientation in space and its stability is impaired by hand tremor. Tracking errors further diminish the stability of digital augmentation. In turn, this instability impairs interaction.

We have based our exploration of handheld AR interaction on **spatial relations**. Indeed, when compared to other forms of handheld device interaction, AR brings additional constraints due to the tight coupling between the physical surroundings, the on-screen content and the handheld device. Indeed, a unique characteristic of AR is the spatial relation between the physical surroundings and the digital objects. We will examine this relation with respect to the different frames of reference involved in interaction.

1.2 Context: international collaborations

This thesis is part of the AMIE[2] international research project. The AMIE project is funded by the French national research agency (ANR) and the Japan Science and Technology Agency (JST). AMIE stands for *Augmented Mobile Interactive Experience: Application to Maintenance Services.*

The aim of this project is to explore interaction within the context of Augmented Reality on handheld devices. The field of application of this project is machine maintenance in production plants. This is a multidisciplinary project involving two complementary academic teams: one dedicated to Sensor-Data fusion techniques for Localization/Registration (CfSR, AIST, Tsukuba, Japan), the other to Human-Computer Interaction (EHCI, LIG, Grenoble, France). Two industrial partners represent the machine maintenance field of application: Digital Electronics (Osaka, Japan) and Schneider Electric (Grenoble, France). The AMIE project therefore focuses on handheld AR systems for production plant maintenance operators, where augmentation is provided based on the knowledge available about the operator's location and the information provided by the production machine automaton.

[2]http://amie.imag.fr/

Handheld AR seems promising for use cases where the users need to access digital information related to physical locations such as production plant machine maintenance. Indeed, during maintenance, the operators need to access dynamic digital information from the machine automaton (e.g., sensor data, actuator state) that is linked to physical parts of the machine. Furthermore, other information such as documentation can also be attached to specific parts of the machine. AR allows such association of digital information with its physical context. Furthermore, the use of a handheld device allows the operators to perform other tasks with the same device (e.g., reading a written documentation), and it is potentially more socially acceptable in an industrial context than wearing a Head-Mounted Display. Yet, it requires the operator to hold the device with her/his hands which can cause fatigue or prevent from using other tools.

The collaborative AMIE project has led to the creation of different demonstrators. In close collaboration with AIST, we conducted demonstrations at the 2011 and 2012 ISMAR conferences. Based on the work presented in this thesis and in collaboration with the other partners, we also developed demonstrators for industrial maintenance scenarios. These demonstrators have been evaluated with maintenance staff in an industrial environment, as well as being presented at AIST and at an industry exhibition in Tokyo in 2013. The work developed in collaboration with the industrial partners is not publishable and therefore not included in this thesis.

1.3 Research Goals and Methods

The overall goal of the research conducted was to improve interaction with handheld AR systems. As stated above, a key characteristic of handheld AR is the spatial relation between the physical surroundings and the digital objects. This spatial relation has a direct impact on interaction, because it makes the on-screen content unstable. In this context, the research question we addressed revolved around improving the interaction precision of handheld AR systems by relaxing spatial constraints while maintaining the digital-physical colocation that is key to AR.

The first goal was to characterize the different entities involved in handheld AR systems, together with their spatial relations, and to organize them in a design space. Based on this first contribution, the second goal was to design novel interaction techniques.

Validating the design framework involved the study of its taxonomic power and its capacity to guide the design of novel interaction techniques. Interaction techniques were evaluated through user experiments. Because this work was part of a collaborative project focusing on AR interaction for machine maintenance services in production plants, we were able to perform in-field evaluations. Finally, designing and evaluating pointing techniques allowed the framework to be validated and refined. Our two goals were therefore complementary and fed off each other in an iterative process (Figure 2).

1.4 Contributions

In line with our research approach, our contributions are three-fold: (1) a design space for handheld AR, (2) pointing techniques to improve precision, and (3) a toolkit and demonstrators for handheld AR systems.

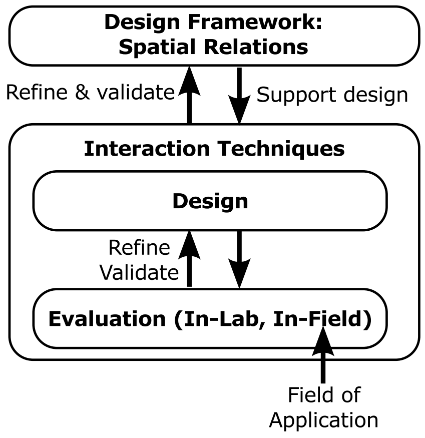


Figure 1.2: Iterative research approach

First, we proposed a design space for handheld AR on-screen content, structured around two components and highlighting the spatial relations in place. This design space allows interaction with handheld AR set-ups to be studied. The aim was to take a conceptual rather than a technical perspective on interaction design parameters.

Secondly, we proposed pointing techniques, in particular *Shift&Freeze* and *Relative Pointing*. Pointing techniques make it possible to perform generic basic tasks that can be used in a wide range of contexts to select a physical location or object, or to select a digital augmentation. The proposed techniques are geared towards improving handheld AR pointing precision. Indeed, pointing precision is limited both by touch screen input and the context of handheld AR itself. We held two sets of two experiments to evaluate the proposed pointing techniques.

Thirdly, as part of the AMIE project, we developed a toolkit. This toolkit has been used to develop demonstrators over the course of the project. Some of the demonstrators are specific to the field of maintenance and allow feedback to be gathered from maintenance operators, while others have been developed to present the interaction techniques designed and the tracking systems developed by AIST at conferences and events.

1.5 Structure of the document

The document is structured as follows:

- In chapter 2, we will first review the design schemes proposed for Mixed Reality and AR systems. We will then present our design space, which focuses on handheld AR, and discuss user interaction in this context with respect to the design space.

- In chapter 3, we will first present a review of existing pointing techniques geared towards pointing with handheld AR systems. We will then present new pointing techniques for handheld AR systems.

- In chapter 4, we will present four experiments we performed in order to evaluate the precision of our pointing techniques.

- In chapter 5, we will describe the toolkit and some of the demonstrators we developed as part of the AMIE project.
- In chapter 6, we will conclude this research thesis by providing a summary of the key findings, before identifying directions for further research.

Part I

Design Space

Chapter 2

Design Space

Design Space for Handheld Augmented Reality On-Screen Content and Interaction

Chapter Content

2.1 Introduction

This chapter presents a design space geared towards defining content and interaction for handheld AR. As discussed in the Introduction (see Chapter 1), interaction with handheld AR applications is not only impacted by handheld device specificities but also by the spatial relations between the handheld device, its on-screen content and the physical surroundings. To study interaction with handheld AR applications, we identified a set of components linked by spatial relations to define a design space.

The need to structure the design of handheld AR (and AR in general) has been highlighted by the organization of workshops focusing on these issues (*Classifying the AR Presentation Space*[1] at the ISMAR 2012 conference and *Designing Mobile Augmented Reality*[2] at the MobileHCI 2013 conference). These workshops confirm the community's growing awareness of the need for design schemes that enable the AR design space to be explored systematically.

To address this need to structure the design, this chapter presents a design space for handheld AR on-screen content. Our design space includes design axes that characterize the AR content on the screen with a particular focus on spatial relations. To describe these relations, our design space breaks down handheld AR on-screen content and highlights the spatial relations between the frames of reference of the components involved. This design space then defines a platform for the study interaction with AR on-screen content.

This design space was partly presented at workshops during ISMAR 2012 [Vincent et al., 2012] and MobileHCI 2013 [Vincent et al., 2013b] and during the French UbiMob 2013[3] event [Vincent et al., 2013d].

Before presenting the state of the art of existing classification and design schemes for the definition of AR systems, we will first describe a sample of handheld AR applications, which we will then use to illustrate the different design schemes presented in this chapter. We will then present our design space for handheld AR on-screen content and illustrate it using a sample of applications from the literature. Based on this design space, we will then study user interaction with handheld AR applications by considering the spatial relations between the frames of reference of the components of our design space.

2.2 Sample handheld AR applications

We will first present three handheld AR applications, which we will then use to illustrate the design axes presented in this chapter.

[1]http://campar.in.tum.de/Chair/IsmarClassifyingARPresentationSpace2012
[2]http://studierstube.icg.tugraz.at/mobilehci2013workshop/
[3]http://ubimob2013.sciencesconf.org/

2.2.1 NaviCam

In 1995, Rekimoto and Nagao proposed *NaviCam*, a seminal prototype handheld AR application [Rekimoto and Nagao, 1995] (Figure 2.1). When a marker is recognized in the live camera images, associated digital information is displayed as a text overlay on the marker. For example, it can provide additional information about a poster or on other objects when the associated marker is recognized. Textual information is overlaid on the camera images without strictly mapping the position of the marker on the screen. This is an example of AR text annotation.

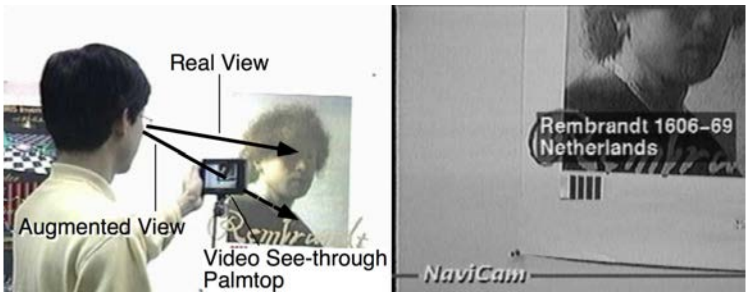


Figure 2.1: *NaviCam* (Figure from [Rekimoto and Nagao, 1995]).

2.2.2 SnapAR

SnapAR [Sukan et al., 2012] is a handheld AR application that allows users to preview 3D models of furniture in the physical context (Figure 2.2). The application allows the user to choose 3D models of furniture, such as a table, and position them on the ground. This application is an example of 3D model augmentation.

However, *SnapAR* offers a very specific feature. It allows the user to take snapshots of the augmented scene from different viewpoints. The user can then choose whether the screen displays the live camera images or a previously recorded snapshot. When a snapshot is displayed, the digital augmentation reflects its current state rather than its state at the time of the snapshot. This makes it possible to move within the augmented scene while remaining stationary in the physical space. Thus, the user can easily observe the augmented scene from different points of view.

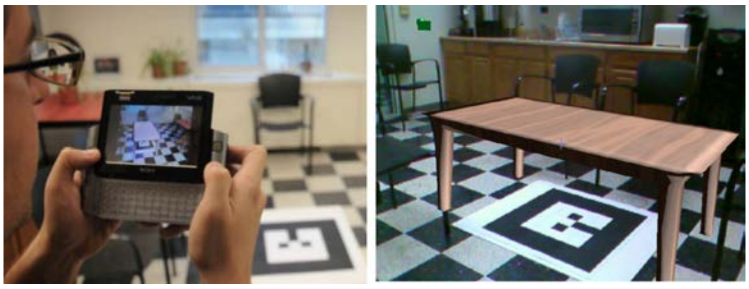


Figure 2.2: *SnapAR* (Figure from [Sukan and Feiner, 2012]).

2.2.3 Touch Projector

Touch Projector [Boring et al., 2010] is an application that allows interaction with remote displays through the camera images of a handheld device (Figure 2.3). It is not strictly a handheld AR application as no digital content is registered to the physical surroundings. However, it is similar to handheld AR applications, as it considers the physical surroundings so as to enable interaction through the camera.

With this application, when a remote screen is recognized in the camera images, the user can interact with the content of the remote screen via the touch-screen of the handheld device. It allows pictures to be moved around on remote screens or dragged from one screen to another.

Furthermore, the user can freeze the video frame to facilitate interaction. By doing so, the user can still move content on the remote screen via the handheld device's touch screen, since the still camera image is updated with a digital copy of the remote screen to reflect its current state. The application also allows the camera zoom to be controlled manually or automatically. The automatic-zoom feature zooms in when a remote screen is recognized, so as to provide a consistent Control-to-Display gain on the remote screen. The application then zooms out when the remote screen is no longer recognized.

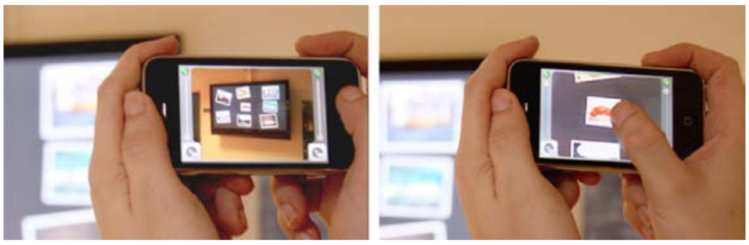


Figure 2.3: *Touch Projector* (Figure from [Boring et al., 2010]).

2.3 State of the art: classification schemes

Our review of classifications and design schemes for AR will be organized as follows. We will first review the existing classification schemes and design spaces of Mixed Reality systems. These schemes encompass AR, because they describe systems that combine physical and digital components. We will then review those classification schemes that are dedicated to AR only.

2.3.1 Classification schemes encompassing AR

We will first review the classification and design schemes dedicated to Mixed Reality systems. Such classification and design schemes have a wider coverage than AR. Indeed, an AR system is defined as being just one type of Mixed Reality system. Mixed Reality systems are systems that incorporate (or link) both physical and digital components [Renevier, 2004]. The classification and design schemes of Mixed Reality systems can therefore also define relevant axes for describing AR systems. Furthermore, such schemes define design axes that highlight the differences and common features of AR and other interaction styles such as Virtual Reality, Tangible

Interfaces and Ubiquitous Computing. However, because such schemes cover a wide range of systems, their design axes characterize systems at a rather abstract level.

2.3.1.1 Rekimoto's comparison of HCI styles

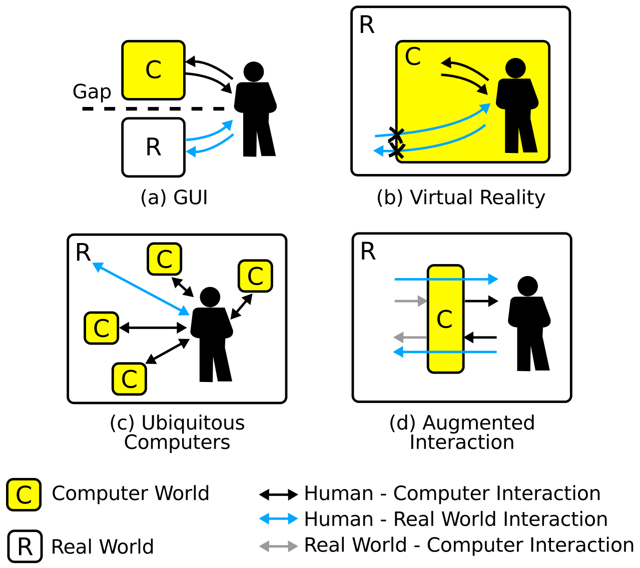

Gap
C
R
(a) GUI
(b) Virtual Reality
(c) Ubiquitous Computers
(d) Augmented Interaction
Computer World
Real World
Human - Computer Interaction
Human - Real World Interaction
Real World - Computer Interaction

Figure 2.4: Comparison of HCI styles by Rekimoto and Nagao (Figure from [Rekimoto and Nagao, 1995]).

When Rekimoto and Nagao proposed *NaviCam* (described in 2.2.1), they compared the proposed interaction style with other HCI styles in terms of human-computer and human-physical world interaction [Rekimoto and Nagao, 1995]:

- On a conventional desktop, human-computer interaction is isolated from human-physical world interaction (Figure 2.4-a).
- In Virtual Reality, a computer-generated world surrounds the user and interaction with the physical surroundings is no longer possible (Figure 2.4-b).
- In ubiquitous environments, the user can interact with both the physical surroundings and the computers embedded in the physical surroundings (Figure 2.4-c).
- "Augmented Interaction" enables and augments the user's interaction with the physical surroundings through a computer (Figure 2.4-d). It means that both the perception and actions of the user in the physical surroundings are mediated and "augmented" by a computer.

This comparison highlights the different iways of combining physical and digital components based on different interaction styles. Our work focuses on the "Augmented Interaction" section of this classification.

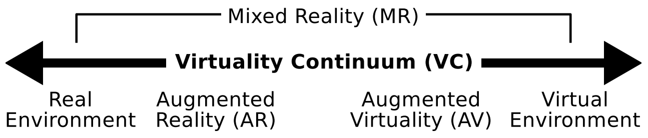


Figure 2.5: The Virtuality Continuum (Figure from [Milgram and Kishino, 1994]).

2.3.1.2 Milgram's taxonomy

Milgram [Milgram and Kishino, 1994] proposed a well-known taxonomic axis, the "Virtuality continuum", which includes AR as one of it classes (Figure 2.5). The authors suggested organizing different Mixed Reality systems along a continuum ranging from the physical environment to a purely virtual environment. In this continuum, AR is where virtual objects "augment" (i.e., add to) a physical environment. The authors also placed Augmented Virtuality in this continuum. Augmented Virtuality denotes systems where the user perceives an environment that is mainly digital, but in which physical objects are embedded.

The authors also proposed three other axes, namely:

- *Extent of World Knowledge* (Figure 2.6): This axis represents the knowledge the system has or requires about the physical environment.

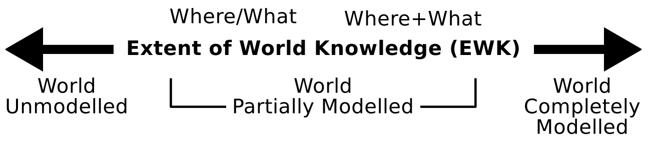


Figure 2.6: The *Extent of World Knowledge* axis (Figure from [Milgram and Kishino, 1994]).

- *Reproduction Fidelity* (Figure 2.7): This axis characterizes the image quality of both digital and physical objects. For digital objects, it represents the level of realism of the rendering. This also applies to physical objects in systems where the view of the physical environment is mediated (i.e., captured by a camera and synthesized on the display). In this case, the axis characterizes the way in which the physical environment captured is rendered to the user.

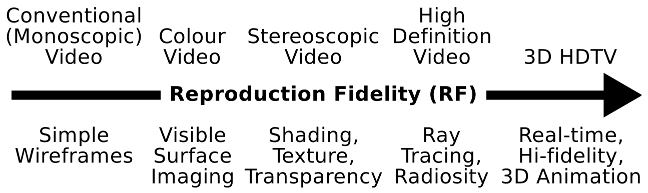

Conventional (Monoscopic) Video
Colour Video
Stereoscopic Video
High Definition Video
3D HDTV
Reproduction Fidelity (RF)
Simple Wireframes
Visible Surface Imaging
Shading, Texture, Transparency
Ray Tracing, Radiosity
Real-time, Hi-fidelity, 3D Animation

Figure 2.7: The *Reproduction Fidelity* axis (Figure from [Milgram and Kishino, 1994]).

- *Extent of Presence Metaphor* (Figure 2.8): This axis refers to the extent to which the user feels immersed within the scene displayed.

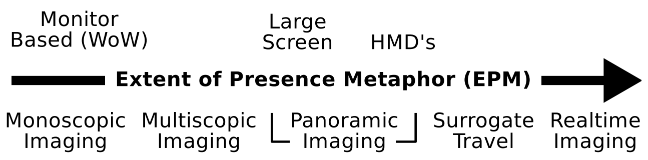


Figure 2.8: The *Extent of Presence Metaphor* axis (Figure from [Milgram and Kishino, 1994]).

Milgram and Colquhoun later extended this work by also considering interaction [Milgram and Colquhoun, 1999]. The authors studied interaction with Mixed Reality systems in terms of *Display Centricity* and *Control-Display Congruence.*

The *Display Centricity* axis defines the distance between the current viewpoint in the scene and the user's nominal viewpoint (Figure 2.9). It spans from egocentric to exocentric presentation.

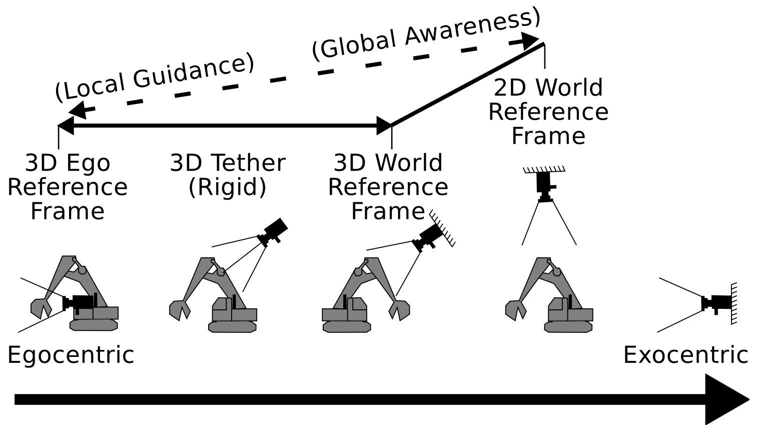


Figure 2.9: The *Display Centricity* axis of Milgram and Colquhoun. (Figure from [Milgram and Colquhoun, 1999]).

The *Control-Display Congruence* axis describes the mapping that takes place between the viewpoint and the user inputs (Figure 2.10). This axis sums up the various spatial transformations between the input space and the display space. It therefore relates to the mental transformations that the user must perform when interacting with an augmented scene. For example, to control a vehicle an input device such as a steering wheel, which provides an ego-referenced control of the viewpoint (i.e., turning left or right), is *highly congruent* with an egocentric point of view. However, this input modality is less congruent with an exocentric top view.

This set of axes provides design elements at the level of abstraction of the entire Mixed Reality systems. However, focusing on user interaction by considering the congruence between user inputs and viewpoint provides more tangible insights for the design of interaction techniques.

2.3.1.3 Dubois' taxonomy

Dubois *et al.* [Dubois et al., 1999] suggested building on Milgram's "Virtuality continuum" [Milgram and Kishino, 1994] (see 2.3.1.2) so as to focus on user interaction

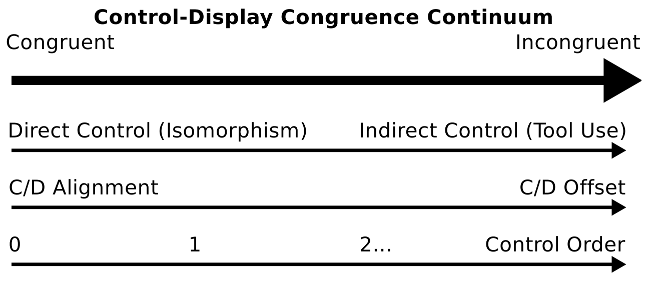

Control-Display Congruence Continuum
Congruent
Incongruent
Direct Control (Isomorphism)
Indirect Control (Tool Use)
C/D Alignment
C/D Offset
0
1
2...
Control Order

Figure 2.10: The *Control-Display Congruence* axes of Milgram and Colquhoun. (Figure from [Milgram and Colquhoun, 1999]).

with the environment and the system. They defined two axes: (1) the *target of the task*, and (2) the *type of augmentation*.

The *target of the task* is a *real* target (i.e., physical) when the user's goal is to modify her/his physical surroundings. The *target of the task* is a *virtual* target (i.e., digital) when the user's goal is to modify digital information contained in the system. By considering the *target of the task*, "Augmented Reality" can be defined as interaction with the physical world augmented by the computer, while "Augmented Virtuality" can be defined as interaction with the computer augmented by physical objects and actions. Unlike Milgram's "Virtuality continuum" [Milgram and Kishino, 1994] (see 2.3.1.2), "Augmented Reality" and "Augmented Virtuality" do not belong to the same continuum. Instead, they define two different continua according to the *target of the task*.

The *type of augmentation* defines whether the *execution* or *evaluation* phases of Norman's Theory of Action [Norman, 1986] are augmented by the computer (for the *real target of the task*) or by physical objects (for the *virtual target of the task*) (Figure 2.11).

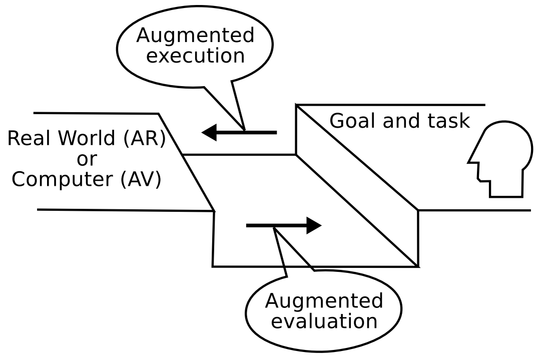


Figure 2.11: Augmented execution and/or augmented evaluation for the real or virtual targets of the task (Figure from [Dubois et al., 1999]).

These two axes (*Target of the task* and *Type of augmentation*) define four classes of Mixed Reality system.

We wil first illustrate the case of the *real target of the task*. Handheld AR applications that overlay digital information on live camera images, such as *NaviCam* (described in 2.2.1) or *SnapAR* (described in 2.2.2), are examples of augmented *evaluation*, because they augment the perception of the physical surroundings. The

DigitalDesk [Wellner, 1993], which enables the user to cut and paste drawings made on real paper using a real pen, is an example of augmented *execution*.

Tangible User Interface are examples of augmented *execution* for a *virtual target of the task*, while more realistic outputs that mimic feedback from the physical world are examples of augmented *evaluation*.

This classification explicitly incorporates user interaction with both the computer and the physical environment. This classification characterizes handheld AR applications as revolving around *real targets of the task*, and *augmented evaluation*.

2.3.1.4 Renevier's taxonomy

Renevier studied collaborative mobile Mixed Reality systems according to the relations in place between the physical world and the digital world (Table 2.1) [Renevier, 2004]. He defined Mixed Reality systems as systems that incorporate both physical and digital components through specific relations (or "augmentations"). Mixed Reality comprises three key components: (1) the user, (2) the objects used during interaction, and (3) physical-digital relations. A relation can either be (1) taking a component from one world and adding it to the other, or (2) manipulating a component in one world with tools from the other world. Relations can be created, accessed, modified or deleted.

Augmentations (i.e., physical-digital relations) are characterized by the following axes:

- *Target of the augmentation*: as proposed in [Mackay, 1998], this axis describes what the system augments: (1) the *user* (who wears devices such as Head-Mounted Displays), (2) the *physical object* (in which input/output devices are embedded), and (3) the *surrounding environment* (which is equipped with sensing and projection devices).
- *Type of augmentation*: either *execution* or *evaluation*, as in [Dubois et al., 1999] (see 2.3.1.3).
- *Temporality*: describes the length of time for which an augmentation exists. An augmentation can either be *permanent* or *transient*.
- *Interaction mode*: describes the way in which the user interacts with an augmented object. this can either be *push*, i.e., relations are automatically managed by the system, or *pull*, i.e., the user can partially or totally control creation, access, modification and deletion of the relations.

Table 2.1: Characteristics of augmentations in a Mixed Reality system [Renevier, 2004].

Axis	Classes
Target of the augmentation	User; Objects; Environment
Type of augmentation	Execution; Evaluation
Temporality	Permanent; Transient
Interaction mode	Push; Pull

For handheld AR applications, the *target of the augmentation* is, at the very least, the *user* who is holding the handheld device. However, the object and the environment can also be instrumented. Handheld AR applications mainly augment evaluation (see 2.3.1.3). The last two axes (i.e., *Temporality* and *Interaction mode*) characterize the physical-digital relation (also known as augmentation) and can be relevant to handheld AR applications.

2.3.1.5 ASUR notation

Dubois *et al.* introduced the *ASUR* notation [Dubois et al., 2001] to describe Mixed Reality systems. This notation is comprised of four kinds of components:

- *User* components (U).
- *System* components (S). Such components can either be *tools* (S_{tool}) or concepts (S_{info}, S_{obj}).
- *Real* object components (R). Such components can either be *tools* (R_{tool}) or *task* objects (R_{task}).
- *Adapter* components (A). Such components transfer data from the physical world to the computer (A_{in}) or from the computer to the physical world (A_{out}).

The *ASUR* notation also serves to define the relations between the different components. The *ASUR++* notation [Dubois et al., 2002] proposes three different types of relations between two components:

- *Exchange of data* (->) for the transfer of information between two components.
- *Physical activity triggering an action* (=>) to describe a trigger that occurs when two components meet a spatial constraint.
- *Physical colocation* (=) to describe the persistent proximity of two components.

Figure 2.12 depicts a handheld AR set-up with touch interaction, e.g., the *Touch Projector* application (described in 2.2.3), based on the *ASUR++* notation. The handheld device is described with three colocated *Adapters*: two input *Adapters* (i.e., the camera and the touch input surface) and one output *Adapter* (i.e., the handheld device's screen). The back-facing camera captures the *Real* object, while the touch surface captures the user's actions. This information is exchanged with the *System*, which sends information to the handheld device's screen, thus providing information to the user.

The *ASUR* notation is a tool for the design and analysis of Mixed Reality systems. Its components are consistent with the key entities highlighted in the definition of AR and in Mixed Reality classification schemes (namely *User*, *System* and *Real* objects). The *Adapters* explicitly describe the boundary between the physical and the digital worlds. Furthermore, the relations allow the spatial relations between the different components to be described. *ASUR* also allows the different components [Graham et al., 2000] and relations that supports the interaction flow to be further described [Dubois and Gray, 2008]. *ASUR* enables us to compare handheld AR systems with other Mixed Reality systems and to explore the design space by considering different modalities. However, the level of abstraction of the *ASUR* description does not make it possible to explore the design space of handheld AR interaction that shares

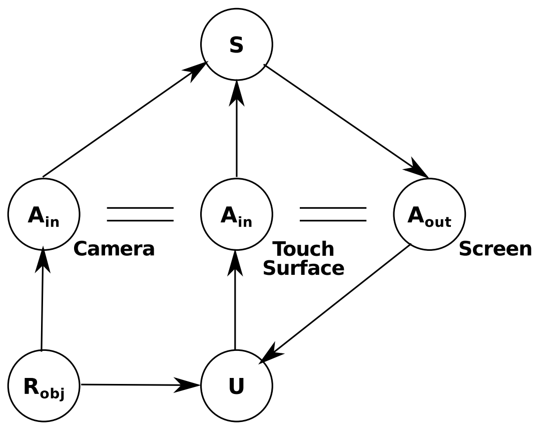


Figure 2.12: ASUR description of a handheld AR system with touch-screen interaction. Three colocated *Adapters* describe the handheld device.

the same modalities. Indeed, in this case the components will be the same for the different types of handheld AR interaction (Figure 2.12).

2.3.2 Classification schemes dedicated to AR

Having reviewed Mixed Reality design schemes, we will now review classification and design schemes that focus on AR only. These inherently offer greater insights into the key elements of AR systems.

In [Hugues et al., 2011], the taxonomies reviewed are classified as functional, technical or conceptual. Similarly, we will first review the functional taxonomies that apply to AR. We will then present a more technically-centered taxonomy, before describing two conceptual taxonomies that describes (1) AR presentation, and (2) AR annotations.

2.3.2.1 Functional taxonomy of AR

Hugues *et al.* [Hugues et al., 2011] proposed a functional classification of AR systems that distinguishes two main functionalities:

- *Augmented perception* of the physical environment: the AR system provides the user with additional information about the physical environment. The authors subdivide this functionality according to the degree of integration of the digital content and the physical environment. It ranges from *Documentation* to *Substitution*. *Documentation* is where digital content and the physical environment are displayed in separate display areas, but their contents is related. *Substitution* is where the digital content replaces the physical environment.

- User immersion in an *artificial environment*: the AR system provides extra information that does not relate to the physical environment. This functionality is broken down into three sub-functionalities: (1) imagine the physical environment as it could be in the future, (2) imagine the physical environment as it was in the past, and (3) imagine an impossible reality (i.e., an imaginary scene).

Table 2.2 presents all the classes of this taxonomy.

This taxonomy extends the description of the *augmented evaluation* class of the *type of augmentation* axis proposed by Dubois *et al.* [Dubois et al., 1999] (See 2.3.1.3) and *Renevier* [Renevier, 2004] (see 2.3.1.4). This taxonomy describes AR as a presentation space, therefore interaction is not studied.

Table 2.2: Functional classification of AR proposed by [Hugues et al., 2011]. VE denotes Virtual Entities and RI denotes Real Images.

Functional classification of AR environments	Augmented Perception	Documentation	Documented Reality
			Documented Virtuality
		Augmented Perception	Augmented Understanding
			Augmented Visibility
		Perceptual Association	Incrustation
			Integration
		Behavioural Association	
		Substitution	
	Artificial Environment	Artificial (future)	VE incrustation on RI
			RI incrustation on VE
		Artificial (past)	VE incrustation on RI
			RI incrustation on VE
		Artificial (impossible)	

2.3.2.2 Technical taxonomy of AR

Normand *et al.* [Normand et al., 2012] proposed a taxonomy of AR systems that mostly focuses on technical concerns.

The main axis is *Tracking*, which defines the requirements of the application in terms of degrees of freedom of the tracking system and tracking system accuracy. It spans from *0D*, where the application does not require camera tracking, but only object detection (such as markers), to *6D*, where an estimation of the camera position and orientation is required. In a second version of their taxonomy [Normand and Moreau, 2012], the authors proposed to further classify AR systems on the *Tracking type* axis (i.e., the type of tracking system, such as Marker-based tracking, or Non-optical systems, such as a compass).

The second axis, *Augmentation type*, describes the type of display and whether the view of the physical world is mediated (i.e., captured by a camera and displayed on a screen, as in handheld AR) or unmediated (i.e., directly perceived by the user with the augmentation projected also known as Spatial AR). The classes featured on this axis are *Optical see-through*, *Video see-through* and *Spatial AR*. In the second version of their taxonomy [Normand and Moreau, 2012], the authors also proposed a *Degree of freedom between frames of reference* axis that describes the relation between the frames of reference of the user, the display, the sensors and the physical environment. This axis encompasses four classes: *Tight* (e.g., Head-Mounted Displays), *Loose* (e.g., handheld AR), *Merged* (e.g., location-based services using GPS positioning) and *No relation* (e.g., fixed-projector set-up).

With regard to the semantic aspects of augmentation, this taxonomy includes a *Temporal Base* axis describing the moment in time the augmentation represents. It is comprised of four classes: past situations ($<t_0$), present information (t_0), foreseeing the future state ($>t_0$) and fully imaginary (∞). These are similar to the *Artificial* *environment* sub-classes proposed in [Hugues et al., 2011] (see 2.3.2.1). It differs from the *Temporality* axis proposed by Renevier [Renevier, 2004] (see 2.3.1.4), which describes the length of time for which an augmentation exists, rather than the moment in time the augmentation represents (i.e., the life expectancy of an augmentation versus the content of an augmentation described in terms of time).

In order to extend the scope of the taxonomy beyond the visual modality, the authors proposed a *Rendering Modality* axis that describes the different output modalities used by the AR system. The authors proposed to describe each output modality (i.e., audio, haptic, olfactory and gustatory) with a degree of freedom axis, as with the *Tracking* axis. In the case of audio, for example, monoaural sounds are classified as *0D*, while binaural sounds relate to a location (i.e., $2D{+}\theta$).

Finally, in the second version of their taxonomy [Normand and Moreau, 2012], the authors defined two additional axes to describe the visual appearance of the augmentation. First, they proposed an *Integration of virtual and real worlds* axis to describe the visual differences (e.g., realism of the colors and lighting, occlusion) between the augmentation and the camera images. This relates to the *Reproduction Fidelity* axis for digital objects proposed by Milgram [Milgram and Kishino, 1994] (see 2.3.1.2). Secondly, they suggested a *Level of abstraction of virtual objects* axis to describe the visual representation of the augmentation. This can range from a text description to a realistic representation using symbols and sketches.

This taxonomy provides axes that are not organized according to an underlying structure. It focuses primarily on technical concerns, i.e., characterization of the tracking system and the display. However, the *Degrees of freedom between frames of reference* axis relates to spatial relations, but it only provides a coarse description because it is geared towards classifying AR set-ups in general. For its part, the *Level of abstraction of virtual objects* axis is an interesting way of further describing digital content by describing its syntax.

2.3.2.3 Classification of the AR presentation space

The conceptual classification proposed by Tönnis *et al.* [Tönnis et al., 2013] considers AR as a presentation space. Thus, interaction with both the physical surroundings and the digital content is also out of the scope of this taxonomy. This classification encompasses the various display devices used to present AR content (Head-Mounted displays, Spatial AR, handheld AR).

The classification is structured around five axes:

- The *Temporality* axis describes whether an augmentation remains available at all times (*continuous*) or if its availability depends on certain conditions, such as the system state or user actions (*discrete*). This axis is similar to the *Temporality* axis proposed by Renevier [Renevier, 2004] (see 2.3.1.4).

- The *Dimensionality* axis describes the visual appearance of the digital augmentation, i.e., whether it is *2D* or *3D* content. This axis is related to the *Reproduction Fidelity* axis presented in [Milgram and Kishino, 1994] (see 2.3.1.2) and with the *Level of abstraction of virtual objects* axis proposed in [Normand and Moreau, 2012] (see 2.3.2.2). It primarily describes the syntax of the digital augmentation.

- The *Viewpoint Reference Frame* axis describes the frame of reference in which information is presented with respect to the user's nominal viewpoint. It can be:

  - *Egocentric*, when information is presented from the user's viewpoint.
  - *Ego-motion*, when information is presented from a viewpoint that is partially related to that of the user.
  - *Exocentric*, when information is presented from a viewpoint that is independent of the user's viewpoint.

  Moreover, the *"ego-impression"* class is proposed for handheld AR systems where the viewpoint in the augmented scene is not that of the user but the user is involved in its control. Because the user is involved in controlling the viewpoint, the impression given is that of an egocentric viewpoint. This axis extends the *Display Centricity* axis proposed by Milgram and Colquhoun [Milgram and Colquhoun, 1999] (see 2.3.1.2).

- The *Mounting* axis describes what the digital augmentation is attached to in the physical environment. *Mounting* describes the location to which the augmentation is anchored in the physical environment. Four classes are proposed for *Mounting* depending on the physical location to which the digital augmentation is mounted (i.e., anchored):

  - *Human*: the augmentation is mounted to a part of the user's body (e.g., her/his hand).
  - *Environment*: the augmentation is mounted to a physical object.
  - *World*: the augmentation's location in the worlds is defined by an absolute coordinate system.
  - *Multiple mountings*: this describes a combinations of the different mounting classes above.

- The *Type of Reference* axis describes the extent to which a digital object relates to a physical object. The three values along this axis are *direct overlay*, *indirect overlay* (for occluded physical objects) and *pure reference* (for physical objects lying outside the view frustum). The authors also described Diminished Reality systems (e.g., [Herling and Broll, 2012]), which remove physical objects from the view of the physical surroundings in the *indirect overlay* class.

The authors also discussed the technical aspects of *Registration.* Tracking systems that provide the geometric transformation required to position the augmentation in the physical space enables 3D registration. However, registration also encompasses rendering. Indeed, the authors state that while 3D registration involves correctly positioning the augmentation in the physical surroundings, it also involves correctly rendering the object in terms of depth perception. This second aspect is not catered for by display devices such as the screen of a handheld device, since the user's eyes need to accommodate to a focal distance that does not correspond to the augmentation's position in 3D space.

There is also a link between *Registration* and the display technology used. This could lead to an extension of the axes relating to the visual rendering of digital augmentation (i.e., *Reproduction Fidelity* [Milgram and Kishino, 1994], see 2.3.1.2) and to its integration in the physical surroundings or its representation (i.e., *Augmented Perception* [Hugues et al., 2011], see 2.3.2.1 and *Integration of the virtual and real worlds* [Normand and Moreau, 2012], see 2.3.2.2), which could also consider the user's depth perception.

This also relates to the more general issue of how computer-generated content is merged with the perception of the physical surroundings. Lindeman and Noma [Lindeman and Noma, 2007] proposed to classify AR systems depending on where the *Mixing* of physical and computer-generated stimuli takes place. *Mixing* locations include the physical *environment* (e.g., projector-based set-ups), the *sensory subsystem* (e.g., retinal displays) and the *computer* (e.g., Video See-Through set-ups such as handheld AR). This classification axis is based on the pathway of stimuli from the physical environment to the human brain. It applies to both direct and mediated AR systems, and to all human senses, not just sight. For the sense of sight, this axis is related to the *Augmentation type* axis [Normand et al., 2012] (see 2.3.2.2), which describes the display system.

The axes are not organized according to a conceptual definition or structure. However, the authors validated their taxonomy by describing a very large number of existing AR systems within the framework of their classification. This taxonomy defines AR as a presentation space, therefore it does not explicitly study interaction.

2.3.2.4 Taxonomy of AR annotations

Wither *et al.* [Wither et al., 2009] defined a conceptual scheme made up of axes that focus on AR annotations only. They proposed the following definition of an AR annotation:

"An Augmented Reality annotation is virtual information that describes in some way, and is registered to, an existing object."

Thus, an AR annotation is comprised of two components (Figure 2.13): (1) a spatially dependent component that links digital content to the physical environment, and (2) a spatially independent component featuring digital content that does not strictly represent its physical counterpart.

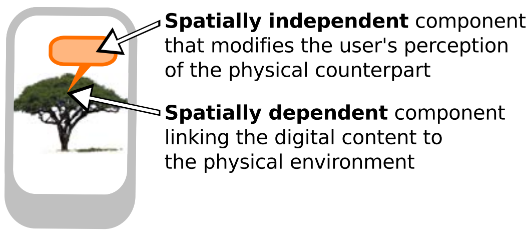

Spatially independent component
that modifies the user's perception
of the physical counterpart
Spatially dependent component
linking the digital content to
the physical environment

Figure 2.13: AR annotation comprised of a spatially dependent component and a spatially independent component.

The authors defined six axes for the characterization of AR annotations. The first two axes describe the spatially dependent component (i.e., the physical location to which the annotation is anchored). The other two axes describe the spatially independent component (i.e., the digital content). The final two axes describe the dynamics of the AR annotations. The siz axes are as follows.

Spatially dependent component:

- The *Location complexity* axis describes the annotation's spatial anchor in the physical world. The simplest annotation location is a 3D point in space. Further along the axis, an annotation can be defined by both a 3D point and a 3D orientation. More complex locations include 2D and 3D areas. This complexity does not relate to the actual content of the annotation.

- The *Location movement* axis describes the extent to which the digital content moves relative to the annotation's physical location. The representation of an AR annotation can be fixed in its location. It can also move from its physical location. This is true, for example, in the case of view management techniques [Bell et al., 2001] that arrange annotations in the screen space in such a way as to avoid clutter.

Spatially independent component:

- The *Semantic relevance* axis describes the semantic distance between the annotation's digital content and the physical object. The semantic relation between the digital content and the object can be further described with the following descriptors: *Names*, *Describes*, *Adds to*, *Modifies* and *Directs to.*

- The *Content complexity* axis encompasses both the quantity of information that the digital content delivers to the user and the complexity of the rendering of the digital content. For example, a point marking physical objects of interest is simple annotation content, while a textured 3D model with audio is complex annotation content. This axis is related to the *Dimensionality* axis in [Tönnis et al., 2013] (see 2.3.2.3), the *Reproduction Fidelity* axis in [Milgram and Kishino, 1994] (see 2.3.1.2) and the *Level of abstraction of virtual objects* axis in [Normand and Moreau, 2012] (see 2.3.2.2).

Dynamicity of annotations:

- The *Interactivity* axis describes the extent to which the user can interact with the annotation. This axis spans from non-interactive static annotations to annotation authoring, where the user can interact, edit and create annotations using the AR system. This axis is related to the *Interaction mode* axis proposed by Renevier [Renevier, 2004] (see 2.3.1.4), but it provides more detailed classes.

- The *Annotation permanence* axis describes how the visibility of the annotations is controlled. Annotations can be permanently visible or their permanence can: change over time, be controlled by the user, depend on spatial constraints or be filtered by the application. This axis extends the *Temporality* axis proposed in [Tönnis et al., 2013] (see 2.3.2.3) and in [Renevier, 2004] (see 2.3.1.4) by describing the way in which augmentation visibility is controlled.

Building on the proposed definition of AR annotation, the scope of this taxonomy is clearly identified, while its axes are structured into three complementary groups. The definition of AR annotation relies on two factors: physical location and digital objects. This taxonomy focuses on AR annotations and does not take into account technical or device-specific considerations. Interaction is described in terms of possible tasks, ratehr than in terms of interaction modalities, which is consistent with the taxonomy's device-agnostic perspective.

2.3.3 Summary

The majority of the design schemes reviewed propose axes that are not explicitly organized according to a common underlying structure or definition. The taxonomy of AR annotations proposed by Wither *et al.* [Wither et al., 2009] is a counterexample, since it is built upon the proposed definition of AR annotation. This lack of structure can make it hard to assess what the design spaces cover (i.e., what is described in detail and what is left aside). In addition, the relations between the different axes are not explicitly defined. While a design space should ideally provide orthogonal axes, it is still possible for meaningful axes to not be strictly orthogonal in practice. Therefore, a common underlying structure is useful to help understand the relations between different axes. It is also useful to know whether different axes describe the same entity in a system or whether an axis describes the relation between two different entities described by other axes.

The Mixed Reality design spaces reviewed consider three types of entity: physical objects, digital objects and the user (either explicitly or through axes describing the user's perception (*Extent of Presence Metaphor* [Milgram and Kishino, 1994]) or interaction (*Display congruence* [Milgram and Colquhoun, 1999], *Target of the task* and *Type of augmentation* [Dubois et al., 1999], *Interaction mode* [Renevier, 2004])). Furthermore, the relations between these entities are strongly highlighted. The comparison of HCI styles carried out by Rekimoto [Rekimoto and Nagao, 1995] is based on the relations involved. Renevier defines an augmentation as a physical-digital relation [Renevier, 2004]. The *ASUR++* notation [Dubois et al., 2001] proposes the *Adapter* component and allows different types of relations between its components to be described. These design schemes allow entire Mixed Reality systems to be described and compared. As such, they remain at a rather high-level of abstraction if we consider the Mixed Reality system that is a handheld AR set-up.

Unlike in Mixed Reality design schemes, the user and the physical objects are not explicitly described by the AR design spaces reviewed. Instead, they describe user interaction, the digital augmentation and the digital augmentation's relation to the physical surroundings. Like Mixed Reality design schemes, AR design spaces remain at a rather high level of abstraction, so as to cover the variety of AR set-ups. In order to remain device agnostic, these design spaces can only describe interaction at a fairly high level of abstraction. Indeed, interaction is explicitly discussed with respect to the type of task (*augmented perception* versus *artificial environment* in [Hugues et al., 2011]) and the degree of control the user has over augmentation (*Interactivity* [Wither et al., 2009]).

Several axes consider the different spatial relations between the frames of reference of the various components. Some axes describe the spatial relation between digital and physical objects: *Mounting*, *Type of reference* [Tönnis et al., 2013], *Location complexity* and *Location movement* [Wither et al., 2009]. Other axes describe the viewpoint in the augmented scene: *Display centricity* [Milgram and Colquhoun, 1999] and *Viewpoint reference frame* [Tönnis et al., 2013]. Finally, some axes describe the spatial mapping of inputs in the display space (*Control-Display Congruence* [Milgram and Colquhoun, 1999]) or sum up the spatial coupling involved (*Degrees of freedom between frames of reference* [Normand and Moreau, 2012]).

In the design space presented in the following section, we will attempt to add some structure to the different axes. We will restrict our study to handheld AR applications. We will proceed in two stages. First, we will propose a design space that describes on-screen content. Secondly, we will use this design space to study interaction.

2.4 Scope of the design space

In order to assess the specificities of handheld AR, we propose a design space that defines the on-screen content of handheld AR applications. This design space is based on two components and two spatial relations. These spatial relations allow user interaction to be studied.

This is a conceptual design space. Thus, technical issues such as tracking methods will not be discussed. Also, our aim is to study interaction independently of a specific field of application or user task. Therefore, the semantics of the design space's different components will not be discussed. Finally, unlike other design spaces for AR and Mixed Reality, this design space is device specific. Indeed, this is a dedicated handheld AR design space, although, as we shall see, some of our axes have a wider scope. Thus, the display device is a fixed parameter: this is the handheld device screen commonly used as a toolglass see-through video system.

One property of our design space is that it establishes a coherent structure by identifying two related components (see 2.5.1 and 2.5.2), which:

- Are characterized by the same two axes (output interaction);
- Are linked by spatial relations between their frames of reference and that of the physical surroundings (see 2.5.4); and
- Define the foundations for studying input interaction by considering viewpoint control and input frames of reference (see 2.6).

We will first present the two components and the two spatial relations that define the on-screen content. We will then study user inputs with respect to these components and spatial relations. When we present our design space we will highlight its descriptive power, firstly, by highlighting links with existing design axes and, secondly, by illustrating the different design elements through existing handheld AR applications.

2.5 Design space for on-screen content

Our design space is built around two components that define the AR on-screen content (Figure 2.14):

- *Representation of the Physical World*

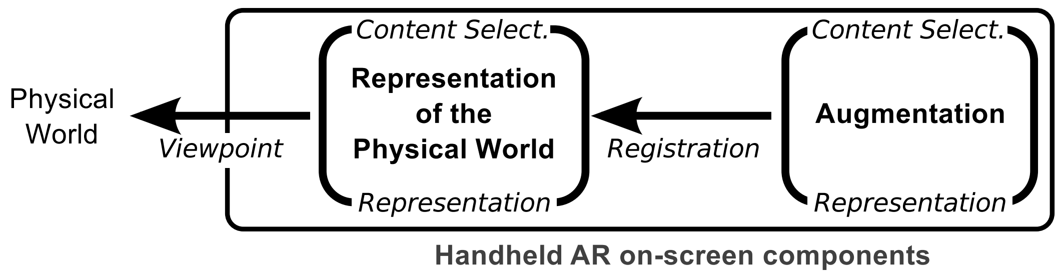


Figure 2.14: Design space for handheld AR. Two components (the Representation of the Physical World and the Augmentation) and two spatial relations (Viewpoint and Registration).

- Digital *Augmentation*

These two components relate to the definition of AR proposed by Azuma [Azuma, 1997] (i.e., 'combines real and virtual', see chapter 1). They also relate to the entities 'Real' and 'Computer' entities used to compare HCI styles in [Rekimoto and Nagao, 1995] (see 2.3.1.1). The physical world is represented, as handheld AR setups belong to the mediated AR systems category. A mediated AR system captures and resynthesizes the physical surroundings to present it to the user (see 2.3.2.2, *Augmentation type* axis).

We will describe each of these two components according to two design axes:

- The *Content Selection* (i.e., what) axis
- The *Mode of Representation* (i.e., how) axis

We will then discuss the spatial relations (i.e., where) between:

- The physical surroundings and their *Representation* (*Viewpoint*, Figure 2.14)
- The *Representation of the Physical World* and the *Augmentation* (*Registration*, Figure 2.14)

2.5.1 Representation of the physical world

The *Representation of the Physical World* defines the elements of the physical surroundings that are displayed on the handheld device's screen. This representation allows the user to locate the viewpoint in the physical surroundings. Indeed, for handheld AR, the viewpoint in the augmented scene is usually the viewpoint of the handheld device's camera and not that of the user. This form of representation also allows the user to locate augmentations overlaid in the physical surroundings. In handheld AR applications, this representation usually comes from the live camera images produced by the handheld device's rear-facing camera, like in *NaviCam* (described in 2.2.1) for example.

The semantic content of the *Representation of the Physical World* is defined by the physical surroundings in which the user is using the AR application. Because we have not focused on any particular field of application, our design space is not intended to describe the contents of the physical surroundings. The design space describes the

Representation of the Physical World along two axes. These axes define whether or not there is a *Content Selection* mechanism, as well as the *Mode of Representation.*

2.5.1.1 Content Selection

As previously explained, our aim here is not to consider the semantic content of the *Representation of the Physical World.* This axis defines whether or not certain parts of the physical world are removed (i.e., filtered) before being displayed on the handheld device. This is possible with so-called Diminished Reality techniques that remove existing physical objects from the camera images. To do so, these techniques either use the knowledge of the background or fill a particular area with a video inpainting system based on the surrounding patterns [Herling and Broll, 2012]. Such systems are classified in the same *indirect overlay* class on the *Type of Reference* axis [Tönnis et al., 2013] (see 2.3.2.3) as augmentation relating to a concealed physical object.

2.5.1.2 Mode of Representation

The *Mode of Representation* axis describes the visual aspects of the *Representation of the Physical World* displayed. This axis is related to the *Reproduction Fidelity* axis applied to physical objects in Milgram and Kishino's taxonomy [Milgram and Kishino, 1994] (see 2.3.1.2).

Live camera images are a common *Mode of Representation*, as in *NaviCam* (described in 2.2.1) for example, but other modes of representation can be used. For example, live camera images can be transformed into a non-photorealistic *Mode of Representation* so as to achieve the same visual representation quality as the digital augmentation [Fischer et al., 2005]. Moreover, a digital model of the physical world can be used to represent them. For example, in *Touch Projector* (described in 2.2.3), when the video is frozen the snapshot of the camera image displayed on the handheld device's screen is updated with a digital copy of the remote screen. The *Mode of Representation* can also be changed to support viewpoints that would otherwise be impossible with live camera images. For example, in *SnapAR* (described in 2.2.2) it is possible to view the augmented scene using previously recorded snapshots. In addition, to overcome a limited camera field of view, Alessandro *et al.* [Alessandro et al., 2010] described animated zoom-out techniques that offer the user an egocentric 360-degree panoramic view or an exocentric map-like top-down view displayed on the handheld device. This issue of changing viewpoints is further discussed in the *Viewpoint control* section (see 2.6.2).

2.5.2 Augmentation

Digital *Augmentation* is the representation of digital content that is not a *Representation of the Physical World.* Such content augments the *Physical World* with extra information and interaction capabilities. Examples of *Augmentation* include text labels overlaid on physical objects, as in *NaviCam* (described in 2.2.1), and 3D models overlaid on fiducial markers, as in *SnapAR* (described in 2.2.2).

As with the *Representation of the Physical World*, the design space does not describe the semantics of *Augmentation*, which depend on the field of application of the AR application or the purpose of the *Augmentation*. However, it is possible to extend this design space with axes from other design spaces so as to describe semantic aspects. Existing axes relating to the semantics of the *Augmentation* include the *Temporal Base* axis described in [Hugues et al., 2011] (see 2.3.2.1) and [Normand and Moreau, 2012] (see 2.3.2.2) and the *Semantic relevance* axis described in [Wither et al., 2009] (see 2.3.2.4).

Our design space describes *Augmentation* along two axes: the *Content Selection* mechanism, if one exists, and the *Mode of Representation*.

2.5.2.1 Content Selection

This axis describes how visible digital information is selected or filtered. In an AR system, an initial selection of visible content is made based on the viewpoint in the augmented scene. This axis does not describe this viewpoint-based visibility selection process. Instead, it describes other possible selection mechanisms performed either by the application or by the user.

Selecting the digital information to be displayed is of particular importance, as it can mitigate information overload and clutter, as well as allowing a better fit to the user's current task. Thus, previous studies have described different design solutions for content selection. For Mixed Reality, the *Temporality* axis in [Renevier, 2004] (see 2.3.1.4) describes whether the augmentation is permanent or transient. Similarly, for AR systems the *Temporality* axis in [Tönnis et al., 2013] (see 2.3.2.3) describes whether or not augmentation remains visible at all times. Julier *et al.* reported on physical methods that use distance and visibility to filter overlaid information [Julier et al., 2002]. For Mixed Reality systems, the *Interaction mode* in [Renevier, 2004] describes whether the augmentation is managed by the system or the user. For AR systems, the *Permanence* axis of Wither *et al.* [Wither et al., 2009] (see 2.3.2.4) describes the content selection according to the following classes: *Permanent*, *Temporally controlled*, *User-controlled*, *Spatially controlled*, and *Information-filtered*.

Closely related to the *Permanence* axis [Wither et al., 2009] (see 2.3.2.4), the *Content Selection* axis of our design space identifies classes according to the level of control the user has over selection:

- *No Selection*: *Augmentation* content always remains visible.
- *Application Selection*: the application changes *Augmentation* content and visibility according to its current state.
- *Spatial Selection*: the application changes *Augmentation* content and visibility according to changes in viewpoint (i.e., implicit user selection).
- *Explicit User Selection*: the user explicitly changes *Augmentation* content and visibility through dedicated interaction.

2.5.2.2 Mode of Representation

The *Mode of Representation* of *Augmentation* describes the visual appearance of the digital content displayed. As with the *Mode of Representation of the Physical*

World, this design issue has been studied in previous work. Milgram *et al.* [Milgram and Kishino, 1994] described the rendering quality of digital content through their *Reproduction Fidelity* axis (see 2.3.1.2). The *Level of abstraction of virtual objects* in [Normand and Moreau, 2012] (see 2.3.2.2), the *Dimensionality* axis in [Tönnis et al., 2013] (see 2.3.2.3) and the *Content Complexity* axis in [Wither et al., 2009] (see 2.3.2.4) are also related to the *Mode of representation* of *Augmentation*.

2.5.3 Distinction between Representation of the Physical World and Augmentation

The distinction between the *Representation of the Physical World* and the *Augmentation* is not always straightforward. Indeed, in some cases the boundary between the two components tends to be blurred.

Milgram *et al.* [Milgram and Kishino, 1994] introduced the following definitions for clarifying the distinction between real (i.e., physical) and virtual:

- Real objects are any objects that have an actual objective existence.
- Virtual objects are objects that exist in essence or effect, but not formally or actually.

Thus, real objects can be directly perceived or sampled and re-synthesized, while virtual objects must be simulated. To better address interaction design, we define the two on-screen components according to their objectives rather than their actual origin: the *Representation of the Physical World* allows the user to locate the current viewpoint and the digital content in the physical surroundings while the *Augmentation* provides extra information.

Applying this distinction is straightforward in cases where 3D models are overlaid on fiducial markers or text annotations are overlaid on physical objects. In the case of AR X-ray vision systems (e.g., [Sandor et al., 2010]) that display occluded objects that exist in the physical world, we consider the display of such occluded physical objects to be part of the *Augmentation*. Indeed, the display of such objects (e.g., the scenery behind a building [Sandor et al., 2010]) provides additional information to the user. The distinction between the *Representation of the Physical World* and the *Augmentation* becomes more complex in cases where the *Representation of the Physical World* is directly transformed:

- For instance, *ClayVision* [Takeushi and Perlin, 2012] is geared towards morphing existing buildings by altering their sizes or appearance. The resulting modified building belongs to both the *Representation of the Physical World* and the *Augmentation*. On the one hand, characteristics such as overall appearance and texture allow the user to map the modified building to its location in the physical world. Such characteristics are part of the *Representation of the Physical World*. On the other hand, characteristics such as modified size or highlighted color provide extra information and are thus considered as *Augmentation*. The distinction here is made on a per-characteristic rather than a per-object basis.
- Some Diminished Reality systems (e.g., [Herling and Broll, 2012]) are another example of direct transformation of live video of the physical surroundings. In

contrast with *ClayVision*, the purpose of camera image filtering is to remove the representation of a physical object from the scene. Thus, such systems do not augment the *Representation of the Physical World.* We categorize them according to the *Content Selection* axis of the *Representation of the Physical World.*

Such considerations are related to the *Integration of virtual and real worlds* axis proposed in [Normand and Moreau, 2012] (see 2.3.2.2).

2.5.4 Spatial relations

Having presented the two components of our design space (namely, the *Representation of the physical world* and the *Augmentation*), we will now present the two spatial relations in place between the *Physical World* and its *Representation* on the screen and between this *Representation of the physical world* and the *Augmentation.*

2.5.4.1 Viewpoint

The *Viewpoint* axis describes the spatial relation between the *Physical World* and its *Representation* on the screen (Figure 2.14). The viewpoint in the augmented scene is characterized by its position and orientation in space as well as by the camera's field of view.

In previous work, the viewpoint in the augmented scene has been described relative to the user's viewpoint (i.e., ego-centric versus exocentric), according to the following axes: *Display Centricity* [Milgram and Colquhoun, 1999] (see 2.3.1.2), *Degrees-of-Freedom between Frames of Reference* [Normand and Moreau, 2012] (see 2.3.2.2) and *Viewpoint Reference Frame* [Tönnis et al., 2013] (see 2.3.2.3). In our design space, we characterize this spatial relation according to the degree of coupling between the viewpoint in the augmented scene and the nominal viewpoint. For handheld AR, the nominal viewpoint is that of the device's camera, which is not the same as the user's viewpoint. The spatial relation between the camera's viewpoint and that of the user is described by the term "ego-impression" [Tönnis et al., 2013] (see 2.3.2.3), where the user has the impression of seeing an egocentric view because she/he controls the viewpoint of the device's camera.

In our design space, the degree of coupling between the viewpoint in the augmented scene and the nominal viewpoint of the device's camera is described by an axis that extends from *Conformal Mapping* to *No Mapping* (Figure 2.15):

- *Conformal Mapping* relates to the cases where the viewpoint is controlled by the handheld device's position and orientation in an absolute manner. For example, using a handheld device with an AR application that overlays digital augmentation on live camera images from the rear-facing camera, such as *NaviCam* (described in 2.2.1), relies on conformal spatial mapping and a fixed field of view (the one of the camera).
- *No Mapping* describes cases where there is no relation between the device pose and the viewpoint. This is the case when the video on the screen is frozen, as proposed in *Touch Projector* (described in 2.2.3) for example. Such video freezing techniques interrupt the spatial relation so as to improve user interaction. With *SnapAR* (see 2.2.2), when a previously recorded snapshot taken

from a different viewpoint is displayed, the *Viewpoint* spatial relation is also interrupted.

- Spatial mapping can also be *relaxed* when the viewpoint is only partially controlled by the device pose. *Touch Projector* (described in 2.2.3) offers an automatic-zooming capability. Zooming makes it possible to obtain a more detailed view and a dynamic ratio between the size of a real object and its on-screen representation. This is an example of *relaxed* mapping.

The *Relaxed Viewpoint* and changing between *Viewpoint* classes is further described

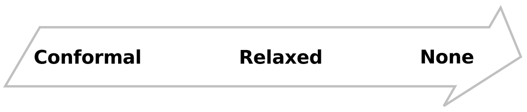


Figure 2.15: Viewpoint: the spatial relation between the Physical World and its Representation on the screen. Three classes.

in the *Viewpoint control* section (see 2.6.2).

2.5.4.2 Registration

The *Registration* axis describes the spatial coupling between the *Augmentation* and the *Representation of the Physical World*.

This spatial relation is very specific to AR and, as such, it has been described in previous work. Wither *et al.* [Wither et al., 2009] (see 2.3.2.4) analyzed this spatial relation in terms of *Location Complexity* and *Location Movement*. Tönnis *et al.* [Tönnis et al., 2013] (see 2.3.2.3) discussed it in terms of *Mounting* and *Type of Reference*. The *Location Complexity* and *Mounting* axes describe the types of physical element to which the augmentation relates.

Orthogonally to such design issues, our design space considers an axis to characterize the degree of spatial coupling between the augmentation and the corresponding physical elements. Our *Registration* axis, presented in Figure 2.16, incorporates the classes of the *Location Movement* (see 2.3.2.4) and *Type of Reference* (see 2.3.2.3) axes and adds new classes. The resulting classes identified are organized along an axis that ranges from:

- *Conformal Mapping*, where the *Augmentation* is exactly mapped onto the object to which it relates to in the *Representation of the Physical World*, to
- *No Mapping*, where the *Augmentation* has no spatial relation with the *Representation of the Physical World*.

In between, there are different forms of *Relaxed* spatial mapping:

- *Partial mapping* refers to cases where some degrees of freedom between the *Augmentation* and the corresponding object in the *Representation of the Physical World* are exactly matched, while others are relaxed. This is the case, for example, for annotations displayed with billboarding and/or a fixed on-screen size. *Partial* mapping can improve text readability.

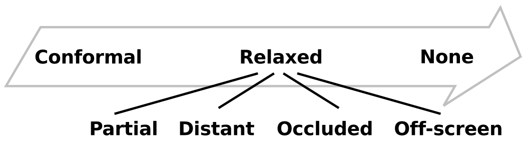


Figure 2.16: Registration: the spatial relation between the Representation of the Physical World and the Augmentation. Three classes defining different levels of registration.

- *Distant mapping* defines *Augmentations*, such as annotations, that are displayed a certain distance from the physical objects they refer to, but are visually linked to them by lines, for instance (e.g., view management [Bell et al., 2001]). Using *Distant* spatial mapping is useful for avoiding visual clutter.
- *Occluded mapping* describes the case of an *Augmentation* where the corresponding physical object is occluded from the current viewpoint by other physical objects, but lies in the view frustum.
- *Off-screen mapping* includes techniques for the visualization of off-screen points of interest, such as the use *3D Arrows* [Schinke et al., 2010] and *Wedges* [Gustafson et al., 2008], as examplified by the handheld AR application that we developed (Figure 2.17).

The main advantage of relaxing the *Registration* spatial mapping is that it provides extra degrees of freedom for the on-screen layout, but this can also reduce the feeling of colocation with the physical world.

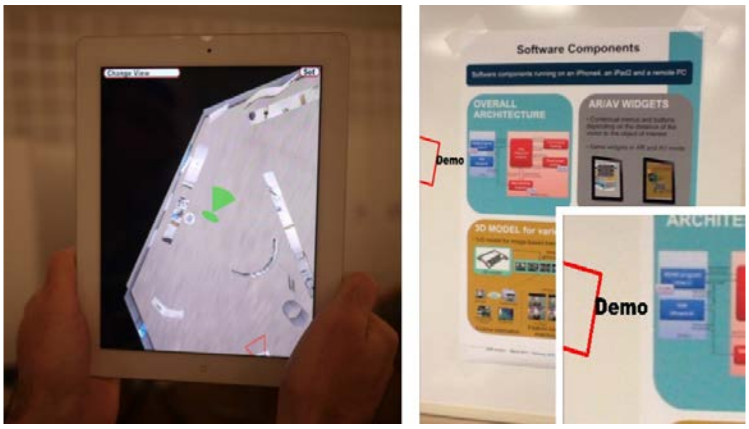


Figure 2.17: Wedge-based [Gustafson et al., 2008] off-screen point-of-interest visualization technique in map view (Left) and in AR mode in front of a poster (Right).

2.6 Studying interaction based on the design space

So far we have described the content displayed on screen in terms of two components (i.e., *Representation of the physical world* and *Augmentation*, see 2.5.1 and 2.5.2) and two spatial relations (i.e., *Viewpoint* and *Registration*, see 2.5.4) (Figure 2.14). We will now explain how this design space serves as a conceptual framework to study interaction.

Firstly, the *Viewpoint* and *Registration* axes describe spatial relations using static values. However, the system and/or user interaction can modify these spatial relations. We will describe the dynamics of these spatial relations in terms of *initiative* and *sustainability* (see 2.6.1).

Secondly, we will consider user input modalities. With a handheld AR set-up, the viewpoint in the augmented scene is commonly the viewpoint of the handheld device's camera. This viewpoint is controlled by the device's position and orientation in space. Also, the de facto standard user input on such devices is the touch-sensitive screen. Therefore, we will successively consider the input modalities involved in: (1) controlling the viewpoint (see 2.6.2); and (2) interacting with the content of the augmented scene (both physical and digital objects) via the touch screen (see 2.6.3).

2.6.1 Characterizing the dynamics of the spatial relations

The different values of the spatial relations presented previously (i.e., *Viewpoint* and *Registration*) describe different degrees of coupling between the on-screen content and the physical surroundings.

On the one hand, these values define a static snapshot at a given moment in time of the degree of coupling supported by a handheld AR application. On the other hand, studying the transitions according to the two spatial relation axes is essential to support improved interaction (e.g., for pointing accuracy), but also to enable "magic" like transitions to other modes of representation [Billinghurst et al., 2001] [Alessandro et al., 2010] or movement within the augmented space, while remaining stationary in the physical world, as in *SnapAR* (described in 2.2.2). Indeed, interaction with AR settings is constrained by the spatial relation with the physical world. However, it is not the physical world that the user is interacting with, so such constraints can be relaxed, at least temporarily. In our design space, we express these transitions along the two axes describing spatial relations. We characterize such transitions in terms of:

- Initiative: ranging from *explicit* user interaction, to *implicit* interaction (on the system's initiative and upon indirect interpretation of user actions), to *automatic* (on the system's initiative)
- Sustainability: ranging from *transient* to *sustained-mode*

Standard interaction modes (e.g., the drawing mode of a graphical editor) are explicit and sustained, while quasi-modes (e.g., holding the Shift key for upper case typing) are explicit and transient. Proxemic interaction [Ballendat et al., 2010] [Ju et al., 2008] adapts on-screen content and interaction capabilities based on spatial relations between users and devices. It is characterized as implicit and transient.

When applied to transitions between spatial mappings, we observe that:

- Modifications to the *Viewpoint* are mostly explicit and sustained: Indeed, the video-freezing technique, as in *Touch Projector* (described in 2.2.3) is implemented as an explicit transition (from conformal to none) triggered by a button between two sustained modes [Guven et al., 2006] [Lee et al., 2009]. In contrast to this explicit transition, the automatic-zoom feature in *Touch Projector* (described in 2.2.3) is an example of implicit and transient transition for the purpose of enhancing interaction. The automatic zoom maintains a fixed Control-to-Display ratio between the touch-screen and the remote screen controlled. The application zooms in when a remote screen is recognized and zooms out when there is no remote screen in the live video.

- Modifications to the *Registration* are mostly implicit and automatic: Indeed, view management [Bell et al., 2001] enhances the mapping between the augmentation and the representation of the physical world by taking into account the visual constraints of projecting objects onto the view plane. Such techniques avoid label clutter and can prevent augmented content from occluding interesting areas of the live video image. To do so, the augmentation is automatically laid out according to both the position of augmented objects in 3D and their on-screen footprint. The mapping of annotations to objects is dynamically and automatically adjusted between partial mapping (billboarding) and distant mapping (linked with a line).

In AR settings, the implicit and temporary relaxation of spatial relations is of particular use. Temporary transitions allow for a better fit between the visual content and the user's current focus and task. Moreover, implicit relaxation does not require the user to perform additional actions to benefit from such transitions.

However, this temporary relaxation of spatial relations to improve interaction in AR settings comes with certain drawbacks that need to be examined. Indeed, if a spatial relation is relaxed for a specific purpose (e.g., freezing the video to support stable interaction), it should be restored when it is no longer necessary. Breaking and restoring spatial relations can disorient users, as observed in [Lee et al., 2009]. An animated transition between the different viewpoints and/or modes of representation, as used in *SnapAR* (described in 2.2.2) or between camera images and top-view mode [Alessandro et al., 2010], and as suggested in [Lee et al., 2009], can minimize the impact of this spatial discontinuity problem.

2.6.2 Viewpoint control

We will now focus on the input modalities that enable the user to control the viewpoint in the augmented scene. For handheld AR set-ups, because the camera is fixed to the device, the modality controlling the viewpoint is commonly the position and orientation of the device in space.

However, it is actually possible to combine this absolute pointing in space with other modalities to control the viewpoint. The position of the camera is constrained by the device's position, but it is possible to use a wide-angle camera and to display only part of the camera image. In our design space, this is the case for *Relaxed Viewpoint*. *ExMAR* [Hwang et al., 2010] allows the user to control the viewpoint with a combination of device position and orientation, and touch strokes on the screen. Moreover, it is possible to use both device position and the user's head position to control the viewpoint. This allows for a viewpoint in the augmented scene that relates to the position of the user's head [Hill et al., 2011] [Baricevic et al., 2012].

In addition to combining modalities to control the viewpoint, it is also possible to display different viewpoints that are either spatially or temporally multiplexed. First, multiple viewpoints can be displayed simultaneously. For example, in [Bell et al., 2002] both an egocentric viewpoint and a top view of a World-in-Miniature are displayed. Second, the user can interactively change the viewpoint over time by switching between the nominal viewpoint and alternative viewpoints. Alessandro *et al.* [Alessandro et al., 2010] discussed the possibility of switching from the camera's viewpoint to an exocentric top-view map using a continuous animation. Furthermore, by using a continuous transition *SnapAR* (described in 2.2.2) allows the user to view the augmented scene from previously recorded viewpoints. With the *MagicBook* [Billinghurst et al., 2001], users can transition between an AR mode and a Virtual Reality mode. In [Tatzgern et al., 2013], the authors designed and evaluated different techniques for object-centric exploration (i.e., orbiting around an object) without moving in the physical space. They proposed both to spatially separate the two viewpoints and to overlay them. They also represented the position of the user's viewpoint in the view from an alternate viewpoint and linked certain feature points between both views in order to help the user to map its current viewpoint in the alternate viewpoint. With all these techniques, the switch between viewpoints is accompanied by a continuous animated transition and a change in the *Mode of Representation of the Physical World.*

Although *Viewpoint* and *Mode of Representation of the Physical World* are two different axes in our design space, both must be considered for the design of viewpoint control. Indeed, the system needs to capture the physical surroundings in order to represent it on the screen. Live camera images are only available in the field of view of the camera. Thus, we can only rely on camera images when the viewpoint remains in the vicinity of the camera's viewpoint, i.e., for small viewpoint changes (e.g., [Baricevic et al., 2012] [Hill et al., 2011]). Larger viewpoint changes (e.g., top view [Alessandro et al., 2010], previous viewpoints, like in *SnapAR* and as described in 2.2.2, or unconstrained viewpoints [Tatzgern et al., 2013]) require the *Mode of Representation of the Physical World* to be changed.

2.6.3 Touch interaction with the augmented scene

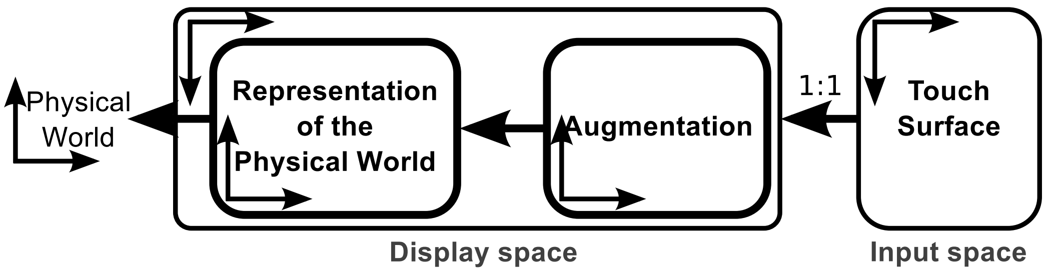


Figure 2.18: Relation between the input space and the display space studied in terms of frames of reference and spatial relations in our design space.

Milgram and Colquhoun [Milgram and Colquhoun, 1999] described the mapping between input and output space in their *Control-Display Congruence* axis (see 2.3.1.2). We will further study interaction by considering the relation between the input space and the display space. Here the touch-sensitive surface is considered as the input space. The touch-screen is the de facto standard user input on handheld devices. In our design space, the display space is defined by two components, namely the *Representation of the Physical World* and the *Augmentation* (Figure 2.14). Techniques for interacting with both physical and digital objects are then designed based on the frames of reference of the different elements of Figure 2.18: (1) the touch-sensitive surface, (2) the display space (i.e., the screen), (3) the *Representation of the Physical World*, and (4) the *Augmentation*. These frames of reference are spatially related and the two spatial relations of our design space, *Viewpoint* and *Registration* (Figure 2.14), are considered in particular.

The viewpoint is controlled by the handheld device's position and orientation in space. The handheld device's position and orientation in mid-air is not self-stabilized (unlike a desktop mouse, for example). It is therefore subject to hand tremor (Figure 2.19), like other freehand interaction techniques such as laser pointers and handheld projectors. As a result, the viewpoint is not stable and neither is the *Representation of the Physical World* in the screen's frame of reference. This may impair user interaction with both physical objects and the digital objects attached to physical objects (i.e., digital targets).

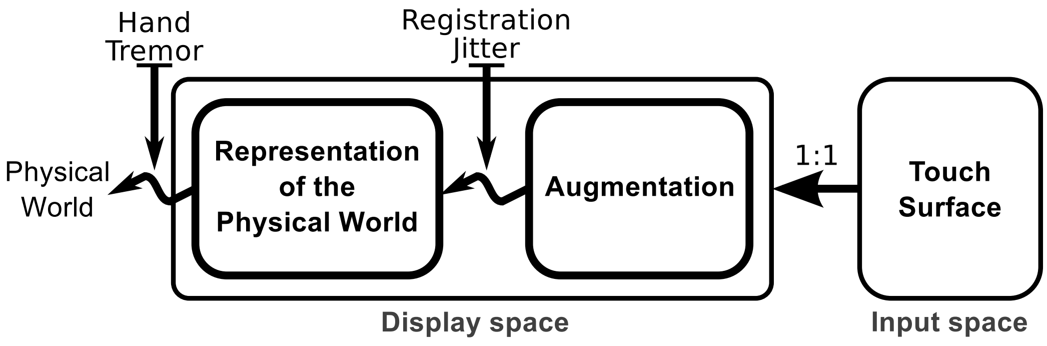


Figure 2.19: Effect of hand tremor and registration jitter on spatial relations in the case of live camera images representing the physical world.

Secondly, in the nominal case of camera images representing the physical surroundings, the *Registration* spatial relation relies on a tracking system that estimates the current position of the camera. However, different types of error [Holloway, 1997], such as static offset, latency and jitter, can impair the accuracy of this tracking system. For example, the accuracy of vision-based tracking can depend on lighting conditions or the availability of feature points to track. Poor tracking conditions can result in a jittery estimation of the current position of the camera. These so-called registration errors impair the *Registration* spatial relation. As a result, the *Augmentation* is not stable relative to the *Representation of the Physical World* (Figure 2.19). This may impair user interaction with the *Augmentation*.

Moreover, the touch-screen provides conformal mapping between the touch surface's frame and the screen. Therefore, due to both hand tremor and possible registration jitter, the on-screen content of a handheld AR application is not stable within the screen frame and therefore not stable within the input frame of reference. To overcome this problem, freeze-frame techniques have been proposed (like that used in *Touch Projector* described in 2.2.3 and [Guven et al., 2006] [Lee et al., 2009]).

When the user takes a snapshot of the live camera images, both the *Representation of the Physical World* and the *Augmentation* are stable within the screen frame and, therefore, the touch input frame. However this comes at the expense of breaking the *Viewpoint* relation between the device's position and orientation and the viewpoint in the augmented scene. As discussed by Lee *et al.* [Lee et al., 2009], such techniques might disorient the user, who may need time to return to the previous viewpoint.

There is an alternative to mitigating the effect of hand tremor. The overall concept involves mapping touch input events into the frame of reference of the physical object's representation, rather than into the screen frame. Lee and Billinghurst proposed the *Snap-to-Feature* [Lee and Billinghurst, 2011] technique, which allows touch inputs to be snapped to edges that are recognized in the live camera image. By stabilizing touch inputs on the representation of an object, this technique allows the user to draw the edges of physical objects more precisely. We adopted a similar approach for the *Relative Pointing* technique we designed, which aims to improve pointing accuracy when positioning digital annotations on a physical object, for example. This interaction technique is described in chapter 3.

2.7 Assessment of the design space

To assess the validity of our design space, we will examine it with respect to the three dimensions proposed by Beaudouin-Lafon for the evaluation of interaction models [Beaudouin-Lafon, 2004]:

- *Descriptive power*: the ability to describe a significant range of existing interfaces
- *Evaluative power*: the ability to help assess multiple design alternatives
- *Generative power*: the ability to help designers create new designs

First, the descriptive and evaluative power of the design space can be assessed based on the illustration of its compound design elements using several existing handheld AR techniques. In this section, we will describe a sample of handheld AR applications with respect to our design space to further illustrate its use (see 2.7.1). When we presented the design space, we discussed relevant existing design axes with regard to our design space (e.g., the *Reproduction Fidelity* axis [Milgram and Kishino, 1994] relating to the *Mode of representation* of both the *Representation of the physical world* and the *Augmentation*). Moreover, certain existing design axes are included in our design space while, as we have highlighted, other could extend the coverage of our design space, in particular for the description of semantic aspects (e.g., the *Semantic relevance* axis [Wither et al., 2009]). Certain existing design axes are outside the scope of our design space. In section 2.7.2 below, we will summarize the relationships of existing design axes with reference to our design space, before organizing them with respect to our design space. This will allow us to show how the design space unifies and extends previous design schemes, but also to highlight the axes that are not included in our design space and why.

Second, we used this design space to support the design of interaction techniques. This emphasizes the generative power of the design space. However, while the design space was useful for the development of new interaction techniques, it is also true that the development of interaction techniques helped build the design space. Indeed, the design and experimental evaluation of new techniques also informed our design space.

In the following paragraphs, we will first describe the handheld AR applications presented in 2.2 with reference to our design space. We will then present a comparison between the axes of our design space and those of other design spaces.

2.7.1 Describing AR applications with the design space

We will consider the three handheld AR applications described in section 2.2. A standard handheld AR application, for which the device is used as a physical magic lens to display augmented digital content, will be described as follows with respect to our design space. The *Viewpoint* is *Conformal*. As regards the *Representation of the physical world*, there is *No Content selection* and the *Mode of representation* is a live camera image. The *Registration* and the description of the *Augmentation* depend on the application.

For 2D text annotations, as for *NaviCam* (described in 2.2.1), the *Registration* is *Relaxed (Partial)*, as the text is not aligned with the physical object to which it relates. For the *Augmentation*, there is *No Content Selection* and the *Mode of representation* is *2D text annotation*.

For 3D model augmentations, as in the case of *SnapAR* (described in section 2.2.2), the *Registration* is *Conformal*, because the 3D model is aligned with the physical surroundings. For the *Augmentation*, there is also *No Content Selection* and the default *Mode of representation* is *3D model*.

In the specific case of *SnapAR*, the value of the *Viewpoint* can change over time, since the application enables the user to change the point of view to one recorded previously. In this case, the *Viewpoint* changes from *Conformal* to *None*, while the *Mode of representation* of the *Representation of the physical world* changes from the camera images to an image recorded previously. This change of viewpoint is *explicit* and *sustained*.

The *Touch Projector* application (described in 2.2.3) is not strictly a handheld AR application, as no digital content is displayed on the handheld device's screen. However, it can be described with our design space, but in this case there will be no *Augmentation*. When entering freeze-frame mode, the *Viewpoint* changes from *Conformal* to *None*. Meanwhile, the *Mode of representation* of the *Representation of the physical world* changes from the camera image to a combination of camera images and digital representation of the remote screen. The manual zoom provides an explicit and sustained change of *Viewpoint* from *Conformal* to *Relaxed*, while the automatic zoom provides an implicit and transient modification.

2.7.2 Comparison with other design schemes

As part of the presentation of our design space, we linked its axes to the most relevant axes from the other design schemes presented in section 2.3. We will now sum up the axes of other design schemes according to their relation to our design space. This will provide a comprehensive list of existing axes that are either covered by our design space or outside the scope of our design space.

As stated in the scope section (see 2.4), this design space does not encompass axes relating to the user's task or to the semantics of the augmentation. Thus, axes from other design schemes relating to either the user's task or the semantics of the augmentation are not included in our design space. Axes describing technical aspects (i.e., tracking systems, display technology) are also outside the scope of our design space. Finally, the axes of our design space describe on-screen content for handheld AR and therefore do not directly describe interaction. Thus, axes of this nature from other design spaces are not included in our design space. However, interaction can be studied with reference to the design space (see 2.6).

Table 2.3: Axes and classes from other design schemes that are outside of the scope of our design space.

Concerns	Axes
Encompass AR	Virtuality continuum [Milgram and Kishino, 1994]
User's task	Target of the task [Dubois et al., 1999] Type of augmentation [Dubois et al., 1999] [Renevier, 2004] Augmented perception vs. Artificial environment [Hugues et al., 2011]
Semantic aspects	Articial environment (past, future, imaginary) [Hugues et al., 2011] Temporal base [Normand et al., 2012] Mounting [Tönnis et al., 2013] Semantic relevance [Wither et al., 2009]
System, tracking and display	Extent of World Knowledge [Milgram and Kishino, 1994] Target of the augmentation [Mackay, 1998] [Renevier, 2004] Tracking DoF and accuracy [Normand et al., 2012] Augmentation type [Normand et al., 2012] Tracking type [Normand and Moreau, 2012] Registration [Tönnis et al., 2013]
Interaction	Extent of Presence Metaphor [Milgram and Kishino, 1994] Control-Display Congruence [Milgram and Colquhoun, 1999] Rendering Modality [Normand et al., 2012] Interactivity [Wither et al., 2009]

The following axes from other design spaces are absent from our design space (Table 2.3):

- Axes that include AR as one of its classes: *Virtuality continuum* [Milgram and Kishino, 1994] (see 2.3.1.2).
- Axes describing the user's task: *Target of the task* [Dubois et al., 1999] (see 2.3.1.3), *Type of augmentation* [Dubois et al., 1999] [Renevier, 2004] (see 2.3.1.3 and 2.3.1.4), *Augmented perception* versus *Artificial environment* [Hugues et al., 2011] (see 2.3.2.1)
- Axes describing semantic aspects: *Temporal base* (*past*, *present*, *future*, *imaginary*) [Hugues et al., 2011] [Normand et al., 2012] (see 2.3.2.1 and 2.3.2.2), *Mounting* [Tönnis et al., 2013] (see 2.3.2.2), *Semantic relevance* [Wither et al., 2009] (see 2.3.2.4)
- Axes describing system, technical and display issues: *Extent of World Knowledge* [Milgram and Kishino, 1994] (see 2.3.1.2), *Target of the augmentation* [Mackay, 1998] [Renevier, 2004] (see 2.3.1.4), *Tracking Degrees of freedom and*

accuracy, *Augmentation type* [Normand et al., 2012] (see 2.3.2.2), *Tracking type* [Normand and Moreau, 2012] (see 2.3.2.2), *Registration* [Tönnis et al., 2013] (see 2.3.2.3)

- Axes that relate to interaction: *Extent of Presence Metaphor* [Milgram and Kishino, 1994] (see 2.3.1.2), *Control-Display Congruence* [Milgram and Colquhoun, 1999] (see 2.3.1.2), *Rendering Modality* [Normand et al., 2012] (see 2.3.2.2), *Interactivity* [Wither et al., 2009] (see 2.3.2.4)

Axes describing semantic aspects could be used to extend our design space so as to describe *Registration* (*Mounting* [Tönnis et al., 2013], see 2.3.2.3) and *Augmentation* (*Temporal base* [Hugues et al., 2011] [Normand et al., 2012], see 2.3.2.1 and 2.3.2.2, and *Semantic relevance* [Wither et al., 2009], see 2.3.2.4) in more detail.

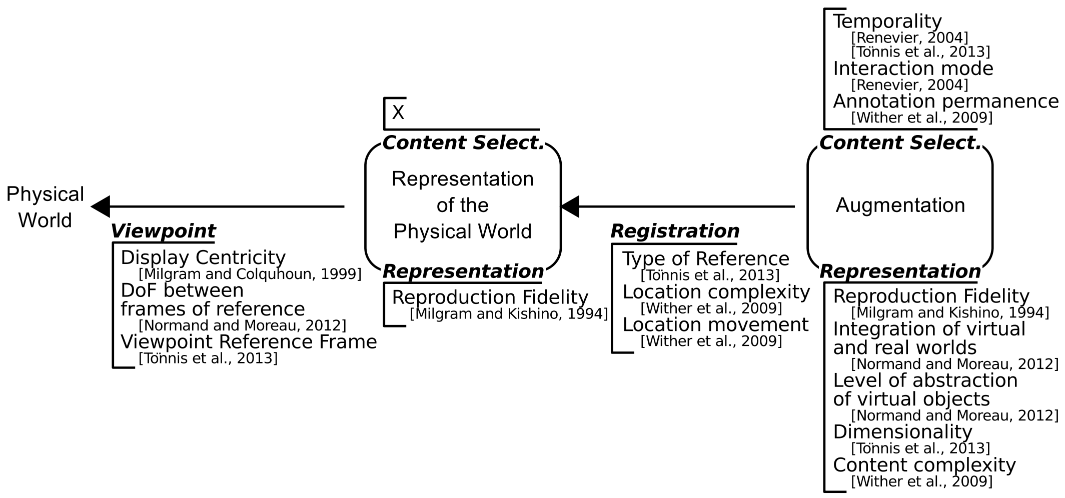


Figure 2.20: Axes from other design schemes that relate to the axes of our design space.

We will now list the remaining axes of the design spaces reviewed according to the axis to which they relate in our design space (Figure 2.20):

- Axes relating to *Viewpoint*: *Display Centricity* [Milgram and Colquhoun, 1999] (see 2.3.1.2), *Degrees of freedom between frames of reference* [Normand and Moreau, 2012] (see 2.3.2.2), *Viewpoint Reference Frame* [Tönnis et al., 2013] (see 2.3.2.3)
- Axes relating to *Representation of the physical world*:
  - No axis relates to *Content selection*
  - Axes relating to *Mode of representation*: *Reproduction Fidelity* [Milgram and Kishino, 1994] (see 2.3.1.2)
- Axes relating to *Registration*: *Type of Reference* [Tönnis et al., 2013] (see 2.3.2.3), *Location complexity*, *Location movement* [Wither et al., 2009] (see 2.3.2.4)

- Axes relating to *Augmentation*:
  - Axes relating to *Content selection*: *Temporality* (*permanent* versus *transient*) [Renevier, 2004] [Tönnis et al., 2013] (see 2.3.1.4 and 2.3.2.3), *Interaction mode* [Renevier, 2004] (see 2.3.1.4), *Annotation permanence* [Wither et al., 2009] (see 2.3.2.4)
  - Axes relating to *Mode of representation*: *Reproduction Fidelity* [Milgram and Kishino, 1994] (see 2.3.1.2), *Integration of virtual and real world*, *Level of abstraction of virtual objects* [Normand and Moreau, 2012] (see 2.3.2.2), *Dimensionality* (*2D*, *3D*) [Tönnis et al., 2013] (see 2.3.2.3), *Content complexity* [Wither et al., 2009] (see 2.3.2.4)

While certain existing axes are outside the initial scope of our design space, the two components and the two spatial relations of our design space allow many axes from other design spaces to be organized.

2.8 Conclusion

In this chapter, we presented a design space for handheld AR on-screen content. This design space is organized around two components, namely the *Representation of the physical world* and the *Augmentation*. The design space also describes two spatial relations. The *Viewpoint* axis describes the spatial relation between the physical surroundings and the *Representation of the physical world*. The *Registration* axis describes the spatial relation between the *Representation of the physical world* and the *Augmentation*.

Based on this design space and its spatial relations, we then studied interaction with handheld AR on-screen content. First, we studied the dynamics of the values along the design axes in terms of initiative and sustainability. We then focused on viewpoint control and touch-based interaction with the augmented scene.

To conclude, we proposed a structured design space enabling the study of interaction techniques for handheld AR applications. We showed its descriptive and comparative powers:

- By describing existing handheld AR applications (in particular the three applications presented in section 2.2)
- By positioning existing design axes (see 2.3) within our design space

We have used this design space to support the design of interaction techniques that we will present in the next chapter (see chapter 3).

To further strenghen the design space, we would need to describe an extensive set of interaction techniques from the literature according to the identified design axes. This would also allow to highlight common trends and reveal unexplored possibilities in handheld AR.

While this design space is primarily dedicated to the specific case of handheld AR set-ups, its components and axes can be relevant for other AR contexts. Indeed,

Display device	Physical World	Representation Physical World	Augmentation
HMD			
- Video		✓	✓
- Video Miniat.	✓	✓	✓
- Optical	✓		✓
Projection-based	✓		✓
Handheld device	✓	✓	✓

Table 2.4: AR display comparison

different display devices used for AR can be compared with reference to the components of our design space (*Representation of the Physical World* and *Augmentation*) and the visibility of the physical surroundings (see Table 2.4):

- With video see-through Head-Mounted Displays (HMDs), a representation of the physical world exists, i.e., the live video produced by the cameras. However, as users cannot directly observe the physical world, the scope for modifying its representation is limited, as it impacts the user's actions in the physical world. For example, freezing the frame might prevent the user from operating safely in the physical world. This limitation does not apply to miniaturized HMDs that also allow direct observation of the physical world.

- With optical see-through HMDs, there is no representation of the physical world, as the physical surroundings are observed directly. Also, users cannot observe the physical world without the digital augmentation.

- With projection-based systems, there is also no representation of the physical world and the physical world cannot be observed in augmented and unaugmented form simultaneously.

- Handheld devices allow both direct observation of the physical world in unaugmented form and observation of the augmented scene on the screen. They therefore allow for a wider range of design possibilities when it comes to modifying the representation of the physical world.

Part II

Pointing Techniques

Chapter 3

Pointing Techniques

Handheld Augmented Reality Pointing Techniques

Chapter Content

3.1 Introduction

Pointing is a basic generic task used in a range of tasks (e.g., placing marks, drawing vector graphic rectangles, selecting an area, drag & drop) and in the manipulation of widgets (e.g., icons, buttons, menus, lists). In the context of AR, two types of targets can be considered: physical locations (i.e., physical objects and 3D positions in space) and digital augmentations. For instance, pointing at physical targets can be part of a task involving the selection of an object in the physical surroundings to access associated information. Another example is the placement of digital marks on a physical object, such as a wall map, so as to add annotations linked to the physical world (Figure 3.1-Left). Pointing at digital targets can be part of a task involving the selection of a digital augmentation to access additional information. This is the case for digital widgets, such as a menu, registered to a physical object (e.g., the poster in Figure 3.1-Right).

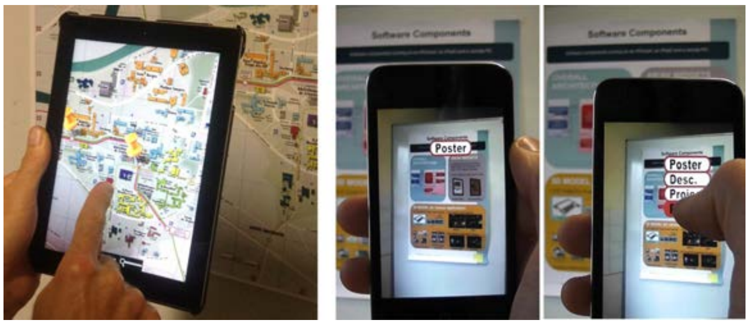


Figure 3.1: Handheld AR pointing tasks: (Left) pointing at a physical location on a remote wall map in order to position an annotation; and (Right) pointing at a digital menu attached to a poster.

As discussed in the previous chapter (see Chapter 2), handheld AR interaction, and therefore pointing, is affected by certain limiting factors. First, interaction via the handheld device's touch-sensitive screen is impaired by touch input issues (i.e., small screen, finger occlusion and ambiguous active point - see 3.2.2.1). With handheld tablets, there is also a trade-off between the device holding method (i.e., one or two hands) and touch interaction capabilities (i.e., available fingers and screen accessibility). Pointing assistance techniques have been proposed to improve pointing on handheld devices (e.g., [Vogel and Baudisch, 2007] [Roudaut et al., 2008] - see 3.2.2.2). However, these techniques do not take into account the specificities of AR.

Indeed, as highlighted in our design space, described in Chapter 2, handheld AR applications rely on the spatial relationships between the physical surroundings, their representation on the screen, the digital augmentation registered and the device's position and orientation in space (see Chapter 2). The *Viewpoint* is impaired by natural hand tremor while *Registration* is impaired by registration jitter. This hinders interaction, because the content with which the user is interacting is not stable on the screen, which is the touch surface's input space.

In this chapter, we will focus on pointing techniques for handheld AR that are geared towards improving precision for small targets. Our aim is to tackle the limitations relating to both touch-based input and the context of handheld AR. We will consider pointing at both physical and digital targets. We will restrict our study to pointing at the surface of objects, rather than the more general case of pointing at any position in space, such as in mid-air or inside/behind objects.

First, we will present an overview of pointing techniques, before discussing further touch-based pointing techniques and handheld AR pointing techniques. We will then present four different techniques and prototypes that we designed to explore handheld AR pointing (Figure 3.2): *AR TapTap*, *Shift&Freeze*, direct interaction with the physical object in front of the camera and *Relative Pointing*. *Shift&Freeze* and *Relative Pointing* have been published at the INTERACT 2013 conference [Vincent et al., 2013c]. Lastly, we will present different use cases for the proposed techniques, which go beyond pointing. It is important to note that the following chapter (see Chapter 4) is devoted to the experiments we held to evaluate two of the proposed techniques, namely *Shift&Freeze* and *Relative Pointing*.

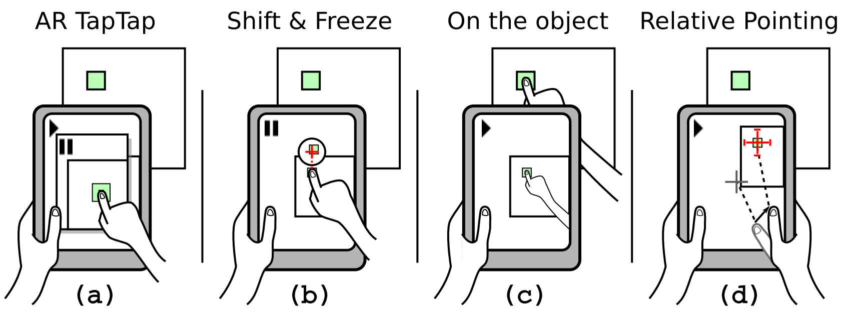


Figure 3.2: Proposed interaction techniques for handheld AR. From left to right: (a) *AR TapTap*; (b) *Shift&Freeze*; (c) Interaction with the object in front of the camera; (d) *Relative Pointing*.

3.2 State of the art

This section presents a review of exisiting pointing techniques, with a particular focus on handheld AR applications. In a handheld AR application, the minimum level of interaction is viewpoint control, which is usually dependent on the position and orientation of the handheld device in mid-air. To interact with the augmented scene, most interaction techniques are based on mid-air viewpoint control, touch-screen input and user's fingers recognition in front of the camera.

This section is organized as follows (Figure 3.3). We will first present an overview of pointing techniques and of the different strategies used by pointing assistance techniques. These techniques are geared towards improving the speed and/or precision of the pointing tasks. We will then characterize touch screen input and review pointing assistance techniques specific to handheld device touch screen. Lastly, we will consider handheld AR pointing by presenting the main input modalities and pointing assistance techniques.

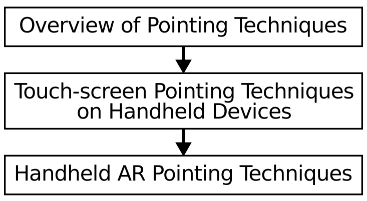

Overview of Pointing Techniques
Touch-screen Pointing Techniques on Handheld Devices
Handheld AR Pointing Techniques

Figure 3.3: Organization of the state of the art of pointing techniques.

3.2.1 Overview of Pointing Techniques

Pointing has been extensively studied in various contexts (e.g., desktop, touch-based handheld devices, virtual environments) and from different perspectives (e.g., human factors, input devices, pointing assistance techniques). A variety of pointing techniques have been proposed in the aim of improving pointing.

Pointing techniques are interaction methods that allow a point or an object to be specified in an interactive space. This interactive space can either be the 2D space comprising the screen (e.g., desktop interaction) or a 3D space (e.g., Virtual Reality set-ups). In handheld AR, pointing is performed in a 3D space, i.e., the physical surroundings augmented with digital content.

In the following section, we will provide a brief recap of the general models used to describe pointing tasks. We will then focus on how input devices control the active point used for pointing, before presenting an overview of existing pointing assistance techniques.

3.2.1.1 Modeling pointing

In HCI, pointing performance is commonly modeled using Fitts' law, which relates Movement Time (MT) to movement Amplitude (A) and target Width (W) according to an Index of Difficulty (ID) [Fitts, 1954] [Soukoreff and MacKenzie, 2004]:

$MT = a + b * ID$ where $ID = log2(1 + A/W)$.

According to Fitts' law, to reduce MT a pointing technique must reduce the ID of the pointing task. This means increasing the target Width W, reducing the movement Amplitude A, or modifying both. In addition, increasing W can improve accuracy when pointing at small targets. Indeed, pointing precision is limited by human and technological factors (the limits of the human motor system when controlling input devices - especially with freehand devices, such as when positioning a handheld device in space - as well as input devices sampling resolution, display resolution and the limits of the human visual system).

The *optimized initial impulse model* [Meyer et al., 1988] models pointing velocity

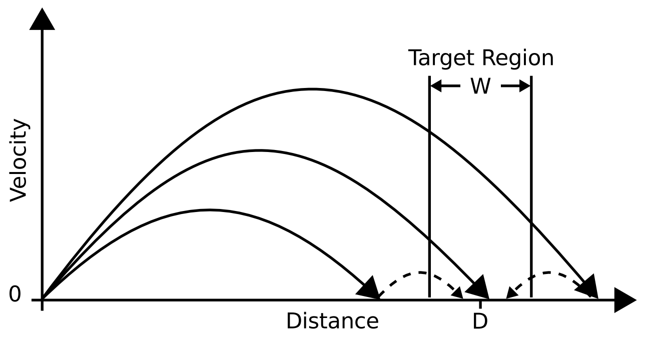


Figure 3.4: *Optimized initial impulse model* [Meyer et al., 1988]. Representation of three aiming movements: the solid curves represent the ballistic phase, while the dashed curves represent the optional corrective phase.

profiles. This model allows for the variability of the end points of pointing sub-movements. With this model, pointing includes the following sub-movements: a large and fast ballistic sub-movement (solid curves in Figure 3.4) and, if the target is missed, a slower corrective sub-movement (dashed curves in Figure 3.4).

3.2.1.2 From input device to active point

The active point is a position in the display space that is controlled by the user and used to interact with the system (e.g., for pointing). It can be represented on the screen by a cursor. With direct touch methods on touch screens, no cursor represents the active point. Interaction with computers, especially via a mouse or touch surface often relies on one or more active points (or an area, ray or volume).

Gilliot [Gilliot, 2014] organized input devices according to criteria characterizing the devices themselves and the control they provide over the active point. Five criteria were identified:

- **Device resistance**: *isometric* (i.e., measuring forces based on infinite resistance, such as with a TrackPoint), *elastic* (i.e., measuring forces based on increasing resistance, such as with a joystick) or *isotonic* (i.e., measuring positions or displacements based on constant resistance, such as with a touchpad or mouse).
- **Type of control**: *position control* (usually for isotonic devices) or *rate control* (usually for isometric and elastic devices).
- **Physical measurement** for isotonic devices: *absolute* (i.e., measurement of position, such as with a touchpad) or *relative* (i.e., measurement of displacement such as with a mouse).
- **Interaction directness**: *direct* (i.e., input and output spaces are colocated, such as with a touch-screen) or *indirect* (i.e., input and output spaces are not colocated, such as with a touchpad and a screen).
- **Active point control**: *absolute* or *relative.*

In the case of handheld AR, we will mainly focus on isotonic absolute input devices such as the handheld device's touch-screen or the absolute position of the handheld device in the physical space, which controls the viewing frustum of the device's back-facing camera. Table 3.1 sums up the different types of input device and display set-up according to the criteria for isotonic absolute input devices.

Because we are primarily interested in isotonic absolute devices, we will only examine the latter two criteria (i.e., *Interaction directness* and *Active point control*) in greater details.

Interaction directness considers the spatial arrangement of the input device and the display screen. With indirect interaction, the input space and the display space are not colocated. This is the case for both absolute inputs (e.g., touchpad) and relative inputs (e.g., mouse). With direct interaction, the input space and the display space are colocated. This is the case for absolute inputs (e.g., the handheld device's touch screen in the context of handheld AR).

Control over the active point is either absolute or relative. With absolute control, the position of the active point depends only on the input position, up to a constant

Control-to-Display gain. With relative control, the position of the active point is a function of the successive input displacements. The transfer function converts input displacements into active point displacements. As we will detail later, this transfer function can serve as leverage for pointing improvements.

Table 3.1: From isotonic input devices to active point control (Table partly reproduced from [Gilliot, 2014]).

Resistance, Type of Ctrl.	Measure	Input/Display	Active Pt. Ctrl.	Example
Isotonic, Position ctrl.	Relative	Not-colocated	Relative	Mouse
	Absolute	Colocated	Absolute	Direct touch
			Relative	*Relative Pointing* (See 3.3.2.2)
		Not-colocated	Absolute	Tablet with stylus
			Relative	Touchpad

3.2.1.3 Pointing assistance

Various approaches have been proposed to improve pointing. First, we will provide two reviews of pointing assistance techniques (the first one is devoted to 2D space and the second to 3D) to highlight the underlying organization of existing techniques. We will then focus on target awareness, a key characteristic of pointing assistance techniques, before further detailing the use of dynamic transfer functions. Indeed, dynamic transfer functions can be applied to handheld AR pointing (for an example, see 3.3.2.2).

In 2D space. In 2004, Balakrishnan reviewed pointing assistance techniques according to factors relating to Fitts' law (see 3.2.1.1), in terms of primarily reducing A, primarily increasing W or modifying both [Balakrishnan, 2004]. Other reviews of pointing techniques are also organized in this way [Grossman and Balakrishnan, 2005] [Tse et al., 2007] [Su et al., 2014]. In computerized systems, the two factors A and W can be applied to two spaces: the input space and the display space. Modifications to A and/or W can be performed in visual space, in motor space or in both visual and motor spaces.

Techniques that primarily reduce A resort to the following approaches: (1) designing widgets that minimize A (e.g., pie menus), (2) bringing possible targets closer to the cursor (e.g., *Drag-and-Pop* [Baudisch et al., 2003]), (3) using multiple cursors (e.g., *Rake Cursor* [Blanch and Ortega, 2009]), and (4) removing any empty space between targets (e.g., object pointing [Guiard et al., 2004]).

Techniques that primarily increase W include: area cursor techniques (e.g., *Bubble cursor* [Grossman and Balakrishnan, 2005]) and techniques that expand targets (e.g., *VTE* [Guillon et al., 2014]).

Techniques that both decrease A and increase W incorporate dynamic transfer functions, such as the speed-dependent transfer functions used on desktops, as well as semantic pointing [Blanch et al., 2004]. Semantic pointing [Blanch et al., 2004] dynamically alters the Control-to-Display gain, so as to provide a continuous gain that is higher in free space and lower on targets. Thus, it decreases A and increases W in motor space only.

Of these techniques, some are applied to both visual space and motor space (e.g., *Bubble cursor* [Grossman and Balakrishnan, 2005], *VTE* [Guillon et al., 2014]), some modify the input space only (e.g., dynamic transfer functions) and some modify visual space only (e.g., fish-eye view).

In 3D space. Argelaguet and Carlos reviewed 3D-object pointing techniques in Virtual environments [Argelaguet and Andujar, 2013]. They described the different techniques according to the type of cursor (usually a ray or a cone) and how it is controlled. They characterized the cursor control based on two characteristics: (1) input degrees of freedom (i.e., which degrees of freedom of the position and orientation of the user's hands and eyes are used) and (2) the transfer function. They then discussed the relation between motor and visual space, which is related to the directness of the interaction. They also characterized: (1) the disambiguation mechanism, which can be manual (i.e., the user chooses a target from those selected), heuristic (i.e., based on heuristics at the time of selection only) or behavioral (i.e., based on heuristics over a period of time prior to selection), (2) how the selection is triggered, and (3) what feedback (in addition to the cursor) is provided by the technique. Feedback involves a trade-off between the level of information provided and the degree of visual distraction.

Target awareness. A key characteristic of pointing assistance techniques is whether or not the technique requires prior knowledge of possible targets to improve pointing. This knowledge is required for *target-aware* techniques, but not for *target-agnostic* techniques.

Indeed, since *target-aware* techniques require knowledge of possible targets to improve pointing, they are only useful in situations where targets are already known by the pointing technique. Furthermore, some *target-aware* techniques (e.g., *VTE* [Guillon et al., 2014], *Bubble Cursor* [Grossman and Balakrishnan, 2005] and *Object Pointing* [Guiard et al., 2004]) prevent pointing in free space (i.e., outside targets). In this case, to enable pointing in free space, the application needs to feature a dedicated mode. Pointing in free space is useful, for example, for area selection or to organize icons on a desktop. Other *target-aware* techniques (e.g., *DynaSpot* [Chapuis et al., 2009] and *Semantic Pointing* [Blanch et al., 2004]) provide target acquisition assistance while also enabling pointing in free space. Finally, the performance of *target-aware* techniques depends on the density of targets [Blanch and Ortega, 2011]. Indeed, when the density of targets increases, the possible benefits of *target-aware* techniques are reduced.

Unlike *target-aware* techniques, *target-agnostic* techniques make it possible to point at every pixel of the screen (or every position in space). In handheld AR, when a digital mark is placed on a physical object, every position on the object is a potential target, because the user may place the mark at any point.

For our work, we focused mainly on *target-agnostic* techniques, as these support a wider set of situations. In the particular context of handheld AR, they allow new annotations to be positioned on physical objects.

Dynamic transfer functions. While it is not possible to dynamically modify the transfer function when the active point is controlled absolutely, with indirect relative pointing, the transfer function that converts input device displacement into cursor displacement can serve to enhance pointing.

Speed-dependent dynamic transfer functions, as used on desktop systems, provide target-agnostic pointing assistance. These transfer function provide a high Control-to-Display (CD) gain at high speed and a low CD gain at low speed. According to the *optimized initial impulse model* [Meyer et al., 1988] (see 3.2.1.1), pointing comprises two phases: a ballistic phase and potentially, a corrective phase. With a dynamic transfer function, a high CD gain during the ballistic phase reduces A, while a low CD gain during the corrective phase increases W. Thus, a dynamic transfer function can improve pointing performance by reducing the ID of the pointing task in motor space.

Casiez *et al.* [Casiez et al., 2008] studied the effect of constant CD gain and speed-dependent dynamic CD gain. During their experiments, they compared different levels of constant CD gain with Windows XP/Vista dynamic transfer functions. Their results indicated that low levels of CD gain were detrimental to pointing performance. This was due to increased clutching and maximum limb speeds. On a desktop, they also found that the dynamic transfer functions they evaluated were 3.3% faster overall than constant CD gains (and 5.6% faster for small targets). This is consistent with the *optimized initial impulse model* (see 3.2.1.1). However, there was more overshooting with dynamic transfer functions than with constant CD gain, thus potentially reducing the benefits of the former. In a further study, Casiez and Roussel [Casiez and Roussel, 2011] evaluated the dynamic transfer functions of Windows, Mac OS X and Xorg. All the dynamic transfer functions outperformed constant CD gain. These dynamic transfer functions displayed differences in performance depending on target size. The results indicated that the Mac OS X dynamic transfer function was 9% slower than the other two (Windows and Xorg) for the experiment's largest targets (i.e., 6 and 9 pixels wide), while it was faster than Xorg's dynamic transfer function for targets 1 pixel wide.

Thus, speed-dependent dynamic transfer functions can improve pointing performance, but the improvement depends on the particular dynamic transfer function used. Since dynamic transfer functions result in the enlargement of targets in motor space at low speed, they can help the acquisition of small targets. For instance, magnification in motor space is one option for assisting pointing at small targets, although the visual size of the targets is also important [Chapuis and Dragicevic, 2011].

It is also possible to use target awareness to assist pointing with a dynamic transfer function. Semantic pointing [Blanch et al., 2004] uses target awareness to reduce the ID of pointing tasks in motor space. To do so, it alters the CD gain to provide a continuous gain that is higher in free space and lower at the targets. In addition, cursor displacement is continuous and it is possible to point at free space, unlike with Object Pointing [Guiard et al., 2004], where the cursor 'jumps' from target to target.

Unlike other target assistance techniques, dynamic modification of the transfer function does not alter visual space. Indeed, target highlighting is not necessary as the target currently selected is under the cursor. Furthermore, dynamic transfer functions can be combined with other pointing assistance techniques. However, because they alter the transfer function dynamically, these approaches cannot be applied to absolute active point control.

3.2.1.4 Summary

Several different techniques have been proposed to improve pointing. An important factor to consider is whether or not these techniques are target aware. Indeed, target-aware techniques require prior knowledge of targets. Our work focuses mainly on target-agnostic techniques, which cater for a wider range of situations.

The most widely adopted pointing assistance technique is the target-agnostic speed-dependent dynamic transfer function (i.e., 'mouse acceleration') used by default on desktops. This provides potential performance benefits. Moreover, it can also be effective for small targets and can reduce the need for a large workspace. However, such improvements are only possible with indirect relative interaction. We used a dynamic transfer function to implement the *Relative Pointing* technique we propose (see 3.3.2.2).

3.2.2 Touch screen pointing techniques on handheld devices

Touch screen input is now the de facto standard input on smart phones and handheld tablets. It is essential to consider this user input for handheld AR interaction for a number of reasons: it is available on current off-the-shelf handheld devices, it has been widely studied, various interaction techniques have been proposed for conventional GUI applications and users are growing used to this input.

We will first characterize touch screen input, before presenting techniques designed to assist pointing on handheld device touch screens.

3.2.2.1 Touch screen input

The touch screen is an absolute direct input device that uses the same space for both display and input purposes. We will now characterize this type of input in terms of interaction capabilities, selection time and precision.

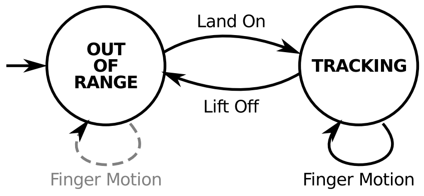


Figure 3.5: Touch input state machine.

Interaction capabilities.

Screen size. The first issue is that handheld devices, smart phones in particular, possess limited touch screen real estate. This limits the amount of information that can be displayed on the screen at any one time. Furthermore, because the screen is used for both touch input and display purposes, touch screens suffer from finger occlusion. This affects precision, as we will discuss later, but it also reduces the visible area of the screen when the user is interacting with the latter.

Expressiveness: one finger. Moreover, in comparison with the mouse state machine, the single-finger touch input state machine (Figure 3.5) lacks a second state where the system is informed of finger motion [Buxton, 1990]. Thus, touch input offers fewer interaction possibilities than mouse input. For example, with touch input it is not possible to provide feed-forward information through "hovering", without triggering system interpretation (e.g., object or command selection). Either the finger is tracked as it comes into contact with the touch surface and triggers a system action as soon as it leaves the surface, or the finger is out of range with no input provided to the system, thus having no effect on the system. There is no intermediate state where a position can be given to the system without triggering system interpretation.

One way of overcoming this limitation is to introduce extra modes or to use a dwell time after which a second tracking mode is activated. Another alternative is to combine touch input with another input modality, based, for instance, on the physical sensors of smart phones (e.g., accelerometers), as proposed in [Hinckley and Song, 2011] and [Scoditti et al., 2011]. Benko *et al.* [Benko et al., 2006] also suggested that a change of input state could be triggered when the finger's contact area changes (interpreted as a change in finger pressure). Finally, touch screens that sense a finger hovering over the screen are now available (e.g., the Sony Xperia sola smart phone[1]). This allows for two states (i.e., finger in contact with the screen and finger above the screen) where the system can sense finger motion as in the case of mouse or tablet with stylus inputs.

Expressiveness: multiple fingers. Touch input can rely on multiple fingers (from both hands), allowing input beyond the two dimensions sensed in the case of a single tracking state when just one finger is used. This allows higher-dimension inputs (as each finger can control one active point on the screen) and/or the triggering of different modes according to the number of fingers used. For example, two-finger-based *pinch-to-zoom* interaction is commonly available on smart-phones to zoom in/out of pictures. This can be combined with one-finger interaction such as *panning*, to allow for additional interaction states. Each finger cannot be controlled independently of the others, thus reducing the number of degrees of freedom actually controlled by the user when interacting with multiple fingers. Wagner *et al.* [Wagner et al., 2012] studied bi-manual multi-touch interaction on handheld tablets and proposed interaction techniques that allow interaction using the fingers of the hand holding the tablet. For instance, in a drawing application, a chord performed with the fingers of the hand holding the tablet can be used to select a drawing tool to be controlled by a finger on the other hand controls.

Form factor has an impact on touch screen interaction.

[1]http://developer.sonymobile.com/knowledge-base/technologies/floating-touch/

Form factor: smart phone. Focusing on one-handed interaction with smartphones, Karlson *et al.* [Karlson and Bederson, 2008] held an experiment in which they asked users to indicate the "easy-to-reach" and "hard-to-reach" regions of the screen. The hard-to-reach regions were located at the top of the screen and at the two bottom corners (Figure 3.6). Bergstrom-Lehtovirta and Oulasvirta proposed a model for the functional range of the thumb (i.e., the area reachable without altering grip) when using a handheld device with one hand [Bergstrom-Lehtovirta and Oulasvirta, 2014]. This model predicts the thumb's functional range according to (1) touch screen size, (2) the user's hand size and (3) the position of the index finger on the back of the device, which reflects the user's grip on the device. The position of the index finger on the back of the device is specified by the orientation of the index finger and the distance over which the index finger extends across the back of the device.

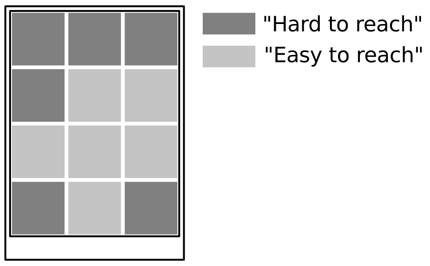


Figure 3.6: Subjective difficulty when it comes to reaching the different regions of the screen with the thumb of the hand holding the device (Figure from [Karlson and Bederson, 2008]). The dark areas are the most difficult to reach.

Form factor: tablet. With tablets, which are larger but also heavier than phones, the trade-off between the device holding method (i.e., one or two handed) and touch interaction (i.e., fingers available for touch interaction and screen accessibility) needs to be taken into account. In [Wagner et al., 2012], the authors studied the combination of device holding method and interaction as a kinematic chain. Interestingly, during their experiments no users held the tablet with both hands while interacting with their thumbs (Figure 3.7-c). During our experiments in a handheld AR context, we frequently observed this holding method (see Chapter 4). When holding the tablet with both hands, neither thumb can access the entire screen.

Selection time. Cockburn *et al.* compared direct touch input with both direct stylus and indirect desktop mouse input for the execution of three separate tasks: a tapping task (i.e., a pointing task), a dragging task (i.e., a pointing task with a finger or stylus touching the screen) and a radial dragging task similar to selecting an item in a pie menu [Cockburn et al., 2012]. Their results indicated that, for tapping, touch was faster than the other two devices. However, it was more error prone for small targets. Unlike with tapping, touch was slower than the other two input devices when it came to the dragging task. All three input devices displayed similar error rates. During the radial dragging task, touch was also found to be slower than the other two input devices. Comparing tapping and dragging revealed

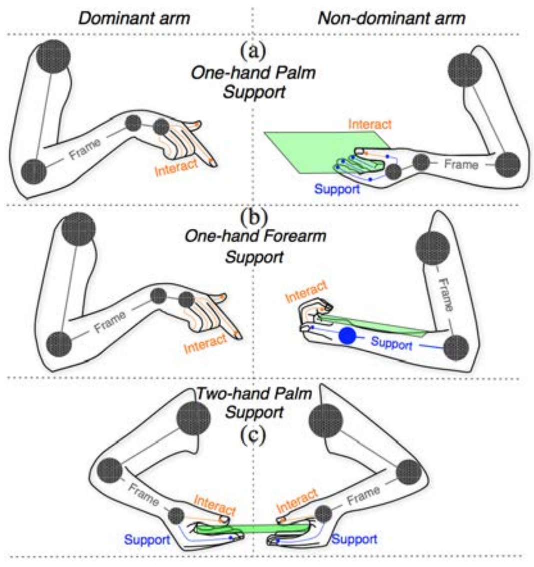


Figure 3.7: Different holding method with distinct trade-offs between interaction and comfort (Figure from [Wagner et al., 2012]).

discrepancies between the selection times achieved during these respective tasks when using touch (dragging being slower than tapping), but not with the stylus and mouse. The authors suggested that finger friction on the touch screen might explain this discrepancy.

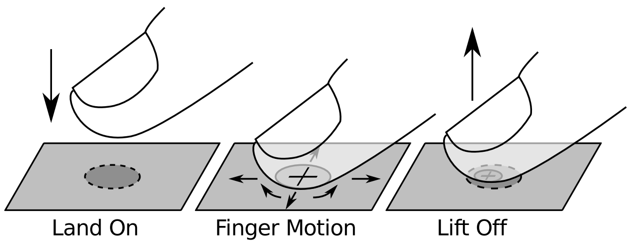


Figure 3.8: Touch input phases: land-on, finger motion and lift-off phases with contact area (ellipse) and active point (crosshair).

Precision. Touch screen input is reasonably fast for tapping (see previous section), but relatively inaccurate. The minimum target size reported for touch inputs is approximately $1cm$ or more [Holz and Baudisch, 2010]. Indeed, there is ambiguity in the active point position sensed by the system, due to the contact area between the finger and the touch screen. The initial explanations put forward related to finger occlusion and the width of the contact area, also known as the "fat finger problem" [Vogel and Baudisch, 2007], which leads to an ambiguous active point. However, touch input precision is impaired differently during the three different phases: land-on, finger motion and lift-off (Figure 3.8).

Holz and Baudisch proposed two models to describe the imprecision of touch inputs in a detailed way. These two models do not consider touch input as being two-dimensional as is the case for current capacitive touch sensors that compute the active point as the centroid of the contact area. Instead, these models consider touch inputs as inputs in space. The first model, the *generalized perceived input point model*, also considers finger yaw, pitch and roll relative to the touch screen and the user [Holz and Baudisch, 2010]. This allows the minimum target size to be reduced up to $4.3mm$, with an optical tracker being used to sense finger angle. However, this approach requires a per-user calibration phase. It is also not directly applicable to capacitive touch screens, since it requires the capture of finger angles in space. The second model they proposed was the *projected center model* [Holz and Baudisch, 2011]. This model is based on the hypothesis that the user places her/his finger on a target according to visual features on the top of her/his finger. At the same time, the touch screen senses the contact area at the bottom of the user's finger. Using this model allowed the authors to reduce the minimum target size to $4.9mm$ without per-user calibration. However, in practice, this approach requires a top view of the touch surface.

While previous studies focused on direct-touch pointing precision, Bérard and Rochet-Capellan [Bérard and Rochet-Capellan, 2012] studied the precision of finger motions using an indirect configuration, by providing a cursor representing the active point. They evaluated linear and rotational finger precision on the touch surface using high-accuracy tracking during a time-constrained small-target pointing task. They found that user precision in the case of linear pointing was around $150dpi$ ($0.17mm$). Their results also indicated that users can reliably point at sectors 2.76 degrees wide. Thus, touch input allows precise adjustment on condition that the active point's position is not ambiguous (i.e., represented by a cursor for example).

Finally, touch screen input precision is impaired upon finger lift-off. In [Scoditti et al., 2011], we compared three different validation triggers on a smart phone: (1) finger lift-off, (2) *touch & hold*, and (3) device tilt. The results indicated that finger lift-off was less accurate (but faster) than the other two triggers. One explanation is that the system may still track the finger for a certain length of time as it is lifted from the screen. This would result in additional finger motion being sensed by the touch screen. Another explanation suggested in [Potter et al., 1988] is that the finger can move on the touch screen during lift-off according to its angular position. We also informally observed similar behaviors among the participants during the experiment reported in [Scoditti et al., 2011]. Lift-off would require further study to enhance our understanding and to enable quantitative measurements to be performed. As proposed in [Scoditti et al., 2011], one solution is to use another modality as a trigger. In the experiments reported in [Holz and Baudisch, 2010] and [Holz and Baudisch, 2011], the trigger was a footswitch rather than finger lift-off. In [Bérard and Rochet-Capellan, 2012], the trigger was controlled by a timeout set by the experimental protocol.

3.2.2.2 Pointing assistance

On touch-based handheld devices, direct-touch interaction is the de facto standard input. As explained in the previous section, while it is quite fast, it is also imprecise. Thus, pointing assistance techniques for touch screen handheld devices are chiefly geared towards improving pointing precision. Another issue addressed is access those regions that are difficult to reach when interacting and holding the device with one hand (see 3.2.2.1 and Figure 3.6).

Roudaut reviewed pointing techniques on handheld devices according to whether they used cursor-based, crossing-based, space partitioning or disambiguation gesture methods [Roudaut, 2010]. Here we will organize the different pointing assistance techniques according to whether or not they require target awareness in order to improve pointing. Target awareness is a key constraint, as it restricts the scope of application of such techniques to more specific use cases. Indeed, it does not assist (or even support) pointing at all positions on the screen (see 3.2.1.3). The original version of certain techniques (i.e., *Thumbspace* [Karlson and Bederson, 2008], *TapTap* and *MagStick* [Roudaut et al., 2008]) are target-aware. However, they can be adapted to target-agnostic situations. Thus, we classed these as target-agnostic techniques, since their use of prior knowledge of targets is optional, rather than mandatory.

Target-aware techniques. Target-aware techniques have been proposed mainly to address precision issues. They are primarily intended for use with dense clusters of small targets, such as marks on a map. Two main approaches have been proposed: space partitioning and disambiguation.

Starburst [Baudisch et al., 2008] addresses precision issues when acquiring small targets, particularly in the case of dense clusters of targets (Figure 3.9-a). *Starburst* is a space partitioning algorithm that is particularly suited to dense clusters of targets where a Voronoi tessellation would lead to small cells inside a the target cluster and large cells for targets on the outskirts of the cluster. Instead, the proposed algorithm allocates space on the outskirts of the cluster to all targets. However, because the cells of the partitioning algorithm must be displayed, this technique leads to a degree of visual overload. In addition, as with other space partitioning techniques, the technique's efficiency is dependent on target density and pointing in free space is not possible.

Unlike space partitioning, other available techniques are based on disambiguation, with the aim still being to improve pointing precision. *Burst* [Gunn et al., 2009] displays nearby targets in a circular menu at the press of a finger (Figure 3.9-b). The circular menu features a disambiguation phase and allows multiple selections by crossing the desired targets. However, in the circular menu, the targets are not presented in context.

Other disambiguation-based target-aware techniques use gestures. *Escape* [Yatani

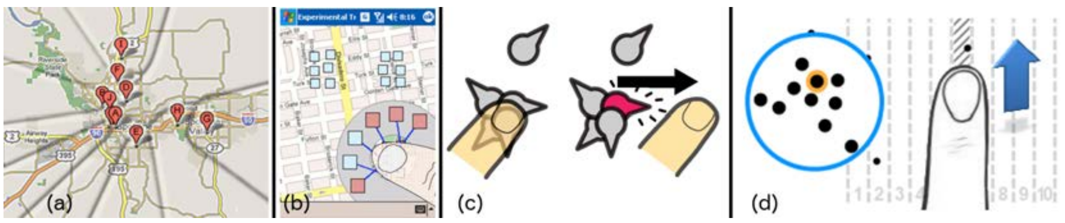


Figure 3.9: Target-aware touch-based pointing approaches for handheld devices. From left to right: (a) *Starburst*, (b) *Burst*, (c) *Escape* and (d) *LinearDragger* (Figures from [Baudisch et al., 2008] [Gunn et al., 2009] [Yatani et al., 2008] [Au et al., 2014]).

et al., 2008] and *Sliding Widgets* [Moscovich, 2009] indicate a direction on each target (Figure 3.9-c). To select a specific target the user must first land her/his finger near the target and then perform a stroke in the direction indicated by the target. Nearby targets should display different directions in order to ensure disambiguation. These techniques require each target to visually advertise its associated direction.

Finally, *LinearDragger* [Au et al., 2014] makes an area selection when the finger lands on the screen (Figure 3.9-d). Targets within the selected area are associated with a target in motor space away from the selected area. Disambiguation is then possible with a stroke in any direction up to the distance associated with the desired target. This technique therefore avoids finger occlusion during the disambiguation phase and minimizes visual overload.

Target-agnostic techniques. We will now present techniques that do not strictly require target awareness to operate. Two main target-agnostic approaches have been proposed: cursor based and zoom based.

One approach to addressing both finger occlusion and active point ambiguity is to use a cursor. Using a cursor allows pointing precision to be improved. Potter *et al.* [Potter et al., 1988] proposed the *Take-Off* technique (also known as *Offset Cursor*), which enables pointing adjustment and avoids finger occlusion by displaying a cursor slightly above the finger position (Figure 3.10-a). Huot and Lecolinet added an adaptive horizontal offset to the cursor position so as to ease access to the borders of the screen [Huot and Lecolinet, 2006]. One drawback of these techniques is that the user does not know the position of the cursor until her/his finger touches the screen.

Building on this technique, *Shift* [Vogel and Baudisch, 2007] extends direct-touch pointing with a precise quasi-mode (Figure 3.10-b). A tap on the screen triggers selection. If the user leaves her/his finger on the screen for a short dwell time, *Shift* enters a precise quasi-mode. When in this mode, *Shift* displays a circular callout showing a copy of the screen area occluded by the finger and places it in a non-occluded location. In the callout, a cursor represents the position of the active point. The user can adjust the position of the cursor by moving her/his finger. Finger lift-off both triggers validation and closes the callout. The authors also proposed to enhance *Shift* with by enabling zooming in the callout and setting a Control-to-Display gain

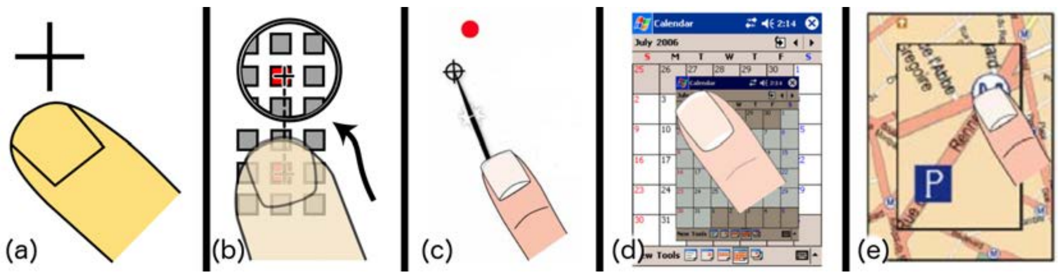


Figure 3.10: Touch-based pointing approaches for handheld devices that do not strictly require target awareness. From left to right: (a) *Offset Cursor*, (b) *Shift*, (c) *MagStick*, (d) *ThumbSpace* and (e) *TapTap* (Figures from [Vogel and Baudisch, 2007] [Roudaut et al., 2008] [Karlson and Bederson, 2008]).

lower than 1:1. *Shift* extends direct touch by making it fast for large targets, and addressing precision issues with its precise quasi-mode.

MagStick [Roudaut et al., 2008] also extends direct-touch pointing by using a telescopic stick metaphor to enable further adjustments (Figure 3.10-c). When the finger touches the screen, it defines a reference point. By dragging the finger away from the reference point, the user can then extend a telescopic stick centered on the reference point, with the finger at one end and the cursor at the other. Finger lift-off both triggers validation and hides the stick and the cursor. In addition, the original technique proposed feaures target-aware magnetization, which attracts the cursor to nearby targets and "bends" the stick. This results in an enlarged target width in motor space as with semantic pointing [Blanch et al., 2004]. *MagStick* extends direct touch by making it fast for large targets and addressing reach and precision issues.

Thumbspace [Karlson and Bederson, 2008] makes it possible to use an area of the touch screen as an absolute indirect touchpad mapped to all screen objects (Figure 3.10-d). The user first needs to define an area on the screen to be used as the touchpad. The user can then activate *ThumbSpace* mode by pressing a physical button on the device. In this mode, a radar view of the screen is displayed in the touchpad area defined previously. The user can then perform absolute pointing in the radar view and carry out adjustments by moving her/his finger, as finger lift-off triggers validation. In its original version, *ThumbSpace* uses a target-aware strategy similar to Object Pointing [Guiard et al., 2004]. However, the *ThumbSpace* approach can be adapted so as to control a cursor without the need for prior target knowledge. *Thumbspace* primarily addresses reach issues, but it can also assist with precision issues by helping to avoid occlusion.

Another approach involves zooming to enlarge the information space to a scale that allows accurate pointing [Albinsson and Zhai, 2003]. When using zooming, the user faces the classic trade-off between zoom level (for accurate pointing) and visible context (to locate the area of interest). The interaction process can be quite tedious on handheld devices with limited screen real estate: zoom in to focus and zoom out for context.

The zoom-based *TapTap* technique [Roudaut et al., 2008] increases pointing precision (Figure 3.10-e). Two taps on the screen are required for pointing. The first tap selects an area of the screen, a zoomed-in version of which is then displayed in a pop-up view at the center of the screen. The second tap enables precise selection within the zoomed area and closes the pop-up view. In its original version, *TapTap* zooms further in on known targets so as to improve pointing. However, target awareness is not required for *TapTap* to work. The drawbacks of this technique is that it is not integrated as an extension of direct touch, as two taps are required even for large targets. *ZoomTap* [Gunn et al., 2009] is based on *TapTap* and allows multiple selection and panning in the zoomed-in pop-up. However, it requires additional user actions to operate. *TapTap* addresses precision issues by using zooming, but it requires two taps even for large targets.

3.2.2.3 Summary

Touch screen input is relatively fast for tapping, but it is also imprecise. This imprecision is due to finger occlusion and to the ambiguous position of the active point. The ambiguity of the active point's position can be reduced by considering other information in addition to the contact area with the touch surface, such as the finger's position in space or the top view of the finger [Holz and Baudisch, 2010]. However, finger motion on the touch surface is precise [Bérard and Rochet-Capellan, 2012]. Finally, touch screen precision can be impaired during finger lift-off.

Furthermore, some regions of the smart phone touch screen are harder to reach when operated with one hand [Karlson and Bederson, 2008]. For tablets, the trade-off is between the device holding position and screen accessibility [Wagner et al., 2012].

A variety of pointing assistance techniques have been proposed primarily to address precision issues relating to both occlusion and an ambiguous active point, but also screen to resolve access issues.

Techniques that require target awareness mainly focus on improving precision, especially for clusters of small targets. They use either a space partitioning approach or a disambiguation step (circular menu or finger stroke). Some of those techniques (*Starburst*, *Escape*, *Sliding Widgets*) use visual feedback to display the effective active area of each target or the direction of the disambiguation gesture to be performed. This requires additional visual artifacts to be permanently displayed, which may be an issue on small screens.

Techniques that do not strictly require target awareness use either cursor-based or zoom-based disambiguation step to overcome precision issues. Some cursor-based techniques relax absolute direct control to a certain extent, but not, to our knowledge, to the point of allowing indirect relative control.

Table 3.2 sums up all the one-handed pointing techniques we reviewed for touch-based handheld devices, according to the target awareness primarily criterion. The Disambiguation column describes each technique's disambiguation step, if there is one. The Visual artifact column indicates whether the techniques rely on additional permanent visual feedback (e.g., *Starburst*'s cells) or temporary visual feedback (e.g., *Shift*'s callout). The table also indicates whether the techniques are compatible with direct touch, meaning that they allow fast but imprecise direct-touch pointing. For example, *TapTap* is not compatible with direct touch, as two taps are required for selection.

Table 3.2: Classification of touch-based pointing assistance techniques on handheld devices.

Target awareness	Disambiguation	Visual artifacts	direct touch compatible	Technique
Required	-	Permanent (Cells)	Yes	Starburst
	Menu	Temporary (Menu)	Yes	Burst
	Gesture	Permanent (Directions)	Yes	Escape
			Yes	Sliding Widgets
		Temporary (Highlight)	Yes	LinearDragger
Optional or None	Cursor	Temporary (Cursor)	No	Offset Cursor
		Temporary (Callout)	Yes	Shift
		Temporary (Radar, Cursor)	Yes	ThumbSpace
		Temporary ('Stick')	Yes	MagStick
	Zoom	Temporary (Pop-up)	No	TapTap

3.2.3 Handheld AR pointing techniques

Pointing in handheld AR applications involves pointing at a 3D position in the augmented scene. Thus, it encompasses pointing both at locations or objects in the physical surroundings and at digital objects registered to physical locations (see Chapter 2). However, several existing pointing techniques are based on pointing in the 2D image plane (e.g. direct touch on the screen, screen-centered crosshair - see 3.2.3.1). These techniques are one dimension short of being able to achieve pointing in 3D space.

In a review of outdoor annotation positioning approaches, Wither *et al.* [Wither et al., 2009] classified pointing techniques as follows:

- The *Model-based* approach, which makes it possible to point at the surface of objects using ray casting. This approach relies on the accuracy of the 3D models of the objects (be they models of physical objects or models used for digital augmentation) and their registration. In addition, pointing is only possible on the surface of physical objects that have been modeled.

- The *Measurement-based* approach, which measures the distance between the user's position and the designated object. This approach restricts pointing at the surface of physical objects, but does not rely on a model. However, it requires specific hardware to measure the distance.

- The *Estimation-based* approach, which requires the user to position a 3D cursor in space. The application can provide extra cues regarding the depth of the cursor and other objects in the viewing frustum, to help the user point at the right depth (e.g., with a top-view representation of the scene) This approach makes it possible to point at any 3D position, not just at objects.

- The *Triangulation-based* approach, which requires the user to perform pointing from two different viewpoints (one of which can be a top-view), so as to specify a 3D position. This approach also makes it possible to point at any 3D position.

Initially, we restricted our study to the model-based approach, with which it is only possible to point at the surface of both physical and digital objects. We did not consider pointing in mid-air or inside/behind objects.

Whether the target is a physical location or a digital augmentation, the user first needs to bring the target into the field of view of the handheld device's camera. This involves an initial pointing task, which is necessary if the user is to further interact with the augmented scene. Because the camera is tightly bound to the handheld device, the camera's viewing frustum is a selection volume controlled in an absolute manner by the handheld device's position and orientation in space. This pointing phase is designated by the term "physical pointing" in [Rohs and Oulasvitra, 2008]. To select a target in the viewing frustum, different input modalities can be used. This second phase is referred to as the "virtual pointing" phase in [Rohs and Oulasvitra, 2008].

In the following section, we will review the modalities used for pointing in handheld AR applications.

3.2.3.1 Input modalities

The techniques proposed for pointing at physical or digital objects in an augmented scene are based on three main modalities:

- Use of touch screen or stylus input
- Use of the device's position and orientation in space to control a screen-centered crosshair
- Use of bare fingers, which are displayed on the handheld device's screen via the camera.

Touch and stylus. Direct touch or stylus input via the screen has previously been used to interact with augmented scenes in handheld AR. For example, *Touch Projector* [Boring et al., 2010] (see 2.2.3) allows users to move pictures on remote screens by directly touching the live video displayed by a handheld device.

Pointing via the touch screen in handheld AR inherits the limitations of touch screens described in 3.2.2. Thus, touch screen-based handheld device pointing techniques are of interest for handheld AR pointing in terms of interaction capabilities, precision and screen accessibility.

However, the handheld AR context brings further limitations. First, competition for screen real estate is even more critical in the context of handheld AR, as the display space is shared by both the representation of the surroundings and the digital augmentation. Second, touch allows absolute pointing on the screen. Due to *Viewpoint* instability, the augmented scene is not only unstable on the screen, but also in the touch input space (see 2.6.3). In addition, it only allows a 2D position in the camera image plane to be specified. This makes it possible to point at the surface of objects using ray casting, but extra information is required to point at any other position such as in mid-air or inside/behind objects.

The touch-based pointing assistance techniques of handheld devices therefore need to be adapted to the context of handheld AR.

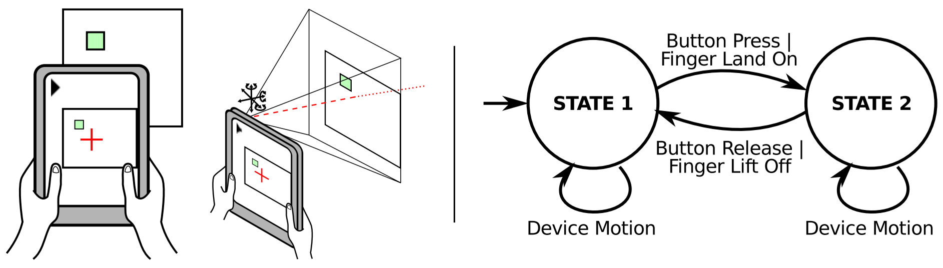


Figure 3.11: Screen-centered crosshair: (Left) Technique presentation; (Right) State machine.

Device position and orientation. Another modality used for pointing in handheld AR applications relies on the device's absolute position and orientation in space (Figure 3.11-Left). A crosshair represents the active point, which is centered on the screen (i.e., centered in the camera image plane) (Figure 3.11-Left and 3.12). As discussed previously, a model-based approach allows the active point to be defiend in 3D space. The device's absolute position and orientation in space controls the active point, while a physical button or a tap on the screen triggers validation.

The screen-centered crosshair technique uses a cursor to represent the active point. Thus, the position of the active point in the image plane is not ambiguous, unlike with direct-touch interaction. Moreover, because the crosshair is always represented on the screen, when an estimate of the camera position and orientation is available two interactive tracking states are defined (Figure 3.11-Right). Unlike with touch input, this enables the use of interaction techniques such as hovering.

Like with other freehand interaction techniques, such as laser pointers and handheld projectors, the device's position and orientation are subject to hand tremor, because the device is not self-stabilized. This was discussed in terms of *Viewpoint* instability in the Design Space chapter (see 2.6.3). In addition, interaction via buttons or the handheld device's touch-screen can further impair stability. These factors reduce the precision of this input modality when pointing in the augmented scene. Stability is probably also affected by the device's form factor and holding method as with laser pointers [Myers et al., 2002].

The screen-centered crosshair technique has been used in several studies [Henze and Boll, 2011] [Liao et al., 2010] [Rohs et al., 2009]. Rohs *et al.* [Rohs and Oulasvitra, 2008] showed that the performance of this technique could be modeled with a two-part Fitts' law. Indeed, in this case Fitts' law can be applied separately for both the physical pointing phase and the digital pointing phase. Their study showed that the process performed by the device (i.e., camera image capture, memory transfer, vision-based tracking and display) impairs the digital pointing phase. Indeed, camera image presentation latency affects pointing at targets when the latter are viewed on the camera image displayed on the handheld device's screen. In a follow-up study, they showed that this two-part Fitts' law also modeled pointing in more realistic conditions when pointing at nearby buildings [Rohs et al., 2011]. In this case, they used an angular version of Fitts' law.

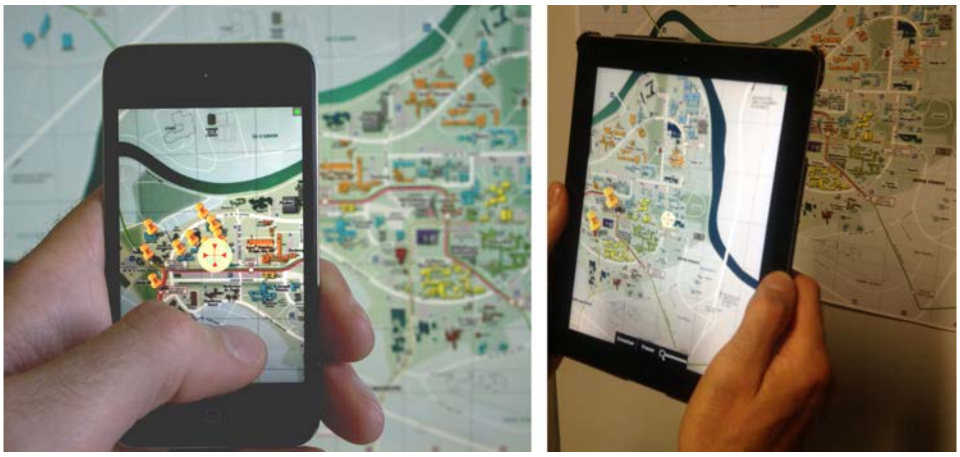


Figure 3.12: Use of screen-centered Crosshair to point at a physical wall map with a smart phone (Left) and a tablet (Right). The cursor is been highlighted for a better picture clarity.

Bare fingers. The third modality involves the use of bare fingers in front of the camera (Figure 3.13). This allows for a single interactive tracking state and a trigger is required for validation.

When comparing the use of bare fingers with touch screen-based interaction, Bai *et al.* [Bai et al., 2012] found that finger interaction was less usable than touch-based techniques. They suggested that this was due to two factors: first, their finger tracking was not sufficiently effective, and, second, they used a dwell time to change the state.

However, an experiment held by Liu *et al.* [Liu et al., 2012] indicated that handheld AR can assist the manipulation of physical buttons and sliders. The authors compared the efficiency of four different ways of providing instructions in order to set buttons, knobs and sliders to specific positions. They compared two baseline conditions (text instructions and image-based instructions) with two handheld AR conditions (AR instructions and AR instructions with additional feedback on the current state of sliders). The results indicated that the handheld AR condition that provided feedback on the current state performed the best.

Such input enables the user to operate directly in her/his physical surroundings and thus Tangible User Interfaces. However, with handheld AR set-ups this form of input requires the user to hold the device with just one hand, which may be awkward with tablets.

Summary Both screen-centered crosshair and touch allow pointing in the camera image plane. Unlike the direct-touch on the screen, crosshair is not impaired by occlusion or an ambiguous active point. Furthermore, it allows for two "tracking" states. However, both modalities suffer from on-screen content instability (see 2.6.3). Bare-finger interaction via the video engage the user with the physical objects, yet, it can be awkward with tablets.

3.2.3.2 Pointing assistance

Various techniques have been proposed to improve pointing and interaction with handheld AR applications.

In the Design Space chapter (see Chapter 2), we discussed the impact on touch interaction of *Viewpoint* instability due to hand tremor (see 2.6.3). Strategies to overcome this limitation include: (1) stabilizing the *Representation of the physical world* on the screen and (2) stabilizing input relative to the frame of the physical

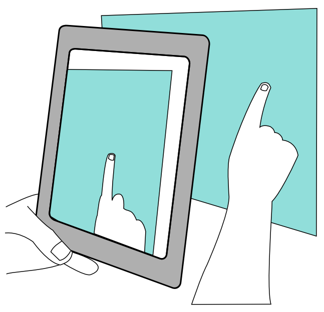


Figure 3.13: Using a bare finger in front of the camera.

object. An orthogonal dimension of these two strategies is zooming. We will discuss the effect of zooming once we have described the two strategies for reducing instability.

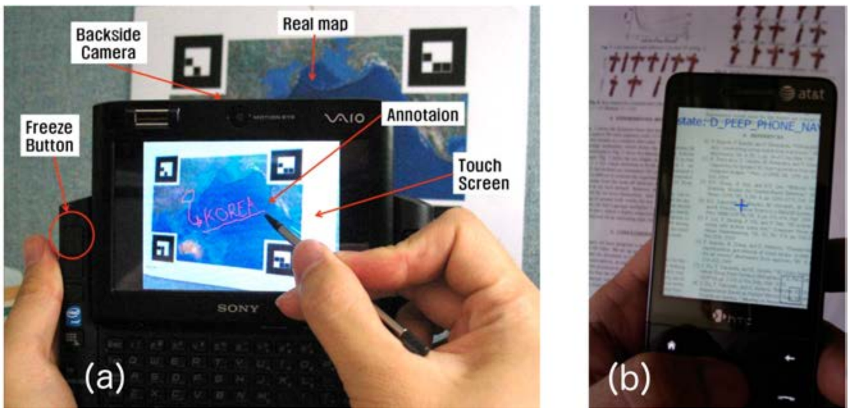


Figure 3.14: From left to right: (a) the handheld AR set-up used by Lee *et al.* [Lee et al., 2009] in their evaluation of the freeze-frame method for drawing; (b) *PACER* displays a digital copy of the document recognized in the camera image (Figures from [Lee et al., 2009] [Liao et al., 2010])

Viewpoint stabilization. To reduce the effect of natural hand tremor on viewpoint position, various techniques and studies have suggested freezing the frame. Indeed, when the live video is paused, the viewpoint becomes steady. Freezing the frame also allows the user to move to a comfortable position for interaction. This approach is obviously not compatible with the screen-centered crosshair or finger interaction via the camera, but it has proven useful in improving pen and touch interaction.

A user study performed by Lee *et al.* [Lee et al., 2009] showed that a video freezing mode improves precision when drawing on a physical object with a pen via the handheld device's camera images. They also noted that some users were disoriented when they returned to the live camera images, as the viewpoint had changed.

Another issue when freezing the frame is that the scene is no longer updated. *Touch Projector* [Boring et al., 2010] (see 2.2.3) overcomes this issue by updating the video snapshot with a digital copy of the remote screen the user is interacting with. Unfortunately, a digital copy of the object of interest is not available in all AR scenarios.

Another way of stabilizing the viewpoint is to use "loose registration", as in *PACER* [Liao et al., 2010]. To interact with paper documents, a digital copy of the document recognized in the camera images is displayed on the handheld device's screen instead of the camera images themselves (Figure 3.14-a). This relaxes tracking requirements and allows the viewpoint position to be filtered. *PACER* allows interaction using both screen-centered crosshair and direct touch. Like *Touch Projector*, this technique requires there to be a digital copy of the object of interest.

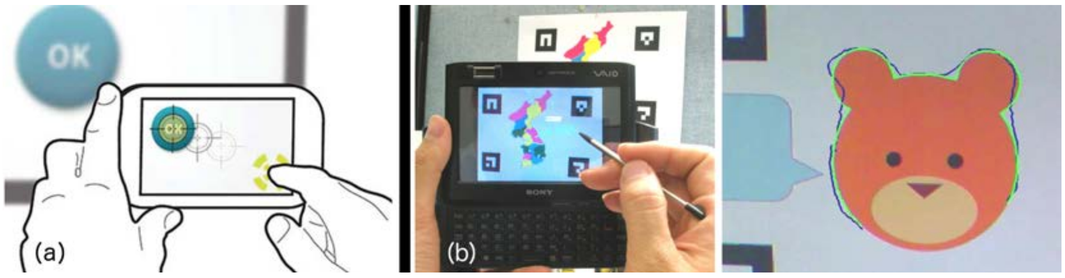


Figure 3.15: From left to right: (a) *Snap Target*: target-aware snapping of a screen-centered crosshair. (b) *Snap-to-feature* snaps stylus inputs to nearby edges recognized in the camera image: (Left) prototype; (Right) input stroke in blue and snapped stroke in green. (Figures from [Baldauf and Fröhlich, 2014] [Lee et al., 2010b])

Input stabilization. A second strategy is to stabilize input in the frame of reference of the physical object (or its representation on the screen), rather than stabilizing the object of interest on the screen, as for viewpoint stabilization. This type of strategy has been applied both to input based on device position and orientation and to on-screen stylus-based input.

For the screen-centered crosshair method, Baldauf and Fröhlich [Baldauf and Fröhlich, 2014] proposed the *Snap Target* technique (Figure 3.15-a). *Snap Target* snaps the crosshair to the closest target less than a threshold distance away in the frame of reference of the physical object (a remote screen in their experiment).

In a controlled experiment, they compared direct touch, Crosshair, and *Snap Target* on both smart phone and tablet form factors, for both target acquisition and dragging tasks. Their results indicated that task completion time and the number of errors per trial both increased when using direct touch for small-target acquisition on the smart phone (Width=$4.32cm$ on the remote screen). This suggests that the touch screen's target acquistion precision is limited. However, the width of the targets on the handheld device's screen were not reported in the paper. For the dragging task, *Snap Target* was faster overall than both direct touch and crosshair, while direct touch was more error prone than both Crosshair and *Snap Target.* Again, direct touch was slower and more error prone during small-target acquisition on the smart phone. This experiment indicated a form factor effect in the case of direct touch that may be explained by the scale factor applied to the camera images displayed, which depends on screen size. *Snap Target* was not as effective as the authors had expected, especially for target acquisition. During the experiment, with dense target environments the authors noted incorrect selections among elderly participants and participants who were overly confident when using the technique and had therefore speeded up. As suggested by the authors, further study and adaptation of target-aware techniques, such as a dynamic area cursor, in the context of handheld AR would of interest.

As regards on-screen stylus input, *Snap-to-feature* [Lee et al., 2010b] snaps stylus strokes to the features of physical objects detected in the live video (Figure 3.15-b). This allows more accurate drawing of the contours of physical objects displayed on the screen without relying on video freezing or a digital copy of the scene. Unlike on-screen content stabilization techniques, which sever the live relationship with the physical surroundings or use a digital copy of the scene, input stabilization enables precision to be improved without losing the live relationship with the surroundings.

In section 3.3.2.2, we propose the *Relative Pointing* technique, which is based on the stabilization of touch input in the frame of the physical object. Unlike *Snap Target*, our technique is not based on target awareness. Unlike *Snap-to-feature*, our technique does not rely on the detection of features of physical objects in the live video.

Zooming. As in other contexts of use, a complementary approach to the two strategies described above is to increase target size on the screen, by either moving closer to the physical object or zooming in to the live video. Zooming is compatible with both screen-centered crosshair and direct input, as well as with other strategies for improving interaction. It can be helpful for small targets, especially when using pointing modalities that lack precision, such as touch input.

The user study performed with *Touch Projector* [Boring et al., 2010] (see 2.2.3) indicated that, overall, automatic zooming was the best performing technique when compared to the fixed focal length, manual zooming and freeze-frame techniques. While zooming improves interaction, it does not steady the camera image. The study also highlighted that a freeze-frame mode combined with automatic zooming outperforms automatic zooming alone when precise manipulation is required.

However, as explained in 3.2.2.2, zooming is a trade-off between zoom level and visible context. Finally, the zoom level is constrained by the camera image quality. The zoom level in a handheld AR context is therefore limited. As in the case of *Touch Projector*, zooming can be combined with other handheld AR pointing assistance strategies, including viewpoint stabilization.

3.2.3.3 Summary

Different input modalities have been used and studied for handheld AR interaction. The use of bare fingers in front of the camera enables to maintain the user in the physical space as it allow tasks to be performed in the physical world. However, it is also limited, as it requires to be combined with device holding. Touch is an input modality widely used by handheld devices in general, and one that has been studied previously (see 3.2.2.1). Device position and orientation already controls the viewpoint of the device's back-facing camera, therefore any other modality needs to be combined with it. These two inputs (touch and device position and orientation) simply allow pointing in the camera image plane on the screen. Thus, additional information is required to position the active point in 3D space, like with the model-based approach. However, pointing in the camera image plane on the screen is subject to viewpoint instability (see Chapter 2).

Interaction improvements are based on viewpoint stabilization (e.g., video freezing) and input stabilization. Furthermore, zooming can be combined with both viewpoint and input stabilization.

The approaches presented address limitations that are specific to the handheld AR context (i.e., viewpoint instability due to hand tremor - see 3.2.3) but not necessarily the limitations of touch inputs (i.e., occlusion, ambiguous active point and reach issues - ee 3.2.2).

3.2.4 Conclusion

Pointing at physical or digital objects using a handheld AR application is a task that involves pointing in 3D space. However, to explore the different alternatives we reviewed pointing in a wider context.

Different techniques have been proposed to improve pointing in different environments (e.g., desktop, Virtual Reality). Pointing assistance techniques are often discussed in terms of performance with respect to Fitts' law. However, it is also important to consider issues relating to small targets. Small targets can be difficult to acquire due to precision issues. A key characteristic of pointing techniques is whether they are target aware or target agnostic, since target-agnostic techniques support a wider range of use cases.

We subsequently focused on touch-based handheld device pointing techniques, as touch input is a candidate for handheld AR pointing. Target-aware techniques are mostly based on space partitioning or a disambiguation step. Target-agnostic techniques use either a cursor or zooming.

Finally, handheld AR pointing can be performed using three main types of input: touch input, device position and orientation in space, and bare fingers in front of the camera. Different interaction improvements have been proposed for handheld AR. First, the viewpoint can be stabilized by freezing the frame (with or without zooming) at the expense of interrupting the live representation of the surroundings on the screen and causing viewpoint discontinuity when returning to the live camera image. It is also possible to stabilize input in the frame of the physical object or in that of its representation on the screen. This allows input stabilization without freezing the frame. This is therefore an interesting avenue that we chose to further explore.

Among the different pointing assistance approaches, we were particularly interested in the use of target-agnostic techniques to support a wide range of use cases. Building on the current state of the art, we proposed different approaches to improving pointing precision in handheld AR applications.

Viewpoint stabilization can be performed to different degrees, live camera images and video freezing being the most extreme case. "Loose registration", as in *PACER* [Liao et al., 2010], is an example of a filtered viewpoint position. Filtering the viewpoint position when using live camera images may be an interesting avenue, but this is an area we did not explore.

Within the scope of handheld AR, other than *PACER* [Liao et al., 2010], we are not aware of other attempts to combine viewpoint stabilization with touch-based pointing assistance techniques. Moreover, the freeze-frame techniques proposed take the form of an explicit mode. The techniques we propose in the following section combine freeze-frame with existing target-agnostic touch screen pointing assistance techniques, either as a once mode or as a quasi-mode.

Looking at existing pointing techniques, it is apparent that more improvement possibilitiies are available for indirect relative active point control. Indeed, indirect relative control allows the transfer function to be modify dynamically so as to improve pointing without modifying visual space. Dynamic transfer functions are a common feature of desktop interaction, as they can improve performance (see 3.2.1). They also reduce the size of the input workspace and/or the need for clutching, since they provide a high Control-to-Display gain at high speed. Finally, they also enlarge targets in motor space at low speed. However, they are only usable with indirect relative active point control. We propose a technique that allows indirect relative pointing within the scope of handheld AR. It uses a dynamic transfer function primarily to reduce the need for finger clutching and to improve the acquisition of small targets.

In the following section, we will detail the different approaches we explored.

3.3 Handheld AR Pointing Techniques

We will organize our exploration of handheld AR pointing techniques with respect to both previous work on pointing techniques and the spatial relations of our design space, as presented in the previous chapter (see Chapter 2 and Figure 3.16). This will enable an analysis of the issues affecting accurate pointing for handheld AR.

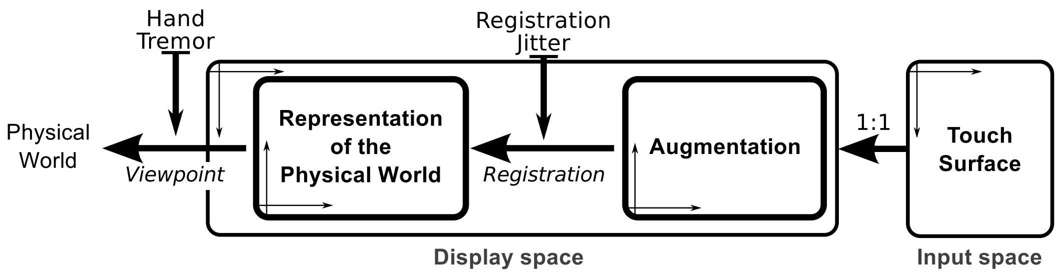


Figure 3.16: The *Viewpoint* and *Registration* spatial relations and the frames of reference of touch input, the screen, the representation of the physical object on the screen and the augmentation in the case of live camera images representing the physical surroundings.

Both the live camera images representing the physical surroundings and the digital augmentation that registered to physical objects are unstable on the screen, since the *Viewpoint* is impaired by hand tremor (Figure 3.16). Direct-touch interaction takes place in the screen frame. The objects with which the user interacts (be they physical or digital) are therefore not stable in the touch input frame (Figure 3.16).

With video freezing, the *Viewpoint* is no longer related to the device's position and orientation and the *Registration* is stationary. The frame of reference of the physical object and that of the digital augmentation are fixed within the still camera image and are therefore stable on the screen (Figure 3.17-b, see 3.2.3). This case is similar to that of GUI interfaces. Thus, existing pointing techniques for handheld devices (see 3.2.2) can be combined with video freezing. We developed two techniques based on this approach: (1) *AR TapTap*, which combines the *TapTap* pointing technique [Roudaut et al., 2008] with video freezing, and (2) *Shift&Freeze*, which combines the *Shift* pointing technique [Vogel and Baudisch, 2007] with video freezing.

Unlike with video freezing, when interacting via live camera images the representation of the physical object on the screen is jittery, because it is impaired by *Viewpoint* instabilities (Figure 3.17-a). In this case, we can consider whether pointing is performed in the frame of reference of the screen or in that of the physical object. Orthogonal to considering the frame of reference used for pointing, we also considered whether pointing is performed with or without an instrument (i.e., a cursor) (Figure 3.18).

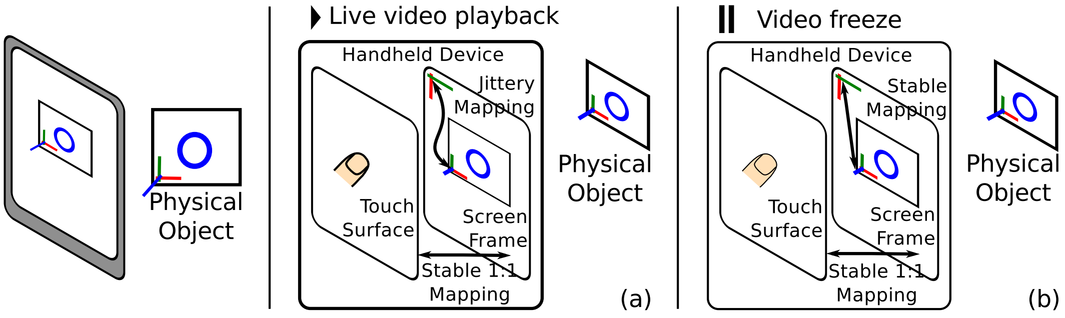


Figure 3.17: Relations between the frames of reference of touch input, the screen and the physical object on the screen: (a) with live camera images; (b) when the video is frozen.

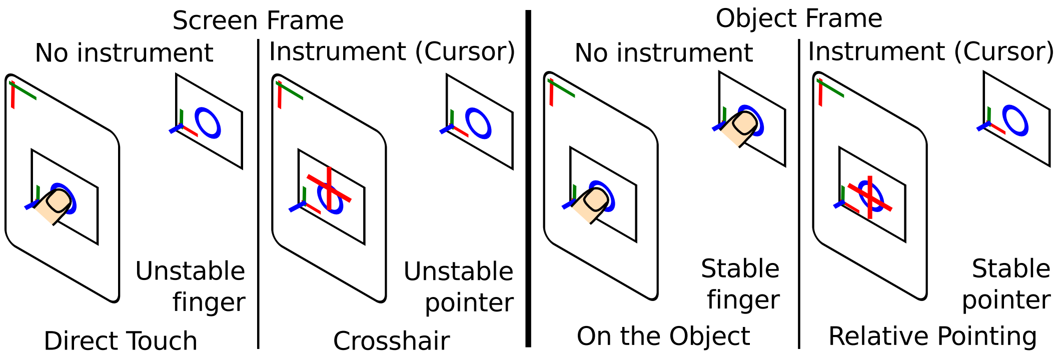


Figure 3.18: Pointing through live camera images: four cases.

Direct touch involves pointing in the screen frame without a cursor. Screen-centered *crosshair* makes use of a cursor and also involves pointing in the screen frame. These two techniques (with and without a cursor) are impaired by hand jitter, as pointing occurs in the screen frame, which displays an unstable image of the physical object of interest (Figure 3.17-a).

If pointing were to be performed in the frame of reference of the physical object rather than in the screen frame, pointing precision would not be impaired by hand jitter. Pointing in the physical object's frame of reference without a cursor instrument requires direct interaction with the physical object. We explored this particular case using a prototype of an augmented light switch.

Finally, our *Relative Pointing* technique involves pointing with a cursor in the physical object's frame of reference. In this case, we use indirect relative mapping of touch input to cursor movement in the frame of reference of the physical object. Because the cursor is located in the frame of reference of the physical object rather than on the screen, its position is not impaired by *Viewpoint* instability relating to hand jitter (see Chapter 2).

To sum up, we explored two approaches to interacting with the augmented scene in a handheld AR set-up (Table 3.3): (1) interacting in the screen frame and (2) interacting in the frame of the physical object. We developed two touch screen-based pointing techniques, namely *AR TapTap* and *Shift&Freeze*, which combine existing pointing techniques for handheld devices (see 3.2.2.2) with video freezing. We also explored the use of fingers between the camera and the physical object, as well as on the physical object itself. Finally, we developed a pointing technique that extends screen-centered *Crosshair* with an indirect relative precise mode, which stabilizes the cursor on the physical object. This section presents these techniques.

Table 3.3: Two adopted approaches to interact with the augmented scene and four techniques.

Main frame for interaction	Proposed Interaction Techniques
Screen frame	AR TapTap
	Shift&Freeze
Physical object's frame	Directly on the object
	Relative Pointing

3.3.1 Pointing in the screen frame

We combined the pointing assistance proposed for touch-based handheld device pointing (see 3.2.2) with video freezing as proposed for handheld AR interaction (see 3.2.3). As explained in 3.2.2.2, target-agnostic pointing assistance techniques on touch-based handheld devices are based on zooming or displaying a cursor.

We combined two of these techniques with video freezing, resulting in two new pointing techniques for handheld AR set-ups. The first, *AR TapTap*, extends *TapTap* (see 3.2.2.2, Figure 3.10-e), a zoom-based pointing technique for touch-based handheld devices, by adding video freezing. The second, *Shift&Freeze*, extends *Shift* (see 3.2.2.2, Figure 3.10-b), a cursor-based pointing technique for touch-based handheld devices, again by adding video freezing. We will now present these two techniques.

3.3.1.1 AR TapTap

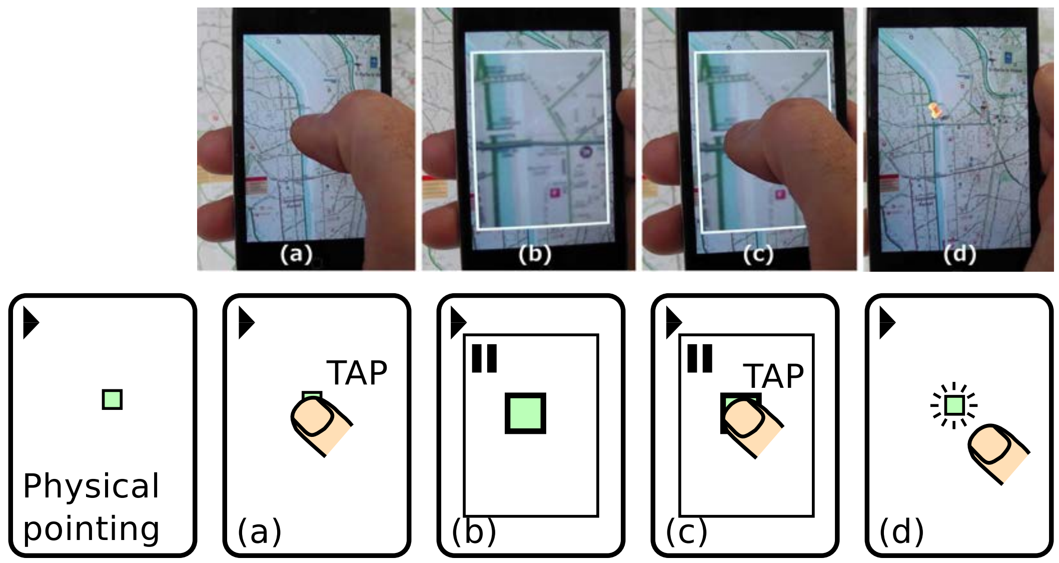


Figure 3.19: Top: Use of *AR TapTap* to place digital marks on a physical wall map. Bottom: *AR TapTap* walk-through. (a) A first tap to freeze and zoom the video (b); a second tap to place a mark (c) and automatically close the frozen and zoomed-in view (d).

AR TapTap uses zooming and video freezing to aid pointing, e.g., to position digital marks on a physical map (Figure 3.19-top). At the bottom of Figure 3.19, a walk-through of pointing with the *AR TapTap* technique is provided.

Scenario:

Physical pointing (see 3.2.3): the user points the handheld device's camera at the target so that the target appears on the screen.

(a) The user performs a first crude tap in the target area.

(b) This opens a pop-up window at the center of the screen displaying a frozen and zoomed-in view of the selected area, while live camera images are displayed in the background.

(c) The user performs a second tap in the pop-up.

(d) This triggers target selection and closes the pop-up view.

When the pop-up view is open (Figure 3.19-c), the user can close it by tapping outside it. This will not cause a new selection to be made.

Matthieu Riegler implemented *AR TapTap* during his Master 1 internship.

Implementation. We developed *AR TapTap* on iOS and ran it on a 4th generation iPod. We used the Vuforia SDK[2] tracking library and OpenGL|ES 1.1[3] as the rendering back-end. The camera images were provided via the tracking library with a resolution of 480x640 pixels. This limits the zoom factor that can be applied in the pop-up window.

Touch interaction. *AR TapTap* is a target-agnostic technique. Unlike *TapTap* [Roudaut et al., 2008], it does not use a target-aware zoom. This technique allows direct interaction on the touch screen and mitigates touch precision issues (relating to occlusion and the ambiguous active point problem, see 3.2.2) by performing pointing in two steps: a crude pointing step, followed by a disambiguation pointing step in a zoomed-in and frozen view. This technique does not provide assistance for hard-to-reach regions of the screen. In addition, in its basic configuration, this technique is not well suited to the tablet form factor when the device is held with both hands, as is usually the case with handheld AR applications.

Inherited from *TapTap*, the interaction method is quite fast. However, two taps are required for all targets, even large ones. This impairs performance when acquiring large targets. It would be preferable to allow direct-touch pointing for large targets and to activate an accurate pointing mode for smaller targets.

Handheld AR characteristics. This technique freezes the frame in the pop-up view to steady the *Viewpoint* in preparation for the second tap.

In our design space, the first tap provokes a transition along the *Viewpoint* axis (from *Conformal* to *None* in Figure 2.15 - see Chapter 2). The second tap, which actually selects the point, also closes the frozen and zoomed-in view and returns the display to the initial *Viewpoint* state (i.e., *Conformal* mapping - live camera images). To enable accurate pointing, *AR TapTap* therefore explicitly modifies the *Viewpoint* with the first tap. This modification is transient, because the second selection tap

[2]https://www.vuforia.com/

[3]http://www.khronos.org/opengles/1_X/

does not serve to change the current mode (from *None* to *Conformal* in Figure 2.15), but to place a mark.

With *AR TapTap*, the freeze-frame mode is a once mode, unlike other techniques that use video freezing [Boring et al., 2010] [Guven et al., 2006] [Lee et al., 2009]. *AR TapTap* adds video freezing to *TapTap* without requiring additional user actions.

Because the frozen view is not displayed in full-screen mode, the live camera images are still visible around the pop-up view. This is an example of on-screen multiplexing of two views with different *Viewpoints*. We had expected this spatial multiplexing of *Viewpoints* to help users locate the current viewpoint when the pop-up view was closed. However, informal tests were inconclusive.

The different limitations we have discussed motivated us to consider extending another touch-based pointing technique: *Shift* [Vogel and Baudisch, 2007].

3.3.1.2 Shift & Freeze

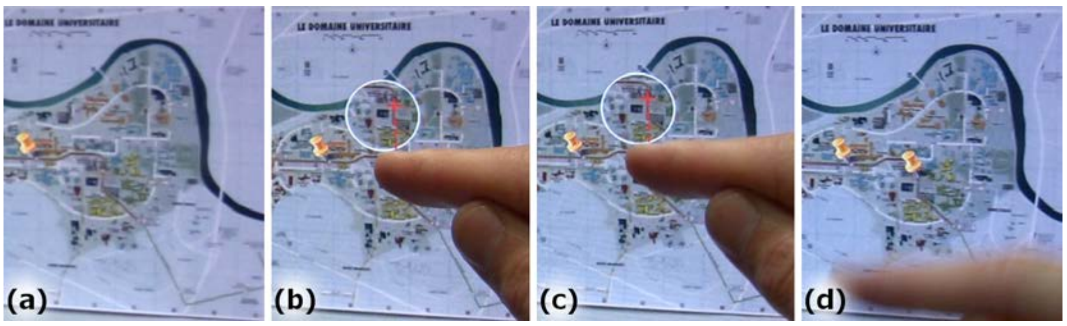


Figure 3.20: Use of *Shift&Freeze* to place digital marks on a physical wall map: after a short dwell time, the video is frozen and a callout is displayed (a-b); while the finger remains on the screen, pointing adjustment is possible (c); on finger lift-off, selection is triggered, the callout is closed and live camera images are restored (d).

Shift&Freeze uses a cursor and video freezing to aid pointing, e.g., to position digital marks on a physical map (Figure 3.20). Figure 3.21 provides a walk-through of the two modes of the *Shift&Freeze* technique.

Scenario 1:

Physical pointing (see 3.2.3): the user points the handheld device's camera at the target so that the target appears on the screen.

(a-b) A short dwell time after the finger makes contact, *Shift&Freeze* enters a precise quasi-mode and the video is frozen. A callout showing the area under the finger is displayed above the finger. In the callout, a cursor indicates the active point.

(c) When in this mode, the video remains frozen and the user can adjust the cursor position by moving her/his finger.

(d) On finger lift-off, the target is selected and live camera images are restored.

Scenario 2:

(e-f) For targets large enough for hand tremor and finger occlusion not to be a problem, selection can be performed with a tap on the screen at the target's position.

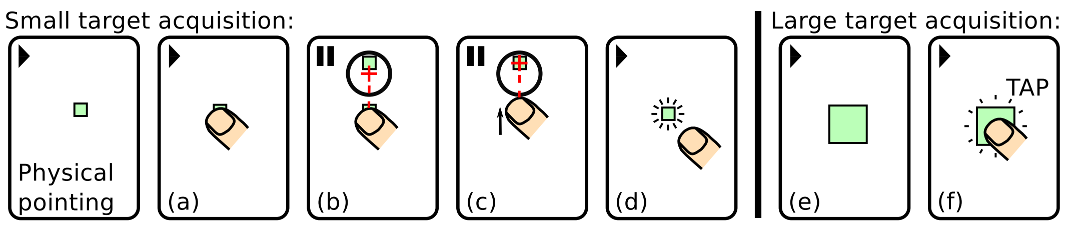


Figure 3.21: *Shift&Freeze* walk-through. (b-d) Small-target acquisition with *Shift* and video freezing; (e-f) Large-target acquisition with direct touch (one tap on the screen).

Implementation. As with *AR TapTap*, we developed *Shift&Freeze* on iOS with the Vuforia SDK[4] tracking library and OpenGL|ES 1.1[5]. We ran it on iPad and the 4th generation of iPod. On all devices, the camera images were provided via the tracking library with a resolution of 480x640 pixels and displayed in full-screen mode (cropped on the iPod).

Touch interaction. To address the limited precision of touch input, we used *Shift* [Vogel and Baudisch, 2007]. *Shift* improves touch screen input precision by using a callout displaying the area under the finger to overcome occlusion. It also overcomes the ambiguous active point issue by displaying a cursor in this callout (see 3.2.2).

In addition, *Shift* possesses the following characteristics. First, it is a target-agnostic technique. We did not consider the use of target awareness to modify the dwell time or the transfer function when in precise mode, as proposed in *Shift* [Vogel and Baudisch, 2007]. Secondly, it extends direct touch, meaning that fast but imprecise pointing is still possible. Unlike *AR TapTap*, overhead interaction is only required when precision is a problem. Finally, it is possible to use *Shift&Freeze* on a handheld tablet while holding it with either one or two hands.

However, *Shift&Freeze* does not provide assistance for hard-to-reach regions during one-handed use or on tablets. Also, since finger lift-off triggers selection and closes the precise mode, finger movements detected when the user lifts her/his finger may impair pointing precision (see 3.2.2).

Handheld AR characteristics. To address viewpoint instability in handheld AR settings, we combined *Shift* with video freezing. Generally speaking, touch-based pointing techniques, *Shift* in particular, are designed for pointing at static targets on the screen. Instead of implementing a separate freeze-frame mode, we enhanced *Shift*'s precise quasi-mode with video freezing. Compared to the original *Shift* technique, no additional user actions are necessary to control video freezing/unfreezing.

Like with *AR TapTap*, in our design space this process leads to a transition along the *Viewpoint* axis between *None* and *Conformal* (see Chapter 2). This transition is transient and only occurs during interaction in precise mode.

However, using video freezing comes at a cost: it interrupts the *Viewpoint* spatial relation. As noted in [Lee et al., 2009], restoring the live camera images leads to viewpoint discontinuity, which may disorient the user. We tested *Shift* informally on

[4]https://www.vuforia.com/
[5]http://www.khronos.org/opengles/1_X/

live camera images, without freezing the frame. This was very hard to control, as both device position and orientation (which are unstable) and finger position caused the designated point to move.

We also informally tested zooming in to the callout. Due to both the limited quality of the camera images and the limited size of the callout, zooming was not usable because of the lack of context displayed in the callout.

Thus, *Shift&Freeze* was more promising than our first attempt with *AR TapTap*, as it extends direct touch and does not rely on zooming. We therefore decided to evaluate *Shift&Freeze* in two controlled experiments that we will describe in the next chapter (see Chapter 4).

3.3.2 Pointing in the object's frame

The second approach we explored instead of pointing in the screen frame is pointing in the physical object's frame (Figure 3.18). We investigated two options for interacting in the object's frame. First, we developed a prototype of a system that allows interacting directly on an object through the live camera image. Second, we designed the *Relative Pointing* technique that uses a cursor controlled in the frame of the object.

3.3.2.1 Interaction through the video

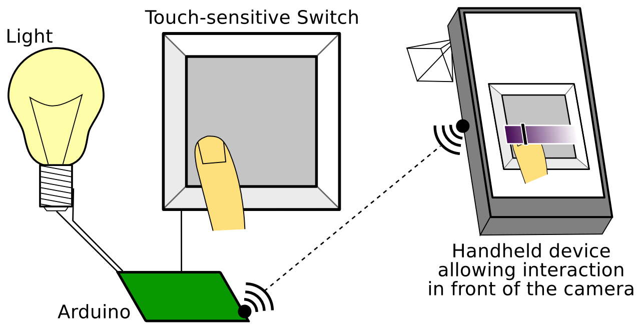


Figure 3.22: Illustration of a light switch with augmented control used in parallel with a handheld AR application.

We explored interaction using bare fingers between the camera and the object, and on the object itself (see 3.2.3.1). The goal was to improve interaction in front of the camera by considering both finger position in the camera image and whether or not the finger is touching the physical object. Indeed, moving the finger in mid-air in front of the camera only provides just a single tracking state.

The use case envisioned involved controlling lights via an augmented light switch. This tyoe of light switch comprises of a touch-sensitive surface that detects finger contact, but not finger position. By default, it acts as an on/off switch upon finger detection. When the light switch is recognized in the handheld device's camera image, more controls (e.g., a slider to control the brightness of the light) are overlaid on the surface of the light switch (Figure 3.22). The user can then interact with this slider by moving her/his finger from side to side between the camera and the light switch. With the finger in mid-air, the user can preview different levels of brightness both on the handheld device's screen and on the light itself. To save the light's new brightness level, the user must touch the light switch at the desired brightness position on the slider.

Remi Di Savino implemented a prototype of this system during his Master 1 internship.

Implementation. We implemented a prototype on which the physical "switch" was comprised of an image for vision-based tracking, overlaid on a sheet of aluminum foil used as a capacitive touch-sensitive surface. The sheet of aluminum foil was connected to an Arduino[6] to detect finger touch (Figure 3.22). The Arduino also controlled an LED. It was connected to the handheld device via the network. We used the Vuforia SDK[7] tracking library to recognize the "switch" and to obtain the finger position, while the Arduino sensed finger contact.

The user was required to hold the handheld device running the AR application, which used vision-based tracking to recognize the image of the "switch". When the image was recognized, a slider was overlaid on it (Figure 3.23). The user could control the slider with her/his finger in mid-air and save the new slider value by touching the "switch".

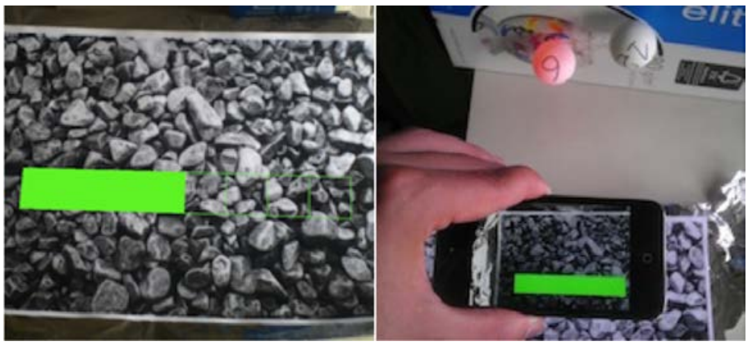


Figure 3.23: Prototype of an augmented light switch: (Left) the slider displayed on the handheld device's screen; (Right) the prototype system.

Interaction and handheld AR characteristics. By using information from both finger tracking in camera images and touch detection on the object's surface, our prototype allowed two input tracking states. Indeed, it entered one tracking state when the finger was detected in mid-air and another when the finger was detected as being in contact with the physical object (Figure 3.24). Touching the object could therefore be used as a trigger. However, one limitation was that the physical object had to be within arm's reach.

While this was only a prototype, it allowed us to gather feedback on interaction using a finger in front of the camera and on the object. First, although finger recognition was quite slow, interaction was possible when the finger moved perpendicular to the cmaera's line of sight. However, moving the finger along the camera's line of sight so as to reach the object (i.e., the physical switch) while keeping the fingertip

[6]http://www.arduino.cc/
[7]https://www.vuforia.com/

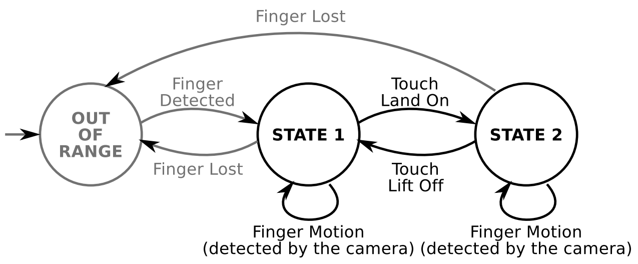


Figure 3.24: State machine of bare-finger input with touch contact detection, to allow two tracking states.

in the same position on the slider in the image plane was difficult to achieve. This made the interaction envisioned fairly awkward.

We suggest that there are different reasons behind these results. First, the latency with which the video was displayed on the screen impaired finger movements. In addition, the live camera images did not provide stereoscopic depth cues to aid depth perception over short distances. Finally, fingertip motion along a defined line (in this case, the camera's line of sight) is probably a difficult gesture to perform. However, further study is required in order to confirm these different factors.

Furthermore, in order to interact with the finger in front of the camera, the user needs to hold the handheld device with only one hand. While this is not a problem for the smart phone form factor, it is much harder with heavier and larger tablet devices. Because of the limitations identified, we did not further investigate this form of interaction. Instead, we proposed a technique that allows pointing in the object frame using touch input, while avoiding interaction on the physical object itself.

3.3.2.2 Indirect Relative Pointing

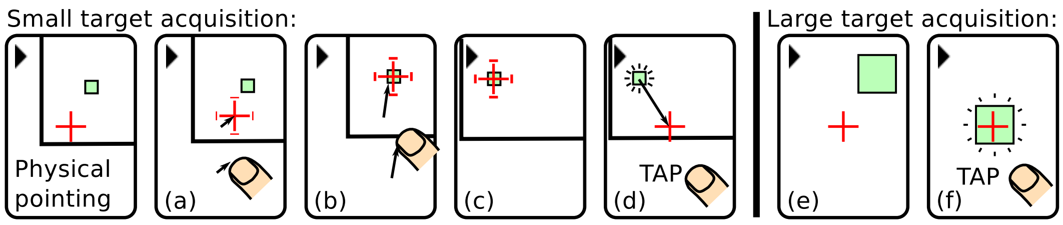


Figure 3.25: *Relative Pointing* walk-through. (a-d) Small-target acquisition with a relative cursor stabilized on the physical object; (e-f) Large-target acquisition with the crosshair technique.

Figure 3.25 provides a walk-through of the two modes of our *Relative Pointing* technique.

Scenario 1:

Physical pointing (see 3.2.3): the user points the handheld device's camera at the target so that the target appears on the screen.

(a) To mitigate instability due to hand tremor, when the user touches the screen and starts moving her/his finger on the screen, a relative pointing mode is activated.

(b) When in this mode, the cursor is no longer bound to the center of the screen. Instead, it is stabilized on the remote physical object at its current position. The user fine-tunes the cursor's position by controlling the cursor indirectly with finger movements on the screen.

(c) On finger lift-off, no particular action is performed.

(d) A position is validated with a tap on the screen. Upon validation, a short animation moves the cursor back to the center of the screen, thus exiting the relative pointing mode.

Scenario 2:

(e-f) When acquiring large enough targets, hand tremor is not a problem. In this case, the user does not need to use the relative pointing mode and can trigger target acquisition at the screen-centered cursor's position with a tap on the screen. This is similar to the screen-centered *Crosshair* technique.

Figure 3.26 illustrates the use of *Relative Pointing* to place annotations on a remote wall map.

To make relative pointing effective in a handheld AR context, we addressed three issues. First, absolute viewpoint control needs to be combined with relative cursor control. Secondly, because this technique uses indirect relative active point control, we can use a dynamic transfer function. Thirdly, we will discuss how pointing at objects with a 3D shape is supported.

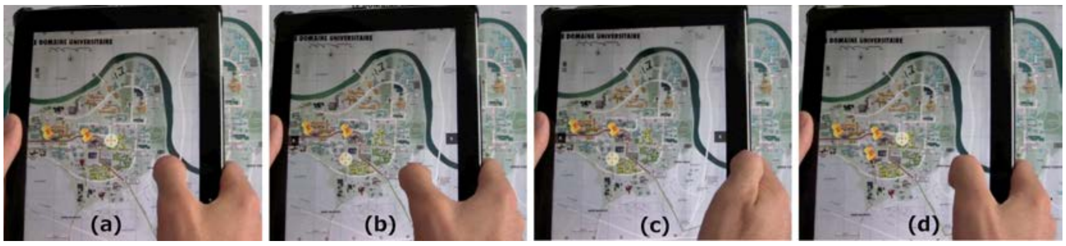


Figure 3.26: *Relative Pointing*: (a) we start with *Crosshair*; (b-c) a finger stroke activates a precise relative pointing mode; (d) a tap on the screen performs selection and restores the *Crosshair* mode. The cursor was highlighted for better picture clarity.

Combining Absolute Physical Pointing and Touch-Based Relative Pointing. Because the device's pose controls the camera's viewpoint, the target in the physical world needs to be placed in the camera's viewing frustum before it can be interacted with (i.e., physical pointing - see 3.2.3). Thus, cursor-based relative pointing needs to be combined with this form of absolute direct pointing in space. We therefore chose to extend the screen-centered *Crosshair* pointing technique, as it already uses a cursor and only relies on device position and orientation for both viewpoint control and pointing. We extended this technique with a relative pointing 'once' mode, in which the cursor is no longer fixed at the center of the screen. Instead, the finger indirectly controls the cursor's position. This mode is triggered by finger movement on the screen. Lifting the finger from the screen does not deactivate the mode. This enables finger clutching and allows the current cursor position to be checked before validation. A tap on the screen triggers validation. It is possible to cancel this relative pointing mode by pressing one of the two cancel buttons that appear on either side of the tablet's screen in relative pointing mode (these buttons are represented by a black square containing a white cross in Figure 3.26-b-c and Figure 3.27). In addition, when tracking is lost, relative pointing mode is deactivated. Finally, the cursor is bound to the screen. In cases where a change in the camera's viewpoint or a finger movement would otherwise make the cursor invisible on the screen (i.e., it would be outside the screen frame), the cursor is automatically moved so that it remains visible on the screen (Figure 3.27).

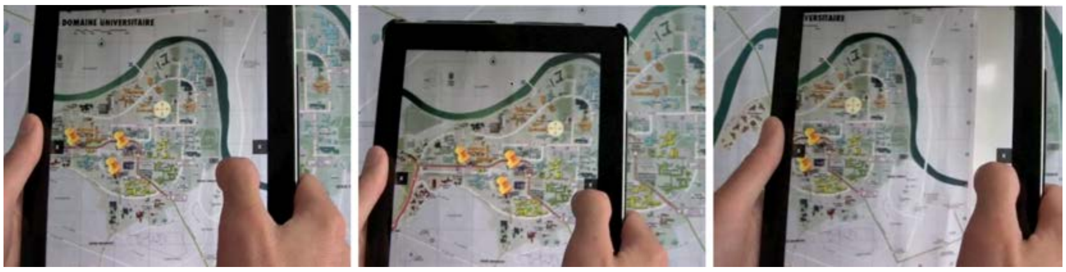


Figure 3.27: In the *Relative Pointing* mode, the cursor is stabilized on the remote physical object: moving the handheld device in space does not move the cursor on the physical object. The cursor was highlighted for better picture clarity.

Transfer function. For indirect relative input, the transfer function (Figure 3.28) that maps input movements to cursor displacement in the visual output space is of particular interest (see 3.2.1).

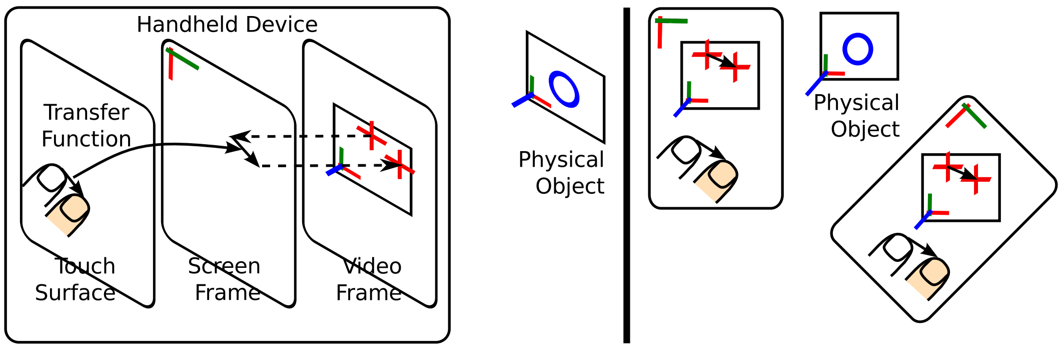


Figure 3.28: With *Relative Pointing*, (Left) the cursor is stabilized in the physical object's frame of reference and relative touch movements are applied on the screen. (Right) The effect of screen rotation on cursor position.

First, a transfer function that maps touch movements in the screen frame directly to cursor displacement in the frame of the physical object is not appropriate. In this case, the relative rotation between the device and the frame of the physical object, the distance between the device and the physical object, and the zoom factor would affect cursor displacement on the screen. Yet, the user views the physical object through the live camera images on the screen. The control loop is therefore between finger movement and cursor displacement on the screen (ratehr than in the frame of the physical object).

With *Relative Pointing*, the transfer function is instead applied in the screen frame. When finger movement input is received, the cursor position is projected from the frame of the physical object onto the screen, the cursor is displaced on the screen, and the new cursor position is projected back onto the physical object (Figure 3.28-Left). This guarantees that the behavior of the cursor on the screen remains consistent when the device is rotated (Figure 3.28-Right) and when the viewpoint or the zoom factor changes. In short, we use the physical object's frame of reference to stabilize the cursor, while the screen's frame of reference is used to apply cursor displacements.

Another question is which transfer function should be used. The transfer function enables interaction improvements such as the ability to adjust the Control-to-Display gain dynamically according to input device speed. The use of dynamic transfer functions is a default feature of mouse and touchpad input in common desktop environments (see 3.2.1). For the technique we developed and some of the experiments we held, we used the touchpad input transfer function from Mac OS X: `osx:touchpad?setting=0.875` [Casiez and Roussel, 2011]. With this configuration, the transfer function allows most of the screen to be reached with fast movements, as well as accurate positioning at lower speeds.

3D-object support. We first developed *Relative Pointing* for pointing at 2D planar physical objects such as wall maps or posters. We then added support for physical objects of different shapes. While the technique supports virtually all shapes of object, we are still using a vision-based tracking library that tracks 2D planar images. Thus, we actually implemented pointing at 3D digital objects overlaid on a 2D physical image.

The main difference between a 2D planar object, such as a wall map, and other types of object is the potential for self-occlusion. Indeed, in *Relative Pointing* mode, the cursor may actually be occluded by the object. The cursor may become occluded because of a viewpoint change. In each new camera image, we therefore check whether or not the cursor is occluded. To test for occlusion, we perform ray casting on the cursor's position on the screen and check whether the cursor is at the closest intersection point. If the cursor is occluded (i.e., it is not positioned at the closest intersection point), we move the cursor to the closest intersection point to ensure that it is always visible. A 3D model of the physical object is required to support *Relative Pointing*, with pointing precision being dependent on the model's accuracy. Figure 3.29 illustrates this process.

On way of overcoming the need for an accurate model could be to use a depth camera, so as to obtain the information about the closest intersection point from the data captured rather than from a model.

Implementation. We first developed *Relative Pointing* on iOS with the Vuforia SDK[8] tracking library and OpenGL|ES 1.1[9]. We ran it on iPad and the 4th generation of iPod. We used libpointing[10] [Casiez and Roussel, 2011] to implement the dynamic transfer function. This implementation allowed us to point at 2D planar images.

To implement pointing at any 3D shape, we incorporated *Relative Pointing* into a toolkit dedicated to handheld AR applications and into the demonstrators we devel-

[8]https://www.vuforia.com/
[9]http://www.khronos.org/opengles/1_X/
[10]http://www.libpointing.org/

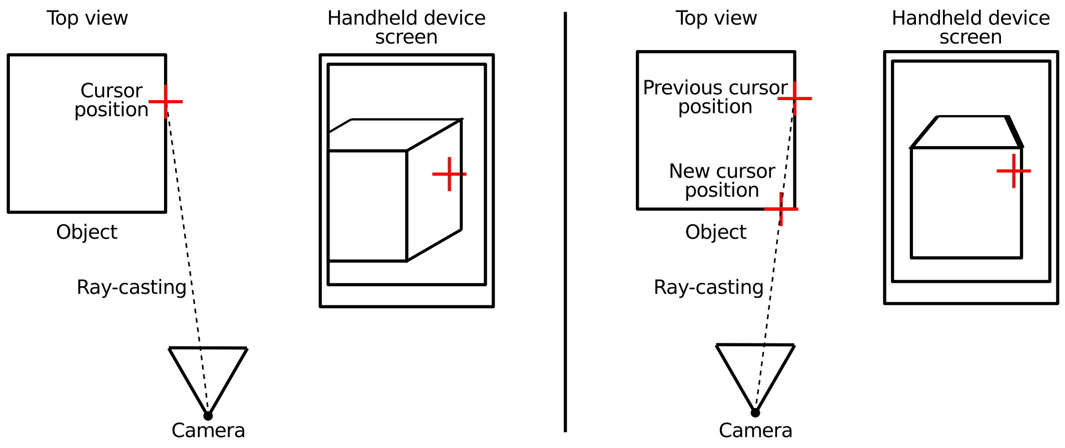


Figure 3.29: A *Relative Pointing* process to support pointing at objects with a 3D shape. Each new camera image is checked for cursor occlusion. If occlusion is detected, the cursor is moved to a non-occluded position on the object, while remaining in the same projected position on the screen. (Left) The cursor is not occluded. (Right) The cursor is occluded, so it is moved to the ray casting's closest intersection point.

oped as part of the AMIE[11] project, which we will present in Chapter 5. This toolkit provides a scene graph that support 3D objects and picking in the 3D augmented scene using ray casting. These are the building blocks required for *Relative Pointing* to work on 3D shapes.

Interaction and handheld AR characteristics. *Relative Pointing* extends *Crosshair* in the same way that *Shift&Freeze* extends direct touch. It therefore requires overhead interaction only when precision matters.

The precise mode uses fully indirect relative cursor control with a dynamic transfer function geared primarily towards reducing the required workspace on the touch surface. This allows the cursor to be controlled with a thumb when holding the tablet with both hands. In addition, tapping to trigger validation and to exit the precise mode eliminates the impace of finger movement upon lift-off (as it is the case for *Shift&Freeze*) (see 3.2.2).

In its basic configuration, *Relative Pointing* only offers one interactive tracking state. In order to provide a second tracking state, an extra mode is required. We will provide an example of such interaction in the next section (see 3.4, where we use a button to change the mode).

Within our design space (see 2), the cursor is registered as *Augmentation* content on the remote physical object. Thus, the cursor's position relative to the *Augmentation* and the *Representation of the physical world* is not impaired by *Viewpoint* instability. This is intended to improve precision without relying on video freezing. Furthermore, the cursor's position is impaired by registration errors, as with *Augmentation*. As a result, in the event of registration jitter, the cursor's position will still share the same frame of reference as the *Augmentation*.

As opposed to other freeze-frame techniques (such as *AR TapTap* and *Shift&Freeze*), *Relative Pointing* does not modify the *Viewpoint* spatial relation.

[11]http://amie.imag.fr/

While *Relative Pointing* maintains a live view of the physical surroundings on the screen, it also prevents the user to move to a more comfortable position. Nevertheless, as opposed to *Crosshair* whose cursor is bound to the center of the screen, the precise mode of *Relative Pointing* can be used while the frame is frozen. Thus *Relative Pointing* is compatible with a modal freeze frame (triggered by a button press for example), which allows the user to interact in a comfortable position.

In the next chapter (see Chapter 4) we will describe experiments geared towards evaluating this technique by modifying registration jitter under normal and altered conditions.

3.4 Beyond pointing

In this chapter, we proposed several handheld AR pointing techniques, most notably *Relative Pointing*. We will now present a number of applications where *Relative Pointing* can be used for different pointing requirements, but also for tasks other than pointing.

3.4.1 Annotation authoring and selection

The proposed pointing techniques (i.e., *AR TapTap*, *Shift&Freeze* and *Relative Pointing* - see 3.3) make it possible to point at both physical and digital targets. First, these techniques can be used to position marks on physical objects, so as to add new annotations anchored to the physical surroundings. This is the examples we used to illustrate these different pointing techniques. However, it is also possible to use these pointing techniques to select existing annotations. In such cases, improving pointing precision is particularly useful for dense environments where target-aware techniques that improve pointing by enlarging known targets are less effective.

Figure 3.30 illustrates the use of *Relative Pointing* to point at both a physical wall map and a digital augmentation.

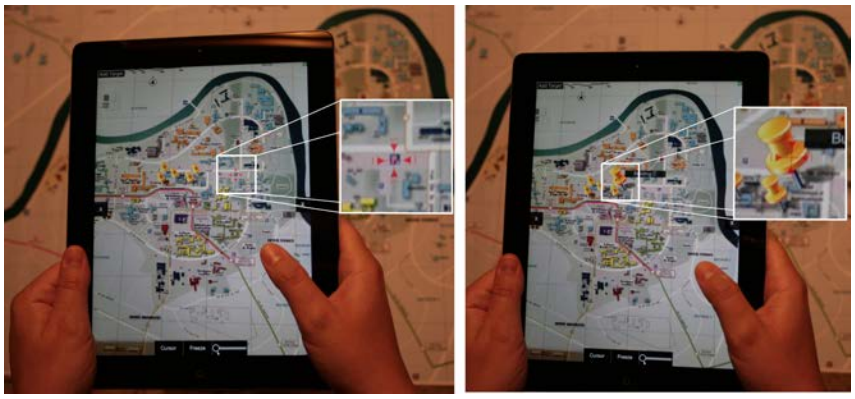


Figure 3.30: (Left) Pointing at a physical target: a parking lot. (Right) Pointing at a digital target: a mark.

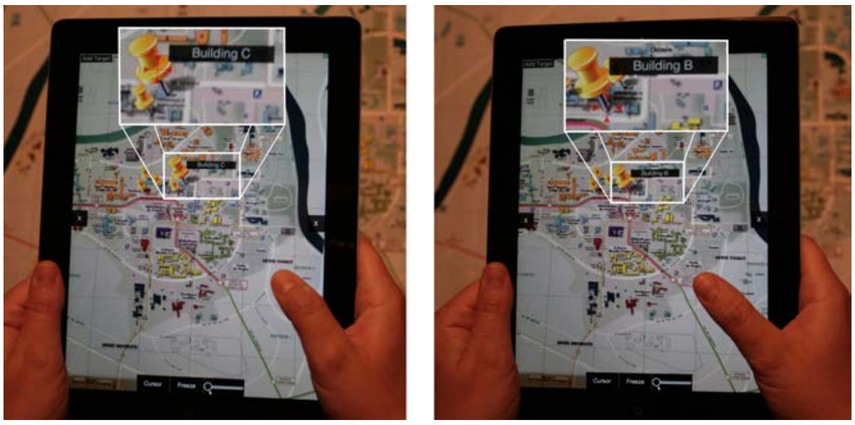


Figure 3.31: Example of semantic zooming: Text information is only displayed while hovering the digital mark.

3.4.2 Semantic zooming

With cursor-based pointing techniques such as *Crosshair* or *Relative Pointing*, it is possible to provide hovering-based interaction. Indeed, the cursor specifies an active point on the screen and, therefore, an active 3D point in the augmented scene at all times.

This is not the case for techniques that build on direct touch, such as *AR TapTap* and *Shift&Freeze*. In this case, when an active point is specified, the user is engaged in interaction. This is a limitation of touch-based interaction (see 3.2.2).

We used hovering to provide additional information when the cursor is over a target. When the cursor is over a digital mark, the mark is enlarged and the text associated with this mark is displayed. Figure 3.31 illustrates this interaction.

While such interaction is possible with *Crosshair*, *Relative Pointing* is more suitable especially for small targets. Indeed, due to hand tremor, it is hard to stabilize the screen-centered *Crosshair* over a small target. This requires the user to keep the handheld device as steady as possible.

With *Relative Pointing*, however, it is possible to place the cursor over a target and to keep it stationary. In this case, the user only needs to keep the target within the camera's viewing frustum to maintain the cursor over the target. This is much easier than keeping the screen-centered *Crosshair* over the target.

3.4.3 Drawing

With *Relative Pointing*, it is also possible to use both thumbs when holding the tablet with both hands. In this case, one thumb operates the cursor while the other can control a button (see Figure 3.32-Left).

This configuration also supports forms of interaction that require two interactive tracking states. To illustrate this, we developed a demonstration that allows drawing on remote 2D physical images (Figure 3.32-Right).

The user controls the drawing with a button and can move the cursor either like a screen-centered *Crosshair* or a relative cursor.

Another example involving the use of two interactive tracking states would be one where area selection could be performed in the augmented scene, at least on 2D planar objects.

3.5 Conclusion

In this chapter, we began by reviewing pointing assistance techniques for touch-based handheld devices and for handheld AR applications, in particular. Among the pointing techniques available, we are particularly interested in target-agnostic techniques, as these techniques can be used in a wider context. The speed-dependent dynamic transfer function is a widely used target-agnostic technique. It allows targets to be enlared at low speeds, while reducing movement amplitude at higher speeds. However, such an approach is only possible with indirect relative active point control.

Finger occlusion and the ambiguous selection point impair touch screen input on handheld devices. Thwe degree to which this is detrimental to pointing precision depends on the phase of the touch gesture. Indeed, finger land-on is imprecise, while finger motion is precise as long as a cursor is displayed. Moreover, finger lift-off can also be imprecise. Finally, on smart phones certain areas of the screen are harder to reach than otehrs, while on handheld tablets there is a trade-off between holding the device and accessing the screen. To address these issues, touch-based target-agnostic pointing assistance techniques follow two strategies: using a cursor and zooming.

In addition, the context of handheld AR brings extra constraints, because on-screen content is related to the physical surroundings (see Chapter 2). Handheld AR pointing is supported by three main modalities: touch screen, device position and orientation, and bare fingers in front of the camera. Handheld AR interaction assistance techniques are based primarily on zooming, viewpoint stabilization (mainly freeze-frame techniques) and input stabilization.

Based on both this review of existing target-agnostic pointing techniques and on our design space (see Chapter 2), we identified two complementary approaches so as to further study pointing techniques for handheld AR applications. The first approach is to combine touch-based pointing techniques with viewpoint stabilization through video freezing. We developed and discussed two techniques that follow this approach: *AR TapTap* and *Shift&Freeze*. *AR TapTap* extends *TapTap* by adding

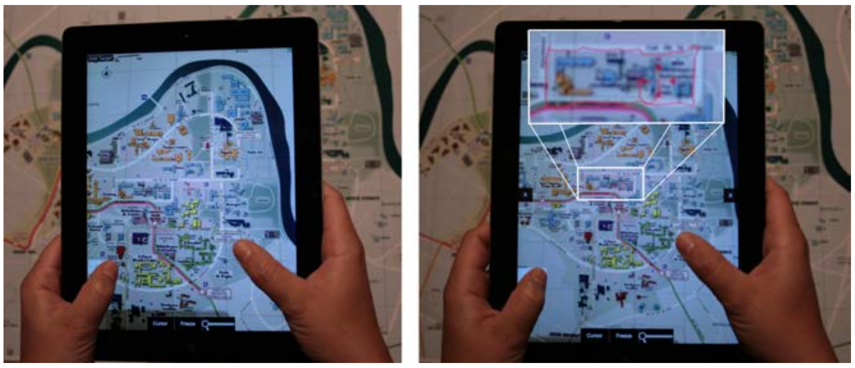


Figure 3.32: (Left) Illustration of the two-hand hold with two thumbs on the screen. (Right) Drawing with *Relative Pointing*.

video freezing and allowing for a spatial multiplexing of a frozen viewpoint and live camera images. However, informal tests were inconclusive regarding the benefits of this multiplexing. *Shift&Freeze* is designed to improve on direct touch. It solves precision issues by freezing the video and using the callout from *Shift*.

The second approach is to explore interaction in the frame of reference of the physical object or its representation on the handheld device's screen. We studied interaction based on the positioning of fingers on the physical object, which were then viewed through the camera. This form interaction is impaired by different factors such as the need to hold the device with only one hand and the fact that finger position is controlled via the camera. We also designed *Relative Pointing*, which extends the screen-centered *Crosshair* technique with a precise 'once' mode in which the cursor is stabilized on the remote physical object. When in this mode, the cursor is controlled with indirect relative finger strokes. We used a dynamic transfer function to enable users to reach most of the screen without clutching, while still allowing precise positioning.

A future direction of handheld AR pointing would be to consider the different types of feedback that the cursor could provide. First, feedback on pointing precision could be provided according to the amount of hand tremor detected by the system. For example, with *Crosshair* the cursor could become an area cursor whose size is controlled by the amount of hand tremor detected by the system. For *Relative Pointing*, the size of such an area cursor could be controlled by the amount of registration jitter. In addition, the cursor could provide certain information about the depth and shape of the object it is currently pointing at. For example, Gómez Jáuregui *et al.* [Gómez Jáuregui et al., 2012] proposed two cursors, *Hand Avatar* and *Torch* cursor, which provide feedback about depth and surface orientation at a specific point in a Virtual environment.

Another direction would be to consider target awareness, as proposed by the *Snap Target* technique [Baldauf and Fröhlich, 2014]. It would also be interesting to explore pointing at any position in space, rather than just on the surface of objects. This would involve exploring pointing in mid-air and inside/behind objects with handheld AR applications.

In the next chapter, we will describe the controlled experiments we performed based on two of the proposed techniques, namely *Shift&Freeze* and *Relative Pointing*. Indeed, these two techniques are the most promising with regards to (1) ease of use, since they build on existing baseline techniques, and (2) precision, since they provide a precise mode.

Chapter 4

Experimental evaluations

Experimental evaluations of Handheld AR Pointing Techniques

Chapter Content

4.1 Introduction

Having designed the pointing techniques described in Chapter 3, we ran experiments to compare some of the proposed techniques with existing techniques. Our goal was to assess the strengths and weaknesses of the proposed techniques under different conditions.

No experiments were conducted with *AR TapTap* (presented in section 3.3.1.1, Figure 4.1-a). Indeed, with this technique pointing tasks require two taps. While this improves accuracy for small targets, it increases the completion time for targets of all sizes. This limitation is inherited from *TapTap* [Roudaut et al., 2008]. Furthermore, the multiplexing of video freezing and live camera images was not a convincing way to alleviate the spatial discontinuity that occurs when leaving freeze-frame mode. Finally, *Shift&Freeze* (presented in section 3.3.1.2, Figure 4.1-b) provides a similar solution. It combines freeze-frame and pointing assistance techniques for touch-based handheld devices. Compared to *AR TapTap*, *Shift&Freeze* allows direct-touch pointing for large targets, as pointing assistance is available on demand. It also scales up better to a tablet's form factor, as *TapTap* was primarily designed for one-handed interaction on smart phones.

It should also be noted that we did not hold experiments based on finger interaction via the video and on the physical object (Figure 4.1-c). Similarly, this solution was not convincing when seeking to perform accurate pointing and would not scale up to a tablet form factor, because the user would have to hold the tablet with one hand while interacting with the other hand in mid-air and on the object.

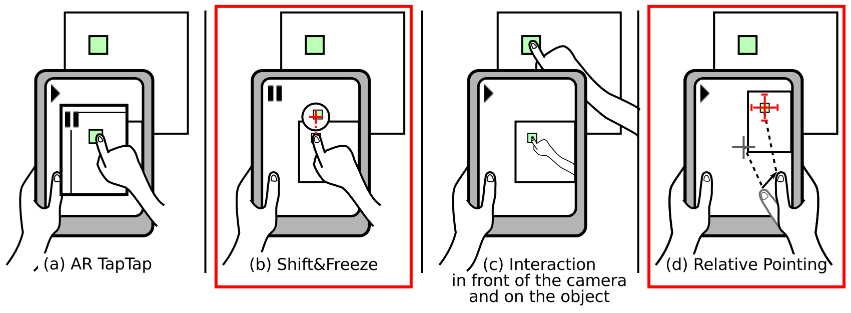


Figure 4.1: Techniques proposed in chapter 3. From left to right: *AR TapTap*, *Shift&Freeze*, interaction in front of the camera and on the object, and *Relative Pointing*. The experiments presented in this chapter are related to the two techniques highlighted: *Shift&Freeze* (b) and *Relative Pointing* (d).

To sum up, our experiments focused on two of our techniques *Shift&Freeze* (see 3.3.1.2, Figure 4.1-b) and *Relative Pointing* (see 3.3.2.2, Figure 4.1-d). Because these techniques are geared towards improving pointing precision, we were particularly interested in the accuracy of small-target acquisition.

Thus, we ran two sets of experiments.

The main goal of the first set of experiments was to compare *Shift&Freeze* and *Relative Pointing* and their baseline techniques, *Direct Touch* and *Crosshair*, respectively. First we will describe a controlled experiment to compare the performance of the four techniques based on an abstract pointing task with small targets. We will then present a second experiment held in a less controlled environment and based on a more realistic task that aimed at collecting user feedback and consolidating the results of the first experiment.

The main goal of the second set of experiments was to explore the effect of registration jitter on *Relative Pointing*. As discussed in the design space chapter (see Chapter 2), the frames of reference of handheld AR's on-screen components (i.e., the *Representation of the physical world* and the *Augmentation*) and the physical surroundings are spatially related (Figure 4.2-Top). These spatial relations (i.e., *Viewpoint* and *Registration*) are impaired by natural hand tremor and registration errors,respectively. This impairs the stability of on-screen content and can therefore be detrimental to interaction. Evaluating the effect of hand tremor on a pointing technique is straightforward. Evaluating the effect of registration errors on a pointing technique requires the camera position and orientation estimated by the underlying tracking system to be altered.

Thus, we compared the two *Relative Pointing* modes: *Crosshair* and the *Relative Pointing*'s precise mode with and without artificial registration jitter. Under registration error conditions, pointing at physical targets is not the same as pointing at digital targets overlaid on a physical object. Indeed, the representation of physical targets on the screen is not impaired by registration jitter, unlike with digital targets overlaid on a physical object (Figure 4.2). We therefore held two experiments to evaluate the effect of registration jitter on *Relative Pointing*. The first controlled experiment focused on pointing at digital targets while the second focused on pointing at physical targets.

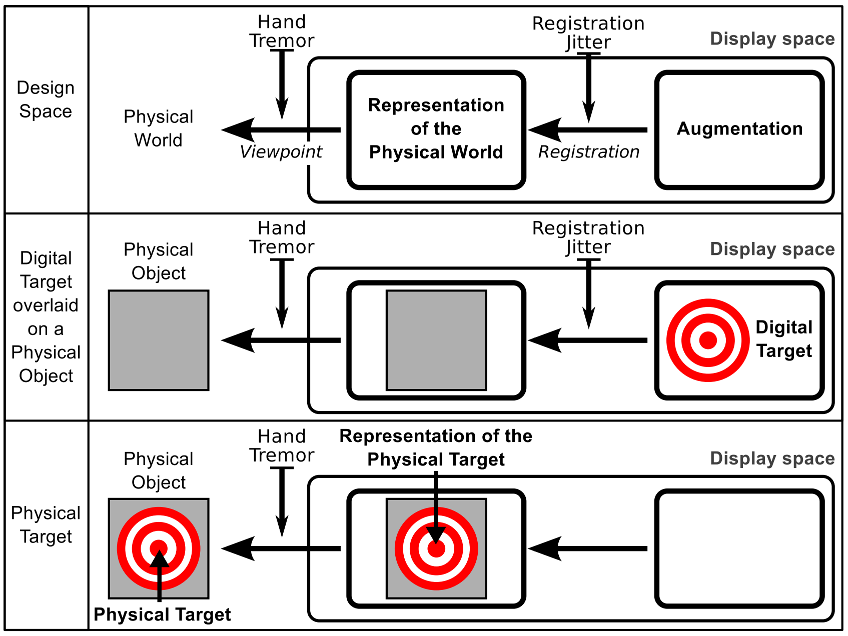


Figure 4.2: Representation of physical targets and digital targets based on the design space presented in chapter 2.

The first set of experiments was published at the INTERACT 2013 conference [Vincent et al., 2013c], while the first experiment of the second set was published at the French IHM 2013 conference [Vincent et al., 2013a].

For all four experiments, we developed ad hoc applications using the OpenGL|ES 1.1[1] rendering back-end andthe Vuforia SDK[2] vision-based tracking library. Statistical analyses were performed with R[3].

4.2 Experiments: Comparison of pointing techniques

Our aim was to compare the techniques we proposed with their baselines, so as to evaluate their benefits in terms of precision. In the context of handheld AR, the precision with which physical targets can be pointed at is subject to hand tremor, which impairs the stability of the representation of the physical world on the handheld device's screen (see Chapter 2, Figure 4.2-Top). The purpose of the experiments we will now present was therefore to evaluate the extent to which the different techniques are affected by hand tremor. The task imposed was to point at physical targets (Figure 4.2-Bottom).

We held a controlled experiment to evaluate the techniques based on the acquisition of small physical targets. We also held an experiment in a less controlled environment that involved performing a more realistic task (i.e., placing marks on a wall map). The goal of this second experiment was to collect user feedback and assess whether the resultsof the first experiment remained valid in another context.

In both experiments, we evaluated and compared the following four handheld AR pointing techniques (two baseline techniques and two proposed techniques, Figure 4.3):

- *Direct Touch*: Absolute pointing on the screen with validation at the press of a finger
- *Crosshair*: Screen-centered crosshair pointing with validation at the press of a finger anywhere on the screen;
- *Shift&Freeze* as described in section 3.3.1.2. Short recap of the technique: Vogel and Baudisch's *Shift* technique [Vogel and Baudisch, 2007] was adapted to the context of handheld AR by adding video freezing while using a precise-pointing quasi-mode. With this technique, the user can either use unaided direct touch pointing or escalate to a precise-pointing quasi-mode by touching the screen for 300ms. When escalation takes place, the camera image is frozen. When escalated, *Shift&Freeze* displays a 44mm wide circular callout containing a copy of the screen area occluded by the finger and placed 22mm above the initial touch position. The callout also shows a cursor representing the active point of the finger. The pointer's position in the callout can be adjusted by moving the finger on the screen (1:1 Control-to-Display ratio) and the validation takes place when the finger is lifted from the screen (finger lift-off). When necessary, the callout position is adjusted so that the distance between the callout center and the callout pointer is no more than 11mm. Finger lift-off triggers validation, closes the callout and returns the screen to the live image from the camera.
- *Relative Pointing* as described in section 3.3.2.2. Short recap of the technique: This technique adds a precise-pointing mode to the screen-centered

[1]http://www.khronos.org/opengles/1_X/
[2]https://www.vuforia.com/
[3]http://www.R-project.org

Crosshair technique. Users can either use the *Crosshair* technique or escalate to the precise mode by moving their finger on the screen instead of performing a tap. When escalated, the cursor is no longer bound to the center of the screen. Instead, the cursor is stabilized in the physical object's frame of reference and its position is controlled by indirect relative touch strokes. Validation is performed with a tap anywhere on the screen, thus making finger clutching possible. The transfer function that converts finger displacement into cursor displacement is applied in the screen frame. For the first set of experiments, we used the dynamic transfer function for trackpads in Mac OS X (`osx:touchpad?setting=0.875` [Casiez and Roussel, 2011]). This transfer function allows most of the screen of a handheld tablet to be reached with rapid movements and without clutching, while enabling accurate positioning at low speed. Unlike in the case of *Relative Pointing*, as presented in section 3.3.2.2, we did not include a cancel button for these experiments.

All the cursor-based techniques (i.e., *Crosshair*, *Shift&Freeze* with callout and *Relative Pointing*) use the same red square cross cursor, which is $7.7mm$ wide with a stroke width of $0.2mm$.

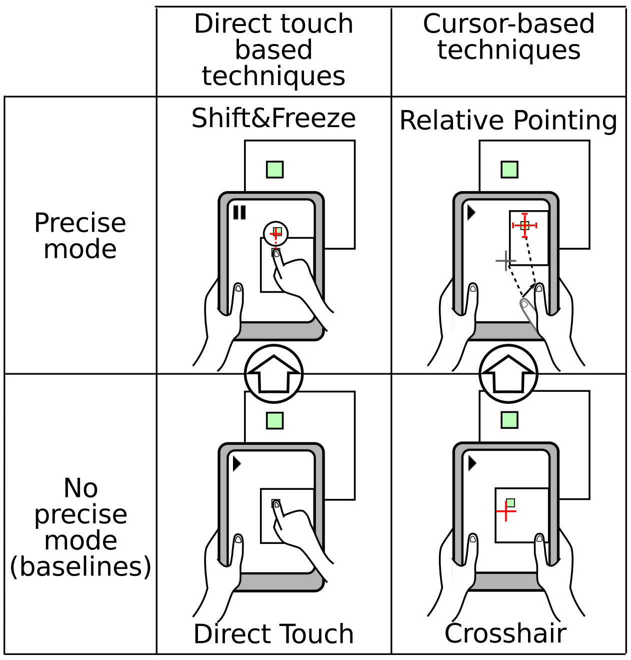


Figure 4.3: The four techniques evaluated: Two baseline techniques (*Direct Touch* and *Crosshair*) and two proposed techniques (*Shift&Freeze* and *Relative Pointing*).

We will now describe the two experiments we performed and their results.

4.2.1 Experiment 1: Small-target acquisition

For this experiment, the task involved pointing at small physical targets. The precision with which physical targets can be pointed at is subject to hand tremor, which impairs the stability of the physical object's representation on the screen (Figure 4.2-Bottom). Our aim was to compare pointing accuracy when pointing at small targets using the two proposed techniques (i.e., *Shift&Freeze* and *Relative Pointing*) and the two baseline techniques on which they expand (i.e., *Direct Touch* and *Crosshair*, respectively). We also wanted to observe whether participants would change the way they use the precise modes depending on target size.

This was a controlled experiment based on an abstract pointing task.

For this experiment, we formulated the following hypotheses:

- H1: *Relative Pointing* and *Shift&Freeze* are more accurate than *Direct Touch* and *Crosshair* but they take longer to operate for small targets.
- H2: *Crosshair* is more accurate than *Direct Touch*. While both techniques are impaired by hand jitter, *Crosshair* is not hindered by finger occlusion.
- H3: *Relative Pointing* and *Shift&Freeze* offer similar accuracy. Both techniques overcome limitations inherent to touch input and hand jitter.

4.2.1.1 Procedure and design

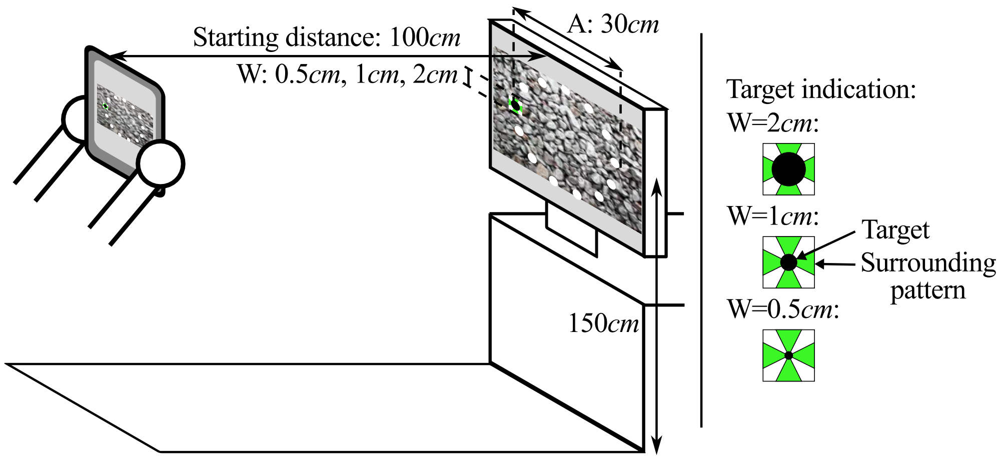


Figure 4.4: (Left) Set-up of experiment 1. (Right) Targets with surrounding pattern.

This experiment was carried out based on the cyclical multi-direction pointing task paradigm of ISO9241-9 [Soukoreff and MacKenzie, 2004]. We used thirteen targets arranged in a circle on a remote screen. Since the handheld tablet application used computer vision to track the device's position and orientation, the targets were overlaid on a background image (image of stones in Figure 4.4). One at a time, the targets were highlighted in black on the remote screen: each target had to be selected by pointing at it on the live camera images displayed on the tablet's screen. In order to ensure that the target highlighted would be clearly visible regardless of its width, it was surrounded by a 3cm wide white and green square pattern (Figure 4.4-Right). The targets always appeared in the same order: top target first, followed by the target opposite and slightly clockwise from the one selected previously, and so on. One block comprised thirteen target selections plus the selection of the first target (i.e., the top target), which was used to indicate the start of each block. The subjects were instructed to hold the tablet in portrait mode, to select the highlighted target as quickly and accurately as possible, and to rest between blocks.

We wished to maintain a consistent distance between the remote screen and the handheld tablet for each participant and block. To do so, before each block participants had to place the handheld tablet 1 meter$\pm 5cm$ away from the remote screen by following the instructions displayed on the tablet screen. These instructions were hidden as soon as subjects started on the block, to avoid disturbing them during the experiment. From this distance, the background image and the targets remained in the camera's viewing frustum. This prevented image tracking failures and avoided the problem of targets being off-screen and needing to be located by users first. In addition, from this distance participants were able to move the device closer or further away without significantly changing the size of the targets on the handheld device's screen.

We used a single movement Amplitude (A) of $30cm$ and three target Widths (W: $0.5cm$, $1cm$ and $2cm$) as shown in Figure 4.4-Right. The Index of Difficulty ($ID = log_2(A/W + 1)$) of these tasks was 5.9 bits for W=$0.5cm$, 4.9 bits for W=$1cm$ and 4.0 bits for W=$2cm$. From a distance of $1m \pm 5cm$, A was within a range of 6.9-7.8cm on the screen of the handheld tablet, W=$0.5cm$ was around $0.1cm$ on the screen, W=$1cm$ was around $0.2cm$ and W=$2cm$ was around $0.5cm$.

The choice of such small target widths was motivated by our goals to observe both (1) the limit of accuracy of the different techniques and (2) how participants would use the precise mode with small targets. Indeed, we designed the techniques with the precise mode (i.e., *Shift&Freeze* and *Relative Pointing*) primarily to improve accuracy over their baseline techniques. Thus we did not evaluate the techniques for large targets in this experiment. This choice also allowed for a reasonable experiment's duration. Yet, this choice resulted in high error rates.

The presentation orders of the four techniques and three Widths were counterbalanced using Latin squares. Each condition was presented three times including once for practice. Thus, the within-subject experimental design was as follows:

4 *Techniques* x 3 *Widths* x 2 *Blocks* x 13 *Selections* = 312 acquisitions per subject; and

4 *Techniques* x 3 *Widths* x 13 *Training Selections* = 156 acquisitions per subject for practice (not including the selection of the first targets).

4.2.1.2 Apparatus and participants

Apparatus. The experiment was conducted on an iPad2 (iOS version 5.1.1, weight: $601g$, screen resolution 1024x768 pixels ($132dpi$)). The system's touch input resolution is the same as the screen's. An ad hoc application was developed with Vuforia SDK[4] version 1.5.9 and libpointing[5] [Casiez and Roussel, 2011]. The application was running at around 30 frames per second. The size of the camera images was 480x640 pixels and the camera images were displayed in full-screen mode. The targets were displayed on a 27" Apple Thunderbolt display with a resolution of 2560x1440 pixels ($109dpi$). The screen was placed vertically so that its center was $1.5m$ above the ground. The ad-hoc application was developed to control the target widths and

[4]https://www.vuforia.com/
[5]http://www.libpointing.org/

highlight the targets on the remote screen. This application was controlled by the handheld tablet application via a Wi-Fi connection.

Participants. Twelve unpaid volunteers (4 female, 8 male; 1 left-handed), ranging in age from 22 to 41 years (mean of 30 years), were recruited from our institution. All the participants had previous experience with touch-based handheld devices (nine on a daily basis) and ten had used a touch-based tablet before. Some of the participants were previously involved in the experiment described in paragraph 4.2.2 below. Indeed, the second experiment was been conducted prior to this one.

4.2.1.3 Results

Of the 3,744 observed target acquisitions, we removed 33 outliers (where the error distance was further than three standard deviations away from the average, as proposed in [Soukoreff and MacKenzie, 2004]). D The average distance from the screen when selections were performed was $1.02m$ (1st quartile: $0.99m$, 3rd quartile: $1.05m$, range: [$0.90m$-$1.18m$]). This indicates that our experimental set-up, which required participants to place the handheld tablet $1m{\pm}5cm$ from the screen before starting each block, was sufficient to confine the distance between the handheld tablet and the remote screen within a fairly small range of $28cm$.

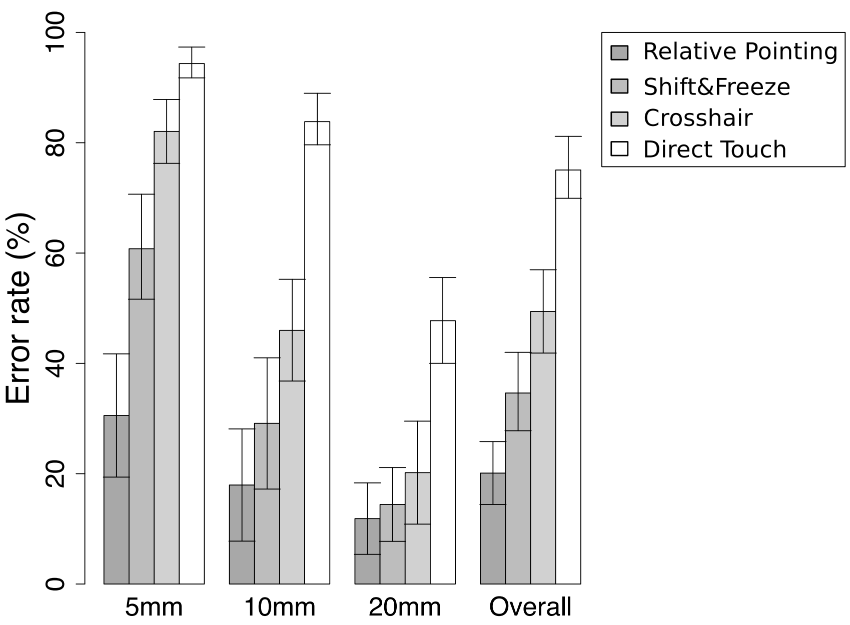


Figure 4.5: Barplots of error rates (%) and 95% confidence intervals of error rates aggregated by block for each participant for each *Technique* and each *Width*.

Errors. A Pearson's Chi-squared independence test between success of target acquisition and the four *Techniques* showed a significant dependence (χ^2=616.0356*[6]). The overall error rate was 44.6%. This high error rate can be explained by the choice

[6]* indicates p<.0001.

of fairly small target *Widths*. The lowest error rate among the three target *Widths* was for *Relative Pointing* (20.1%), followed by *Shift&Freeze* (34.6%) and *Crosshair* (49.4%), with the highest error rate being observed for *Direct Touch* (75.0%) (Figure 4.5).

We performed a 4 x 3 (*Technique* x *Width*) within-subject analysis of variance on error rate aggregated by participants. The *Technique* ($F_{3,143}$=50.835*) and *Width* ($F_{2,143}$=57.286*) main effects as well as the *Technique* x *Width* interaction ($F_{6,143}$=3.397; $p<.01$) were found to be significant. A post-hoc Tukey multiple mean comparison found significant differences for all the comparisons. Differences between *Relative Pointing* and *Shift&Freeze* and between *Shift&Freeze* and *Crosshair* were found to be significant with $p<.05$. All other differences were found to be significant with $p<.0001$.

We also performed a 4 x 3 (*Technique* x *Width*) within-subject analysis of variance on the median distance between target center and selection point aggregated by participant. Significant main effects were found for *Technique* ($F_{3,143}$=42.605*) and *Width* though with $p<.05$ ($F_{2,143}$=4.389). *Technique* x *Width* interaction was not significant. A post-hoc Tukey multiple mean comparison found significant differences for all comparisons except between *Relative Pointing* and *Shift&Freeze* and between *Shift&Freeze* and *Crosshair* (with $p<.0001$ for all significant comparisons except for the *Relative Pointing*-*Crosshair* comparison, where $p<.01$).

Duration. The overall median for selection duration was 2.1 seconds, while the medians for target acquisition duration for each *Technique* were 2.7 seconds for *Relative Pointing*, 2.5 seconds for *Shift&Freeze*, 2.2 seconds for *Crosshair* and 1.0 second for *Direct Touch* (Figure 4.6).

We performed a 4 x 3 (*Technique* x *Width*) within-subject analysis of variance on the median of target acquisition durations aggregated by participant. Significant main effects were found for *Technique* ($F_{3,143}$=67.781*) and *Width* ($F_{2,143}$=17.478*). The effect of the *Technique* x *Width* interaction was also significant, though with $p<.05$ ($F_{6,143}$=2.827). A post-hoc Tukey multiple means comparison found significant differences for all but two comparisons (with at least $p<.01$). Again, the differences between *Relative Pointing* and *Shift&Freeze* and between *Shift&Freeze* and *Crosshair* were not significant.

Precise mode. Participants used the precise modes of both *Shift&Freeze* and *Relative Pointing*. They used the precise mode almost all of the time when target *Width* was $0.5cm$ and $1cm$, and 60% of the time when *Width* was $2cm$. Table 4.1 sums up precise-mode usage percentages for the two techniques.

We looked more closely at individual participant behavior when acquiring the larger targets (i.e., *Width*=$2cm$). We found that all but one of the participants behaved almost exacly the same when it came to using the precise mode. Indeed, they either used the precise mode almost all of the time or almost never. In the case of *Shift&Freeze*, for the 26 targets with *Width*=$2cm$ (i.e., the two blocks of 13 targets with *Width*=$2cm$), 4 participants almost never used the precise mode except for a maximum of one target, while 8 participants used it for at least 24 targets. For *Relative Pointing*, 5 participants never used the precise mode and 6 participants

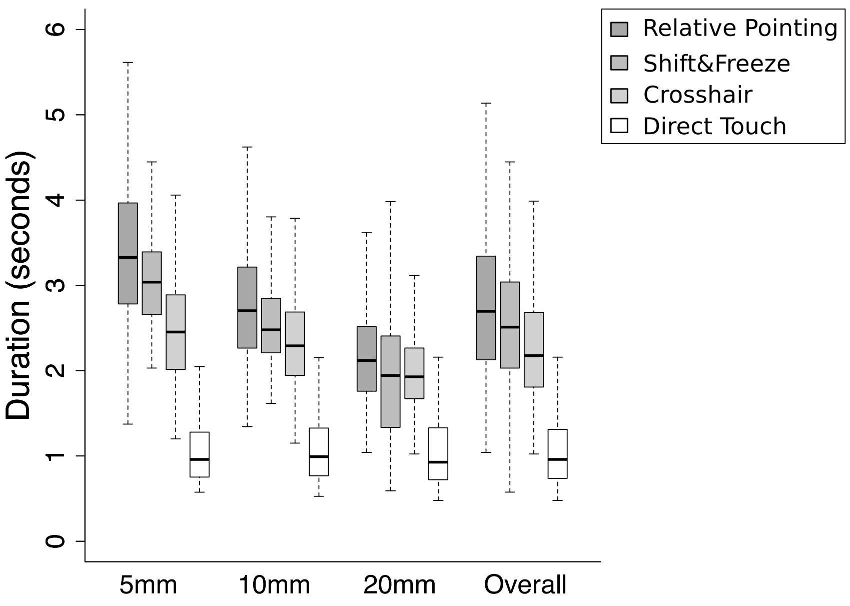


Figure 4.6: Boxplots of selection durations (seconds) for each *Technique* and each *Width*.

used it for at least 24 targets. One participant used the *Relative Pointing*'s precise mode for 10 targets, but did not use it for the other 14 targets.

This shows that most of the participants behaved the same throughout the two blocks of targets of *Width*=2*cm*. Yet, some participants (4/12 for *Shift&Freeze* and 5/12 for *Relative Pointing*) almost never used the precise mode for the largest targets, but used it for the smaller targets. This indicates that they adjusted their behavior according to the size of the targets.

Table 4.1: Percentage of usage of the precise modes of *Shift&Freeze* and *Relative Pointing* according to target size.

Technique	Overall	W=0.5cm	W=1cm	W=2cm
Overall	81%	95%	87%	60%
Shift&Freeze	83%	91%	91%	66%
Relative Pt.	78%	99%	83%	52%

4.2.1.4 Discussion

The tasks chosen were quite difficult to perform, which resulted in high error rates for all the techniques. Nevertheless, it can be observed that the different techniques offer different trade-offs between duration and error rate (Figure 4.5 and Figure 4.6).

We expected *Relative Pointing* and *Shift&Freeze* to be more accurate but longer to operate than *Direct Touch* and *Crosshair* (H1). We also expected *Crosshair* to be more accurate than *Direct Touch* (H2), and *Relative Pointing* and *Shift&Freeze* to offer similar accuracy (H3).

Relative Pointing was significantly more accurate and longer to operate than the two baseline techniques (i.e., *Direct Touch* and *Crosshair*), but this was not the case for *Shift&Freeze*. Indeed, *Shift&Freeze* did not differ significant from *Crosshair*. These results partly support hypothesis H1.

Crosshair was significantly more accurate than *Direct Touch*. In fact, *Direct Touch* is not suited for such small target widths, as indicated by the high error rates recorded and the lack of difference between the durations recorded for all target widths. This supports hypothesis H2.

The non-significant difference between the durations recorded for *Relative Pointing* and *Shift&Freeze* suggests that both techniques offer similar performance. However, there was a significant difference between the error rate of *Shift&Freeze* and that of *Relative Pointing*. This may be explained by the fact that, with *Shift&Freeze*, validation takes place on finger-lift. Lifting the finger may lead to inaccuracy (see 3.2.2). In the case of *Relative Pointing*, finger lift-off does not trigger validation, but only stops cursor displacement. This allows the user to check the position of the cursor. Finger lift-off would need to be studied further to better understand this difference. This partly supports hypothesis H3.

While participants held the tablet with both hands when using *Relative Pointing* and *Crosshair*, they adopted different strategies with *Direct Touch* and *Shift&Freeze*. For *Direct Touch* and *Shift&Freeze*, participants used two different strategies: (1) holding the tablet with one hand and interacting using a finger on the other hand (9/12 for *Direct Touch* and 6/12 for *Shift&Freeze*), and (2) holding the tablet with both hands and interacting with their thumb (3/12 for *Direct Touch* and 6/12 for *Shift&Freeze*). This highlights the trade-off that takes place between holding the tablet and interacting on the screen when using *Direct Touch*-based techniques. Unlike in the experiment reported in [Wagner et al., 2012] on the different ways of holding the handheld tablet, here some of the participants did hold the tablet with both hands and interacted only with their thumb.

One flaw in this experiment was that the choice of selection mode for *Shift&Freeze* and *Relative Pointing* was left to the participants. This resulted in different strategies being employer, as some users always used the precise mode while others used the mode according to the difficulty of the task. This was particularly true for the largest target W= $2cm$. Ultimately, though, our goal was to evaluate two techniques featuring two modes.

4.2.2 Experiment 2: User experience

In this second experiment, the task again involved pointing at physical targets. However, the context was more realistic: digital marks were placed on a wall map (Figure 4.7). Moreover, the tasks were performed in a less controlled way than in the previous experiment. The goal of this second experiment therefore complemented that to the previous one. Indeed, its objective was to collect user feedback and to assess whether the results produced by the first experiment were still valid in another context. We also wanted to observe whether participants would use the precise mode differently depending on the target and how they would adjust the distance they stood from the targets depending on the target size and the technique. We expected participants to maintain a greater distance when pointing for larger targets and when using techniques featuring a precise mode.

For this second experiment, we formulated the following hypotheses:

- H1: *Relative Pointing* and *Shift&Freeze* are preferred over *Crosshair* and *Direct Touch*. This is due to the additional precise mode offered by our two techniques. Moreover, this precise mode does not prevent the use of *Crosshair* or *Direct Touch* as basic modes.
- H2: In a tablet form factor, indirect cursor-based techniques are preferred over direct pointing techniques. Thus, *Relative Pointing* is preferred over *Shift&Freeze* and *Crosshair* is preferred over *Direct Touch*. This is due both to the finger occlusion affecting direct touch input and the trade-off between holding the tablet and screen accessibility.

4.2.2.1 Procedure and design

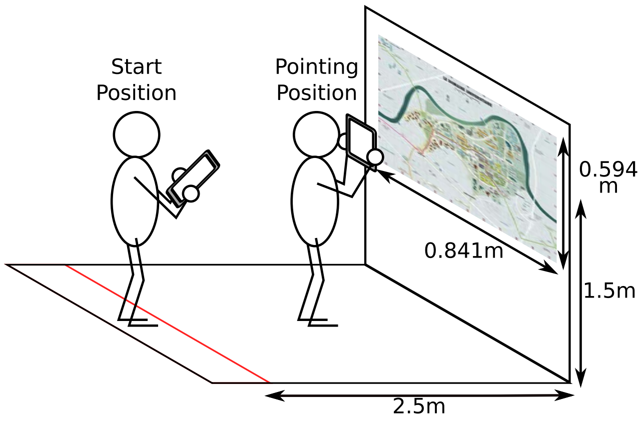


Figure 4.7: Experimental set-up: Participants started 2.5m away from the physical wall map and could then move freely to perform the pointing tasks.

For each of the four techniques, we explained the technique to the participants before giving them the chanve to try it freely. Participants then performed five different pointing tasks in which they placed AR marks on a physical wall map using a handheld tablet. Participants were allowed to place multiple marks on each target, thus allowing them to make multiple attempts at correctly placing the marks. Participants then pressed a button to finish the current task. They were advised that only the last mark placed on each target would be taken into account. Each task was repeated three times.

We used a map of our campus site in A1 format and in landscape orientation (841mmx594mm). It was placed vertically on a wall with the center of the map 1.5m above the floor (Figure 4.7). The targets for the five tasks are shown in Figure 4.8. Tasks 1 and 2 involved placing a mark on a single target. Tasks 3, 4 and 5 involved placing marks on two different targets.

The five tasks were as follows:

- Task 1: Placing a mark on a building at the center of the map (height: 1.49m from the ground, dimensions: 3x2.5cm)
- Task 2: Placing a mark on a bus stop in the top right-hand corner of the map (height: 1.63m from the ground, diameter: 0.5cm)

- Task 3: Placing two marks: the first on a parking lot on the left-hand side of the map (height: 1.50m from the ground, dimensions: 2.1x1.9cm) and the second on a park on the right-hand side (height: 1.51m from the ground, dimensions: 4x4.2cm), a distance of 46cm from the first target
- Task 4: Placing two marks: the first on a triangle indicating the entrance to a building (height: 1.53m from the ground, dimensions: 0.2x0.2cm) and the second on a triangle indicating another entrance to the building (height: 1.52m from the ground, dimensions: 0.2x0.2cm), a distance of 2cm from the first target
- Task 5: Placing two marks: the first on a tram stop on the left-hand side of the map (height: 1.50m from the ground, diameter: 1cm) and the second on a bus stop on the right-hand side of the map (height: 1.47m from the ground, diameter: 0.6cm), a distance of 36cm from the first target

We choose these tasks so that there would be a wide range of different target sizes, from very small (0.2cm wide in Task 4) to fairly large (4cm wide in Task 3). We also varied the target acquisition conditions by introducing tasks that included two targets to be acquired with different distances between the targets.

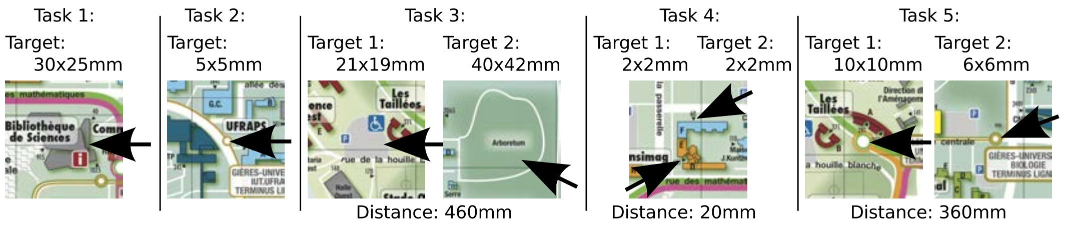


Figure 4.8: Targets used for each task. Tasks 1 and 2 involved of placing one mark while tasks 3, 4 and 5 involved of placing AR marks on 2 targets.

In each test, participants started off by standing 2.5m from the wall map. They were instructed to move around the room freely and to hold the tablet in portrait mode. A debriefing questionnaire and interview concluded experiment conducted using each technique.

Before starting the experiments based on the four techniques, the participants began to perform each of the five tasks once with no interaction, by simply locating the targets on the video on the tablet screen. The aim was to help them become acquainted with the tasks and the experimental set-up (in particular, form factor and video quality). This also mitigated the different degrees of knowledge the participants had of the campus map. After this initial training, all participants started with the *Direct Touch* technique. The presentation orders of the other three techniques were then counterbalanced across participants using Latin square. We used this design so that all participants would have *Direct Touch*, the de facto standard interaction, as a common baseline. The experiments lasted approximately one hour including a debriefing discussion.

The experimental design was as follows:

4 *Techniques* x (2 Tasks x 1 *Target* + 3 Tasks x 2 *Targets*) x 3 *Trials* = 96 target acquisitions per participants.

4.2.2.2 Apparatus and participants

Apparatus. We used a similar handheld tablet to that employed in the previous experiment (see 4.2.1.2). Unlike in the previous experiment, the tracked image was a physical wall map, rather than an image of stones displayed on the screen.

Participants. Twelve unpaid volunteers (4 female, 8 male; 1 left-handed and 1 ambidextrous), ranging in age from 22 to 45 years (mean of 27 years), were recruited from our institution. Given that they worked on the campus, they already knew the area represented on the map. In order to mitigate any differences between participants, we allowed them to practice freely at the beginning of the experiment and, prior to each test, we presented each task with the targets highlighted. All the participants had previous experience with touch-based handheld devices (seven on a daily basis) and nine had used a handheld tablet before.

4.2.2.3 Results

User preference. Having performed the tasks using each technique, we asked participants to complete a questionnaire and rate the techniques.

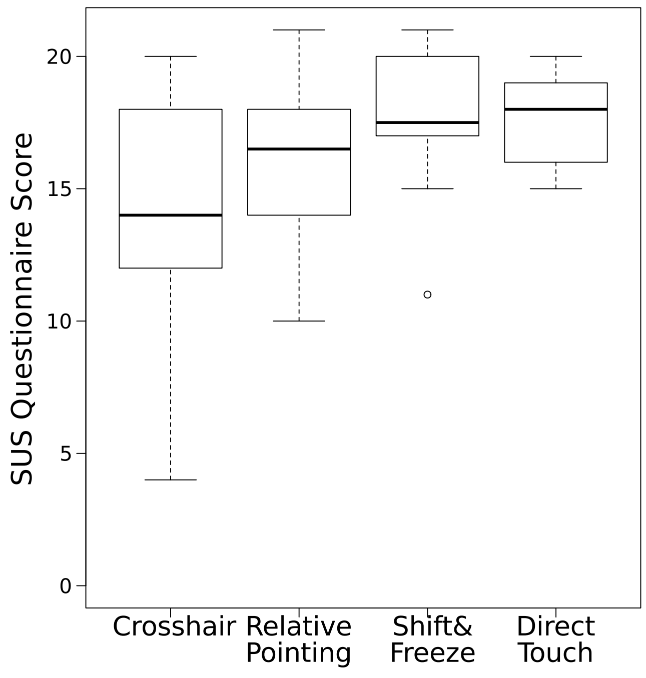


Figure 4.9: Overall SUS questionnaire scores for each technique.

The questionnaire comprised seven questions taken from the System Usability Scale (SUS) questionnaire [Brooke, 1996]. We did not include the following questions from the SUS questionnaire, because they are relevant for applications but not for pointing techniques alone: Question 4 on the need for technical support, question 5 on the integration of the different functionalities and question 6 on system inconsistency. The answers given used a four-point Likert scale and were combined to produce an overall usability score ranging from 0 (low) to 21 (high). The overall median score was 17/21. *Crosshair* had the lowest median score (14/21), followed by *Relative Pointing* (16.5/21), *Shift&Freeze* (17.5/21) and *Direct Touch* (18/21) (Figure 4.9). The score differences were not statistically significant (Kruskal-Wallis rank sum test of score by technique: χ^2=6.651, p>.05).

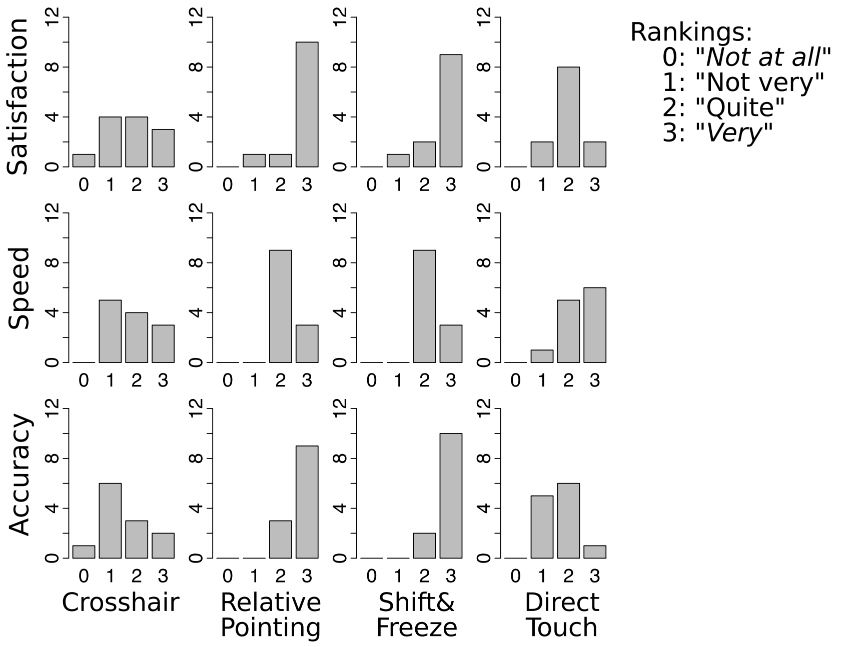


Figure 4.10: Histograms of overall satisfaction, speed and accuracy ratings for each technique.

We then asked participants to rate each technique according to three criteria: (1) overall satisfaction, (2) speed and (3) accuracy. The answers given were on a four-point Likert scale (Figure 4.10) ranging from "Very" (i.e., "Very satisfying", "Very fast", "Very precise") to "Not at all" ("Not satisfying at all", "Not fast at all", "Not precise at all"). Kruskal-Wallis rank sum tests found significant differences between the techniques when it came to overall satisfaction (χ^2=14.3897, p<.01) and accuracy (χ^2=24.2827*[7]) ratings. A post-hoc pairwise comparison of overall satisfaction and accuracy ratings showed significant differences (with p<.05 for overall satisfaction and p<.01 for accuracy) for all pairwise comparisons, except for the comparisons between *Shift&Freeze* and *Relative Pointing* and between *Crosshair* and *Direct Touch*. Table 4.2 sums up the means of the satisfaction and accuracy ratings.

Finally, during the experiment debriefing discussions, we asked participants which techniques they found to be the fastest and the most precise, and which techniques they preferred overall (multiple answers were allowed). Six participants said *Relative Pointing* was the fastest, four said *Direct Touch*, three said *Crosshair* and one said *Shift&Freeze*. All but one participant said *Shift&Freeze* was the most precise and five

[7]* indicates p<.0001.

Table 4.2: Means of satisfaction and accuracy rating by techniques.

Rank Means	Crosshair	Relative Pointing	Shift&Freeze	Direct Touch
Satisfaction	1.75	2.75	2.67	2
Accuracy	1.5	2.75	2.83	1.67

said *Relative Pointing*. Eight participants preferred *Relative Pointing* and six preferred *Shift&Freeze*. Two more participants would also have preferred *Shift&Freeze* if it had provided zooming and a cancel option.

Precise mode. The participants made use of the precise modes provided by the two proposed techniques, *Shift&Freeze* and *Relative Pointing*. Overall, the precise modes of both techniques were used for 73% of the pointing tasks. We further analyzed precise-mode usage by examining its usage separately for tasks involving large targets (i.e., Task 1 and Task 3) and for tasks involving smaller targets (i.e., Task 2, Task 4 and Task 5). For small targets, the precise modes were used almost all of the time (97%), while they were used for 37% of the pointing tasks involving large targets. The results were similar for both techniques. Table 4.3 provides a summary of precise-mode usage for the two techniques. This shows that participants adjusted their behavior according to target size, thus confirming that participants found the two modes useful and that they used them deliberately. Iy also confirms that the precise modes of our two techniques (both *Shift&Freeze* and *Relative Pointing*) are particularly suited to small targets.

Distance to the map. We measured the distance of the handheld tablet from the map when the participants selected each target. Of the 1,152 target acquisitions collected, we removed 6 cases where selection was not actually performed. The distances from the map at which participants acquired the targets ranged from $8cm$ to $234cm$, with a median of $30cm$ (1st quartile: $23cm$, 3rd quartile: $45cm$) (Figure 4.11-Left).

The difference of $37cm$ between the smallest distance and the 3rd quartile is fairly low compared to the distance of $2.5m$ at which participants started each task. This indicates that most of the participants walked approximately the same distance for all tasks and all pointing techniques. This results is surprising given that we expected the participants to adjust the distance they walked according to the difficulty of the task. Only one participant clearly adjusted his distance from the map according to both target size and ease of use of the technique. He did so to such an extent that he did not walk at all for large-target selection using *Relative Pointing* (as he felt more comfortable with this technique).

Table 4.3: Percentage of usage of the precise modes of *Shift&Freeze* and *Relative Pointing* according to the size of targets.

Technique	Overall	Large targets	Small targets
Overall	73%	37%	97%
Shift&Freeze	73%	37%	97%
Relative Pt.	72%	36%	96%

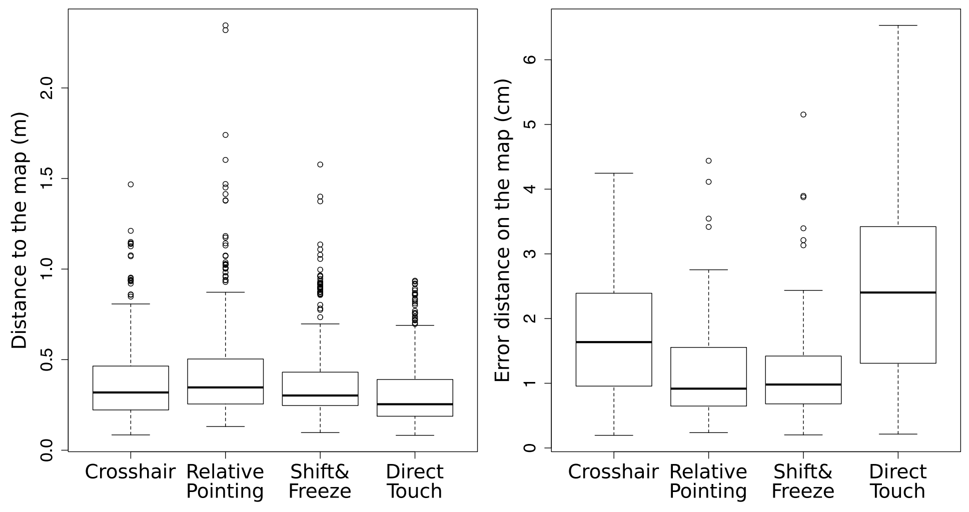


Figure 4.11: (Left) Boxplots of distances from the map at which participants selected targets for each *Technique*. (Right) Boxplots of distances between the selection point and the target's center in Task 4 ($2mm$ targets).

Selection precision. We looked at the spread of selection points around the small targets in Task 4 ($2mm$ wide). Of the 288 target selections, we removed 7 outliers noted during the experiment. The overall median of the distances between the point selected and the target on the map was $1.7cm$. The median for *Direct Touch* ($2.4cm$) was more than twice that of *Shift&Freeze* ($1.0cm$) and that of *Relative Pointing* ($0.9cm$). The *Crosshair* median ($1.6cm$) lay in between (Figure 4.11-Right). This confirms that the precision improvement we observed with *Shift&Freeze* and *Relative Pointing* in the previous experiment (see 4.2.1) still holds in a less controlled environment.

4.2.2.4 Discussion

The spread of selection points around small targets indicates that *Relative Pointing* and *Shift&Freeze* had a higher accuracy than the two baseline techniques as in the previous experiment. It also indicates that *Direct Touch* is the least precise technique and that *Crosshair* has an intermediate degree of accuracy.

We expected participants to prefer *Relative Pointing* and *Shift&Freeze* to *Crosshair* and *Direct Touch* (H1). We also expected participants to prefer indirect cursor-based techniques (i.e., *Crosshair* and *Relative Pointing*) to direct pointing techniques (i.e., *Direct Touch* and *Shift&Freeze*) (H2).

Shift&Freeze and *Relative Pointing* were indeed preferred over *Direct Touch* and *Crosshair*. Participants gave the highest ratings in terms of accuracy and overall satisfaction to *Shift&Freeze* and *Relative Pointing*. Moreover, participants never mentioned either *Direct Touch* or *Crosshair* when asked for their preferred technique or for the most precise technique. These results support hypothesis H1.

Crosshair received the lowest SUS scores, while *Shift&Freeze* was almost unanimously declared to be the most precise technique. In addition, *Shift&Freeze* and

Relative Pointing received almost the same overall preference rating. Thus, indirect pointing techniques were not preferred over *Direct Touch*-based techniques (i.e., *Direct Touch* and *Shift&Freeze*), even though the tablet form factor was presumably less convenient for *Direct Touch*-based techniques. These results refute hypothesis H2.

Some of our participants were so accustomed to *Direct Touch* interaction that they were tempted to tap on the cursor when using the two indirect pointing techniques (i.e., *Crosshair* and *Relative Pointing*).

The user feedback collected via the questionnaires was consistent with the informal comments made by the participants during the interviews.

Some participants complained about the handheld tablet form factor. The first reason is that its size and weight make it more suited to being held with both hands. However, as already explained, this impairs access to the screen when using the *Direct Touch* and *Shift&Freeze* techniques. The second reason is that the tablet is not held the same for AR applications as it is for other applications. Indeed, the user must to maintain the camera's focus while interacting with the screen. Some participants felt that they could not hold the tablet securely because it was slippery. They suggested adding grips to the device. In addition, the screen borders were not wide enough to allow all participants to hold the tablet with their thumb on the side of the screen. This resulted in accidental touch inputs and discomfort when trying to hold the tablet with one hand and interact with the other one.

4.2.3 Summary

We learned the following from the two experiments performed to compare the two proposed techniques, i.e., *Shift&Freeze* and *Relative Pointing*, and their two baseline techniques, *Direct Touch* and *Crosshair*, respectively.

Overall, participants preferred the two proposed techniques (*Shift&Freeze* and *Relative Pointing*) to the two baseline techniques (*Direct Touch* and *Crosshair*). We expected participants to prefer *Relative Pointing* to *Shift&Freeze* because it allows the tablet to be held with both hands, but this was not the case. This is probably due to the fact that people are used to *Direct Touch* interaction on touch-based handheld devices.

The experiments indicated that there was no significant difference between *Relative Pointing* and *Shift&Freeze* in terms of error distance and duration. Yet, the error rate was significantly different. Indeed, *Relative Pointing* was more accurate than *Shift&Freeze.*

In the following experiments, we further investigated *Relative Pointing* under altered registration conditions and compared it with its baseline, i.e., *Crosshair.*

4.3 Experiments: The effects of registration errors

The results of the previous set of experiments indicated that *Relative Pointing* mitigates the effect of hand tremor and thus improves pointing precision, while not relying on freezing the frame as in the case of *Shift&Freeze.* In this second set of experiments, we focused on *Relative Pointing* and evaluating its precision under altered registration conditions. We also expected *Relative Pointing* to mitigate the effects of altered registration conditions.

For this purpose, we held two additional experiments aimed primarily at evaluating the effect of registration jitter on the precise modes of both *Relative Pointing* and *Crosshair* (the baseline technique extended by *Relative Pointing*). We ran two experiments that involved pointing at digital targets and then at physical targets. According to our design space (see Chapter 2), the position of the digital targets on the screen is impaired by both natural hand tremor and registration jitter, while the position of physical targets on the screen is impaired by hand tremor only (Figure 4.12).

The goal of the first experiment was to compare the effect of registration jitter on both *Crosshair Relative Pointing*'s precise mode when pointing at digital targets overlaid on a physical object. We also compared the effect of device form factor during this experiment. The goal of the second experiment was to compare the effect of registration jitter on *Crosshair* and *Relative Pointing*'s precise mode when pointing at physical targets rather than digital ones. We also compared the effect of camera image latency during this experiment.

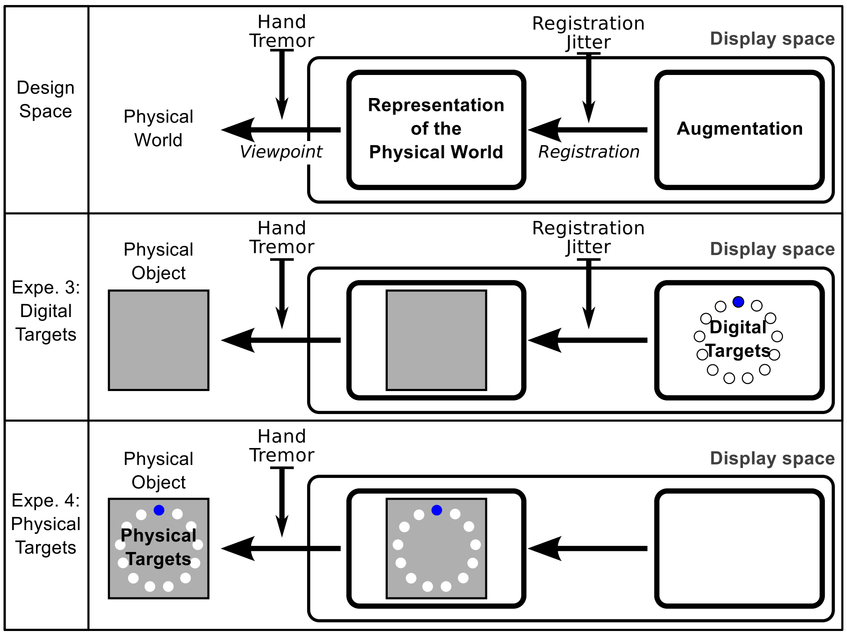


Figure 4.12: Parameters when pointing at physical targets and digital targets, based on the design space presented in chapter 2.

Before presenting these two experiments, we will first review previous experiments held to evaluate the effect of altered registration conditions on user performance with AR set-ups.

4.3.1 Experiments conducted on registration errors

Registration errors, such as fixed offset error, latency or jitter, are key issues for AR set-ups.

A fixed offset error results in the augmentation always being misaligned with the representation of the physical world by the same offset. This therefore affects the *Registration* spatial relation (see Chapter 2 and Figure 4.12-Top).

Latency errors are due to the tracking system taking a long time to compute the current point of view. This can result in two types of latency. First, the camera image may not be displayed until the tracking system has computed an estimation of the point of view. In this case, the augmentation is registered to the corresponding camera image. However, there will be a degree of latency when it comes to displaying the augmented scene on the screen. This impairs the *Viewpoint* spatial relation because the representation of the physical world on the screen does not exaclty relate to the device's current position (see chapter 2 and Figure 4.12-Top). Second, when rendering the current camera image, the application does not wait for the corresponding estimation of the point of view. In this case, the information used to register the augmentation is older than the camera image presented. This impairs the *Registration* spatial relation, particularly when the point of view is changing.

Finally, registration jitter is due to the inaccuracy of the underlying tracking system, which provides a noisy estimation of the point of view. This also impairs the *Registration* spatial relation.

Registration errors have been studied and evaluated experimentally. In his survey of AR, Azuma [Azuma, 1997] discussed registration errors in terms of static and dynamic errors. Holloway [Holloway, 1997] proposed a model to analyze the registration errors of an optical see-through head-mounted display used for surgery planning. Experimental evaluations of registration errors tend to follow two strategies: (1) a Virtual Reality set-up is used to simulate an AR set-up, and (2) an AR set-up is used and its characteristics altered.

4.3.1.1 Virtual Reality set-up

Some protocols use an immersive Virtual Reality (VR) set-up to simulate a Head-Mounted Display AR set-up. This allows precise control of the different parameters of the simulated Head-Mounted Display AR set-up, which would otherwise be impossible. We are not aware of any experiments having been performed using such a set-up to evaluate the effect of fixed offset error. Ventura *et al.* [Ventura et al., 2009] experimented with the effects of different fields of view and the duration of registration dropouts, while performing a target following task using X-ray vision. They found that both field of view and dropout duration had a significant effect. Using a similar set-up, Ragan *et al.* [Ragan et al., 2009] evaluated the effects of both latency and jitter when guiding a ring along a crooked path. They observed effects of both latency and jitter. Their results suggested that jitter was the dominant type of error. Lee *et al.* [Lee et al., 2010a] also found that latency had an effect during a ring-guiding task. In addition, they studied the effect of the VR environment's latency and found that it had a significant impact on the task's performance. Therefore, simulating an AR set-up in a VR environment might have a significant effect on the results.

Each of these studies simulated an AR Head-Mounted Display set-up, rather than a handheld AR set-up. One experiment conducted using a VR simulation of a handheld AR is reported in [Baricevic et al., 2012]. However, this experiment did not focus on registration conditions, as its aim was to evaluate different viewpoints in the augmented scene: the viewpoint of the device's camera or that of the user.

4.3.1.2 Augmented Reality set-up

Past experiments have used an AR set-up and introduced artificial registration errors. Livingston and Ai [Livingston and Ai, 2008] held an outdoor experiment based on a target following task using X-ray vision. Durin the experiment, they varied fixed orientation errors, latency and registration jitter. They found that high latency impaired performance and that the effect of fixed orientation offset error and registration jitter were not as important as expected. However, users believed that registration jitter was most detrimental. Robertson and MacIntyre [Robertson and MacIntyre, 2007] evaluated the effectiveness of digital graphic context as a means to overcome registration errors, by placing a brick at the position indicated by the augmentation. They found graphic context to be useful. Coffin *et al.* [Coffin et al., 2011] evaluated the impact of recovery density on registration recovery time for key frame-based and model-based tracking mechanisms. Unlike the other studies described in this section, which used Head-Mounted Displays, this study was conducted with a handheld tablet device.

4.3.1.3 Summary

To further explore the capabilities of *Relative Pointing*, we carried out two experiments to compare screen-centered *Crosshair* with the precise mode of *Relative Pointing*. We based these experimental studies on a handheld AR set-up rather than a Virtual Reality set-up to avoid the possible effects of Virtual Reality simulation. We chose to evaluate the effect of registration jitter, as in [Ragan et al., 2009] and [Livingston and Ai, 2008], because we expected this type of registration error to be detrimental to both pointing adjustment accuracy and the user's visual perception. In the second experiment, we evaluated the effect of additional camera image presentation latency. Indeed, such latency can also be detrimental to performance.

In the first experiment, we evaluated pointing at digital targets overlaid on a physical image. We also considered both smart phone and tablet form factors. In the second experiment, we evaluated pointing at physical targets. We also considered the effect of additional camera image presentation latency.

4.3.2 Techniques and pointing tasks tested

The aim of this set of experiments was to evaluate the *Relative Pointing* technique under altered registration conditions. We will now detail the techniques evaluated, since they differ from the techniques presented in the previous experiments (see 4.2). We will then present the differences between the two pointing tasks performed during these experiments.

4.3.2.1 Techniques

The *Relative Pointing* technique presented in the previous chapter on Pointing Techniques (see 3.3.2.2) is comprised of two modes: a default mode corresponding to the screen-centered *Crosshair* technique and a precise mode with indirect relative cursor control. In order to better evaluate the difference between the precise mode of *Relative Pointing* and its default mode (i.e., *Crosshair*) we separated these two modes in the two following two experiments.

The *Relative Pointing* technique used in this set of experiments therefore differs from the full technique we presented in the previous chapter on Pointing Techniques (See 3.3.2.2) and evaluated in previous experiments (see 4.2). The full *Relative Pointing* technique includes both screen-centered *Crosshair* and *Relative Pointing* modes. In the previous experiment we evaluated *Relative Pointing* as a complete interaction technique. In this experiment we focused on the indirect relative pointing mode only, where finger touch input controls cursor displacements in a relative manner. For this experiment we therefore simplified the *Relative Pointing* technique at its core: participants could not choose between absolute and relative pointing mode and only the relative pointing mode was available. Thus, it was not meant to be a complete interaction technique that could be used as is.

We also used a 1:1 Control-to-Display (CD) gain, as this is a baseline that a well designed dynamic transfer function should better. Furthermore, a 1:1 CD gain was sufficient to perform the pointing tasks involved in this experiment with a single finger stroke on the screen. Furthermore, since *Relative Pointing* is intended to allow pointing adjustment to be performed, movements should be of limited amplitude.

Finally, we chose to trigger the validation on finger lift-off so as to simplify *Relative Pointing*. While this limits cursor displacement, because it does not allow clutching, we designed the experiment so that this method would be sufficient and the participants would not need clutching. Whether performing validation on finger lift-off or with a tap on the screen (as used in the previous experiments) is a trade-off between faster interaction (no tap required to validate the selection) and richer interaction (finger clutching, cancellation, preview of the position selected before validation).

Thus, the two elementary pointing techniques that we evaluated in these two experiments were as follows:

- *Crosshair*: A screen-centered crosshair indicates the pointing position. Validation takes place on finger lift-off with a tap anywhere on the screen.
- *Relative Pointing*: Only the precise mode of this technique was used. The cursor is always bound to the augmented scene attached to the physical image. The cursor is initially placed at the center of the physical image. Finger strokes control the cursor displacements with a 1:1 CD gain on the screen. Finger lift-off triggers the validation, meaning that neither finger clutching nor cancellation are possible. We will subsequently refer to this method, where only precise mode is used, as *Relative Pointing*.

For both *Crosshair* and *Relative Pointing*, the cursor was a red square cross with filled triangles at each end. The cursor was $7.7mm$ wide with a $0.19mm$ stroke width on the tablet and $6.2mm$ wide with a $0.16mm$ stroke width on the phone.

4.3.2.2 Tasks

We evaluated these two basic techniques under altered registration conditions when pointing at both digital targets and physical targets. Indeed, under registration jitter conditions, there is a difference between digital targets attached to a physical object and actual physical targets. The position of digital targets on the screen is affected by registration jitter, while the position of physical targets is not (Figure 4.13). However, the system's knowledge of a physical target's position is affected by registration jitter. Thus, unlike with digital targets, in the case of physical targets under registration jitter conditions, there is a difference between the targets displayed in the camera image and the position of these targets as known to the system.

In the case of digital targets, as in experiment 3 (see 4.3.3), the *Relative Pointing* cursor is subject to the same registration jitter as the targets (Figure 4.13-Top). Therefore, even if the cursor is not stable on the screen, it will be stable relative to the targets. This is not the case with *Crosshair* as the cursor is fixed on the screen.

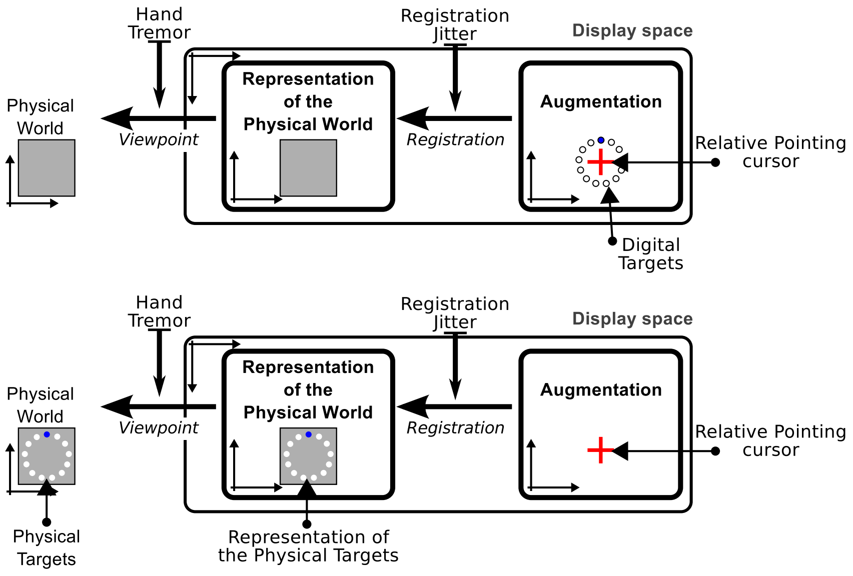


Figure 4.13: (Top) The *Relative Pointing* cursor and the digital targets share the same frame of reference, i.e., *Augmentation*. Thus, they are both impaired by *Registration jitter*. (Bottom) The *Relative Pointing* cursor is subject to *Registration jitter*, unlike the representation of the physical targets on the screen.

In the case of physical targets, as covered in experiment 4 (see 4.3.4), the *Relative Pointing* cursor provides feedback about registration jitter, since the cursor position is subject to the same registration jitter as any augmentation (Figure 4.13-Bottom). Unlike *Relative Pointing*, the *Crosshair* technique does not provide feedback on the registration jitter while pointing at physical targets, unless some augmentation is displayed (because augmentation is impaired by registration jitter).

We also wished to explore the effects of camera image presentation latency. This is an altered condition that is orthogonal to registration jitter. Such latency impairs the *Viewpoint* spatial relation while registration jitter impairs the *Registration* spatial relation (Figure 4.13). Thus, camera image presentation latency can impair pointing at physical targets. We therefore evaluated such latency during experiment 4 only (see 4.3.4).

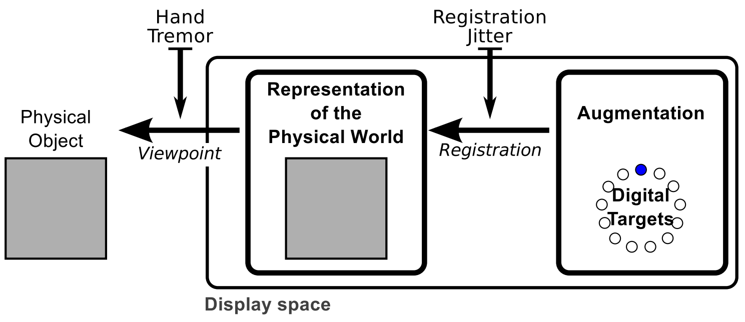


Figure 4.14: Digital target position on the screen is subject to both hand tremor and registration jitter.

4.3.3 Experiment 3: Digital target acquisition

The task performed during this experiment involved pointing at digital targets. The position of digital targets on the handheld device's screen is subject to both natural hand tremor and registration jitter (Figure 4.14).

During this experiment, we varied the following conditions:

- Two levels of registration jitter: (1) that of the default set-up, and (2) extra artificial jitter.
- Two device form factors: (1) a handheld tablet, and (2) a one-handed handheld device (i.e., a smart phone).
- Two basic pointing techniques: the precise mode of *Relative Pointing* (see 4.3.2) and *Crosshair*.

In this experiment, we studied pointing at digital targets registered to a physical image placed on a wall. Our aim was to compare the two techniques solely during the digital pointing phase, when targets are already visible on the handheld device screen.

We formulated the following hypotheses:

- H1: Registration jitter impairs the accuracy of *Crosshair*. The cursor is fixed on the screen, while the targets are unstable in the screen frame.
- H2: Registration jitter does not impair the accuracy of *Relative Pointing*. The cursor is subject to the same registration jitter as the targets (see 4.3.2). Therefore, even if the cursor is not stable on the screen, it is stable relative to the targets. However, registration jitter may be detrimental to visual perception and may therefore still impair pointing accuracy.
- H3: Overall, *Relative Pointing* is more accurate than *Crosshair*. The stabilization provided by *Relative Pointing* also allows for natural hand tremor.

4.3.3.1 Procedure and design

This experiment was carried by applying the cyclical multi-direction pointing task paradigm of ISO9241-9 [Soukoreff and MacKenzie, 2004], adapted to a handheld

AR set-up (Figure 4.15). An image ($24.7cm$ x $17.3cm$) was placed vertically on the wall with its center located $1.5m$ from the ground. This image (the stones shown in Figure 4.15) had no meaningful content, as the pointing task performed was in no particular context. It simply provided a background area with good features for the vision-based tracking system we used where to overlay the digital targets. On the screen of the handheld device, 13 digital targets arranged in a circle were overlaid on this physical image. The targets to acquire were highlighted in blue and always in the same order: top target first, followed by the target opposite and slightly clockwise from the one selected previously, and so on. In the event of a target acquisition failure, the target in question turned dark red, otherwise it reverted to white. The aim was to provide participants with an immediate feedback on their success or failure.

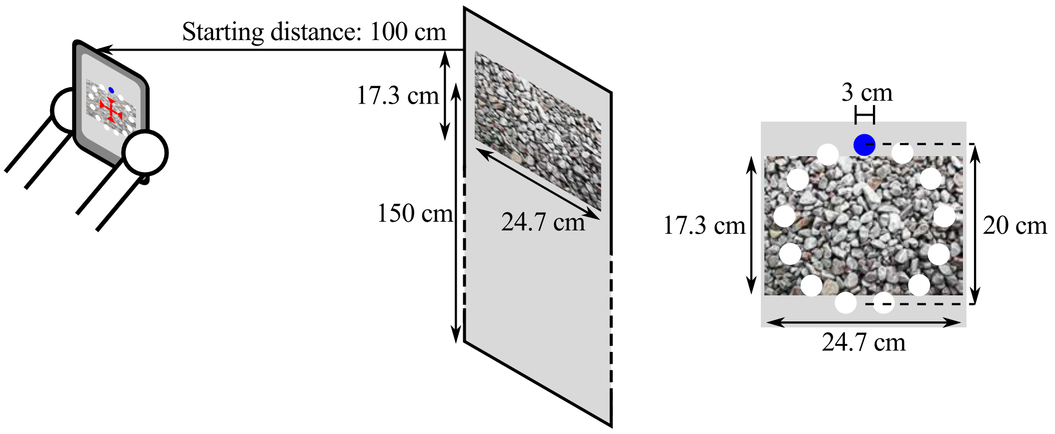


Figure 4.15: Set-up of experiment 3.

Participants were first given a presentation on the handheld techniques and the task. They were also asked to complete a short questionnaire. They then performed the experiment using the first technique, before performing it using the second technique. The initial presentation and questionnaire took around 10 minutes, while the experiment itself also lasted around 10 minutes.

Participants performed the experiment while standing in front of the physical image. When using the handheld tablet, participants were instructed to hold the device in portrait mode with both hands and to interact with their thumbs. When using the phone, participants were instructed to hold the device in portrait mode with their dominant hand and to interact only with this hand. Participants were also instructed to select the highlighted target as quickly and accurately as possible and to rest between blocks.

We wanted to have all the participants to interact at a consistent distance from the remote image, whether they were using the tablet or the phone. Therefore, before each block, participants had to place the handheld device $1m \pm 5cm$ away from the physical image according to the indications displayed on the screen. These instructions were hidden as soon as participants acquired the first target. The experimental results enabled us to check the actual distance from the remote image when the task was being performed.

We tested two registration jitter conditions: (1) The underlying tracking system's default set-up, and (2) Extra artificial translational noise added to the relative position of the physical image in the physical image plane (with a pseudo-normal distribution of mean 0 and standard deviation $5mm$). This is consistent with Ragan *et al.* [Ragan et al., 2009], who varied the translational jitter's standard deviation between 0 and $11.43mm$. The artificial noise was generated using the polar form of the Box-Muller transform (let x and y be two random numbers with a uniform distribution, where $(x^2.y^2) \in]0,1[$, we generated two normal random variables $a = x\sqrt{-2\frac{\ln s}{s}}$ and $b = y\sqrt{-2\frac{\ln s}{s}}$ where $s = x^2 + y^2$) and the pseudo random function arc4random.

We used a single movement Amplitude (A) of $20cm$ on the physical image and a single target width (W) of $3cm$ on the physical image. The Index of Difficulty of this task is ID$=log_2(A/W+1) = 2.9$ bits. From $1m{\pm}5cm$, on the screen of the phone, A was within a range of [1.7-1.9]cm and W was within a range of [0.25-0.3]cm. On the screen of the tablet, A was within a range of [4.6-5.2]cm and W was within a range of [0.7-0.8]cm.

With these A and W values on the screen and a 1:1 CD gain for *Relative Pointing*, it was possible to reach the targets with a single thumb stroke on both devices. With this set-up, the physical image on the wall remained in the camera's field of view at all times while performing the task.

We used a mixed experimental design with repeated measurements. *Device* was a between-subjects independent variable. Half of the participants performed the experiment with a handheld tablet and the other half with a smart phone-sized handheld device. *Technique* and *Registration jitter* were within-subject independent variables. The presentation orders of both *Technique* and *Registration jitter* were counterbalanced across participants using Latin square. Based on a single target Width ($3cm$), a single movement Amplitude ($20cm$) and a single initial distance between the remote image and the handheld device ($1m{\pm}5cm$), the within-subject experimental design was as follows:

2 *Techniques* x 2 *Registration jitter* x 2 *Blocks* x 12 *Targets* = 96 target acquisitions per subject.

For each *Technique*, participants first performed two blocks for practice (one for each *Registration jitter* condition), resulting in 48 extra acquisitions:
2 *Techniques* x 2 *Registration jitter* x 12 *Targets* = 48 target acquisitions per subject for practice.

4.3.3.2 Apparatus and Participants

Apparatus. We used an iPad2 (weight: $601g$, screen resolution: 1024x768 - $132dpi$) for the *tablet* condition and an iPod4 (weight: $88g$, screen resolution: 960x640 - $326dpi$) for the *phone* condition. Each device's touch input resolution was the same as that of its screen. The application used for the experiment was based on Vuforia SDK[8] version 1.5.9. This application runs at around 30 frames per second on the iPad2 and 26 frames per second on the iPod4. The images retrieved from the camera had a resolution of 480x640 pixels and were displayed in full-screen mode (cropped on the iPod4).

[8]https://www.vuforia.com/

Participants. The 24 individuals who took part in the experiment were unpaid right-handed Computer Science undergraduates.

Twelve of the participants (one female; age: [21-29] years, mean: 23 years) performed the experiment with a handheld *tablet*. Of thesem ten used a touch-based handheld device on a daily basis and two had never used one, while five had used a handheld tablet before and one had used an AR application before.

The twelve other participants (three females, age: [21-27] years, mean: 23 years) performed the experiment with a *phone*. All had previous experience with touch-based handheld devices (eleven on a daily basis), eight had used a handheld tablet before and four had used an AR application previously.

4.3.3.3 Results

We monitored the distance between the physical image and the handheld device when the target acquisitions were performed. The overall average distance from the physical image was $99.5cm$ (1st quartile: $97cm$, 3rd quartile: $102cm$, range: [$89cm$-$113cm$]). This indicates that the constraint of placing the handheld device $1m \pm 5cm$ away from the physical image before starting each block was sufficient to confine the distance between the handheld device and the physical image to a fairly small range of $24cm$.

We explored the effects of the *Technique*, *Registration jitter* and *Device* factors by analyzing two dependent variables: *Errors* and *Duration*. Table 4.4 summarizes the values of the dependent variables for each condition. We recorded 2,304 target acquisitions and discarded no observations during the analysis described below.

We checked for failures in the vision-based method used to track the background image during the experiment. When tracking failed during the experiment, participants could not perform target acquisition until tracking had successfully resumed. Tracking failed 68 times in total over the 2,304 target acquisitions recorded. On average, tracking resumed after 0.30 seconds±0.33[9]. Most of the tracking failures (58 out of 68) were observed with *Crosshair* on the *phone*. These may have resulted from the lower computational power and frame rate of the *phone* compared to the *tablet*. The failure may also have resulted from the fact that *Crosshair* requires the participants to move the device in order to interact, unlike *Relative Pointing*.

Table 4.4: Error rates, duration (mean ± standard deviation).

Tablet			
Technique	Jitter	Error rate (%)	Duration (s)
Crosshair	Default	8	1.51 ± 0.39
	Artif.	22	1.71 ± 0.48
Relative Pt.	Default	3	1.62 ± 0.57
	Artif.	4	1.87 ± 0.73

Phone			
Technique	Jitter	Error rate (%)	Duration (s)
Crosshair	Default	14	2.11 ± 0.79
	Artif.	30	2.25 ± 0.81
Relative Pt.	Default	6	1.55 ± 0.42
	Artif.	7	1.75 ± 0.44

[9] m+/-sd gives the mean m and standard deviation sd.

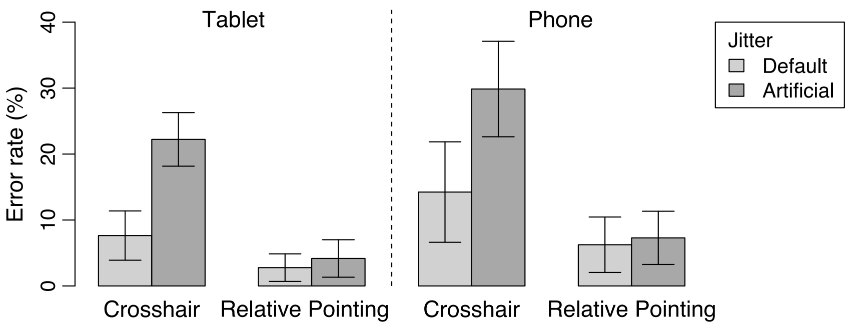


Figure 4.16: Barplots of error rates (%) and 95% confidence intervals of error rates aggregated by participant for each *Technique* and each level of *Registration jitter* for each *Device*.

Errors. The overall error rate was 12%. Table 4.5 summarizes the error rate for each factor. Figure 4.16 lists the error rates separately for both *Devices*, for each *Technique* and level of *Registration jitter*.

We tested the dependence between errors and the different factors using Pearson's Chi-squared test with Yates' continuity correction. We did not find a significant dependence between errors and *Blocks* (χ^2=0.504, p=.48).

Accoss all the observations, significant dependences were found for *Technique* (χ^2=97.583* [10]), *Registration jitter* (χ^2=36.054*) and *Device* (χ^2=14.511, p<.001). We further analyzed the dependence between errors and *Registration jitter* for each *Technique* on each *Device*. For *Crosshair*, we found a significant dependence between errors and *Registration jitter* on both *Devices* (tablet: χ^2=22.977*, ϕ=0.20; phone: χ^2=19.556*, ϕ=0.18). With *Relative Pointing*, no significant dependence was found (tablet: χ^2=0.466, p=.49, ϕ=0.03; phone: χ^2=0.11, p=.74, ϕ=0.01). A post-hoc power analysis indicated a power of 0.67 for small effect size (0.1) and a power of 0.99 for medium effect size (0.3).

Table 4.5: Error rates, duration (mean ± standard deviation) for each factor.

Factor	Error rate (%)	Duration (s)
Overall	12	1.80 ± 0.65
Crosshair	18	1.90 ± 0.71
Relative Pt.	5	1.70 ± 0.57
Default Jitter	8	1.70 ± 0.61
Artif. Jitter	16	1.90 ± 0.67
Tablet	9	1.68 ± 0.57
Phone	14	1.90 ± 0.70

Duration. The overall mean target acquisition duration was 1.80±0.65 seconds[11]. Table 4.5 summarizes the mean duration and standard deviation for each factor. Figure 4.17 lists the durations separately for both *Devices*, for each *Technique* and level of *Registration jitter*.

[10]* indicates p<.0001.

[11]m+/-sd gives the mean m and standard deviation sd.

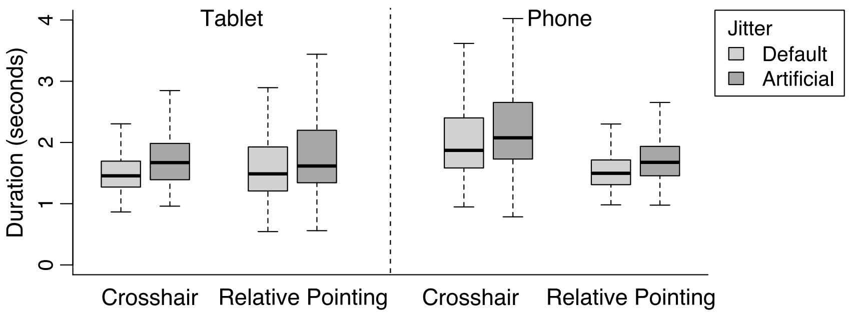


Figure 4.17: Boxplots of target acquisition durations (seconds) for each *Technique* and each level of *Registration jitter* for each *Device*.

A paired t-test found a significant difference between *Blocks* (t_{1151}=7.559*) with the target acquisitions of second block being faster than those of the first (95% confidence interval (CI): [0.09-0.16] seconds). Unlike the error rate, this indicates a learning effect.

We performed a 2 x 2 x 2 (*Technique* x *Registration jitter* x *Device*) mixed-design analysis of variance on the median aggregate repetition duration with participant as a fixed factor. We found a significant effect for *Technique*, with $p<.05$ ($F_{1,22}$=5.894), and *Registration jitter* ($F_{1,22}$=33.266*). The *Technique* x *Device* interaction was also found to be significant ($F_{1,22}$=12.340; $p<.01$). The *Device* main effect and other interactions were not found to be significant.

For *Technique*, a paired t-test found a mean difference of 0.21 seconds (95% CI: [0.05-0.37] *s*; t_{47}=2.749; $p<.01$). For *Registration jitter*, a paired t-test found a mean difference of 0.16*s* (95% CI: [0.11-0.22] *s*; t_{47}=5.876*). To further study the *Technique* x *Device* interaction, we ran paired t-tests separately for both *Devices*. For the *tablet*, the paired t-test was not significant (t_{23}=1.223; p=.23). For the *phone*, we found a mean difference of 0.53*s* (95% CI: [0.31-0.73] *s*; t_{23}=5.155*), with *Crosshair* being slower than *Relative Pointing*.

4.3.3.4 Discussion

In this experiment, we expected *Registration jitter* to impair the accuracy of *Crosshair* (H1), but not to impair the accuracy of *Relative Pointing* (H2). We also expected *Relative Pointing* to be more accurate overall than *Crosshair* (H3).

Registration jitter impaired both the accuracy and the target acquisition duration of *Crosshair* as indicated by its significantly higher error rate with artificial jitter and the significant *Registration jitter* main effect on target acquisition duration. This supports hypothesis H1.

This experiment did not show that *Registration jitter* had a significant effect on the error rate of *Relative Pointing*. However, like with *Crosshair*, *Registration jitter* significantly impaired target acquisition duration when using *Relative Pointing*, as indicated by the *Registration jitter* main effect on target acquisition duration. This partly supports hypothesis H2. Indeed, the accuracy of *Relative Pointing* was not significantly impaired by *Registration jitter*, unlike that of *Crosshair*. But this does not signify that *Registration jitter* has no effect on the accuracy of *Relative Pointing*.

The overall error rate of *Relative Pointing* was lower than that of *Crosshair*. In addition, *Relative Pointing* was faster overall than *Crosshair*. This supports hypothesis H3.

Firstly, with *Crosshair*, both error rate and acquisition duration varied across the different conditions. Our hypotheses can explain such variations, but other effects could also have interfered. Indeed, *Crosshair* had a rather high error rate across all conditions. This might indicate that *Crosshair* was operating close to its precision limit. If we consider the effect of artificial jitter to be a reduction of the target width, and if *Crosshair* was indeed being used at its precision limit, then part of the increase in error rate may be due to the precision limit. *Crosshair* performed worse on the *phone* than on the *tablet*. This might be related to the tracking failures observed mainly when using *Crosshair* on the *phone* (58 out of the 68 failures we observed overall) and to the lower processing power and frame rate of the *phone* we used. This might also be due to the way in which the device is held (one-handed vs. two-handed). However, our experiment does not allow us to draw any conclusions on this point.

Secondly, with *Relative Pointing*, the results suggest that both error rates and acquisition durations varied less across the different *Registration jitter* and *Device* conditions than was the case with *Crosshair*. As explained in hypothesis H2, the cursor was stable within the frame of reference of the targets. This explains the stability recorded across the different *Registration jitter* conditions. The small variation across the *Devices* can be explained by the fact that the difference between the devices can be interpreted as a change in the scale of the pointing task in motor space. Indeed, the differences between the devices in terms of camera and screen size resulted in different scales of both movement distance and target width on the screen. This led to pointing tasks in motor space that had different scales but a similar Index of Difficulty.

In the form evaluated, *Relative Pointing* did not allow finger clutching. Adding finger clutching would ensure access to the entire screen. It would also require an additional action, such as a tap, to perform validation, which would increase the target acquisition duration. A dynamic transfer function could also be used to improve the performance of *Relative Pointing* and reduce the number of clutches.

This experiment compared the impact of both *Registration jitter* and *Device* form factor on two cursor-based pointing techniques for handheld AR: (1) screen-centered *Crosshair*, and (2) the precise mode of *Relative Pointing* in the frame of reference the physical object. Our evaluation indicated that the latter was less error prone than the former. In addition, the accuracy of *Relative Pointing* was less sensitive to registration jitter and device form factor than that of *Crosshair*.

4.3.4 Experiment 4: Physical target acquisition

The two techniques we compared in this experiment were the same as in the previous experiment (see 4.3.2), in which we used only one target size. This led to rather high error rates for *Crosshair* and the presumption that part of this error rate may have been dut to the small target size. In this experiment, we used the same target size as in the previous experiment (i.e., $3cm$), but we also used a larger target size.

Following the experiment on the effect of registration jitter on digital target acquisition, we investigated the differences between *Crosshair* and *Relative Pointing*

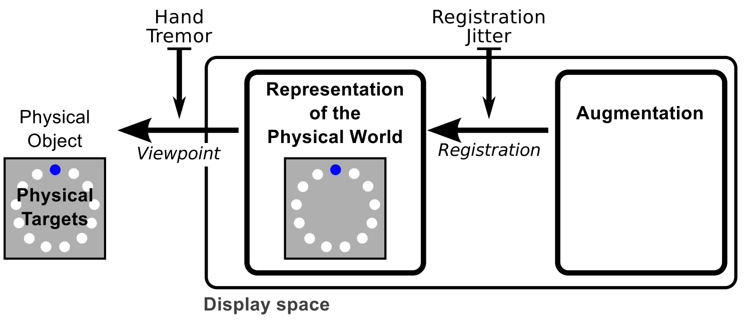


Figure 4.18: The position of physical targets on the screen is subject to natural hand tremor but not to registration jitter.

for physical target acquisition (Figure 4.18). When it comes to registration jitter, differences are apparent between digital targets attached to a physical object (such as circles overlaid on a 2D image on a wall, as in the previous experiment) and physical targets. Indeed, the representation of physical targets on the screen is not impaired by registration jitter, while the positions of targets known to the system are subject to registration jitter (see 4.3.2).

The *Crosshair* technique does not provide feedback on registration jitter in the case of physical target acquisition, unless some augmentation is displayed (as augmentation is impaired by registration jitter). With *Relative Pointing*, feedback on registration jitter is provided, because the cursor position is subject to the same registration jitter as any augmentation.

Thus, we expected the accuracy of *Crosshair* when pointing at physical targets to be subject to registration jitter, as this technique provides no feedback on registration jitter. We also expected the accuracy of the precise mode of *Relative Pointing* not to be subject to registration jitter, since the cursor would provide a certain amount of feedback on the registration jitter condition. Users could use this feedback to adjust their behavior according to the registration jitter condition.

We also evaluated the effect of camera image presentation latency, as this is an altered condition that is orthogonal to registration jitter. Such latency impairs the *Viewpoint* spatial relation and can therefore impair pointing at physical targets (Figure 4.18). We expected *Crosshair* to be more sensitive to such latency than *Relative Pointing*.

This experiment was held on a handheld tablet alone, so as to reduce the number of conditions.

We formulated the following hypotheses:

- H1: When pointing at physical targets, registration jitter impairs the accuracy of *Crosshair* more than that of *Relative Pointing*. *Crosshair* provides no feedback on the registration jitter condition, therefore participants cannot adjust their behavior. With *Relative Pointing*, unlike the physical targets represented on the screen, the cursor is subject to registration jitter (see 4.3.2). The cursor therefore provides feedback on the registration jitter condition. However, registration jitter can be detrimental to visual perception and may therefore still impair pointing accuracy.

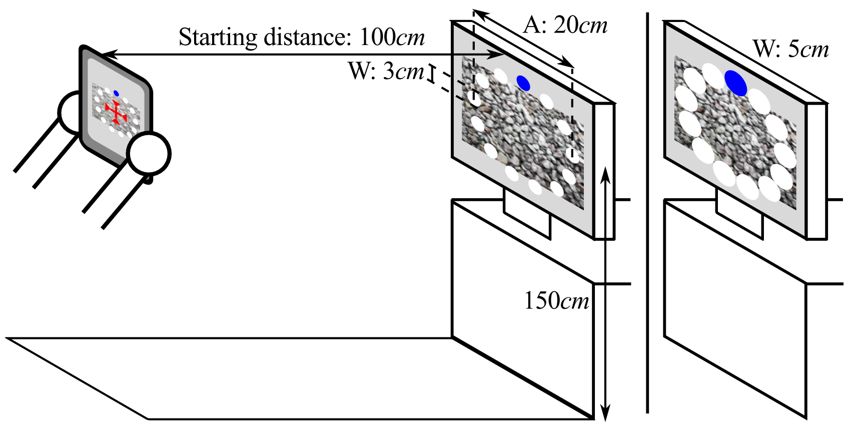


Figure 4.19: Set-up of experiment 4.

- H2: Camera image presentation latency affects the target acquisition duration of *Crosshair* more than that of *Relative Pointing*. Cursor displacement in *Relative Pointing*, which are controlled by finger strokes, are not affected by such latency, unlike the control method for screen-centered *Crosshair*.

4.3.4.1 Procedure and design

The procedure followed was similar to that of the previous experiment. The main difference was that the targets were physical rather than digital (i.e., the targets were visible on the remote image, not just visible on the handheld device's screen as in the previous experiment). To make this possible, the physical image recognized by the vision-based tracking was displayed on a remote screen rather than being printed on paper. Targets were then displayed on this remote screen (Figure 4.19).

This experiment was carried out by applying the cyclical multi-direction pointing task paradigm of ISO9241-9 [Soukoreff and MacKenzie, 2004], adapted to a handheld AR set-up with physical targets (Figure 4.19). A remote screen displaying a textured image (the same as in the previous experiment), with 13 targets arranged in a circle, was positioned with its center 1.5m from the ground. The targets to acquire were highlighted in blue and always in the same following order: top target first, followed by the target opposite and slightly clockwise from the one selected previously, and so on. In the event of a target acquisition failure, the target in question turned dark red, otherwise it reverted to white. The ain was to provide participants with an immediate feedback on their success or failure.

Participants performed the task while standing in front of the remote screen, holding the device in portrait mode. Before each block, participants had to place the handheld device 1m±5cm away from the physical image according to the instructions displayed on the screen. These instructions were hidden as soon as participants acquired the first target. The experimental results enabled us to check the actual distance from the remote image when the task was being performed.

We tested two registration jitter conditions, as in the previous experiment: (1) The underlying tracking system's default set-up, and (2) Extra artificial translational noise added to the relative position of the physical image (with a pseudo-normal distribution of mean 0 and standard deviation 5mm).

We also tested two camera image presentation latency conditions: (1) The default set-up of the underlying operating system and tracking system as is (within a range of [0.08-0.12]s), and (2) Extra camera image presentation latency equivalent to a 2-frame delay (an extra delay of approximately 67ms, as the frame rate was around 30 frames/s, giving an overall latency within a range of [0.15-0.18]s). In the apparatus section, we will detail camera image latency is measured (see 4.3.4.2).

We did not combine the registration jitter and latency conditions, which resulted in three *Conditions* being evaluated: (1) The system as is (*Default* condition), (2) Extra artificial translational noise (*Jitter* condition), and (3) Extra camera images display latency (*Latency* condition).

We used a single movement Amplitude (A) (20cm on the remote screen, as in the previous experiment) and two target Widths (W) (3cm, as in the previous experiment, and 5cm on the remote screen). The Index of Difficulty ($ID = log_2(A/W+1)$) of these tasks was 2.9 bits for W=3cm and 2.3 bits for W=5cm. From a distance of 1m±5cm, A was within a range of [4.6-5.2]cm on the screen of the tablet, W=3cm was within a range of [0.7-0.8]cm on the screen and W=5cm was within a range of [1.1-1.3]cm.

We used a within-subject experimental design with repeated measurements. *Technique*, *Condition* and *Width* were the independent variables. The presentation orders of both *Technique* and *Condition* were counterbalanced across participants using Latin square. For each *Technique*, participants performed all three *Conditions* first with W=3cm and then with W=5cm.

The within-subject experimental design was as follows:

2 *Techniques* x 2 *Widths* x 3 *Conditions* x 12 *Targets* = 144 acquisitions per subject.

For each *Technique*, participants first performed three practice blocks (one for each *Condition*) with the smaller target width (i.e., 3cm), resulting in 72 extra acquisitions:
2 *Techniques* x 3 *Conditions* x 12 *Targets* = 72 acquisitions per subject for practice.

4.3.4.2 Apparatus and Participants

Apparatus. As in the previous experiment, we used an iPad2 (with an updated operating system: iOS version 7.0.3) and a similar ad-hoc application (using Vuforia SDK[12] version 2.6.8).

Measurement of latency. We wished to measure the effective camera image presentation latency, which is the delay between the moment the camera image is captured and the moment it is displayed on the handheld device's screen. To do so, we displayed one new pattern per frame on a remote screen (Figure 4.20-Left). We then took pictures of both the handheld device's screen and the remote screen (Figure 4.20-Right). In the pictures, we counted the difference between the black square displayed on the remote screen and that displayed on the handheld device's screen.

[12]https://www.vuforia.com/

This allowed us to measure the camera image presentation latency as a number of remote screen's frames. Due to camera exposure time (of both that of the handheld device's camera and that of the camera taking the pictures) and possibly also screen persistence (both that of the handheld device's screen and that of the remote screen), there is not just one dark square in the pictures (on both screens). Here, we used the difference between the oldest square on the remote screen and the most recent square on the handheld device's screen to compute a conservative estimate of the camera image presentation latency of the handheld device's AR application.

We used a monitor with a 120 Hz refresh rate. We took pictures at $1/250^{\text{th}}s$.

We took 10 pictures with the experimental application for each of the two latency conditions. We measured camera image latencies within a range of [0.08-0.12]s for the *Default* condition and within a range of [0.15-0.18]s for the *Latency* condition. The difference between the two conditions was consistent with the extra delay of two frames at $30Hz$ (i.e., $67ms$).

This is a crude method of measuring camera presentation latency. A more precise measurement would require more latency samples to be gathered, so as to build a histogram of latencies. This would require automated vision-based processing of the captured images as in [Sielhorst et al., 2007]. Nonetheless, our measurement provides a first estimate of the end-to-end camera image presentation latency.

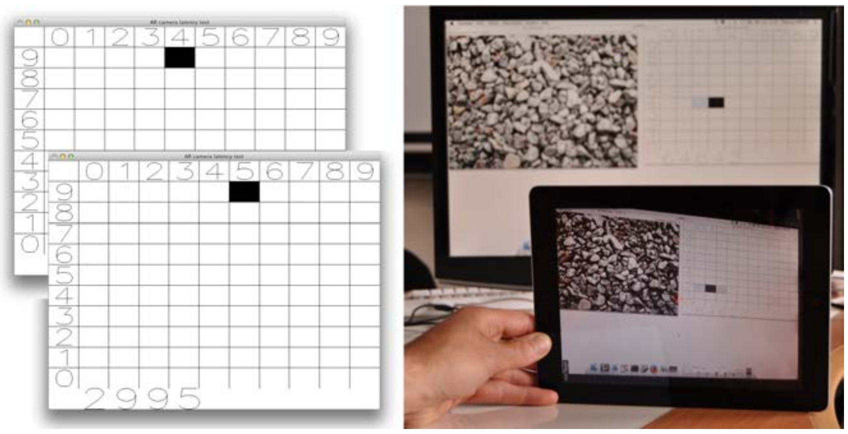


Figure 4.20: Camera image latency test: (Left) Frame displayed on the remote screen. (Right) Picture of both the handheld device and the remote screen.

Participants. Fifteen unpaid participants from our institution (1 left-handed; 8 females; age: [20-49] years, mean: 30.8 years), who did not participate in the previous experiment, took part in this experiment. All but one participant used a touch-based handheld device on a regular or daily basis, eleven participants had used a handheld tablet before and eight had used an AR application.

4.3.4.3 Results

The distance between the remote screen and the tablet during target acquisitions ranged from $88cm$ to $113cm$ (1st quartile: $96cm$, 3rd quartile: $103cm$).

We explored the effect of *Technique*, *Width* and *Condition* by analyzing the *Errors* and *Duration* dependent variables. To explore the effect of *Condition*, we separately analyzed the effect of *Registration jitter* and camera image *Latency* by comparing them against the *Default* condition.

We recorded 2,160 target acquisitions and kept them all for the analysis.

The overall error rate was 4.3%. The overall mean target acquisition duration was 1.64±0.52 seconds[13]. Table 4.6 summarizes the values of the dependent variables for each condition. Table 4.7 summarizes the dependent variables for each factor.

Table 4.6: Error rates, duration (mean ± standard deviation).

Technique	Condition	Width	Error rate (%)	Duration (s)
Crosshair	Default	3cm	6.7	1.79 ± 0.47
		5cm	1.1	1.43 ± 0.33
	Jitter	3cm	16.7	1.80 ± 0.47
		5cm	4.4	1.42 ± 0.52
	Latency	3cm	8.3	2.09 ± 0.62
		5cm	1.7	1.57 ± 0.41
Relative Pt.	Default	3cm	2.2	1.65 ± 0.39
		5cm	0.5	1.34 ± 0.51
	Jitter	3cm	4.4	1.90 ± 0.47
		5cm	1.1	1.49 ± 0.41
	Latency	3cm	2.7	1.75 ± 0.37
		5cm	2.2	1.47 ± 0.62

First we will detail the results for *Registration jitter*, which were obtained by comparing the *Registration jitter* and *Default* conditions. We will then detail the results for camera image presentation *Latency*, which were obtained by comparing the *Latency* and *Default* conditions. Finally, we will discuss the subjective impressions of the participants.

Table 4.7: Error rates, duration (mean ± standard deviation) for each factor.

Factor	Error rate (%)	Duration (s)
Overall	4.3	1.64 ± 0.52
Crosshair	6.5	1.68 ± 0.53
Relative Pt.	2.2	1.60 ± 0.51
W:3cm	6.8	1.83 ± 0.49
W:5cm	1.8	1.45 ± 0.48
Default	2.6	1.55 ± 0.47
Jitter	6.7	1.65 ± 0.51
Latency	3.7	1.72 ± 0.57

Registration jitter results. We compared target acquisition with and without artificial *Registration jitter*, in terms of *Errors* and *Duration* (Figure 4.21-Left).

Errors. We tested the dependence between *Errors* and the different factors using Pearson's Chi-squared test with Yates' continuity correction. Across all the *Default* and *Registration jitter* observations, we found significant dependences for *Technique*

[13]m+/-sd gives the mean m and standard deviation sd.

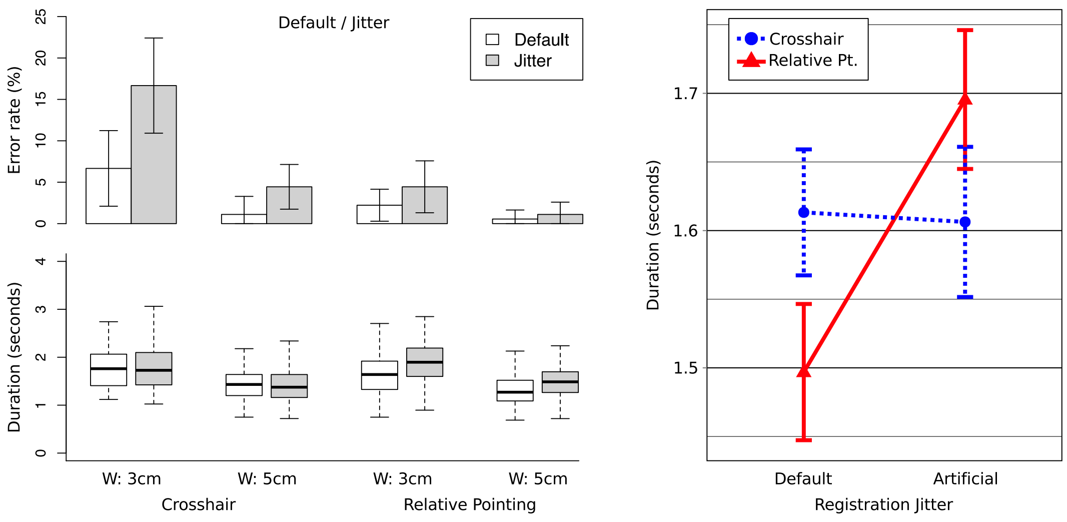


Figure 4.21: Comparison between *Default* and artificial *Registration Jitter*. (Left) Barplots of error rates (%) and 95% confidence intervals of error rates aggregated by participant, and boxplots of target acquisition durations (in seconds) for each *Technique* and each target *Width*. (Right) *Technique* x *Registration Jitter* interaction graph with mean and 95% confidence intervals computed without aggregation by participant.

(χ^2=20.29*[14]), *Registration jitter* (χ^2=12.27, p<.001) and *Width* (χ^2=25.04*). We further analyzed the dependence between *Errors* and *Registration jitter* for each *Technique*. We found a significant dependence for *Crosshair* (χ^2=10.96, p<.001, ϕ=0.12) but not for *Relative Pointing* (χ^2=1.09, p=0.30, ϕ=0.04). A post-hoc power analysis indicated a power of 0.76 for small effect size (0.1) and a power of almost 1 for medium effect size (0.3).

Duration. We performed a 2 x 2 x 2 (*Technique* x *Registration jitter* x *Width*) within-subject analysis of variance on the median aggregate repetition duration with participant as a fixed factor. We found significant main effects for *Width* ($F_{1,14}$=232.8*) and *Registration jitter* ($F_{1,14}$=15.71, p<.01). We also found significant interactions for *Technique* x *Registration jitter* ($F_{1,14}$=11.99, p<.01) and *Registration jitter* x *Width*, though with p<.05 ($F_{1,14}$=4.678).

Regarding *Width*, a paired t-test found a mean difference of 0.38 seconds (95% confidence interval (CI): [0.33-0.43] s; t_{59}=16.284*). For *Registration jitter*, we found a mean difference of 0.09 seconds (95% CI: [0.04-0.14] s; t_{59}=3.712; p<.001). To further analyze the *Technique* x *Registration jitter* interaction we ran a pairwise t-test with Holm correction. The only significant difference was found between the *Default* condition and the artificial *Registration jitter* condition of *Relative Pointing*, with a mean of differences of 0.18 seconds (95% CI: [0.13-0.24] s; t_{59}=6.74*). Figure 4.21-Right shows this interaction for data without aggregation by participant.

Discussion. We expected *Registration jitter* to impair the accuracy of *Crosshair* more than that of *Relative Pointing* (H1), as the latter provides feedback on the registration jitter condition.

[14]* indicates p<.0001.

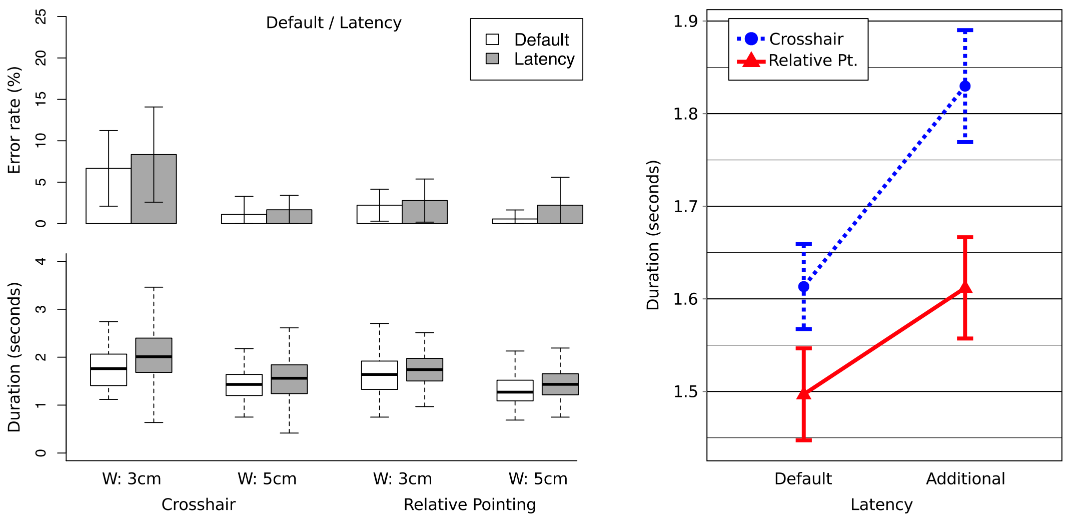


Figure 4.22: Comparison between *Default* and additional camera image presentation *Latency*. (Left) Barplots of error rates (%) and 95% confidence intervals of error rates aggregated by participant, and boxplots of target acquisition durations (in seconds) for each *Technique* and each target *Width*. Note: The white barplots and box plots represent the same data as in Figure 4.21. (Right) *Technique* x *Latency* interaction graph with mean and 95% confidence intervals computed without aggregation by participant.

The results indicate that the error rate of *Relative Pointing*'s precise mode was less affected by *Registration jitter* than that of *Crosshair*. However, *Registration jitter* impaired target acquisition duration in the case of *Relative Pointing*. This suggests that, with *Relative Pointing*, participants adjusted the speed-accuracy trade-off according to *Registration jitter*. These results support hypothesis H1 and are consistent with those of the previous experiment, although in this experiment we used two target *Widths* and the targets were physical rather than digital.

In the previous experiment, which was based on digital targets, our explanation for the greater accuracy of *Relative Pointing* under the artificial *Registration jitter* condition was the fact that the cursor was stabilized in the frame of reference of the digital targets. For this experiment, we suggest a different explanation. Indeed, under the artificial jitter condition the *Relative Pointing* cursor is no longer stabilized in the frame of reference of the physical targets. However, the cursor provides feedback on the registration jitter condition. Participants again appeared to use this information to adjust their behavior when using *Relative Pointing*.

Camera image latency results. We compared target acquisition, with and without additional camera image presentation *Latency*, in terms of *Errors* and *Duration* (Figure 4.22-Left).

Errors. We tested the dependence between *Errors* and the different factors using Pearson's Chi-squared test with Yates' continuity correction. Across all the *Default* and *Latency* observations, we found significant dependences for *Technique*, though with $p<.02$ (χ^2=6.490) and *Width* (χ^2=14.035, $p<.001$), but we did not find a significant dependence on *Latency* (χ^2=1.100; p=0.29). A post-hoc power analysis indicated a power of 0.97 for small effect size (0.1).

Duration. We performed a 2 x 2 x 2 (*Technique* x *Latency* x *Width*) within-subject analysis of variance on the median aggregate repetition duration with participant as a fixed factor. All main effects were significant: *Latency* ($F_{1,14}$=44.93*), *Width* ($F_{1,14}$=240.4*) and *Technique*, though with p<.05 ($F_{1,14}$=7.332). The *Technique* x *Latency* interaction was also significant, though with p<.05 ($F_{1,14}$=5.784).

We further analyzed main effects using paired t-tests. Regarding *Technique*, *Relative Pointing* was on average 0.18 seconds faster than *Crosshair* (95% CI: [0.13-0.23] *s*; t_{59}=6.98*). Regarding *Width*, large targets were on average 0.37 seconds faster to acquire than small targets (95% CI: [0.32-0.42] *s*; t_{59}=14.22*). Regarding *Latency*, the *Default* condition was on average 0.18 seconds faster than the condition with additional *Latency* (95% CI: [0.13-0.23] *s*; t_{59}=6.983*).

We analyzed the *Technique* x *Latency* interaction using a pairwise t-test with Holm correction. All differences were significant except between *Crosshair* in the *Default* condition and *Relative Pointing* both with and without additional *Latency*. Figure 4.22-Right shows this interaction for data without aggregation by participant. This figure indicates not only that *Crosshair* was slower than *Relative Pointing*, but that it was also more impaired by *Latency*. For *Crosshair*, the mean duration difference between the *Default* and *Latency* conditions was 0.23 seconds (95% CI: [0.16-0.32] *s*; t_{29}=6.22*), while it was 0.12 seconds for *Relative Pointing* (95% CI: [0.06-0.18] *s*; t_{29}=3.86, p<.001).

Discussion. We expected camera image presentation *Latency* to be more detrimental to target acquisition duration with *Crosshair* than with *Relative Pointing* (H2). In the case of the latter, cursor control was not affected by the extra latency.

Overall, we found that *Relative Pointing* was less error prone and faster than *Crosshair* in the conditions tested. The results indicate that additional *Latency* did not strongly impair the error rates of either technique. Unlike error rate, duration was impaired by *Latency* for both techniques, and *Crosshair* was more sensitive to *Latency* than *Relative Pointing.* This only partly supports hypothesis H2, as *Latency* has an effect on the target acquisition duration of *Relative Pointing.* However, this effect is smaller than for *Crosshair.*

We did not expect *Relative Pointing* to be sensitive to camera image presentation *Latency* as it should primarily impair absolute pointing in the physical space, as with *Crosshair.* Further study is required to explain this effect.

User preferences. After the experiment, we asked participants which technique they found to be the fastest, the most accurate and which one they preferred overall over the course of the experiment. Out of the fifteen participants, eight said that *Crosshair* was faster and six said that *Relative Pointing* was faster. One participant said that both techniques seemed equally fast. Most of the participants (13/15) said that *Relative Pointing* was the more precise technique and only two participants said *Crosshair.* Eleven participants preferred *Relative Pointing* overall, while three preferred *Crosshair.* One participant had no preference for either.

Unlike in the previous experiment, some participants complained about the 1:1 Control-to-Display gain we used. One participant said that all the errors she made with *Relative Pointing* were due to her finger strokes reaching the edge of the screen. Two other participants said that they would have preferred a higher Control-to-Display gain.

4.3.4.4 Discussion

As in the case of the experiment based on digital targets, this experiment indicated that, when acquiring physical targets, the accuracy *Relative Pointing*'s precise mode was less sensitive to registration jitter than that of *Crosshair*. This can be explained by the fact that, unlike *Crosshair*, *Relative Pointing* provides feedback on the registration jitter condition. Indeed, with *Relative Pointing*, participants adjusted the speed-accuracy trade-off according to the registration jitter that wa translated on the screen in the form of cursor movements.

Adding camera image presentation latency was detrimental to both techniques in terms of target acquisition duration. However, *Relative Pointing* was less sensitive to such latency than *Crosshair*.

4.4 Conclusion

We have presented two sets of two experiments (Table 4.8) that provide some insight into the proposed techniques, namely *Shift&Freeze* and *Relative Pointing*.

Table 4.8: Two sets of two experiments.

Comparative studies: *Relative Pointing* and *Shift&Freeze*
1. Controlled experiment
2. More realistic context

Controlled experiments of *Relative Pointing*: Degrading conditions
1. Pointing at digital targets
2. Pointing at physical targets

The aim of the first set of experiments was to compare the two proposed techniques with two baseline techniques, namely *Direct Touch* and *Crosshair*. These experiments were held using a handheld tablet. The first experiment was a controlled experiment based on an abstract pointing task geared towards comparing the accuracy of the different techniques when acquiring small targets. The second experiment was held in a less controlled environment and based on a more realistic task. Its aim was to collect participant feedback and back-up the results of the first experiment in a less controlled context.

The results indicated that participants tended to use the precise mode offered by *Shift&Freeze* and *Relative Pointing*. These two techniques were preferred over the baseline techniques. *Shift&Freeze* was preferred over *Relative Pointing*, when we had expected the handheld tablet form factor to be detrimental to *Shift&Freeze* due to the trade-off between holding the device and accessing the touch screen. The fact that participants were accustomed to *Direct Touch* interaction might explain this result.

The results also indicated that *Shift&Freeze* and *Relative Pointing* were more precise than the two baseline techniques. However, they took longer to operate. *Direct Touch* was the worst technique in terms of precision.

After this first set of experiments, we focused on *Relative Pointing.* We evaluated its two modes (i.e., *Crosshair* and the precise mode of *Relative Pointing*) under an artificial registration jitter condition and using both digital and physical targets. The results indicated that *Relative Pointing* was less sensitive to registration jitter than *Crosshair* for both types of target. For digital targets, this can be explained by the fact that the cursor was subject to the same registration jitter as the digital targets. This meant that the cursor was stable relative to the digital targets. This was not true for the physical targets. However, the *Relative Pointing* cursor provided feedback on the registration jitter condition, which users could use to adjust the speed-accuracy trade-off.

Moreover, we held the first experiment on both tablet and smart phone form factors. Again, the results indicated that *Relative Pointing* was less sensitive to device form factor than *Crosshair.*

In the second experiment, we evaluated both techniques with additional camera image presentation latency. The results indicated that both techniques were affected by such latency, but that, again, *Relative Pointing* was less sensitive than *Crosshair.*

Additional experiments could be performed to evaluate the techniques we proposed in greater depth. It would be interesting to further investigate the effect of different dynamic transfer functions when using *Relative Pointing.* It would also be useful to evaluate the different techniques when it comes to drawing on physical objects, so as to further assess the strengths and weaknesses of each technique. Indeed, while pointing requires the discrete specification of a single point, drawing is based on continuous strokes.

However, the experimental results backed up our belief that both *Shift&Freeze* and *Relative Pointing* improved the pointing precision for handheld AR applications, which was our primary aim when designing the techniques (see Chapter 3). These are not target-aware techniques, therefore they can be applied to different contexts. Unlike *Shift&Freeze*, *Relative Pointing* is not based on a freeze-frame technique. Freezing the frame breaks the *Viewpoint* spatial relation, and there is a *Viewpoint* discontinuity when returning to the live camera image. Instead of breaking the *Viewpoint* relationship, *Relative Pointing* uses the frame of the physical object to stabilize the cursor. This allows dynamic physical objects to be supported and therefore enables use cases such as interaction with a public display via the handheld device camera. This would not be possible with *Shift&Freeze*. *Relative Pointing* therefore improves pointing precision while maintaining the spatial relation between the on-screen content and the physical surroundings.

Thus, we consider *Relative Pointing* to be a useful pointing technique for handheld AR when accuracy matters.

Those experimental studies highlighted the fact that *Relative Pointing* is more resilient to both *Viewpoint* and *Registration* instabilities as the cursor shares the frame of reference of the *Augmentation* and provides an implicit feedback about registration jitter. It would be very interesting to organize existing handheld AR techniques according to which feedback they provide and how they are resilient to instabilities that can affect the two spatial relations of our design space (see Chapter 2).

Part III

Development of Interaction Techniques

Chapter 5

Toolkit and demonstrators

Chapter Content

5.1 Introduction

This thesis is part of the international research project AMIE: *Augmented Mobile Interactive Experience: Application to Maintenance Services* ("Projet ANR blanc international", see Chapter 1). The AMIE project's members are AIST-Tsukuba and our own team as academic partners as well as two industrial partners, Schneider France and Digital Japan.

The project is devoted to handheld AR. AIST boasts expertise in localization, registration and tracking, while our team focuses on interaction techniques. The two industrial partners have defined a use case scenario revolving around machine maintenance in a production plant. Their role is to specify their machine maintenance requirements and to have their maintenance staff evaluate the proposed solution. This provides a real-life context for the techniques designed.

As part of this project we developed two kinds of demonstrator. First, we developed demonstrators for a use case revolving around machine maintenance in a production plant. These demonstrators were presented to maintenance staff internal to the project. We also presented a demonstrator linked up to a fake machine at an industry event (see 5.3.3.3).

We also developed demonstrators for more generic scenarios in conjunction with AIST, an academic partner of the project. These demonstrators were presented at the ISMAR 2011 and ISMAR 2012 conferences (see 5.3.3.1 and 5.3.3.2) and during events held at AIST.

To support the development of these demonstrators, we developed a prototype toolkit dedicated to handheld AR applications. This toolkit has been used and updated throughout the project and the development of the demonstrators.

In this chapter, we will first describe the overall architecture of the toolkit and the primitives it provides. We will then describe the features and interaction we included in the demonstrators, before describing some of the demonstrators themselves.

5.2 Handheld AR toolkit

Our aim was to provide a library of reusable functionalities for the development of AR applications on handheld devices. Thus, our objectives were to support both 2D and 3D graphic rendering, to support different tracking systems and to provide a lightweight and portable library that could run interactively on handheld devices.

To provide a clearer understanding of our toolkit, before describing the latter we will first describe the main components of an application built using our toolkit. We will then explain the implementation choices made before describing the primitives provided by the toolkit.

5.2.1 Software components

Figure 5.1 presents the different software components of an application built using our toolkit. We will now describe the role of the following components:

- Application
- Application container
- Scene-graph
- Tracking

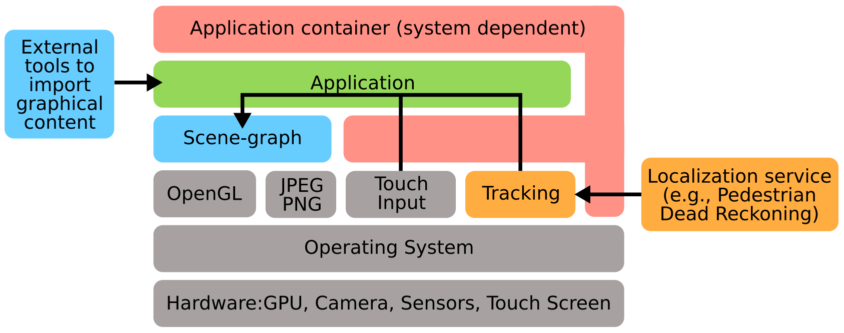

Application container (system dependent)
External tools to import graphical content
Application
Scene-graph
OpenGL
JPEG PNG
Touch Input
Tracking
Localization service (e.g., Pedestrian Dead Reckoning)
Operating System
Hardware:GPU, Camera, Sensors, Touch Screen

Figure 5.1: Software architecture of an application built with the toolkit.

5.2.1.1 Application

The 'Application' component contains the application's system independent code. Its role is to manage the scene graph's life cycle and to perform application-specific behaviors. This component first needs to initialize the 'Scene graph' and then to update its content according to application events. The 'Application' is also in charge of receiving touch input events, tracking events and camera images. It can then update and manage the scene graph accordingly.

The 'Application' uses the functionalities of the 'Application container', the 'Scene graph' and the 'Tracking' system.

5.2.1.2 Application container

The 'Application container' implements operating system-specific functionalities in order to provide the 'Application' with an abstraction of the device resources. An AR application running on a handheld device requires the operating system and devices to provide the following minimum resources:

- Access to a rear-facing camera image (for AR)
- A windowing system and an OpenGL context for rendering
- User input events (i.e., touch events)

These resources are available via system-dependent Application Programming Interfaces (API). Therefore, the 'Application container' is system specific. It provides the 'Application' with an abstraction of the hardware and the operating system.

To seamlessly integrate an application with different operating and tracking systems, the 'Application container' must be in charge of the main loop of the application. The 'Application container' is therefore started first and calls upon the 'Application' according to the life cycle of the application and user events.

5.2.1.3 Scene graph

The 'Scene graph' manages on-screen content and supports the management of interaction. This component provides a scene graph structure that supports both 3D graphics (i.e., meshes) and 2D vector graphics (SVG-like arbitrary paths, text, bitmaps). It also provides certain higher-level ready-to-use widgets such as buttons or menus. It supports picking in a scene for interaction purposes. This component is portable as it relies on standard APIs (i.e., OpenGL) and portable libraries. Device specific dependencies such as management of the OpenGL context are part of the 'Application container'. The 'Scene-graph' must be updated by the 'Application', which is in charge of its life cycle. We will describe the structure of the 'Scene graph' in more depth in section 5.2.3.

5.2.1.4 Tracking support

Our toolkit supports different libraries and systems for estimating the camera position and orientation. The 'Tracking' component is in charge of providing access to such functionalities. Depending on the way in which camera position and orientation are estimated, the underlying library may require access to camera images (for vision-based tracking). The 'Application' can retrieve the current position and orientation estimate from the 'Tracking' system and can then use this information to update the 'Scene graph'. This provides an abstraction of the underlying tracking system.

5.2.2 Implementation

We developed the toolkit with two main goals in mind, both of which governed the choices we made. The first objective of the toolkit was to provide a portable set of reusable components for the development of portable handheld AR applications. The second objective related to the context of handheld AR, which requires applications to run interactively on handheld devices.

To provide a toolkit that would be portable, we minimized dependencies. To ensure that applications would be able to run interactively on handheld devices, we aimed for simple and optimized scene graph implementation.

The toolkit was written in C and we chose OpenGL|ES 1.1 [Aaftab Munshi, 2008] as the rendering backend. We used widely supported and easily portable dependencies (e.g., libpng[1] and ligjpeg[2]).

We based the description of the graphical primitives on open standards: (1) Collada [Barnes and Finch, 2008] for 3D graphical primitives, and (2) SVG [Dahlström et al., 2011] for 2D graphical primitives.

External tools enable such graphics to be imported into the scene graph. These tools were written in Python.

For iOS, the 'Application container' was developed in Objective-C using system-specific APIs. For Windows and Mac OS X, it was developed in C++. It uses the GLFW[3] library to create a window with an OpenGL context and to manage input events. It also uses OpenCV[4] to retrieve camera images.

This implementation allows us to run the toolkit, and therefore our demonstrators, on iOS (for iPad and iPod support), Windows (for Windows tablet support) and Mac OS X (mainly for development purpose).

[1]http://www.libpng.org/pub/png/libpng.html
[2]http://libjpeg.sourceforge.net/
[3]http://www.glfw.org/
[4]http://opencv.org/

5.2.3 Scene graph

A common approach to 2D GUI toolkits is to offer a set of predefined widgets and layouts. User interfaces are then organized in a tree of layout nodes (or views) with widgets as the leaves. This provides a high level of granularity and provides a standardized look and feel. A common approach to 3D toolkits is to offer a scene-graph (i.e., a direct acyclic graph) structure with nodes to arrange objects in 3D space. However, the direct acyclic graph structure can also be used to structure 2D graphics, such as with SVG [Dahlström et al., 2011].

Our toolkit uses a scene graph structure. The toolkit supports both low-level graphic primitives, such as 3D meshes and 2D vector graphic paths, and higher-level widgets such as buttons and menus. It is possible to use low-level graphic primitives (e.g., 2D paths, text) within other primitives. This eases the development of new widgets. For example, a button may use graphic primitives such as text, 2D paths or images to perform rendering, while also managing picking and its own state.

In the following sections, we will describe the key functionalities and primitives available to the scene graph. Finally, we will describe the external tools that make it possible to embed graphical content supported by the scene graph in the applications developed.

5.2.3.1 Functionalities

The scene graph provides the application with two main functionalities. First, it is in charge of rendering graphical content on the screen using OpenGL|ES 1.1 as a graphic backend. The scene graph is not in charge of creating and managing the OpenGL context. Indeed, this relies on system dependent APIs and is therefore managed by the 'Application container' component. Second, the scene graph offers picking support to manage user input events.

Rendering. Rendering is implemented using a scene graph traversal, where each node is asked to render its content. A rendering context that is common to all rendering operations is updated along the traversal.

The standard rendering sequence is as follows:

The application updates the scene graph according to its current state. For example, the application can update the content of a text field displaying the value of a sensor.

The application prepares the OpenGL context and provides the scene graph with the following information:

- The size of the viewport (i.e., the area on the screen in which the rendering is performed);
- In the case of AR content, the current camera image and the extrinsic camera parameters (i.e., camera position and orientation).

The application then calls upon the scene graph's rendering function.

Picking. The scene graph supports two types of picking: (1) a crude and fast type, and (2) a full ray casting.

The crude picking method enables interactive widgets, such as buttons and menus to update their own state. For each interactive primitive, the picking system checks whether the picking position on the screen lies within the bounding box of an interactive primitive projected on the screen. Interaction with only one widget is allowed at a time and picking is performed in the opposite order to the scene graph. Thus, for co-planar overlaid widgets the topmost widget will process the events. One limitation of picking using the scene graph structure is that it does not use depth information to select the interactive widget. The scene graph therefore needs to be carefully designed. In addition, the picking method only informs an object of the picking status (picked or not picked) and does not provide information about the position picked on the object. Nevertheless, this picking method is efficient. Only interactive primitives are checked: this check is based on the bounding box rather than the complete geometry of the primitive. This is efficient and sufficient to support interactive widgets such as buttons and menus.

A second picking method is also available, which implements a ray casting in the scene. However, this requires more computations, since each triangle of 3D meshes needs to be tested for intersection. This picking method is used by the precise mode of the *Relative Pointing* technique (see 3.3.2.2) when pointing at 3D shapes. For each new camera image, *Relative Pointing* performs a ray casting in order to check whether or not the cursor is occluded from the new point of view.

5.2.3.2 Primitives

To describe the augmented scene's content, the toolkit provides a set of primitives allowing both 3D and 2D content creation. These primitives manage both the picking and rendering functionalities. The primitives available make it possible to (1) describe the rendering context, (2) describe graphical content, (3) describe widgets, and (4) group and lay out other primitives.

All primitives have the following common basic attributes:

- A visibility flag checked during graph traversal to enable/disable the rendering and picking behavior of the primitive and its children
- A transformation matrix that stores a transformation in space to be applied to the primitive.

Rendering primitives. These primitives (namely *Viewpoint*, *Camera* and *Light*) are in charge of setting-up the rendering parameters.

The root primitive of a scene graph must be a *Viewport*. The *Viewport* represents the on-screen area in which the scene graph is actually rendered. A single *Viewport* can render multiple scene graphs, which are rendered as different layers. For example, this makes it possible to support a background layer showing the camera image, a 3D augmentation layer that is overlaid on the camera image and a screen layer that displays widgets attached to the screen.

The *Camera* primitive is useful for rendering a scene with a perspective projection. For example, to render the 3D augmentation overlaid on a camera image, the perspective projection must match that of the camera. The *Camera* primitive can be placed anywhere in the scene. *Camera* can either be defined by its field of view or by the intrinsic camera parameters provided by the calibration of the handheld device's camera.

Light primitives control the lighting of the virtual scene. If the scene contains no light node, then lighting is disabled. There are currently two *Light* primitives: (1) *Ambient Light*, which is defined by its color intensity, and (2) *Directional Light*, which is defined by its color intensity and its current orientation in the scene.

Graphic primitives. These primitives (namely *Mesh*, *Material*, *MeshInstance*, *2D Shape*, *Image* and *Text*) actually draw both 3D and 2D content on the screen.

The 3D *Mesh*, *Material* and *MeshInstance* primitives describe 3D meshes as a set of vertices and can also provide their color, normal and texture coordinates. The *Material* primitive describes the texture to use while the *Mesh* primitive describes the geometry of the 3D object. *MeshInstance* matches a *Material* with a *Mesh*. This allows a single geometry (which can take up a fair amount of memory) to be reused with different textures.

The *2D Shape* primitive describes a 2D vector graphic path defining a fill area and a stroke. This primitive can be initialized with a vector graphic path description similar to an SVG path description [Dahlström et al., 2011].

The *Image* primitive describes an image loaded from a file. It enables the rendering of resource images stored as JPEG or PNG files within the scene graph. In figure 5.2, the images of the buttons in the top left-hand corner are rendered with *Image* primitives.

The *Text* primitive describes a line of text. It enables the rendering of a line of text within the scene graph. In figure 5.2, the text in the menus on the right-hand side is rendered with *Text* primitives.

Widgets. These primitives (namely *Button*, *Menu* and *Wedge*) provide higher-level reusable widgets. They embed interaction behavior and update their state when user input events take place. The widgets proposed can be placed either in the screen's 2D space or in the augmented scene's3D space.

The *Button* (Figure 5.2-Left) and *IconButton* (Figure 5.2-Left) primitives describe buttons with either a text label (for the *Button*) or an image (for *IconButton*).

The *Menu* primitive describes a single-level menu that can be reduced to its title or expanded (Figure 5.2-Right).

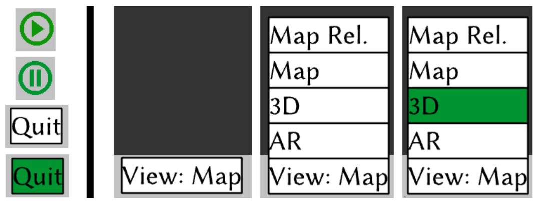


Figure 5.2: Sample buttons and menus: (Left) *Button* (bottom) and *IconButton* (top). (Right) *Menu*: Closed, open and with an item selected.

The *Wedge* primitive (Figure 5.3) describes an off-screen point of interest with a text label that remains visible on the screen (in Figure 5.3 the text label is "Demo") and a triangle indicating the direction and distance of the off-screen object. This is a version of the *Wedge* technique [Gustafson et al., 2008] proposed for 2D maps adapted for an AR context. The *Wedge* technique provides localization information by drawing part of a triangle on the edge of the screen, which points toward an off-screen target. From the portion of the triangle that is visible on-screen, users can infer the off-screen location of the point of interest. This primitive is useful for finding off-screen points of interest.

To adapt this technique to 3D, we first project the location of the points of interest onto the image plane. We then use the *Wedge* technique with the 2D positions obtained for the points of interest, as shown in Figure 5.3. This technique prompts the user to turn the device in the direction of the points of interest. This technique only provides information in the image plane. The user does not know the exact location of the corresponding point of interest in the physical surroundings. Nonetheless, this provides sufficient information to retrieve nearby points of interest.

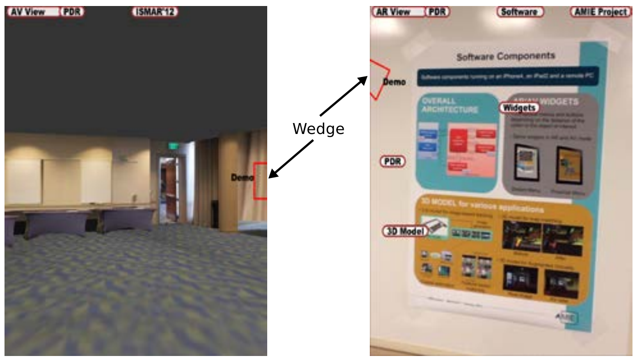


Figure 5.3: *Wedge* (red triangle) indicating an off-screen point of interest: (Left) on a 3D model of the physical surroundings, and (Right) in AR mode. The text label is "Demo" indicating the demo booth.

Layout primitives. These primitives (namely *Node*, *Anchor* and *POI Frame*) are in charge of grouping primitives and managing transformation between the 3D space of the augmented scene and the 2D space of the screen.

The *Node* primitive has a set of children. It allows for multiple primitives to share a common transformation in space (e.g., translate, scale, rotate).

The *Anchor* primitive enables the changes from 3D scene coordinates to 2D screen coordinates, before rendering its children. This is a convenient node that allows billboarding rendering, for example (i.e., adjusting a primitive's orientation so that it is parallel to the screen plane).

Like the *Anchor* primitive, the *POI Frame* primitive enables the change from 3D scene coordinates to 2D screen coordinates, before rendering its children. In addition, it places the rendering of its children in a surrounding shape with a small arrow indicating its position in the augmented scene (Figure 5.4).

Extension. This set of primitives can be extended. More primitives (e.g., new widgets) can be added, provided that they support the two main functionalities of the scene graph (i.e., rendering and picking). New primitives can either stem from a fresh implementation or reuse existing primitives (usually graphic primitives).

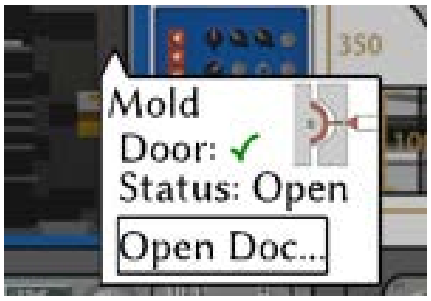


Figure 5.4: *POI Frame* component with *Text* and *Image* primitives and an 'Open Doc..' *Button* widget.

Our toolkit lacks support for layout algorithms. To remedy this issue, standard 2D layouts such as flow or grid layouts would be useful. Moreover, laying out augmentations on the screen both according to their 3D positions and their position and footprint on the screen would be of use for AR applications. Such a layout is known as view management [Bell et al., 2001] and helps avoid visual clutter. While the *Anchor* node allows changing from 3D scene coordinates to 2D screen coordinates, it does not take into account the position of other rendered primitives. A 'view management' node would place its children on the screen according to their position in the 3D scene and in a way that avoids visual clutter on the screen.

5.2.3.3 Importing graphics

We developed external tools to convert graphical content (both 3D and 2D) into source code that defines scene graph nodes. This source code can then be integrated in the application. Collada [Barnes and Finch, 2008] files can be converted into a graph of *Nodes* and *3D Meshes* with *Material* (see 5.2.3.2). Collada is a comprehensive format and we only support a subset of Collada's specifications. However, tools such as Collada Refinery[5] allow Collada primitives to be converted into other Collada primitives. It is also possible to convert an SVG [Dahlström et al., 2011] path into *2D Shapes* nodes (see 5.2.3.2).

The external tools convert the input file into source code that contains the structures describing the graphical content. This approach, which involves the offline conversion of files into source code, is one solution. It is efficient because loading content at application start-up does not require content files to be parsed. This is limited to static resources, since they must be integrated into the application during development. To allow for resources to be loaded at runtime, a parser would need to be integrated in the toolkit.

5.3 Demonstrators

We developed demonstrators with the different partners of the AMIE project, both to explore handheld AR for machine maintenance and for presentation at conferences and events.

[5]https://collada.org/mediawiki/index.php/COLLADA_Refinery

We will first describe the different tracking systems we used. We will then present the key features developed for our demonstrators. Finally, we will describe some of the demonstrators.

5.3.1 Tracking systems

Throughout the project, the demonstrators used different systems and libraries to estimate camera position and orientation so as overlay digital graphics on live camera images. Because the 'Tracking' component (see 5.2.1.4) provides the same API for all tracking systems, changing the tracking system had a minimal impact on the remaining of the demonstrators. The development of different demonstrators with different tracking systems proved the portability of our toolkit, at least from the perspective of the tracking system.

5.3.1.1 Optitrack

Early prototypes used an external Optitrack system[6] to estimate camera position and orientation. The Optitrack system uses infrared cameras placed at fixed positions in the environment to estimate the position and orientation of rigid bodies fitted with infrared reflective spheres. We attached rigid bodies to both the handheld device and the object to be augmented, in order to compute their relative positioning. Such a solution enables for 6DoF tracking but only within the limits of the workspace covered by the infrared cameras. Furthermore, this is not a standalone solution as the environment must be suitably equipped.

5.3.1.2 Pedestrian Dead Reckoning

We then used a Pedestrian Dead Reckoning (PDR) method developed by AIST [Kourogi and Kurata, 2003]. This PDR method uses the self-contained sensors of handheld devices to estimate the user's locomotion. The handheld device running PDR had to be placed on the user's waist or in her/his pocket. We therefore used two handheld devices, one for PDR and the other for the graphical user interface. PDR provides a 2D estimation of the user's position relative to their starting point and their direction relative to the North. Thus, the method does not provide 6DoF camera positioning and does not enable to register computer graphics on the live camera images provided by the handheld device's camera. However, it allows navigation on a map or in a virtual representation of the physical surroundings (i.e., Augmented Virtuality). We used this tracking method and a virtual model of the demo room for a demonstration at the ISMAR 2011 conference (See 5.3.3.1). PDR can also be used as a fallback when other tracking systems (e.g., Optitrack, vision-based camera position estimation) are not available. For example, we combined the PDR and Optitrack systems to extend the area in which the demonstrators could be used.

5.3.1.3 Vision-based tracking

We also developed demonstrators with two vision-based tracking libraries.

[6]http://www.naturalpoint.com/optitrack/

First, in iOS, we used Vuforia SDK[7], a third-party vision-based image tracking library. This allows the demonstrators to run on an iPad on a standalone basis.

Second, AIST developed a vision-based library that provides 6DoF camera position estimation and runs on Windows. We held a first demo at the ISMAR 2012 conference (see 5.3.3.2) with the handheld device sending camera images via the Wi-Fi network to a PC running the tracking library. This allowed the images recognized to be augmented on the handheld device's screen. The demonstrator also used PDR information when the vision-based tracking library did not detect a known image. This allowed users to be guided to points of interest. Such an approach was obviously limited by the bandwidth of the network, since we were sending camera images to a remote PC.

We then ported our toolkit and demonstrators to Windows. This allowed tighter integration of the vision-based tracking library in the end-user application. Thus, we were able to run the demonstrator on a standalone basis on a Windows tablet.

5.3.2 Supported features

The demonstrators we developed during the AMIE project include different features:

- Different representation modes
- Contextual menus
- Annotation authoring

5.3.2.1 Representation modes

Our main aim was to present of digital content using AR, but we also incorporated different representation modes into our demonstrators. By using a 3D model of the physical surroundings, we were also able to develop an Augmented Virtuality (as defined in [Milgram and Kishino, 1994]) mode and a top-view map mode. The interaction techniques (i.e., menus, *Wedges*, *Relative Pointing*) we developed work seamlessly in all three modes.

Augmented Virtuality. In Augmented Virtuality mode, the physical surroundings are represented on the screen by a 3D model rather than live camera images (Figure 5.5-b). This allows the physical surroundings to be represented and extra information and widgets to be overlaid, even if 6DoF camera positioning of the camera is not available. For example, when only PDR data is available, it is still possible to present what the user sees in front of him on the screen (Figure 5.5-b). In addition, when registration is impaired by static offsets, with Augmented Virtuality, the augmentation's registration on the 3D model remains correct, even if the viewpoint is not correct, unlike with live camera images in AR mode. Indeed, in AR mode, the representation of the physical world on the screen is not impaired by registration errors, unlike the digital augmentation (see chapter 2). However, because the 3D model is recorded a priori, it might not be up-to-date with the current state of the physical surroundings as it only represents what has been modeled.

[7]https://www.vuforia.com/

The 3D models of the physical surroundings and the objects were created with an interactive 3D indoor modeler developed at AIST [Ishikawa et al., 2013] (Figure 5.5-a). This modeler enables the off-line interactive creation of 3D models from photographs based on the estimation of geometric constraints.

The interactive modeler exports the 3D models in Collada format. We then convert them into source code describing the 3D graphical primitives of our scene graph, so as to embed them in the application. In Augmented Virtuality mode, we render these 3D models instead of the camera images. Toggling the rendering on the 3D model only requires its visibility flag to be changed (see 5.2.3.2).

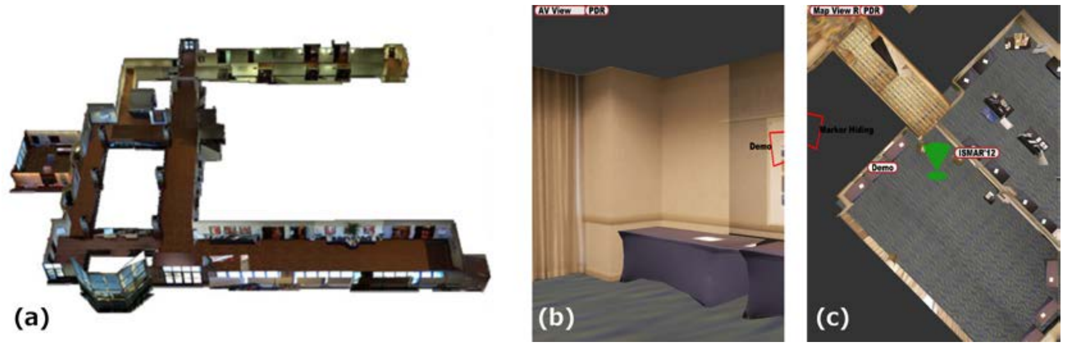


Figure 5.5: Different representation modes. From left to right: (a) 3D model used to represent the physical surrounding (Figure from [Ishikawa et al., 2013]), (b) Augmented Virtuality mode, and (c) Top-view map mode.

Top-view map. The top-view map mode (Figure 5.5-c) is also based on a model of the physical surroundings. This model can either be a 3D model of the physical surroundings as in Augmented Virtuality mode, or a 2D map of the building when a 3D model of the physical surroundings is not available.

We developed the top-view map for guidance (based on PDR) when users are far from the point of interest.

Like in Augmented Virtuality mode, in map mode we render the 3D model instead of the camera images. The difference is that in top-view mode, the viewpoint is placed above the user's current position and looks downward.

5.3.2.2 Contextual menus

In handheld AR, controlling the viewpoint essentially filters the information visible through the viewing frustum in the augmented scene. As depicted by the class *Spatial selection* class of the *Content Selection* axis of our design space's *Augmentation* (see 2.5.2), the position of the viewpoint can be used to further filter the *Augmentation* displayed. For example, DiVerdi et al. [DiVerdi et al., 2004] suggested altering the content of widgets, such as a weather station window according to the distance from which they are seen.

We also explored this kind of *Spatial selection* when designing our so-called 'Contextual Menus' technique. The technique augments a physical object with menus so as to access digital content relating to different parts of the physical object. The menus displayed vary according to the distance between the handheld device and the

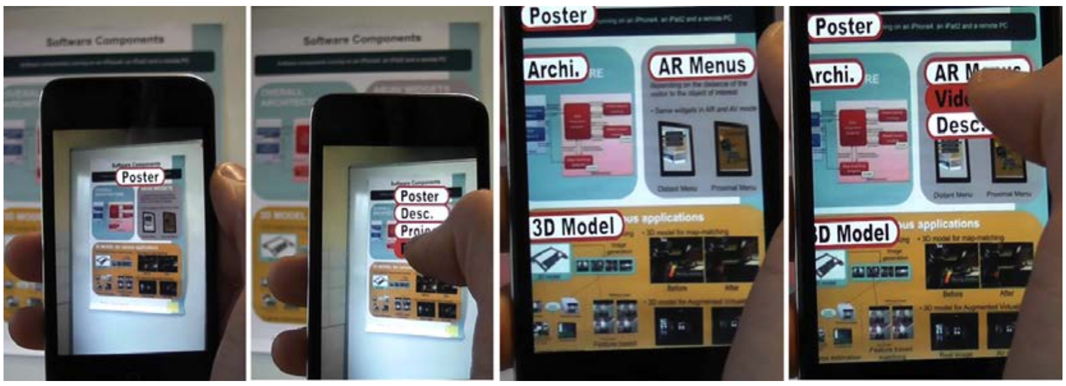


Figure 5.6: Contextual Menus attached to a poster: (Left) From far away, a single global menu is attached to the poster. (Right) From close-up, proximal menus are attached to subparts of the poster and the global menu is available on top of the screen.

physical object. Indeed, the menus attached to a physical object form a hierarchy of menus, rather than a hierarchical menu (Figure 5.6). From far away, the 'distal' global menu is overlaid on the physical object (Figure 5.6-Left). From a closer distance, the global menu is available at the top of the screen and 'proximal' menus are overlaid on different parts of the physical object (Figure 5.6-Right). This allows menus to match the on-screen footprint of objects of interest. In addition, the distal menu is available even when the handheld device is close to the physical object. This is expected to cater for the needs of users, because even when they are focusing on a specific part of the physical object they may need general information. However, our technique does not allow proximal menus to be accessed from far away: this would mean moving in the augmented space while remaining stationary in the physical space.

The Contextual Menus are based on the menus provided by the toolkit. We toggle the visibility flags of the menus according to the user's location.

5.3.2.3 Annotation authoring

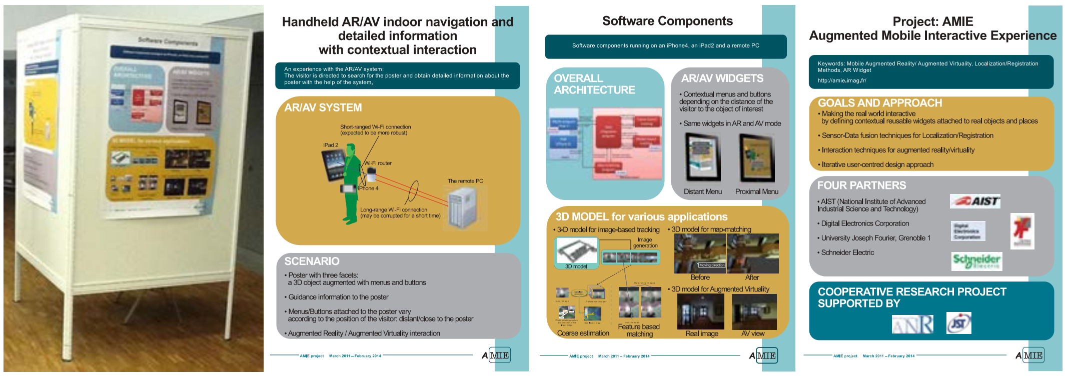
Another feature we developed was annotation authoring. The user can enter an annotation authoring mode in which she/he can position new annotations using either direct touch, *Crosshair* or the *Relative Pointing* technique presented in chapter 3 (see 3.3.2.2). In this mode, users can insert annotations, edit their text or delete annotations.

To compute the position of the annotation created, the different pointing techniques perform ray casting on the 3D model of the physical surroundings (the same model as that used in Augmented Virtuality mode). Newly created annotations are comprised of a *POI Frame* and a *Button*.

5.3.3 Demonstrations

Using the generic features presented in the previous section, we performed several demonstrations during the course of the AMIE project. The physical objects augmented were a poster (presenting the project or a production machine) and a physical mock-up of a production machine. Here we will describe three of the demonstrations.

5.3.3.1 PDR-based Handheld Augmented Virtuality


Handheld AR/AV indoor navigation and detailed information with contextual interaction

An experience with the AR/AV system:
The visitor is directed to search for the poster and obtain detailed information about the poster with the help of the system.

AR/AV SYSTEM

Short-ranged Wi-Fi connection (expected to be more robust)
iPad 2
Wi-Fi router
iPhone 4
The remote PC
Long-range Wi-Fi connection (may be corrupted for a short time)

SCENARIO
- Poster with three facets: a 3D object augmented with menus and buttons
- Guidance information to the poster
- Menus/Buttons attached to the poster vary according to the position of the visitor: distant/close to the poster
- Augmented Reality / Augmented Virtuality interaction

AMIE project March 2011 – February 2014

Software Components

Software components running on an iPhone4, an iPad2 and a remote PC

OVERALL ARCHITECTURE

AR/AV WIDGETS
- Contextual menus and buttons depending on the distance of the visitor to the object of interest
- Same widgets in AR and AV mode

3D MODEL for various applications
- 3-D model for image-based tracking
- 3D model for map-matching
- 3D model for Augmented Virtuality

AMIE project March 2011 – February 2014

Project: AMIE
Augmented Mobile Interactive Experience

Keywords: Mobile Augmented Reality/ Augmented Virtuality, Localization/Registration Methods, AR Widget
http://amie.imag.fr/

GOALS AND APPROACH
- Making the real world interactive by defining contextual reusable widgets attached to real objects and places
- Sensor-Data fusion techniques for Localization/Registration
- Interaction techniques for augmented reality/virtuality
- Iterative user-centred design approach

FOUR PARTNERS
- AIST (National Institute of Advanced Industrial Science and Technology)
- Digital Electronics Corporation
- University Joseph Fourier, Grenoble 1
- Schneider Electric

COOPERATIVE RESEARCH PROJECT SUPPORTED BY

AMIE project March 2011 – February 2014

Figure 5.7: Poster booth and the three posters used in the demonstration at ISMAR 2011.

The goal of the demonstration was to provide users with information to guide them towards a set of three posters presenting the AMIE project. The three posters (Figure 5.7) were augmented with menus and buttons. The menus and buttons provided access to extra information, such as the web sites of the partners and a video presenting the authoring tool for 3D indoor models.

The demonstration used PDR to localize the user (their 2D position and their direction relative to the North). PDR does not enable AR mode to be used, as the position and orientation of the camera are not known. However, it supports Augmented Virtuality and map modes.

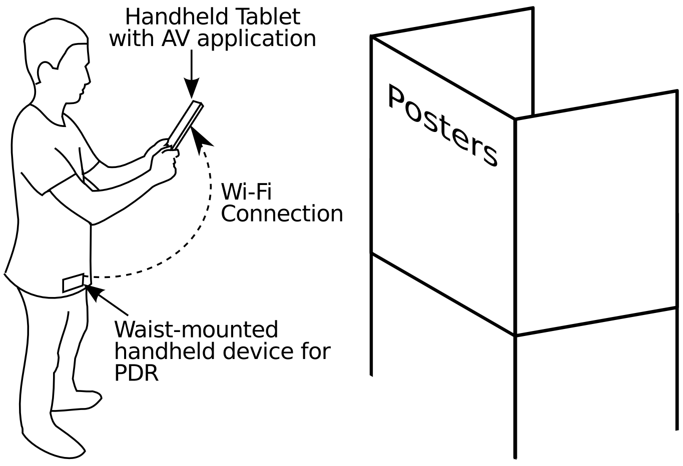

Handheld Tablet with AV application
Wi-Fi Connection
Waist-mounted handheld device for PDR
Posters

Figure 5.8: Set-up of the demonstration held at ISMAR 2011.

Participants were equipped with a waist-mounted handheld device used for PDR (Figure 5.8). They held a handheld tablet displaying the demonstration room either in map mode (Figure 5.9-Left) or from a first-person point of view in the 3D model (Figure 5.9-Right). Guidance to the posters was provided using *Wedges* (see 5.2.3.2

and figure 5.3). The posters were augmented with buttons and menus. We used contextual menus that included a global 'distal' menu common to all three posters and detailed proximal menus for each part of the posters (Figure 5.6).

This demonstration was presented at the ISMAR 2011 conference.

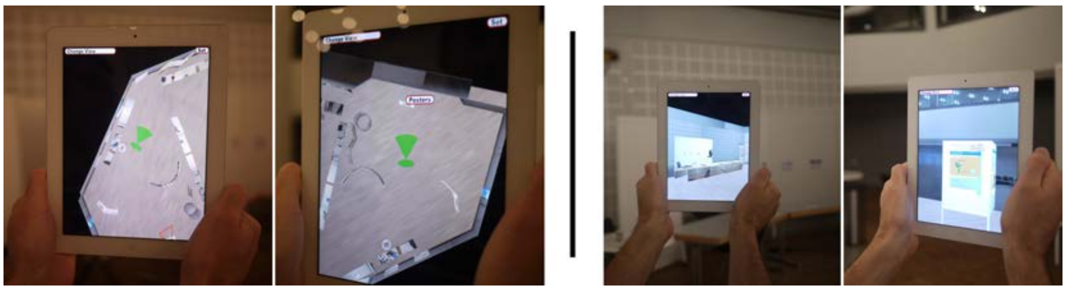


Figure 5.9: Pictures of the demonstration held at ISMAR 2011: (Left) Two pictures of top-view map mode in two different positions. (Right) Two pictures of Augmented Virtuality mode in two different positions.

5.3.3.2 Handheld AR/AV combining PDR and frame-based tracking

The goal of this demonstration was the same as the previous one, except that this time both AR and Augmented Virtuality were possible (Figure 5.11). Indeed, this demonstration combined PDR relative tracking with vision-based keyframe recognition running on a remote PC (Figure 5.10). Vision-based tracking enabled augmented content to be registered on the live camera images when posters were recognized. This demonstration was presented at the ISMAR 2012 conference.

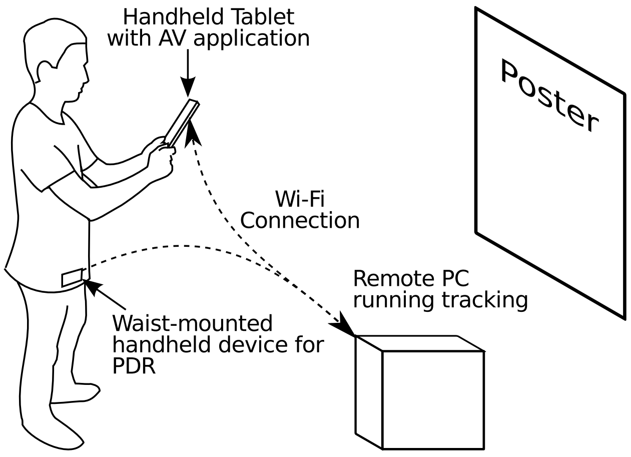


Figure 5.10: Set-up of the demonstration held at ISMAR 2012.

5.3.3.3 Maintenance Demonstration

We held a demonstration at an industrial event in Japan in November 2013. Like with the previous demonstrations, this demonstration covered AR, Augmented Virtuality and map modes. Unlike the previous demonstrations, which augmented posters, this

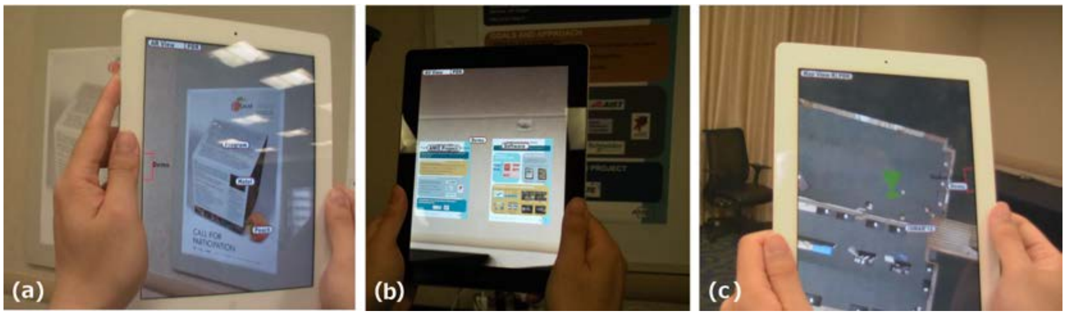


Figure 5.11: Pictures of the demonstration held at ISMAR 2012. From left to right: (a) AR mode in front of a poster, (b) Augmented Virtuality mode in front of posters, and (c) Map mode.

demonstration augmented a mock-up of a plastic injection machine. The augmentations provided information about the machine (Figure 5.12). For the different parts of the machine, the augmentations reported the current (dummy) states of the corresponding sensors (e.g., whether or not a door is opened, the internal temperature). The augmentations also provided access to the associated documentation.

This demonstration ran on a Windows tablet PC on a standalone basis and used the vision-based tracking library developed by AIST. This demonstration proves the portability of our toolkit, since the two previous demonstrations ran on iOS only, while this one runs on both Windows and iOS.

In the following version of this demonstrator, we integrated the positioning of new annotations on the machine.

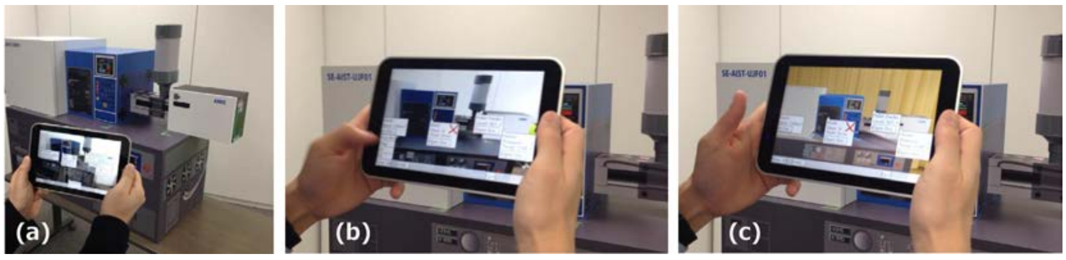


Figure 5.12: Maintenance demonstration. From left to right: (a) Mock-up machine and Windows tablet, (b) AR view of the mock-up machine with augmentations, and (c) Augmented Virtuality mode, showing the 3D model of the machine with augmentations.

5.3.3.4 Other demonstrations

Other demonstrations sharing similar features but using different models and posters were held at various events including:

- 19th International Displays Workshop in 2012
- Location Business Japan 2014
- 25th Design Engineering & Manufacturing Solutions Expo in 2014

As part of the AMIE project, we developed other demonstrators for industrial production plant machine maintenance scenarios. These were tested and discussed during a focus group with maintenance staff in order to check the usefulness and usability of the handheld AR applications in the context of machine maintenance.

5.4 Conclusion

As part of the AMIE project geared towards exploring handheld AR interaction in the context of machine maintenance in production plants, we developed a prototype toolkit to support the development of handheld AR applications. This toolkit is based on a scene graph structure and aims to be a lightweight and portable solution that runs interactively on handheld devices. It offers both a set of low-level primitives and higher-level widgets allowing new ad hoc widgets to be designed and existing ones to be reused. The portability of the toolkit has been proved since the demonstrators run on both Windows and iOS and use different tracking systems.

We used this toolkit throughout the project to develop demonstrators for an industrial use case, for the evaluation of handheld AR with maintenance staff and for demonstrations during conferences and events. The demonstrators developed supported three modes: AR mode, Augmented Virtuality mode and top-view map mode. We adapted the widgets registered according to the distance at which the user was standing from the physical object in question. From far away, a single overview menu was displayed, while from a closer distance, menus or buttons were registered to more specific parts of the object. Finally, we used *Relative Pointing* (see chapter 3) for annotation authoring, so as to position newly created annotations. This functionality is useful in dense physical environments where there are several buttons or a set of electrical switches.

Although both the toolkit and the demonstrators are prototypes, they have proved to be useful during the project. The overall architecture has successfully minimized the impact of changing the tracking system and the operating system on the demonstrators. The toolkit has eased the development of demonstrators and made it possible to determine the essential basic building blocks for handheld AR applications. The demonstrators have also allowed feedback to be obtained from users and maintenance staff internal to the project.

The design and development of the toolkit was mostly driven by practical and technical considerations to ease the development of demonstrators during the AMIE project. We should have positioned our toolkit relatively to the other toolkits developed for AR but also relatively to GUI toolkits and to other solutions allowing to provide handheld AR content such as AR web-browser. Moreover, while this toolkit has been used by colleagues at Schneider to develop other demonstrators during the AMIE project, it has not been evaluated.

Chapter 6

Conclusion

Conclusions and Prospects for Future Work

Chapter Content

6.1 Introduction

Within the context of handheld Augmented Reality (AR), we focused on the spatial constraints and relations of interaction techniques. By handheld AR, we mean the use of a handheld device as a physical magic lens that merges digital information with the representation of the physical surroundings displayed on the handheld device's screen. Interaction in such context provides an opportunity to present digital information in the physical context to which it belongs. Yet, digital/physical spatial relations can impair the user's interaction with the system. Our aim was therefore to explore how such spatial relations can be relaxed so as to improve interaction while maintaining the digital-physical colocation, which lies at the core of AR.

We addressed this question by providing two contributions.

First, we characterized the entities comprising the on-screen content of handheld AR applications, together with their spatial relations so as to study the implications for interaction.

Secondly, we designed different techniques geared towards improving pointing precision when using handheld AR applications. Pointing is a basic generic task. Indeed, pointing is useful for several tasks, such as adding a new annotation anchored to the physical world and selecting an existing annotation. Pointing in the context of handheld AR is impaired not only by the touch screen's limitations, but also by the relationship between the on-screen content and the physical surrounding.

We also developed a prototype toolkit. We used the toolkit to develop demonstrators both for presentation at conferences and for industrial machine maintenance scenarios.

Figure 5 sums up our contributions.

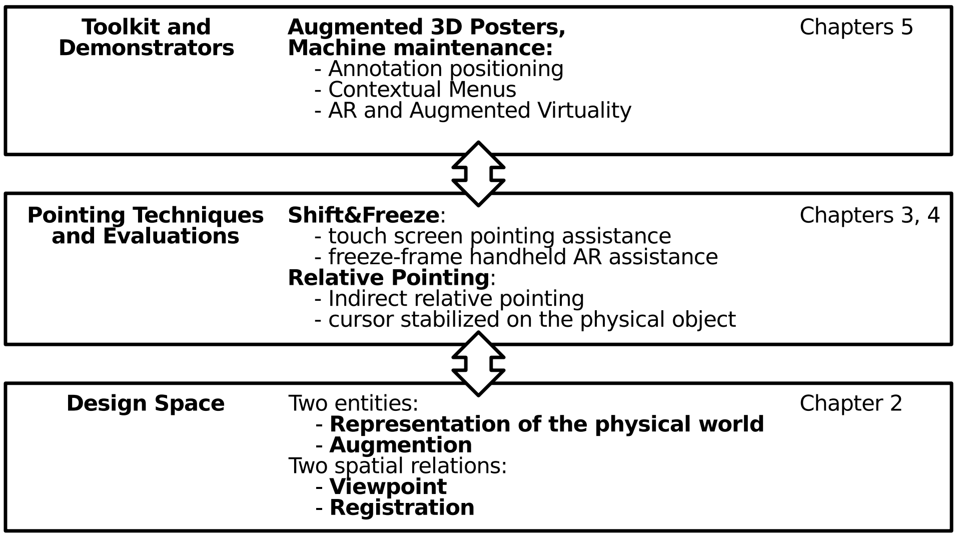

Toolkit and Demonstrators
Augmented 3D Posters, Machine maintenance:
- Annotation positioning
- Contextual Menus
- AR and Augmented Virtuality
Chapters 5
Pointing Techniques and Evaluations
Shift&Freeze:
- touch screen pointing assistance
- freeze-frame handheld AR assistance
Relative Pointing:
- Indirect relative pointing
- cursor stabilized on the physical object
Chapters 3, 4
Design Space
Two entities:
- Representation of the physical world
- Augmention
Two spatial relations:
- Viewpoint
- Registration
Chapter 2

Figure 6.1: Summary of contributions

Below, we will sum up our contributions in terms of: (1) the characterization of handheld Augmented Reality (AR) entities and the spatial relations between their frames of reference, (2) the design and evaluation of pointing techniques geared towards overcoming both the limitations of the handheld AR context and issues relating to touch screen input precision, and (3) the toolkit and demonstrators we developed as part of the AMIE project. We will then present the prospects for future work.

6.2 Contributions

Our contributions are three-fold: (1) a design space for handheld AR, (2) the design and evaluation of pointing techniques, and (3) a prototype toolkit to support the development of handheld AR applications (as part of the AMIE project that explored the use of a handheld AR set-up for machine maintenance in production plants).

6.2.1 Design space

We proposed a design space for handheld AR on-screen content (see Chapter 2) that includes two components (Figure 6):

- The *Representation of the physical world*, which defines the elements of the physical surroundings that are displayed on the handheld device's screen (i.e., camera images). It allows the viewpoint to be mapped in the physical surroundings.
- The digital *Augmentation*, which is the representation of digital content that is not a *Representation of the Physical World* (e.g., 3D models, text annotations).

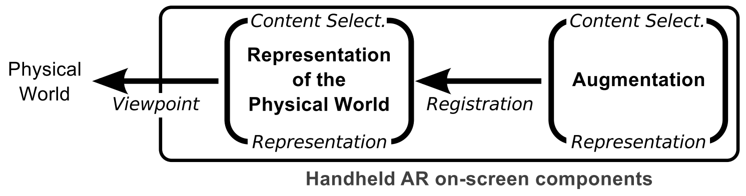


Figure 6.2: Design space for handheld AR on-screen content and interaction, organized based on two components: the *Representation of the Physical World* and the digital *Augmentation*. Two spatial relations link the two components to the physical surroundings: *Viewpoint* and *Registration*.

This augments the view of the *Physical World* with additional information and interaction capabilities.

The design space also describes the two spatial relations between the frames of reference of these two components and the physical world:

- *Viewpoint*, which describes the spatial relation between the physical surroundings and the *Representation of the physical world*. The *Viewpoint* is not steady, as it is subject to natural hand tremor and thus impairs the stability of the *Representation of the physical world* on the screen.
- *Registration*, which describes the spatial relation between the *Representation of the physical world* and the *Augmentation*. *Registration* can be hindered by camera position and orientation estimation errors, thus impairing the positioning of the *Augmentation* relative to the *Representation of the physical world*.

This design space and the spatial relations it defines between the different frames of reference allow studying interaction with handheld AR on-screen content. with reference to the design space, we discussed viewpoint control, touch screen-based interaction and the dynamics of the spatial relations in terms of initiative and sustainability.

The key contribution of this design space lies in the fact that it structures axes from other design spaces and allows studying interaction techniques. Moreover, in a field still driven by technology, an abstract design space of this type enable studying AR in a way that is independent of the technology. This need for conceptual design spaces was highlighted during the *Classifying the AR Presentation Space*[1] workshop at the ISMAR 2012 conference and the *Designing Mobile Augmented Reality*[2] workshop at the MobileHCI 2013 conference.

6.2.2 Pointing techniques

Because the *Viewpoint* in handheld AR is subject to natural hand tremor, on-screen content is not stable. This limits the precision with which both physical and digital objects can be pointed at. Based on the state of the art of both touch screen pointing assistance techniques for handheld devices and handheld AR interaction techniques, we aimed to improve handheld AR pointing precision. We explored handheld AR

[1]http://campar.in.tum.de/Chair/IsmarClassifyingARPresentationSpace2012
[2]http://studierstube.icg.tugraz.at/mobilehci2013workshop/

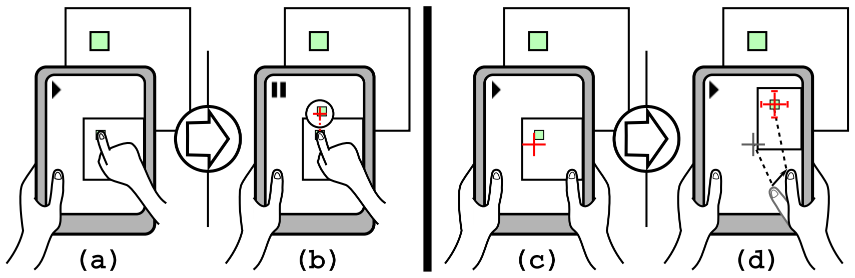


Figure 6.3: The two proposed techniques: (Left) *Shift&Freeze* builds on *Direct Touch* (a) with a precise quasi-mode that uses the callout from Shift [Vogel and Baudisch, 2007] combined with freeze-frame (b); (Right) *Relative Pointing* builds on *Crosshair* (c) with a precise mode, during which the cursor is stabilized on the physical object and controlled with indirect relative touch strokes (d).

pointing techniques through our design space and adopted two approaches: (1) combining touch-based pointing techniques with handheld AR pointing techniques, and (2) stabilizing input within the frame of reference of the physical object.

In particular, we designed and evaluated two techniques:

- *Shift&Freeze*, which builds on the *Shift* [Vogel and Baudisch, 2007], a touch-based pointing technique by adding freeze-frame. Thus, it combines touch-based pointing assistance with freeze-frame to cope with issues relating to the handheld AR context (Figure 7-a and b).

- *Relative Pointing*, which enhances pointing with a screen-centered *Crosshair* by adding a precise mode, during which the cursor is stabilized on the physical object and controlled with indirect relative strokes on the screen (Figure 7-c and d).

The experiments we held indicated that these two techniques, namely *Shift&Freeze* and *Relative Pointing*, were preferred overall, as well as being more accurate than the two baseline techniques they extend (*Direct Touch* and *Crosshair*, respectively). We further evaluated *Relative Pointing* under the following conditions: (1) registration jitter conditions for both digital and physical targets, (2) tablet and smart-phone devices, and (3) camera image presentation latency. The results indicated that *Relative Pointing* was less sensitive to registration jitter, device form factor and camera image presentation latency than *Crosshair*. We see *Relative Pointing* as a promising alternative to freezing the frame.

The proposed techniques are geared towards improving pointing precision in the context of handheld AR. They can be applied in a range of situations because: (1) pointing is an elementary interaction task, (2) precision is required for some tasks, yet, (3) precision is not always required. The proposed techniques extend baseline techniques that remain useful when there is no need for high pointing precision.

6.2.3 Demonstrators and toolkit

As part of the AMIE project, we developed a lightweight and portable toolkit to ease the development of handheld AR applications. The toolkit has been used throughout the project to develop several demonstrators for presentation both to maintenance staff in production plants and during conferences and events.

The toolkit has proved its worth for the development of demonstrators. In particular, it has minimized the impact of operating system and tracking system changes. Indeed, the demonstrators developed run on both the iOS and Windows operating systems and can use different tracking systems depending on the operating system.

6.3 Prospects for future work

We have identified a number of directions in which the contributions presented could be evolved.

6.3.1 Evaluations

The first possibility would be to further assess the differences between interaction in the image plane (as with *Direct Touch* and *Crosshair*) and interaction via input stabilized within the frame of the physical object (as with *Relative Pointing*). Although we have previously held experiments based on the execution of pointing tasks under different conditions, it would be useful to compare the different techniques by performing other tasks such as drawing on the surface of physical objects. This would reveal additional characteristics of the different techniques. It would also be interesting to run experiments over a longer period of time and in specific context, so as to study user acceptance and possible flaws in the different techniques in more complex situations. Finally, contextual menus (see chapter 5) were used to evaluate handheld AR in an industrial context, which is one example of an AR application. However, a more specific evaluation would be required to better assess such a generic AR widget.

6.3.2 Changing the viewpoint

The pointing techniques we proposed were intended to improve precision by addressing the touch screen's limitations and *Viewpoint* instabilities due to hand tremor. We also evaluated *Relative Pointing* under *Registration* error conditions. Thus, we studied the impairments affecting handheld AR spatial relations. However, in our work we did not challenge the spatial relations themselves (i.e., *Viewpoint* and *Registration*).

View management techniques [Bell et al., 2001], which arrange annotations on the screen according to their position in 3D space, so as to avoid clutter. This modifies the *Registration* relation.

It has also been proposed that the *Viewpoint* relation be modified by: (1) using head tracking to display the augmented scene on the screen according to the user's point of view, rather than that of the handheld device's camera [Hill et al., 2011] [Baricevic et al., 2012], and (2) allowing switching between alternative viewpoints, such as a top view [Alessandro et al., 2010] or previously recorded snapshots of other viewpoints [Sukan et al., 2012]. The ability to change the viewpoint in handheld AR would make it possible to access areas occluded by objects and mitigate the precision issues that can occur when the surface of physical and digital objects are almost parallel to the camera's line of sight.

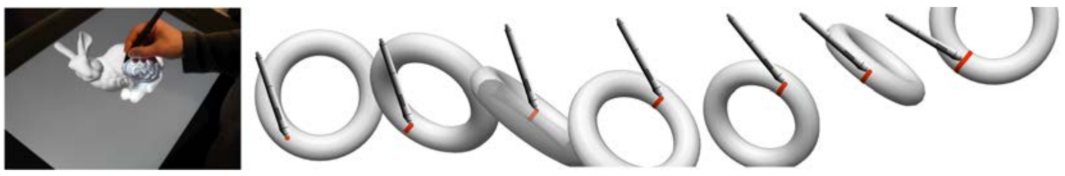


Figure 6.4: Direct drawing on 3D shapes with automated camera control: (Left) Drawing on a 3D shape on a tablet display with a stylus; and (Right) The camera viewpoint is adjusted to keep the stylus perpendicular to the shape, making it possible to draw across occluded and initially back-facing areas with a fairly constant precision.

As part of a collaboration on the topic of 3D authoring tools for painting on 3D objects, we started to investigate automated viewpoint changes during interaction [Ortega and Vincent, 2014]. For tablet displays with a stylus, we proposed a form of assistance for direct drawing on 3D shapes based on automated camera control, so as to keep the surface of the 3D shape perpendicular to the stylus when drawing (Figure 8). This technique makes it possible to draw across faces occluded by the object itself (i.e., self-occlusion) and back faces, while maintaining a fairly constant level of drawing precision. A similar approach could be used in the context of handheld AR to allow pointing and drawing on surfaces that are almost parallel to the camera's line of sight. Using automated camera control in the context of handheld AR would generate additional issues. First, the camera only captures images from a single viewpoint at a time. However, it is possible either to use a previously recorded 3D model of the scene, as we did during the AMIE project and as in [Tatzgern et al., 2013], or to construct such a model from scratch while using the system using a depth camera, for example. Secondly, the user must understand viewpoint displacements to be able to map the current viewpoint in its physical surroundings.

Another research avenue relating to changes in viewpoint is based on contextual menus (see chapter 5) or level-of-detail interfaces [DiVerdi et al., 2004]. While we previously proposed to explore changes in viewpoint to improve access to elements of the physical surroundings such as surfaces that are almost parallel to the camera's line of sight, this would focus more specifically on accessing the digital content registered to the physical surroundings.

6.3.3 Other AR set-ups

Our work can also be extended to other types of AR set-up such as see-through Head-Mounted Displays (HMDs) and projector-based Spatial AR systems. We focused on handheld AR set-ups, in particular, but our contributions can be adapted to a other AR set-ups.

The entities and dimensions of the design space presented in chapter 2 can be adapted to cover other AR set-ups (see 2.8). These includes optical see-through HMDs and Spatial AR, where no representation of the physical world is displayed by the system. The potential to relax the viewpoint also depends on the display device. For example, with video see-through HMDs, because the user cannot see the physical surroundings directly, altering the viewpoint might prevent her/him from operating safely in the physical world.

The *Relative Pointing* technique can be adapted to other AR set-ups such as handheld projector-based systems. Such set-ups are also subject to hand tremor, but, provided that the handheld projector is combined with a touch-sensitive surface, the *Relative Pointing* technique can be used.

6.3.4 Cognition of AR content

Because AR relies on the colocation of digital information with physical objects, a promising avenue would be to further study how users perceive and understand such a combination of spatial information. It would be particularly useful to build on existing studies of human perception and cognition of 3D space, so as to better understand the mental representation we have of our physical surroundings and how digital information links up with this representation depending on how it is presented to users.

This research area is directly related to the topics addressed by the French working group "INTERCO3D"[3] (INTERaction and COgnition in real and virtual 3D spaces).

The "Living Book of Anatomy - LBA"[4] project, in which the EHCI team is involved, is also related to this research area. The goal of the LBA project is to study the effect of embodiment on the learning of anatomy. Its objective is to teach anatomy with a handheld AR application that overlays a dynamic and personalized anatomical model onto parts of the student's body (e.g., the knees). Knowledge of anatomy sometimes requires good visuospatial skills and the handheld AR approach adopted may help students better understand and remember anatomical structures and their dynamics.

[3]http://www.irit.fr/INTERCO3D/
[4]https://persyval-lab.org/en/sites/lba

Synthèse en français

Interaction en Réalité Augmentée sur dispositifs mobiles : Relations Spatiales.

Chapter Content

Avant-propos

Ce chapitre fournit une traduction de l'introduction et de la conclusion ainsi qu'un résumé de chaque chapitre en français.

Introduction

Domaine

Cette thèse s'inscrit dans le domaine de l'Interaction Homme-Machine (IHM) et est consacrée à l'interaction dans le cadre de systèmes de Réalité Augmentée sur dispositifs mobiles.

Les systèmes de **Réalité Augmentée** (RA) ont pour objectif de combiner la perception qu'a l'utilisateur des informations numériques avec la perception qu'il a de son environnement physique. En 1992, Caudell et Mizell ont baptisé "Réalité Augmentée" les systèmes qui affichent des informations numériques alignées sur le monde physique [Caudell and Mizell, 1992]. Azuma [Azuma, 1997] [Azuma et al., 2001] a défini les systèmes de RA comme étant des systèmes qui :

- Combinent des objets réels et des objets virtuels dans un environnement réel ;
- S'exécutent de manière interactive et en temps réel ;
- Font correspondre (autrement dit, alignent) les objets réels et virtuels entre eux.

Cette définition de la RA lui offre un large champ d'application car elle ne limite pas la RA à une technologie ou une modalité d'interaction particulière. Bien que l'augmentation de différents sens (tels que l'ouïe ou le toucher) soit possible, la combinaison de la perception d'objets numériques avec celle d'objets physiques a été essentiellement étudiée pour le sens visuel.

Dans ce travail, nous nous intéressons à **l'augmentation visuelle**.

De plus, cette définition ne limite par la RA à des dispositifs particuliers pour l'interaction en entrée et en sortie. D'ailleurs, plusieurs types de dispositifs d'affichage peuvent être utilisés dans des systèmes de RA : casques munis d'écrans semi transparents, casques munis de caméras et d'écrans, dispositifs mobiles embarquant une caméra ou encore systèmes de projection. Ces dispositifs variés offrent différents types de contrôle du point de vue dans la scène augmentée ainsi que différentes manières d'interagir avec son contenu. La RA sur dispositifs mobiles est le cas où un dispositif mobile, comme un téléphone portable ou une tablette tactile, est utilisé pour afficher une scène augmentée. Dans ce cas, la vue de l'environnement physique capturée par la caméra du dispositif mobile est affichée à l'écran. Des objets numériques sont superposés aux images prises par la caméra et sont alignés sur les objets physiques (Figure 1). Par exemple, *NaviCam* [Rekimoto, 1995], un système pionnier de RA sur dispositifs mobiles, ajoute des informations textuelles sur les images prises par la caméra qui sont en relation avec les objets physiques visualisés à l'écran. Le système utilise un algorithme de vision par ordinateur pour détecter des marqueurs dans les images de la caméra et permettre l'affichage des informations correspondantes.

Dans nos travaux, nous nous sommes intéressés aux systèmes de **RA sur dispositifs mobiles**. En effet, il est désormais possible d'exécuter des applications de RA sur les dispositifs mobiles en utilisant des librairies de vision par ordinateur actuellement disponibles (Vuforia SDK[5] par exemple). De plus, par rapport aux casques permettant de fournir du contenu de RA, le dispositif mobile est moins intrusif et permet une utilisation plus occasionnelle des applications de RA. Cela peut favoriser une meilleure intégration des systèmes de RA avec d'autres tâches.

Pour permettre l'alignement du contenu numérique avec l'environnement physique de l'utilisateur, un système de RA sur dispositifs mobiles doit utiliser un composant lui permettant d'estimer la position et l'orientation de la caméra dans l'environnement physique (comme par exemple des algorithmes de vision par ordinateur). De

[5]https ://www.vuforia.com/

plus, le rendu d'objets numériques qui s'intègre naturellement dans un environnement physique est demandeur en terme de dispositif d'affichage et de rendu graphique (par exemple, affichage à différentes profondeurs et illumination réaliste des objets numériques). Les systèmes de RA sur dispositifs mobiles (et de RA en général) sont des défis pour les systèmes d'estimation de la position de la caméra ainsi que pour les dispositifs d'affichage et les techniques de rendu.

En plus de ces défis, **l'interaction de l'utilisateur** avec les systèmes de RA doit aussi être étudiée. En effet, un ensemble de contraintes peuvent limiter l'interaction avec de tels systèmes.

Tout d'abord, les systèmes de RA sur dispositifs mobiles héritent des spécificités des dispositifs mobiles en terme d'interaction. La taille de l'écran est limitée et l'interaction directe avec le doigt sur l'écran tactile (modalité d'interaction communément utilisée) est limitée par le problème des 'gros doigts' [Roudaut et al., 2008]. Lorsque l'utilisateur interagit avec l'écran tactile, ses doigts cachent en partie l'écran. Dans le cas de l'interaction directe sur l'écran, la position active qui est effectivement utilisée par le système pour l'interaction est ambigüe car cette position est cachée par le doigt et n'est pas représentée par un curseur. De plus, l'interaction sur écran tactile ne permet pas des interactions telles que le survol [Buxton, 1990].

Dans le cas spécifique de la RA sur dispositifs mobiles, les images prises par la caméra et le contenu numérique sont affichés simultanément à l'écran (Figure 1) alors que sa taille est limitée.

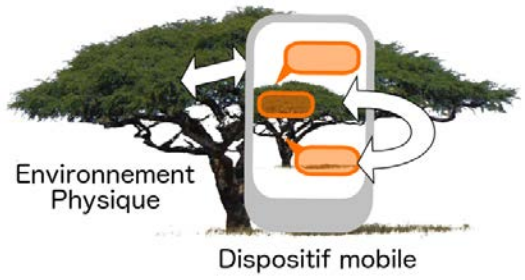


FIGURE 1: RA sur dispositifs mobiles : Les images prises par la caméra représentant l'environnement physique et l'augmentation numérique se partagent la place à l'écran. La relation spatiale entre l'environnement physique et le contenu affiché à l'écran est une caractéristique centrale de la RA, mais peut aussi être une contrainte pour l'interaction.

De plus, la relation spatiale entre le contenu affiché à l'écran et l'environnement physique limite également l'interaction tactile. Le point de vue sur la scène augmentée est contrôlé par la position et l'orientation du dispositif mobile dans l'espace dont la stabilité est affectée par les tremblements naturels de la main. Les erreurs d'estimation de la position de la caméra peuvent aussi rendre instable l'augmentation numérique à l'écran. Ces instabilités peuvent limiter l'interaction car le contenu affiché à l'écran n'est pas stable.

Nous sommes partis des **relations spatiales** pour explorer l'interaction avec les systèmes de RA sur dispositifs mobiles. En comparaison avec l'interaction sur dispositifs mobiles, le contexte de la RA apporte des contraintes supplémentaires qui sont liées au fort couplage entre l'environnement physique, le contenu affiché à l'écran et le dispositif mobile. En effet, la relation spatiale entre l'environnement physique et les objets numériques est propre à la RA. Nous avons examiné cette relation au regard des différents référentiels mis en œuvre durant l'interaction.

Contexte : collaboration internationale

Ce travail s'inscrit dans le projet international AMIE[6]. Le projet AMIE est financé par l'Agence Nationale de la Recherche française (ANR) et l'agence pour la science et les technologies japonaise (JST).

AMIE est l'abréviation de *Augmented Mobile Interactive Experience : Application to Maintenance Services* (Expérience interactive augmentée et mobile : Application à la maintenance).

L'objectif de ce projet est l'étude de l'interaction dans le contexte de la RA sur dispositifs mobiles. Le domaine d'application de ce projet est la maintenance de machines de production. Ce projet est pluridisciplinaire. Les deux partenaires académiques de ce projet sont : une équipe de recherche s'intéressant à la combinaison d'informations venant de plusieurs sources et à la localisation (CfSR, AIST, Tsukuba, Japon), et une équipe de recherche en Interaction Homme-Machine (IIHM, LIG, Grenoble, France). Les deux partenaires industriels de ce projet sont liés au domaine de la maintenance de machines de production : Digital Electronics (Osaka, Japon), et Schneider Electric (Grenoble, France). Le projet AMIE est donc dédié à l'utilisation de la RA sur dispositifs mobiles par les opérateurs de maintenance des machines de production. Dans ce cadre, l'augmentation fournie à l'utilisateur s'appuie sur la position de l'utilisateur et les informations renvoyées par l'automate de la machine.

La RA sur dispositifs mobiles semble une solution intéressante pour des scénarios où les utilisateurs ont besoin de visualiser des informations numériques qui correspondent à des emplacements physiques tel que la maintenance de machines de production. En effet, dans le cas de la maintenance, les opérateurs ont besoin d'accéder à des informations numériques dynamiques fournies par l'automate de la machine (telles que les données des capteurs ou l'état des actionneurs) qui correspondent à des parties et des emplacements physiques de la machine. De plus, d'autres informations comme la documentation peuvent être associées à des parties spécifiques de la machine. La RA permet cette association d'informations numériques et d'emplacements physiques. D'autre part, l'utilisation d'un dispositif mobile permet aux opérateurs d'effectuer d'autres tâches (comme accéder à l'état du stock pour commander une pièce) avec le même dispositif. Par ailleurs, l'usage d'un dispositif mobile est potentiellement plus acceptable socialement dans un contexte industriel qu'un casque munis d'écrans, même si la miniaturisation tend à limiter ce facteur. Cependant, l'opérateur doit tenir le dispositif mobile dans ces mains ce qui peut engendrer de la fatigue ou encore l'empêcher d'utiliser d'autres outils.

Ce projet collaboratif a donné lieu à différents démonstrateurs. Avec AIST, nous avons présenté des démonstrations durant les conférences ISMAR en 2011 et en 2012. Avec l'ensemble des partenaires du projet, nous avons aussi développé des démonstrateurs pour des scénarios de maintenance industrielle. Ces démonstrateurs ont donné lieu à des évaluations auprès d'opérateurs de maintenance et à une démonstration durant un séminaire industriel à Tokyo en 2013. Les travaux réalisés en collaboration avec les partenaires industriels sont confidentiels et ne sont donc pas inclus dans ce mémoire.

[6]http ://amie.imag.fr/

Objectifs et méthode

L'objectif de nos travaux est d'améliorer l'interaction en RA sur dispositifs mobiles. Une caractéristique centrale de ces systèmes réside dans la relation spatiale entre l'environnement physique, le contenu numérique et le dispositif. Cette relation spatiale a un impact sur l'interaction car elle rend le contenu affiché à l'écran instable. Dans ce contexte, la question est comment améliorer l'interaction en RA sur dispositifs mobiles en relâchant les contraintes spatiales sans pour autant perdre la co-localisation physique-numérique propre à la RA.

Un premier objectif a été de caractériser les différents composants d'un système de RA sur dispositifs mobiles et leurs relations spatiales et de les organiser dans un espace conceptuel. En s'appuyant sur ce premier travail, un deuxième objectif a été de proposer des nouvelles techniques de pointage.

La validation de l'espace conceptuel demande l'étude de son pouvoir taxonomique et de ses capacités à aider la conception de nouvelles techniques d'interaction. Les techniques de pointage ont été évaluées avec des expériences contrôlées. Comme ces travaux s'inscrivent dans un projet collaboratif s'intéressant à l'utilisation de la RA sur dispositifs mobiles pour la maintenance de machines de production, nous avons pu effectuer des évaluations en contexte. Le développement et l'évaluation des techniques de pointage ont permis de faire évoluer l'espace conceptuel. Les deux objectifs étaient complémentaires et se sont mutuellement complétés (Figure 2).

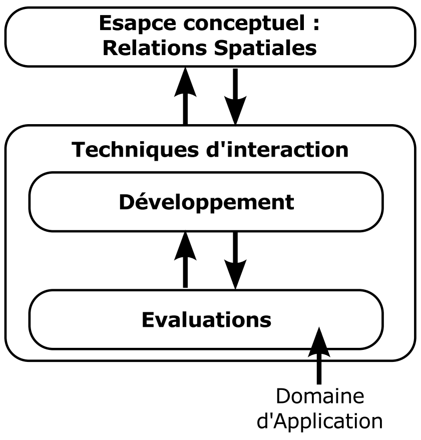


FIGURE 2: Approche itérative

Contributions

Nous proposons les contributions suivantes : (1) un espace conceptuel pour la RA sur dispositifs mobiles, (2) des techniques de pointage améliorant la précision, et (3) une boîte à outils et des démonstrateurs.

Premièrement, nous proposons un espace conceptuel du contenu affiché à l'écran dans le cas de la RA sur dispositifs mobiles. Cet espace conceptuel est structuré autour de deux composants et il met en avant les relations spatiales entre ces composants et l'environnement physique. Cet espace conceptuel permet d'étudier l'interaction de l'utilisateur avec le contenu affiché à l'écran. Cet espace fournit un point de vue conceptuel plutôt que technique sur l'interaction.

Deuxièmement, nous proposons des techniques de pointage, et en particulier *Shift&Freeze* et *Pointage Relatif*. Les techniques de pointage peuvent être utilisées dans une variété de contexte pour sélectionner un objet physique ou numérique ou pour sélectionner une position dans l'environnement physique. Les techniques que nous proposons visent à améliorer la précision du pointage en RA sur dispositifs mobiles. Nous avons évalué les techniques proposées au cours de quatre expériences.

Troisièmement, dans le cadre du projet AMIE, nous avons développé une boîte à outils permettant l'affichage et l'interaction en RA sur dispositifs mobiles. Cette boîte à outils a été utilisée pour développer des démonstrateurs tout au long du projet. Certains de ces démonstrateurs sont spécifiques au domaine d'application du projet et d'autres ont été développés pour être présentés durant des conférences ou des événements publiques.

Structure du document

Ce mémoire est organisé comme suit :

- Dans le chapitre 2, nous dressons d'abord un état de l'art des espaces conceptuels décrivant les systèmes de RA. Nous présentons ensuite notre espace conceptuel dédié à la RA sur dispositifs mobiles et nous étudions l'interaction utilisateur au regard de cet espace conceptuel.
- Dans le chapitre 3, nous dressons d'abord un état de l'art des techniques de pointage orienté vers le pointage en RA sur dispositifs mobiles. Nous présentons ensuite de nouvelles techniques de pointage en RA sur dispositifs mobiles.
- Dans le chapitre 4, nous présentons quatre expériences que nous avons effectuées pour évaluer la précision des techniques de pointage proposées.
- Dans le chapitre 5, nous décrivons la boîte à outils et certains démonstrateurs que nous avons développés durant le projet AMIE.
- Dans le chapitre 6, nous concluons ce mémoire avec un résumé des points clés et nous proposons des perspectives à nos travaux.

Partie I : Espace conceptuel

Chapitre 2 : Espace conceptuel

Après avoir dressé un état de l'art des espaces conceptuels de Réalité Mixte et de Réalité Augmentée, dans ce chapitre nous proposons un espace conceptuel décrivant le contenu visuel affiché à l'écran dans le cadre de la Réalité Augmentée sur dispositifs mobiles. Cet espace conceptuel est organisé autour de deux composants (Figure 3) :

- La *Représentation du monde physique* qui décrit le contenu à l'écran qui permet à l'utilisateur de se repérer dans l'environnement physique.
- L'*Augmentation* qui fournit des informations supplémentaires par rapport à la représentation du monde physique.

Nous proposons de décrire ces deux composants avec deux axes :

- La *sélection du contenu* qui décrit comment le contenu affiché est choisi.
- Le *Mode de représentation* qui décrit l'aspect visuel du contenu affiché.

Notre espace décrit également deux relations spatiales :

- Le *Point de vue* qui lie la *Représentation du monde physique* à l'environnement physique.
- L'*Ancrage* qui lie l'*Augmentation* à la *Représentation du monde physique.*

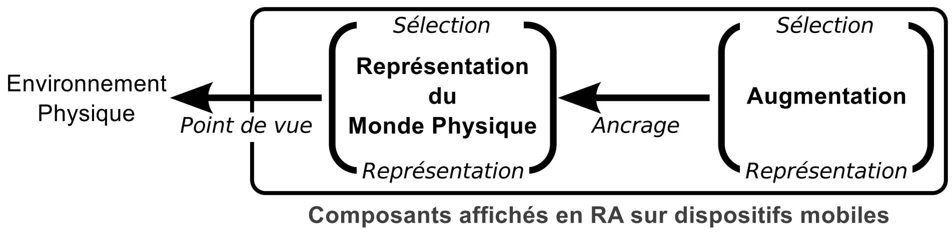


FIGURE 3: Espace conceptuel pour l'interaction en Réalité Augmentée sur dispositifs mobiles : Deux composants (la *Représentation de l'environnement physique* et l'*Augmentation*) et deux relations spatiales (le *Point de Vue* et l'*Ancrage*).

Avec cette description du contenu affiché à l'écran, nous étudions l'interaction utilisateur en Réalité Augmentée sur dispositifs mobiles. Nous nous intéressons tout d'abord à la dynamique des deux relations spatiales en terme d'initiative et de permanence. Nous focalisons ensuite sur le contrôle du *Point de vue* et sur l'interaction avec l'écran tactile.

Partie II : Techniques de pointage

Chapitre 3 : Techniques de pointage

Après avoir dressé un état de l'art des techniques de pointage, ce chapitre présente notre étude sur le pointage en Réalité Augmentée sur dispositifs mobiles au travers de quatre techniques d'interaction.

Tout d'abord, nous présentons deux techniques de pointage qui combinent une technique de pointage sur dispositifs mobiles à écran tactile avec la mise en pause de la vidéo. Premièrement, *AR TapTap* qui étend *TapTap* [Roudaut et al., 2008] avec la mise en pause de la vidéo (Figure 4-a). *TapTap* est une technique de pointage sur dispositifs mobiles basée sur un pointage en deux étapes (deux 'taps') et le changement du niveau de zoom. Deuxièmement, *Shift&Freeze* qui étend *Shift* [Vogel and Baudisch, 2007] avec la mise en pause de la vidéo (Figure 4-b). *Shift* est une technique de pointage basée sur l'utilisation d'une bulle qui affiche la partie de l'écran cachée par le doigt et contient un curseur.

Ensuite, nous présentons deux techniques où l'interaction est effectuée dans le référentiel de l'objet physique plutôt que dans celui de l'écran comme c'est le cas pour les deux techniques précédentes. Premièrement, nous avons développé un prototype d'interaction avec le doigt entre la caméra et l'objet, ainsi que sur l'objet (Figure 4-c). Deuxièmement, nous proposons d'étendre le pointage avec un viseur au centre de l'écran en lui ajoutant un mode précis dans lequel le curseur est stabilisé sur l'objet physique et contrôlé de manière indirecte avec l'écran tactile (Figure 4-d).

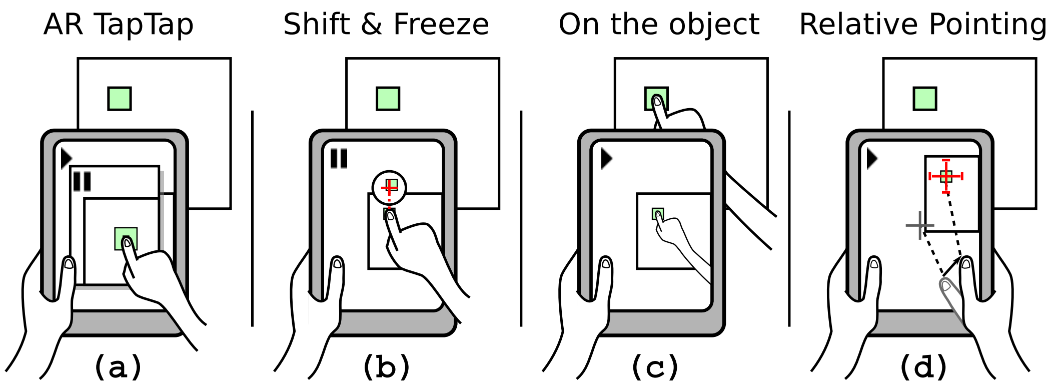


FIGURE 4: Deux techniques combinent une technique de pointage existante avec la mise en pause de la vidéo : (a) *AR TapTap* et (b) *Shift&Freeze*. Deux autres techniques proposent d'interagir dans le référentiel de l'objet physique : (c) interaction devant la caméra et (d) *Pointage relatif* indirect.

Chapitre 4 : Evaluations

Des quatre approches explorées dans le chapitre précédent, nous en avons retenu deux (*Shift&Freeze* et *Pointage Relatif*) que nous avons évalués au cours de quatre expériences utilisateurs.

Les deux premières expériences avaient pour objectif de comparer les techniques *Shift&Freeze* et *Pointage Relatif* avec les techniques de base qu'elles étendent (interaction directe et viseur au centre de l'écran). Une expérience utilisait une tâche de pointage abstraite tandis que l'autre utilisait une tâche de pointage plus réaliste. Les résultats indiquent que les participants ont préféré les deux techniques proposées aux deux techniques de base et que les deux techniques proposées sont plus précises que les techniques de base.

Nous avons ensuite effectué deux expériences supplémentaires visant à évaluer le *Pointage Relatif* dans des conditions dégradées d'estimation de la position de la caméra. Pour ce faire, nous avons comparé le mode précis de *Pointage Relatif* avec le pointage avec un viseur pour l'acquisition de cibles physiques et de cibles numériques ancrées sur un objet physique. Durant ces expériences, nous avons fait varier le bruit de l'estimation de la position de la caméra, mais aussi le type de dispositif (téléphone ou tablette) et le délai entre la prise de vue par la caméra et l'affichage de l'image à l'écran. Globalement, le mode précis de *Pointage Relatif* est moins sensible aux différents facteurs que nous avons fait varier que le pointage avec un viseur.

Partie III : Développement de techniques d'interaction

Chapitre 5 : Boîte à outils et démonstrateurs

Dans le cadre du projet AMIE[7], dans lequel s'inscrivent nos travaux, nous avons développé une boîte à outils pour faciliter le développement d'application de Réalité Augmentée sur dispositifs mobiles. Cette boîte à outils fournit des primitives permettant de décrire un graphe de scène qui gère le rendu à l'écran et l'interaction. La boîte à outils fournit des primitives de bas niveau comme des graphiques vectoriels ou des modèles 3D et des primitives de plus haut niveau comme des menus.

Cette boîte à outils a été utilisée tout au long du projet pour développer des démonstrateurs pour des scénarios de maintenance de machines de production ainsi que pour présenter durant des conférences et des événements. Ces démonstrateurs proposent en plus de la Réalité Augmentée, un mode carte (vue de dessus) et un mode 'Virtualité Augmentée' où la représentation de l'environnement physique est effectuée avec un modèle 3D de cet environnement construit à priori. Ces démonstrateurs utilisent également des menus contextuels qui proposent d'afficher différents menus en fonction de la position de l'utilisateur par rapport à l'objet augmenté. De loin, un menu global est attaché à l'objet. De plus près, le menu global est affiché en haut de l'écran et des autres menus apparaissent sur les sous-parties de l'objet.

Conclusion

Dans le contexte de la Réalité Augmentée (RA) sur dispositifs mobiles, nous nous sommes intéressés aux relations et contraintes spatiales pour l'interaction. Par RA sur dispositifs mobiles, nous entendons l'utilisation d'un dispositif mobile comme une loupe magique qui permet de combiner des informations numériques avec la représentation de l'environnement physique affichée sur l'écran du dispositif mobile. L'interaction dans ce contexte permet de présenter des informations numériques dans l'environnement physique auquel elles se rapportent. Cependant, les relations spatiales physique/numérique peuvent nuire à l'interaction de l'utilisateur avec le système. Notre objectif était donc d'explorer comment ces relations spatiales peuvent être relâchées pour améliorer l'interaction tout en maintenant la co-localisation physique/numérique propre à la RA.

Nous nous sommes intéressés à cette question à deux niveaux.

Premièrement, nous avons proposé une caractérisation des composants constituant le contenu affiché à l'écran dans le cas de la RA sur dispositifs mobiles ainsi que leurs relations spatiales. Cette caractérisation permet d'examiner les effets sur l'interaction.

Deuxièmement, nous avons proposé des techniques permettant d'améliorer la précision du pointage en RA sur dispositifs mobiles. En effet, le pointage en RA sur dispositifs mobiles hérite des problèmes interactionnels des écrans tactiles et est impacté par les relations entre ce qui est affiché à l'écran et l'environnement physique.

Finalement, nous avons développé un prototype de boîte à outils qui a été utilisé pour développer des démonstrations présentées durant des conférences, mais aussi pour des scénarios de maintenance de machines de production.

La Figure 5 résume nos contributions.

[7]http ://amie.imag.fr/

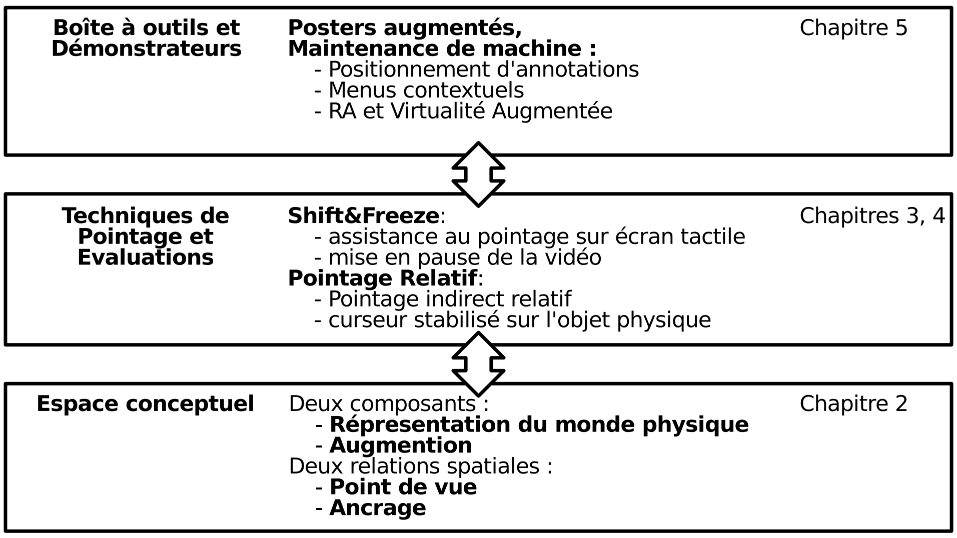

Boîte à outils et Démonstrateurs
Posters augmentés, Maintenance de machine :
- Positionnement d'annotations
- Menus contextuels
- RA et Virtualité Augmentée
Chapitre 5
Techniques de Pointage et Evaluations
Shift&Freeze:
- assistance au pointage sur écran tactile
- mise en pause de la vidéo
Pointage Relatif:
- Pointage indirect relatif
- curseur stabilisé sur l'objet physique
Chapitres 3, 4
Espace conceptuel
Deux composants :
- Réprésentation du monde physique
- Augmention
Deux relations spatiales :
- Point de vue
- Ancrage
Chapitre 2

FIGURE 5: Résumé des contributions

Nous résumons nos contributions avant de proposer des perspectives à ces travaux.

Contributions

Nous avons proposé les contributions suivantes : (1) un espace conceptuel pour la RA sur dispositifs mobiles, (2) des techniques de pointage, et (3) un prototype de boîte à outils permettant le développement d'applications de RA sur dispositifs mobiles (dans le cadre du projet AMIE qui s'intéresse à l'utilisation de le RA sur dispositifs mobiles pour la maintenance de machines de production).

Espace conceptuel

Nous avons proposé un espace conceptuel pour le contenu affiché à l'écran en RA sur dispositifs mobiles qui comprend deux composants (Figure 6) :

- La *Représentation du monde physique* qui se compose des éléments de l'environnement physique affichés à l'écran (les images prises par la caméra en général). Ce composant permet à l'utilisateur de comprendre où est le point de vue dans l'environnement physique.
- L'*Augmentation* numérique qui se compose du contenu numérique qui n'est pas la *Représentation du monde physique* (par exemple des modèles 3D ou des annotations textuelles). Ce composant augmente la vue du monde physique avec des informations et/ou des capacités d'interaction supplémentaires.

L'espace conceptuel décrit également les deux relations spatiales entre les référentiels de ces deux composants et l'environnement physique :

- Le *Point de vue*, qui décrit la relation spatiale entre l'environnement physique et la *Représentation du monde physique*. Le *Point de vue* n'est pas stable à cause des tremblements naturels de la main, ce qui rend instable la *Représentation du monde physique* à l'écran.

- L'*Ancrage*, qui décrit la relation spatiale entre la *Représentation du monde physique* et l'*Augmentation.* L'*Ancrage* peut être affecté par les erreurs d'estimation de la position de la caméra, ce qui affecte la position de l'*Augmentation* par rapport à la *Représentation du monde physique.*

Cet espace conceptuel et les relations spatiales qu'il définit entre les différents référentiels permettent d'étudier l'interaction avec le contenu affiché à l'écran en RA sur dispositifs mobiles, telle que le contrôle du point de vue, l'interaction avec l'écran tactile et la dynamique des relations spatiales. Un point important de cet espace est de fournir une structure pour ses différents axes. De plus, il définit un point de vue conceptuel sur la RA sur dispositifs mobiles.

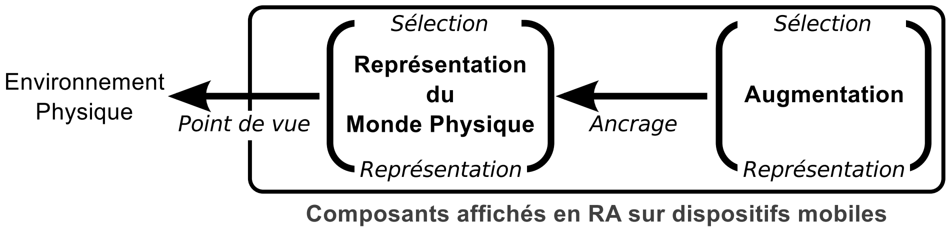


FIGURE 6: Espace conceptuel du contenu affiché à l'écran en Réalité Augmentée sur dispositifs mobiles, articulé autour de deux composants : la *Représentation du monde physique* et l'*Augmentation* numérique. Deux relations spatiales relient ces deux composants et l'environnement physique : le *Point de Vue* et l'*Ancrage.*

Techniques de Pointage

Comme le *Point de vue* en RA sur dispositifs mobiles est sensible aux tremblements de la main, le contenu à l'écran n'est pas stable. Ceci limite la précision du pointage sur des objets physiques ou numériques. En s'appuyant sur l'état de l'art des techniques de pointage sur dispositifs mobiles et des techniques facilitant l'interaction en RA sur dispositifs mobiles, nous avons proposé des techniques visant à améliorer le pointage en RA sur dispositifs mobiles. Nous avons adopté deux approches complémentaires : (1) combiner une technique de pointage sur dispositifs mobiles avec la mise en pause de la vidéo, et (2) stabiliser les entrées utilisateur dans le référentiel de l'objet physique pointé.

En particulier, nous avons développé et évalué les deux techniques suivantes :

- *Shift&Freeze*, qui étend *Shift* [Vogel and Baudisch, 2007], une technique de pointage sur dispositif mobile, avec la mise en pause de la vidéo (Figure 7-a et b).

- *Pointage Relatif*, qui étend le pointage avec un viseur centré sur l'écran en ajoutant un mode précis. Dans ce mode précis, le curseur est stabilisé sur l'objet physique et contrôlé de manière indirecte et relative avec l'écran tactile (Figure 7-c et d).

Les évaluations que nous avons réalisées indiquent que ces deux techniques, *Shift&Freeze* et *Pointage relatif*, étaient préférées par les utilisateurs et plus précises que les deux techniques qu'elles étendent (interaction directe sur l'écran tactile

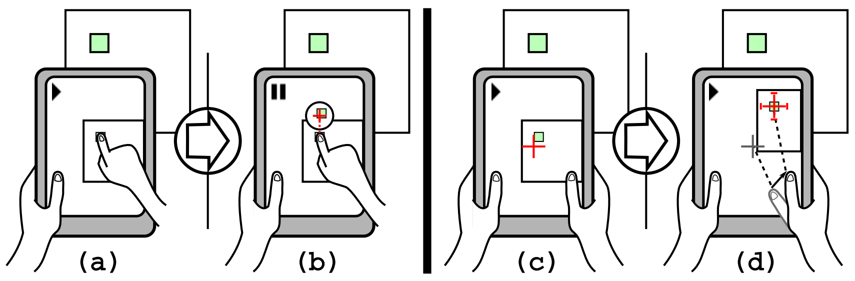


FIGURE 7: Les deux techniques proposées : (A gauche) *Shift&Freeze* qui étend l'interaction directe (a) avec un quasi-mode précis utilisant une bulle comme Shift [Vogel and Baudisch, 2007] et la mise en pause de la vidéo (b). (A droite) *Pointage relatif* qui étend le viseur (c) avec un mode précis dans lequel le curseur est stabilisé sur l'objet physique et contrôlé indirectement avec l'écran tactile (d).

et viseur). Nous avons aussi évalué *Pointage relatif* dans différentes conditions : (1) avec du bruit dans l'estimation de la position de la caméra lors du pointage sur des cibles physiques et numériques, (2) sur téléphone et sur tablette, et (3) avec de la latence supplémentaire entre la prise de vue et l'affichage des images de la caméra. Les résultats indiquent que le mode précis de *Pointage relatif* est moins sensible aux changements de conditions que le viseur qu'il étend. Nous concluons que *Pointage relatif* est une alternative valide et prometteuse à la mise en pause de la vidéo.

Démonstrateurs et boîte à outils

Dans le cadre du projet AMIE, nous avons développé une boîte à outils simple et portable pour faciliter le développement d'applications de RA sur dispositifs mobiles. La boîte à outils a été utilisée durant le projet pour développer plusieurs démonstrateurs qui ont été présentés à des opérateurs de maintenance et durant des conférences et des événements publiques.

La boîte à outils a été utile pour développer les démonstrateurs, en particulier pour minimiser l'impact du changement de système d'exploitation et du système d'estimation de la position de la caméra.

Perspectives

Nous avons identifié des pistes de recherche se basant et étendant nos contributions.

Evaluations

La première possibilité serait de réaliser des expériences avec d'autres tâches que le pointage comme par exemple le dessin sur surface d'objets. Ceci permettrait d'étudier plus en avant les différences entre l'interaction dans le plan de l'image (comme avec l'interaction tactile en direct ou le viseur) et l'interaction dans le référentiel de l'objet physique (comme avec *Pointage relatif*). Il serait aussi intéressant de réaliser des évaluations sur une plus longue période de temps et dans un contexte d'utilisation effectif pour étudier leur acceptabilité et pour détecter d'éventuels problèmes quand des techniques sont utilisées dans des situations plus complexes. Finalement, les menus contextuels n'ont pas été évalués en tant que tels et il serait intéressant de les évaluer en contexte.

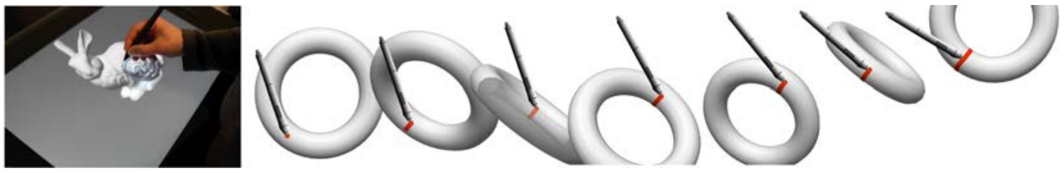


Figure 8: Dessin en direct sur des modèles 3D avec un contrôle automatique de la caméra. (A gauche) Dessin sur un modèle 3D avec un écran à stylet. (A droite) Le point de vue de la caméra est modifié pour que le stylet reste perpendiculaire à la surface du modèle 3D pendant le dessin. Cela permet de continuer à dessiner sur des surfaces non visibles depuis le point de vue initial et de conserver la même précision tout au long du dessin.

Changer de point de vue

Les techniques de pointage que nous avons proposées avaient pour but d'améliorer la précision du pointage en traitant les limitations de l'écran tactile et l'instabilité du *Point de vue*. Nous avons également évalué la technique *Pointage relatif* dans des conditions d'*Ancrage* dégradées. Nous avons donc étudié les bruits possibles affectant les relations spatiales en RA sur dispositifs mobiles. Cependant, nous n'avons pas complètement remis en cause les relations spatiales elles-mêmes.

Les techniques de 'view management' [Bell et al., 2001], qui mettent en page les annotations à l'écran en fonction de leur position dans l'espace et de leur taille de manière à éviter les recouvrements, modifient effectivement la relation spatiale d'*Ancrage*.

Des modifications de la relation spatiale *Point de vue* ont aussi été proposées : (1) en utilisant la position de la tête de l'utilisateur pour afficher à l'écran un point de vue correspondant à celui de l'utilisateur [Hill et al., 2011] [Baricevic et al., 2012], et (2) en autorisant le passage à des points de vue différents, comme une vue de dessus [Alessandro et al., 2010] ou des points de vue enregistrés précédemment [Sukan et al., 2012]. La possibilité de changer de point de vue en RA sur dispositifs mobiles rendrait possible l'accès à des zones qui resteraient autrement cachées. Cela permettrait aussi de limiter les problèmes de précision qui apparaissent quand la surface des objets (physiques ou numériques) est presque parallèle à la ligne de mire de la caméra.

Au cours d'une collaboration sur le sujet du dessin sur des modèles 3D dans les outils de création de contenu 3D, nous avons exploré l'utilisation du changement de point de vue dans une scène 3D durant l'interaction [Ortega and Vincent, 2014]. Sur écran avec stylet, nous avons proposé de combiner le dessin au stylet sur la surface de modèles 3D avec des mouvements automatiques de la caméra permettant de conserver la surface du modèle 3D perpendiculaire au stylet au point de contact (Figure 8). Cette technique permet de continuer à dessiner sur des surfaces non-visibles depuis le point de vue initial et de conserver une précision de dessin équivalente tout au long du trait.

Une stratégie de changement de point de vue similaire pourrait être appliquée dans le contexte de la RA sur dispositifs mobiles. Ceci permettrait de pointer et de dessiner sur des surfaces presque parallèles à la ligne de mire de la caméra. Un changement automatique du point de vue dans le contexte de la RA sur dispositifs mobiles pose des problèmes supplémentaires. Tout d'abord, à un instant donnée, la caméra ne capture les images que d'un seul point de vue. Cependant, il est possible d'utiliser un modèle 3D de l'environnement physique enregistré à priori, comme nous l'avons fait durant le projet AMIE. Il est aussi possible de construire un modèle de l'environnement physique en direct avec une caméra de profondeur par exemple. De plus, l'utilisateur doit pouvoir comprendre les changements de point de vue afin de pouvoir comprendre la position du point de vue dans son environnement physique.

Une autre perspective en rapport au changement de point de vue est liée aux menus contextuels (Chapitre 5) et à la présentation d'interfaces avec plusieurs niveaux de détails [DiVerdi et al., 2004]. Il s'agirait ici de changer de point de vue pour accéder à des informations numériques plutôt qu'à des surfaces peu ou pas visibles depuis le point de vue initial.

Autres dispositifs

Nos travaux pourraient aussi être étendus à d'autres types de dispositifs de RA comme les casques munis d'écrans et de caméras ou les systèmes utilisant des projecteurs.

Les composants et les axes de notre espace conceptuel présenté au Chapitre 2 pourraient être adaptés à d'autres dispositifs que les dispositifs mobiles. Dans le cas des casques munis d'écrans semi transparents et des systèmes utilisant des projecteurs, il convient de considérer l'absence de représentation du monde physique. Dans le cas des casques munis de caméras, les modifications du point de vue sont plus limitées que sur dispositifs mobiles car l'utilisateur ne peut pas observer l'environnement physique directement.

La technique *Pointage relatif* pourrait aussi être adaptée à d'autres dispositifs comme par exemple à un système de RA utilisant un pico-projecteur tenu par l'utilisateur. Ce type de système est aussi affecté par les instabilités du point de vue liées aux tremblements naturels de la main, mais si le pico-projecteur est combiné à une surface tactile, la technique *Pointage relatif* peut être utilisée.

RA et Cognition

Comme la RA se base sur la co-localisation d'informations numériques avec des objets physiques, il serait intéressant d'étudier comment les utilisateurs perçoivent et comprennent cette combinaison d'informations dans l'espace. Cette perspective en directement liée au sujet du groupe de travail "INTERCO3D"[8] (INTERaction, COgnition et 3D dans les espaces physiques et numériques).

Le projet "Livre d'Anatomie Vivant - LBA"[9], auquel participe l'équipe IIHM est aussi lié à cette problématique. L'objectif de ce projet est d'étudier l'effet de l'encorporation (c'est à dire l"'embodiment") sur l'apprentissage de l'anatomie : pour cela le projet vise la conceptin d'une application de RA sur dispositifs mobiles qui affiche un modèle dynamique et personnalisé des structures anatomiques sur une zone du corps de l'étudiant (le genou par exemple). Certaines connaissances en anatomie nécessitent une bonne habilité visuo-spatiale et la présentation en RA sur le corps de l'étudiant des structures anatomiques et de leur dynamique peut peut-être favoriser leur compréhension et leur mémorisation.

[8]http ://www.irit.fr/INTERCO3D/
[9]https ://persyval-lab.org/en/sites/lba

List of Acronyms

2D: Two Dimensions
3D: Three Dimensions
A: movement Amplitude
AIST: national institute of Advanced Industrial Science and Technology
AMIE: Augmented Mobile Interactive Experience
ANR: Agence National de la Recherche
API: Application Programming Interface
AR: Augmented Reality
ASUR (ASUR++): Adapter-System-User-Real object notation
AV: Augmented Virtuality
CD: Control-to-Display gain
CfSR: Center for Service Research unit
CI: Confidence Interval
Collada: Collaborative Design Activity format
DoF: Degrees of freedom
EHCI: Engineering Human-Computer Interaction research group
GLFW: OpenGL Framework library
GUI: Graphical User Interface
HCI: Human-Computer Interaction
HMD: Head-Mounted Display
iOS: Apple iPad|iPhone|iPod Operating System
ID: Index of Difficulty
IIHM: Ingénierie de l'Interaction Homme-Machine research group (see EHCI)
INTERCO3D: INTERaction and COgnition in real and virtual 3D spaces French working group
ISMAR: International Symposium on Mixed and Augmented Reality
JST: Japan Science and Technology agency
LBA: Living Book of Anatomy
LED: Light-Emitting Diode
libjpeg: Joint Photographic Experts Group library
libpng: Portable Network Graphics library
LIG: Laboratoire d'Informatique de Grenoble
MobileHCI: international conference on Human-Computer Interaction with Mobile devices and services
MT: Movement Time
OpenCV: Open Computer Vision library
OpenGL|ES: Open Graphics Library for Embedded System
OS: Operating System

PDR: Pedestrian Dead Reckoning
POI: Point of interest
SDK: Software Development Kit
SVG: Scalable Vector Graphics format
UbiMob: journées francophones Mobilité et Ubiquité (French workshop on Mobility and Ubiquity)
Vuforia SDK: Vuforia Software Development Kit
VR: Virtual Reality
W: target Width

List of Figures

List of Tables

List of Publications

International refereed conference papers

- Ortega, M. and Vincent, T. Direct drawing on 3D shapes with automated camera control. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2014), April 26-May 1, Toronto, Ontario, Canada. ACM (2014), pp. 2047-2050.

- Vincent, T., Nigay, L. and Kurata, T. Precise pointing techniques for handheld Augmented Reality. In Proceedings of the 14th IFIP TC13 Conference on Human-Computer Interaction (INTERACT 2013), September 2-6, Cape Town, South Africa. IFIP-Springer (2013), pp. 122-139. (Honorary Mention of the Program Committee)

National refereed conference papers

- Vincent, T., Nigay, L. and Kurata, T. Handheld Augmented Reality: Effect of registration jitter on cursor-based pointing techniques. In Actes de la 25ème conférence francophone sur l'Interaction Homme-Machine (IHM 2013), November 13-15, Bordeaux, France. ACM (2013), pp. 1-6. (Best Paper Award)

- Scoditti, A., Vincent, T., Coutaz, J., Blanch, R., Mandran, N. TouchOver: Decoupling positioning from selection on touch-based handheld devices In Actes de la 23ème conférence francophone sur l'Interaction Homme-Machine (IHM 2011), October 24-27, Nice-Sophia Antipolis, France. ACM (2011), pp. 37-40.

Other conference papers

- Coutrix, C., Delamare, W., Guillon, M., Kurata, T., Leitner, F., Nigay, L., Vincent, T. Techniques de Pointage à Distance : Cibles Numériques et Cibles Physiques In Actes des 10èmes journées francophones Mobilité et Ubiquité (UbiMob 2014), June 5-6, Sophia Antipolis, France. (2014), 5 pages.

- Vincent, T., Pelurson, S., Regazzoni, V., Kurata, T. and Nigay, L. Relations spatiales en Réalité Augmentée sur dispositifs mobiles. In Actes des 9èmes journées francophones Mobilité et Ubiquité (UbiMob 2013), June 5-6, Nancy, France. INRIA (2013), 6 pages.

Workshop papers

- Vincent, T., Nigay, L. and Kurata, T. Handheld Augmented Reality: Spatial Relationships and Frames of Reference. In Workshop on Designing Mobile Augmented Reality at the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI 2013), August 27-30, Munich, Germany. (2013), 4 pages.
- Vincent, T., Nigay, L. and Kurata, T. Classifying handheld Augmented Reality: Three categories linked by spatial mappings. In Workshop on Classifying the AR Presentation Space at the 11th IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2012), November 5-8, Atlanta, Georgia, USA. (2012), 6 pages.

Doctoral Consortium

- Vincent, T. Spatial Relations in Handheld Augmented Reality Interaction. Doctoral Consortium of the 14th IFIP TC13 Conference on Human-Computer Interaction (INTERACT 2013), September 2-6, Cape Town, South Africa. (2013), 6 pages.

Demonstrations

- Vincent, T., Nigay, L. and Kurata, T. Handheld Augmented Reality Pointing Techniques. Demonstration at the 25ème conférence francophone sur l'Interaction Homme-Machine (IHM 2013), November 13-15, Bordeaux, France. (2013).
- Makita, K., Kourogi, M., Vincent, T., Okuma, T., Nishida, J., Ishikawa, T., Nigay, L., Kurata, T. Handheld AR/AV system using PDR localization and image based localization with virtualized reality models. Demonstration at the 11th IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2012), November 5-8, Atlanta, Georgia, USA. (2012).
- Kourogi, M., Makita, K., Vincent, T., Okuma, T., Nishida, J., Ishikawa, T., Nigay, L., Kurata, T. Handheld AR/AV indoor navigation and detailed information with contextual interaction. Demonstration at the 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2011), October 26-39, Basel, Switzerland. (2011).
- Scoditti, A., Vincent, T., Coutaz, J., Blanch, R., Mandran, N. TouchOver: Decoupling Positioning from Selection on Touch-based Handheld Devices. Demonstration at the 23ème conférence francophone sur l'Interaction Homme-Machine (IHM 2011), October 24-27, Nice-Sophia Antipolis, France. (2011).

Bibliography

[Aaftab Munshi, 2008] Aaftab Munshi, J. L. (2008). Opengl es common/common-lite profile specification version 1.1.12. Full specification, The Khronos Group Inc.

[Albinsson and Zhai, 2003] Albinsson, P.-A. and Zhai, S. (2003). High precision touch screen interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 105–112. ACM.

[Alessandro et al., 2010] Alessandro, M., Dünser, A., and Schmalstieg, D. (2010). Zooming interfaces for augmented reality browsers. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, pages 161–170. ACM.

[Argelaguet and Andujar, 2013] Argelaguet, F. and Andujar, C. (2013). A survey of 3d object selection techniques for virtual environments. *Computers & Graphics*, 37:121–136.

[Au et al., 2014] Au, O. K.-C., Su, X., and Lau, R. W. (2014). Lineardragger: A linear selector for target acquisition on touch screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 2607–2616. ACM.

[Azuma, 1997] Azuma, R. T. (1997). A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385.

[Azuma et al., 2001] Azuma, R. T., Baillot, Y., Behringer, R., Feiner, S., Julier, S., and MacIntyre, B. (2001). Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47.

[Bai et al., 2012] Bai, H., Lee, G. A., and Billinghurst, M. (2012). Freeze view touch and finger gesture based interaction methods for handheld augmented reality interfaces. In *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*, IVCNZ '12, pages 126–131. ACM.

[Balakrishnan, 2004] Balakrishnan, R. (2004). "beating" fitts' law: Virtual enhancements for pointing facilitation. *International Journal of Human-Computer Studies*, 61(6):857–874.

[Baldauf and Fröhlich, 2014] Baldauf, M. and Fröhlich, P. (2014). Snap target - investigating an assistance technique for mobile magic lens interaction with large displays. *International Journal of Human-Computer Interaction*, 30(6):446–458.

[Ballendat et al., 2010] Ballendat, T., Marquardt, N., and Greenberg, S. (2010). Proxemic interaction: designing for a proximity and orientation-aware environment. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 121–130. ACM.

[Baricevic et al., 2012] Baricevic, D., Lee, C., Turk, M., Höllerer, T., and Bowman, D. A. (2012). A hand-held ar magic lens with user-perspective rendering. In *Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality*, pages 197–206. IEEE.

[Barnes and Finch, 2008] Barnes, M. and Finch, E. L. (2008). COLLADA - digital asset schema release 1.5.0. Specification, The Khronos Group Inc.

[Baudisch et al., 2003] Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B. B., and Zierlinger, A. (2003). Drag-and-pop and drag-and-pick: techniques for accessing remote screen content on touch- and pen-operated systems. In *Proceedings of Interact*, INTERACT '03, pages 57–64. IOS Press.

[Baudisch et al., 2008] Baudisch, P., Zotov, A., Cutrell, E., and Hinckley, K. (2008). Starburst: A target expansion algorithm for non-uniform target distributions. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '08, pages 129–137. ACM.

[Beaudouin-Lafon, 2004] Beaudouin-Lafon, M. (2004). Designing interaction, not interfaces. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '04, pages 15–22. ACM.

[Bell et al., 2001] Bell, B., Feiner, S., and Höllerer, T. (2001). View management for virtual and augmented reality. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, pages 101–110. ACM.

[Bell et al., 2002] Bell, B., Höllerer, T., and Feiner, S. (2002). An annotated situation-awareness aid for augmented reality. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, UIST '02, pages 213–216. ACM.

[Benko et al., 2006] Benko, H., Wilson, A. D., and Baudisch, P. (2006). Precise selection techniques for multi-touch screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 1263–1272. ACM.

[Bérard and Rochet-Capellan, 2012] Bérard, F. and Rochet-Capellan, A. (2012). Measuring the linear and rotational user precision in touch pointing. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '12, pages 183–192. ACM.

[Bergstrom-Lehtovirta and Oulasvirta, 2014] Bergstrom-Lehtovirta, J. and Oulasvirta, A. (2014). Modeling the functional area of the thumb on mobile touchscreen surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 1991–2000. ACM.

[Betsworth et al., 2013] Betsworth, L., Rajput, N., Srivastava, S., and Jones, M. (2013). Audvert: Using spatial audio to gain a sense of place. In *Proceedings* *of the 14th IFIP TC 13 international conference on Human-computer interaction*, volume LNCS 8120 of *INTERACT '13*, pages 455–462. IFIP-Springer.

[Billinghurst et al., 2001] Billinghurst, M., Kato, H., and Poupyrev, I. (2001). The magicbook: A transitional ar interface. *Computers & Graphics*, 25(5):745–753.

[Blanch et al., 2004] Blanch, R., Guiard, Y., and Beaudouin-Lafon, M. (2004). Semantic pointing: Improving target acquisition with control-display ratio adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 519–526. ACM.

[Blanch and Ortega, 2009] Blanch, R. and Ortega, M. (2009). Rake cursor: Improving pointing performance with concurrent input channels. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1415–1418. ACM.

[Blanch and Ortega, 2011] Blanch, R. and Ortega, M. (2011). Benchmarking pointing techniques with distractors: adding a density factor to fitts' pointing paradigm. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1629–1638. ACM.

[Boring et al., 2010] Boring, S., Baur, D., Butz, A., Gustafson, S., and Baudisch, P. (2010). Touch projector: Mobile interaction through video. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2287–2296. ACM.

[Brooke, 1996] Brooke, J. (1996). Sus - a quick and dirty usability scale. In Jordan, P. W., Thomas, B., Lyall, M. I., and Weerdmeester, B., editors, *Usability Evaluation in Industry*, chapter 21. London: Taylor and Francis.

[Buxton, 1990] Buxton, W. (1990). A three-state model of graphical input. In *Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction*, INTERACT '90, pages 449–456. North-Holland Publishing Co.

[Casiez and Roussel, 2011] Casiez, G. and Roussel, N. (2011). No more bricolage!: methods and tools to characterize, replicate and compare pointing transfer functions. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 603–614. ACM.

[Casiez et al., 2008] Casiez, G., Vogel, D., Balakrishnan, R., and Cockburn, A. (2008). The impact of control-display gain on user performance in pointing tasks. *Human-Computer Interaction*, 23(3):215–250.

[Caudell and Mizell, 1992] Caudell, T. P. and Mizell, D. W. (1992). Augmented reality: An application of heads-up display technology to manual manufacturing processes. In *Proceedings of 1992 IEEE Hawaii International Conference on Systems Sciences*, pages 659–669. IEEE Computer Society.

[Chapuis and Dragicevic, 2011] Chapuis, O. and Dragicevic, P. (2011). Effects of motor scale, visual scale and quantization on small target acquistion difficulty. *ACM Transactions on Computer-Human Interaction*, 18(3):13:1–13:32.

[Chapuis et al., 2009] Chapuis, O., Labrune, J.-B., and Pietriga, E. (2009). Dynaspot: Speed-dependent area cursor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1391–1400. ACM.

[Cockburn et al., 2012] Cockburn, A., Ahlström, D., and Gutwin, C. (2012). Understanding performance in touch selections: Tap, drag and radial pointing drag with finger, stylus and mouse. *International Journal of Human-Computer Studies.*

[Coffin et al., 2011] Coffin, C., Lee, C., and Höllerer, T. (2011). Evaluating the impact of recovery density on augmented reality tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 93–101. IEEE Computer Society.

[Dahlström et al., 2011] Dahlström, E., McCormack, C., Dengler, P., Schepers, D., Jackson, D., Watt, J., Fujisawa, J., Grasso, A., Lilley, C., and Ferraiolo, J. (2011). Scalable vector graphics (svg) 1.1 (second edition). W3C recommendation, W3C.

[DiVerdi et al., 2004] DiVerdi, S., Hollerer, T., and Schreyer, R. (2004). Level of detail interfaces. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '04, pages 300–301. IEEE Computer Society.

[Dubois and Gray, 2008] Dubois, E. and Gray, P. (2008). A design-oriented information-flow refinement of the asur interaction model. In *Engineering Interactive Systems*, volume 4940 of *Lecture Notes in Computer Science*, pages 465–482. Springer Berlin / Heidelberg.

[Dubois et al., 2002] Dubois, E., Gray, P., and Nigay, L. (2002). Asur++: A design notation for mobile mixed systems. In *Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction*, MobileHCI '02, pages 123–139. Springer Berlin / Heidelberg.

[Dubois et al., 2001] Dubois, E., Nigay, L., and Troccaz, J. (2001). Consistency in augmented reality systems. In Little, M. and Nigay, L., editors, *Engineering for Human-Computer Interaction*, volume 2254 of *Lecture Notes in Computer Science*, pages 111–122. Springer Berlin / Heidelberg.

[Dubois et al., 1999] Dubois, E., Nigay, L., Troccaz, J., Chavanon, O., and Carrat, L. (1999). Classification space for augmented surgery, an augmented reality case study. In Sasse, A. and Johnson, C., editors, *Proceedings of Interact'99*, INTERACT '99, pages 353–359. IOS Press.

[Fischer et al., 2005] Fischer, J., Bartz, D., and Straßer, W. (2005). Stylized augmented reality for improved immersion. In *Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality*, VR '05, pages 195–202, 325. IEEE.

[Fitts, 1954] Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391.

[Gilliot, 2014] Gilliot, J. (2014). *Interactions multi-points indirectes sur grands écrans.* PhD thesis, Université de Lille 1.

[Gómez Jáuregui et al., 2012] Gómez Jáuregui, D. A., Argelaguet, F., and Lecuyer, A. (2012). Design and evaluation of 3d cursors and motion parallax for the exploration of desktop virtual environments. In *IEEE Symposium on 3D User Interfaces*, 3DUI '12, pages 69–76. IEEE Computer Society.

[Graham et al., 2000] Graham, T. C. N., Watts, L. A., Calvary, G., Coutaz, J., Dubois, E., and Nigay, L. (2000). A dimension space for the design of interactive systems within their physical environments. In *Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques*, DIS '00, pages 406–416. ACM.

[Grossman and Balakrishnan, 2005] Grossman, T. and Balakrishnan, R. (2005). The bubble cursor: Enhancing target acquisition by the bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 281–290. ACM.

[Guiard et al., 2004] Guiard, Y., Blanch, R., and Beaudouin-Lafon, M. (2004). Object pointing: A complement to bitmap pointing in guis. In *Proceedings of Graphics Interface 2004*, GI '04, pages 9–16. Canadian Human-Computer Communications Society.

[Guillon et al., 2014] Guillon, M., Leitner, F., and Nigay, L. (2014). Static voronoi-based target expansion technique for distant pointing. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '14, pages 41–48. ACM.

[Gunn et al., 2009] Gunn, T. J., Zhang, H., Mak, E., and Irani, P. (2009). An evaluation of one-handed techniques for multiple-target selection. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, CHI EA '09, pages 4189–4194. ACM.

[Gustafson et al., 2008] Gustafson, S., Baudisch, P., Gutwin, C., and Irani, P. (2008). Wedge: Clutter-free visualization of off-screen locations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 787–796. ACM.

[Guven et al., 2006] Guven, S., Feiner, S., and Oda, O. (2006). Mobile augmented reality interaction techniques for authoring situated media on-site. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '06, pages 235–236. IEEE Computer Society.

[Henze and Boll, 2011] Henze, N. and Boll, S. (2011). Who's that girl? handheld augmented reality for printed photo books. In *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part III*, INTERACT'11, pages 134–151. Springer-Verlag.

[Herling and Broll, 2012] Herling, J. and Broll, W. (2012). Pixmix: A real-time approach to high-quality diminished reality. In *Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '12, pages 141–150. IEEE.

[Hill et al., 2011] Hill, A., Schiefer, J., Wilson, J., Davidson, B., Gandy, M., and MacIntyre, B. (2011). Virtual transparency: Introducing parallax view into video see-through ar. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 239–240. IEEE Computer Society.

[Hinckley and Song, 2011] Hinckley, K. and Song, H. (2011). Sensor synaesthesia: Touch in motion, and motion in touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 801–810. ACM.

[Holloway, 1997] Holloway, R. L. (1997). Registration error analysis for augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):413–432.

[Holz and Baudisch, 2010] Holz, C. and Baudisch, P. (2010). The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 581–590. ACM.

[Holz and Baudisch, 2011] Holz, C. and Baudisch, P. (2011). Understanding touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2501–2510. ACM.

[Hugues et al., 2011] Hugues, O., Fuchs, P., and Nannipieri, O. (2011). *New Augmented Reality Taxonomy: Technologies and Features of Augmented Environment*, chapter 1, pages 47–63. Springer.

[Huot and Lecolinet, 2006] Huot, S. and Lecolinet, E. (2006). Spiralist: A compact visualization technique for one-handed interaction with large listes on mobile devices. In *Proceedings of the 4th Nordic Conference on Human-Computer Interaction*, NordiCHI '06, pages 445–448. ACM.

[Hwang et al., 2010] Hwang, S., Jo, H., and hee Ryu, J. (2010). Exmar: Expanded view of mobile augmented reality. In *Proceedings of the 9th annual IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '10, pages 235–236. IEEE Computer Society.

[Ishikawa et al., 2013] Ishikawa, T., Thangamani, K., Kourogi, M., Gee, A., Mayol-Cuevas, W., Hyun, J., and Kurata, T. (2013). Interactive 3-d indoor modeler for virtualizing service fields. *Virtual Reality*, 17(2):89–109.

[Ju et al., 2008] Ju, W., Lee, B. A., and Klemmer, S. R. (2008). Range: exploring implicit interaction through electronic whiteboard design. In *Proceedings of the 2008 conference on Computer Supported Cooperative Work*, CSCW '08, pages 17–26. ACM.

[Julier et al., 2002] Julier, S., Baillot, Y., Brown, D. G., and Lanzagorta, M. (2002). Information filtering for mobile augmented reality. *IEEE Computer Graphics and Applications*, 22(5):12–15.

[Karlson and Bederson, 2008] Karlson, A. K. and Bederson, B. B. (2008). One-handed touchscreen input for legacy applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1399–1408. ACM.

[Kourogi and Kurata, 2003] Kourogi, M. and Kurata, T. (2003). Personal positioning based on walking locomotion analysis with self-contained sensors and a wearable camera. In *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '03, pages 103–112. IEEE Computer Society.

[Lee et al., 2010a] Lee, C., Bonebrake, S., Höllerer, T., and Bowman, D. A. (2010a). The role of latency in the validity of ar simulation. In *Proceedings of the 2010 IEEE Virtual Reality Conference*, VR '10, pages 11–18. ACM.

[Lee and Billinghurst, 2011] Lee, G. A. and Billinghurst, M. (2011). A user study on the snap-to-feature interaction method. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 245–246. IEEE Computer Society.

[Lee et al., 2010b] Lee, G. A., Yang, U., Kim, Y., Jo, D., and Kim, K.-H. (2010b). Snap-to-feature interface for annotation in mobile augmented reality. In *Augmented Reality Super Models Workshop at The 9th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '10 workshop.

[Lee et al., 2009] Lee, G. A., Yang, U., Kim, Y., Jo, D., Kim, K.-H., Kim, J. H., and Choi, J. S. (2009). Freeze-set-go interaction method for handheld mobile augmented reality environments. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, VRST '09, pages 143–146. ACM.

[Liao et al., 2010] Liao, C., Liu, Q., Liew, B., and Wilcox, L. (2010). Pacer: fine-grained interactive paper via camera-touch hybrid gestures on a cell phone. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2441–2450. ACM.

[Lindeman and Noma, 2007] Lindeman, R. W. and Noma, H. (2007). A classification scheme for multi-sensory augmented reality. In *Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology*, VRST '07, pages 175–178. ACM.

[Liu et al., 2012] Liu, C., Huot, S., Diehl, J., Mackay, W. E., and Beaudouin-Lafon, M. (2012). Evaluating the benefits of real-time feedback in mobile augmented reality with hand-held devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2973–2976. ACM.

[Livingston and Ai, 2008] Livingston, M. A. and Ai, Z. (2008). The effect of registration error on tracking distant augmented objects. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 77–86. IEEE Computer Society.

[Mackay, 1998] Mackay, W. E. (1998). Augmented reality: Linking real and virtual worlds: A new paradigm for interacting with computers. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI 1998, pages 13–21. ACM.

[Meyer et al., 1988] Meyer, D. E., Abrams, R. A., Kornblum, S., Wright, C. E., and Smith, J. E. K. (1988). Optimality in human motor performance: ideal control of rapid aimed movements. *Psychological Review*, 95(3):340–370.

[Milgram and Colquhoun, 1999] Milgram, P. and Colquhoun, H. (1999). *A taxonomy of real and virtual world display integration*, volume Mixed Reality : Merging Real and Virtual Worlds, chapter 1, pages 1–26. Ohmsha/Springer-Verlag.

[Milgram and Kishino, 1994] Milgram, P. and Kishino, F. (1994). A taxonomy of mixed reality visual displays. *IEICE Transactions on Information Systems*, E77-D(12).

[Moscovich, 2009] Moscovich, T. (2009). Contact area interaction with sliding widgets. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, pages 13–22. ACM.

[Myers et al., 2002] Myers, B. A., Bhatnagar, R., Nichols, J., Peck, C. H., Kong, D., Miller, R., and Long, A. C. (2002). Interacting at a distance: measuring the performance of laser pointers and other devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '02, pages 33–40. ACM.

[Norman, 1986] Norman, D. A. (1986). Cognitive engineering. In Norman, D. A. and Draper, S. W., editors, *User Centered System Design, New Perspectives on Human-Computer Interaction*, pages 31–61. Lawrence Erlbaum Associates.

[Normand and Moreau, 2012] Normand, J.-M. and Moreau, G. (2012). Dof-based classification of augmented reality applications. In *Workshop on Classifying the Augmented Reality Presentation Space at ISMAR '12*.

[Normand et al., 2012] Normand, J.-M., Servières, M., and Moreau, G. (2012). A new typology of augmented reality applications. In *Proceedings of the 3rd Augmented Human International Conference*, AH '12, pages 18:1–18:8. ACM.

[Ortega and Vincent, 2014] Ortega, M. and Vincent, T. (2014). Direct drawing on 3d shapes with automated camera control. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 2047–2050. ACM.

[Potter et al., 1988] Potter, R. L., Weldon, L. J., and Schneiderman, B. (1988). Improving the accuracy of touch screens: An experimental evaluation of three strategies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, pages 27–32. ACM.

[Ragan et al., 2009] Ragan, E., Wilkes, C., Bowman, D. A., and Höllerer, T. (2009). Simulation of augmented reality systems in purely virtual environments. In *Proceedings of the 2009 IEEE Virtual Reality Conference*, VR '09, pages 287–288. IEEE Computer Society.

[Rekimoto, 1995] Rekimoto, J. (1995). The magnifying glass approach to augmented reality systems. In *Proceedings of the 1995 International Conference on Artificial Reality and Tele-Existence and Conference on Virtual Reality Software and Technology*, ICAT/VRST '95, pages 123–132.

[Rekimoto and Nagao, 1995] Rekimoto, J. and Nagao, K. (1995). The world through the computer: Computer augmented interaction with real world environments. In *Proceedings of the 8th annual ACM symposium on User interface and software technology*, UIST '95, pages 29–36. ACM.

[Renevier, 2004] Renevier, P. (2004). *Systèmes mixtes collaboratifs sur supports mobiles : Conception et réalisation.* PhD thesis, Université Joseph Fourier - Grenoble 1.

[Robertson and MacIntyre, 2007] Robertson, C. M. and MacIntyre, B. (2007). An evaluation of graphical context as a means for ameliorating the effect of registration error. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, pages 99–110. IEEE Computer Society.

[Robinson et al., 2009] Robinson, S., Eslambolchilar, P., and Jones, M. (2009). Sweep-shake: Finding digital resources in physical environments. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '09, pages 12:1–12:10.

[Rohs and Oulasvitra, 2008] Rohs, M. and Oulasvitra, A. (2008). Target acquisition with camera phones when used as magic lens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1409–1418. ACM.

[Rohs et al., 2011] Rohs, M., Oulasvitra, A., and Suomalainen, T. (2011). Interaction with magic lens: Real-world validation of a fitt's law model. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2725–2728. ACM.

[Rohs et al., 2009] Rohs, M., Schleicher, R., Schöning, J., Essl, G., Naumann, A., and Krüger, A. (2009). Impact of item density on the utility of visual context in magic lens interactions. *Journal Personal and Ubiquitous Computing*, 13(8):633–646.

[Roudaut, 2010] Roudaut, A. (2010). *Conception et Evaluation de Techniques d'Interaction pour Dispositifs Mobiles.* PhD thesis, Télécom ParisTech.

[Roudaut et al., 2008] Roudaut, A., Huot, S., and Lecolinet, E. (2008). Taptap and magstick: improving one-handed target acquisition on small touch-screens. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '08, pages 146–153. ACM.

[Sandor et al., 2010] Sandor, C., Cunningham, A., Dey, A., and Mattila, V.-V. (2010). An augmented reality x-ray system based on visual saliency. In *Proceedings of IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '10, pages 27–36. IEEE.

[Schinke et al., 2010] Schinke, T., Henze, N., and Boll, S. (2010). Visualization of off-screen objects in mobile augmented reality. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, pages 313–316. ACM.

[Scoditti et al., 2011] Scoditti, A., Vincent, T., Coutaz, J., Blanch, R., and Mandran, N. (2011). Touchover: Decoupling positioning from selection on touch-based handheld devices. In *Actes de la 23ème conférence francophone sur l'Interaction Homme-Machine*, IHM '11, pages 37–40. ACM.

[Sielhorst et al., 2007] Sielhorst, T., Sa, W., Khamene, A., Sauer, F., and Navab, N. (2007). Measurement of absolute latency for video see through augmented reality. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, pages 1–4. IEEE Computer Society.

[Soukoreff and MacKenzie, 2004] Soukoreff, R. W. and MacKenzie, I. S. (2004). Towards a standard for pointing device evaluation, perspectives on 27 years of fitts' law research in hci. *International Journal of Human-Computer Studies*, 61:751–789.

[Su et al., 2014] Su, X., Au, O. K.-C., and Lau, R. W. (2014). The implicit fan cursor: A velocity dependent area cursor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 753–762. ACM.

[Sukan and Feiner, 2012] Sukan, M. and Feiner, S. (2012). Using augmented snapshots for viewpoint switching and manipulation in augmented reality. In *Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts*, CHI EA '12, pages 1095–1098. ACM.

[Sukan et al., 2012] Sukan, M., Feiner, S., Tversky, B., and Energin, S. (2012). Quick viewpoint switching for manipulating virtual objects in hand-held augmented reality using stored snapshots. In *Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '12, pages 217–226. IEEE.

[Takeushi and Perlin, 2012] Takeushi, Y. and Perlin, K. (2012). Clayvision: The (elastic) image of the city. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2411–2420. ACM.

[Tatzgern et al., 2013] Tatzgern, M., Grasset, R., Veas, E., Kalkofen, D., Seichter, H., and Schmalstieg, D. (2013). Exploring distant objects with augmented reality. In *Proceedings of the 5th Joint Virtual Reality Conference*, JVRC '13, pages 49–56. Eurographics Association.

[Tönnis et al., 2013] Tönnis, M., Plecher, D. A., and Klinder, G. (2013). Representing information - classifying the augmented reality presentation space. *Computers & Graphics*, 37(8):997–1011.

[Tse et al., 2007] Tse, E., Hancock, M., and Greenberg, S. (2007). Speech-filtered bubble ray: Improving target acquisition on display walls. In *Proceedings of the 9th International Conference on Multimodal Interfaces*, ICMI '07, pages 307–314. ACM.

[Ventura et al., 2009] Ventura, J., Jang, M., Crain, T., Höllerer, T., and Bowman, D. A. (2009). Evaluating the effects of tracker reliability and field of view on a target following task in augmented reality. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, VRST '09, pages 151–154. ACM.

[Vincent et al., 2012] Vincent, T., Nigay, L., and Kurata, T. (2012). Classifying handheld augmented reality: Three categories linked by spatial mappings. In *Workshop on Classifying the Augmented Reality Presentation Space at ISMAR '12*.

[Vincent et al., 2013a] Vincent, T., Nigay, L., and Kurata, T. (2013a). Handheld augmented reality: Effect of registration jitter on cursor-based pointing techniques. In *Actes de la 25ème conférence francophone sur l'Interaction Homme-Machine*, IHM '13, pages 1:1–1:6. ACM.

[Vincent et al., 2013b] Vincent, T., Nigay, L., and Kurata, T. (2013b). Handheld augmented reality: Spatial relationships and frames of reference. In *Workshop Designing Mobile Augmented Reality at MobileHCI 2013*, MobileHCI '13 workshop.

[Vincent et al., 2013c] Vincent, T., Nigay, L., and Kurata, T. (2013c). Precise pointing techniques for handheld augmented reality. In *Proceedings of the 14th IFIP TC 13 international conference on Human-computer interaction - Volume Part I*, volume LNCS 8117 of *INTERACT '13*, pages 122–139. IFIP-Springer.

[Vincent et al., 2013d] Vincent, T., Pelurson, S., Regazzoni, V., Kurata, T., and Nigay, L. (2013d). Relations spatiales en réalité augmentée sur dispositifs mobiles. In *Actes des 9èmes journées francophones Mobilité et Ubiquité*, UbiMob 2013. INRIA.

[Vogel and Baudisch, 2007] Vogel, D. and Baudisch, P. (2007). Shift: A technique for operating pen-based interfaces using touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 657–666. ACM.

[Wagner et al., 2012] Wagner, J., Huot, S., and Mackay, W. E. (2012). Bitouch and bipad: Designing bimanual interaction for hand-held tablets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2317–2326. ACM.

[Wellner, 1993] Wellner, P. (1993). Interacting with paper on the digitaldesk. *Communications of the ACM*, 36(7):87–96.

[Wither et al., 2009] Wither, J., DiVerdi, S., and Höllerer, T. (2009). Annotation in outdoor augmented reality. *Computers & Graphics*, 33(6):679–689.

[Yatani et al., 2008] Yatani, K., Partridge, K., Bern, M., and Newman, M. W. (2008). Escape: A target selection technique using visually-cued gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 285–294. ACM.

Interaction en Réalité Augmenté sur dispositif mobile : Relations spatiales

Nous nous intéressons à l'interaction dans le contexte de la Réalité Augmentée sur dispositifs mobiles. Le dispositif mobile est utilisé comme une lentille magique qui 'augmente' la perception de l'environnement physique avec du contenu numérique. Nous nous sommes particulièrement intéressés aux relations spatiales entre le contenu affiché à l'écran et l'environnement physique. En effet la combinaison des environnements physique et numérique repose sur ces relations spatiales, comme l'adaptation de l'augmentation numérique en fonction de la localisation de l'utilisateur. Mais ces relations spatiales définissent aussi des contraintes pour l'interaction. Par exemple l'effet des tremblements naturels de la main rend instable la vidéo affichée sur l'écran du dispositif mobile et par conséquent a un impact direct sur la précision d'interaction. La question est alors, comment peut-on relâcher ces contraintes spatiales pour améliorer l'interaction sans pour autant casser la co-localisation physique-numérique. Nous apportons trois contributions.

- Tout d'abord, nous avons établi un espace de conception décrivant le contenu affiché à l'écran dans le contexte de la Réalité Augmentée sur dispositifs mobiles. Cet espace conceptuel met en exergue les relations spatiales entre les différents repères des composants le structurant. Cet espace permet d'étudier systématiquement la conception de techniques d'interaction dans le contexte de la Réalité Augmentée sur dispositifs mobiles.
- Deuxièmement, nous avons conçu des techniques de pointage améliorant la précision du pointage en Réalité Augmentée sur supports mobiles. Ces techniques de pointage ont été évaluées lors d'expériences utilisateur.
- Enfin, dans le cadre du projet en partenariat AIST-Tuskuba, Schneider France et Schneider Japon dans lequel s'inscrit cette thèse, nous avons développé une boîte à outils pour le développement d'applications de Réalité Augmentée sur dispositifs mobiles. Cette boîte à outils a été utilisée pour développer plusieurs démonstrateurs.

Handheld Augmented Reality Interaction: Spatial Relations

We explored interaction within the context of handheld Augmented Reality (AR), where a handheld device is used as a physical magic lens to 'augment' the physical surrounding. We focused, in particular, on the role of spatial relations between the on-screen content and the physical surrounding. On the one hand, spatial relations define opportunities for mixing environments, such as the adaptation of the digital augmentation to the user's location. On the other hand, spatial relations involve specific constraints for interaction such as the impact of hand tremor on on-screen camera image stability. The question is then, how can we relax spatial constraints while maintaining the feeling of digital-physical collocation. Our contribution is three-fold.

- First, we propose a design space for handheld AR on-screen content with a particular focus on the spatial relations between the different identified frames of reference. This design space defines a framework for systematically studying interaction with handheld AR applications.
- Second, we propose and evaluate different handheld AR pointing techniques to improve pointing precision. Indeed, with handheld AR set-up, both touch-screen input and the spatial relations between the on-screen content and the physical surrounding impair the precision of pointing.
- Third, as part of a collaborative research project involving AIST-Tsukuba and Schneider-France and Japan, we developed a toolkit supporting the development of handheld AR applications. The toolkit has been used to develop several demonstrators.

Discipline : **Informatique**, spécialité : **Interaction Homme-Machine**
Mots Clés : **Réalité Augmentée, Interaction sur Dispositifs Mobiles, Techniques de Pointage, Espace de Conception.**

Thèse préparée au **Laboratoire d'Informatique de Grenoble**.