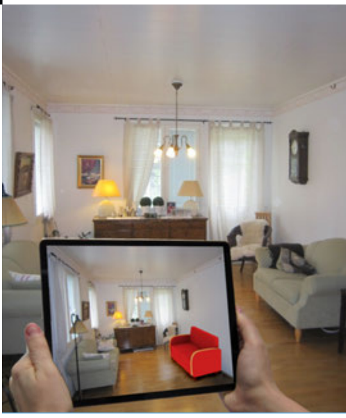
VISIONS • SCIENCE • TECHNOLOGY • RESEARCH HIGHLIGHTS

3

Theory and applications of marker-based augmented reality



Sanni Siltanen

VTT SCIENCE 3

Theory and applications of marker-based augmented reality

Sanni Siltanen

ISBN 978-951-38-7449-0 (soft back ed.)
ISSN 2242-119X (soft back ed.)

ISBN 978-951-38-7450-6 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 2242-1203 (URL: http://www.vtt.fi/publications/index.jsp)

JULKAISIJA – UTGIVARE – PUBLISHER

VTT
PL 1000 (Vuorimiehentie 5, Espoo)
02044 VTT
Puh. 020 722 111, faksi 020 722 4374

VTT
PB 1000 (Bergsmansvägen 5, Esbo)
FI-2044 VTT
Tfn. +358 20 722 111, telefax +358 20 722 4374

VTT Technical Research Centre of Finland
P.O. Box 1000 (Vuorimiehentie 5, Espoo)
FI-02044 VTT, Finland
Tel. +358 20 722 111, fax + 358 20 722 4374

Kopijyvä Oy, Kuopio 2012

Theory and applications of marker-based augmented reality

[Markkeriperustaisen lisätyn todellisuuden teoria ja sovellukset].
Sanni Siltanen. Espoo 2012. VTT Science 3. 198 p. + app. 43 p.

Abstract

Augmented Reality (AR) employs computer vision, image processing and computer graphics techniques to merge digital content into the real world. It enables real-time interaction between the user, real objects and virtual objects. AR can, for example, be used to embed 3D graphics into a video in such a way as if the virtual elements were part of the real environment. In this work, we give a thorough overview of the theory and applications of AR.

One of the challenges of AR is to align virtual data with the environment. A marker-based approach solves the problem using visual markers, e.g. 2D barcodes, detectable with computer vision methods. We discuss how different marker types and marker identification and detection methods affect the performance of the AR application and how to select the most suitable approach for a given application.

Alternative approaches to the alignment problem do not require furnishing the environment with markers: detecting natural features occurring in the environment and using additional sensors. We discuss these as well as hybrid tracking methods that combine the benefits of several approaches.

Besides the correct alignment, perceptual issues greatly affect user experience of AR. We explain how appropriate visualization techniques enhance human perception in different situations and consider issues that create a seamless illusion of virtual and real objects coexisting and interacting. Furthermore, we show how diminished reality, where real objects are removed virtually, can improve the visual appearance of AR and the interaction with real-world objects.

Finally, we discuss practical issues of AR application development, identify potential application areas for augmented reality and speculate about the future of AR. In our experience, augmented reality is a profound visualization method for on-site 3D visualizations when the user's perception needs to be enhanced.

Keywords augmented reality, AR, mixed reality, diminished reality, marker-based tracking, tracking, markers, visualization

Markkeriperustaisen lisätyn todellisuuden teoria ja sovellukset

[Theory and applications of marker-based augmented reality].
Sanni Siltanen. Espoo 2012. VTT Science 3. 198 s. + liitt. 43 s.

Tiivistelmä

Lisätty todellisuus yhdistää digitaalista sisältöä reaalimaailmaan tietokonenäön, kuvankäsittelyn ja tietokonegrafiikan avulla. Se mahdollistaa reaaliaikaisen vuorovaikutuksen käyttäjän, todellisten esineiden ja virtuaalisten esineiden välillä. Lisätyn todellisuuden avulla voidaan esimerkiksi upottaa 3D-grafiikkaa videokuvaan siten, että virtuaalinen osa sulautuu ympäristöön aivan kuin olisi osa sitä. Tässä työssä esitän perusteellisen katsauksen lisätyn todellisuuden teoriasta ja sovelluksista.

Eräs lisätyn todellisuuden haasteista on virtuaalisen tiedon kohdistaminen ympäristöön. Näkyviä tunnistemerkkejä eli markkereita hyödyntävä lähestymistapa ratkaisee tämän ongelman käyttämällä esimerkiksi 2D-viivakoodeja tai muita keinonäön keinoin tunnistettavia markkereita. Työssä kerrotaan, kuinka erilaiset markkerit ja tunnistusmenetelmät vaikuttavat lisätyn todellisuuden sovelluksen suorituskykyyn, ja kuinka valita kuhunkin tarkoitukseen soveltuvin lähestymistapa.

Kohdistamisongelman vaihtoehtoiset lähestymistavat eivät vaadi markkereiden lisäämistä ympäristöön; ne hyödyntävät ympäristössä olevia luonnollisia piirteitä ja lisäantureita. Tämä työ tarkastelee näitä vaihtoehtoisia lähestymistapoja sekä hybridimenetelmiä, jotka yhdistävät usean menetelmän hyötyjä.

Oikean kohdistamisen lisäksi ihmisen hahmottamiskykyyn liittyvät asiat vaikuttavat lisätyn todellisuuden käyttäjäkokemukseen. Työssä selitetään, kuinka tarkoituksenmukaiset visualisointimenetelmät parantavat hahmottamiskykyä erilaisissa tilanteissa, sekä pohditaan asioita, jotka auttavat luomaan saumattoman vaikutelman virtuaalisten ja todellisten esineiden vuorovaikutuksesta. Lisäksi työssä näytetään, kuinka häivytetty todellisuus, jossa virtuaalisesti poistetaan todellisia asioita, voi parantaa visuaalista ilmettä ja helpottaa vuorovaikutusta todellisten esineiden kanssa lisätyn todellisuuden sovelluksissa.

Lopuksi käsitellään lisätyn todellisuuden sovelluskehitystä, yksilöidään potentiaalisia sovellusalueita ja pohditaan lisätyn todellisuuden tulevaisuutta. Kokemukseni mukaan lisätty todellisuus on vahva visualisointimenetelmä paikan päällä tapahtuvaan kolmiulotteiseen visualisointiin tilanteissa, joissa käyttäjän havainnointikykyä on tarpeen parantaa.

Avainsanat augmented reality, AR, mixed reality, diminished reality, marker-based tracking, tracking, markers, visualization

Preface

First of all, I would like to thank the VTT Augmented Reality Team for providing an inspiring working environment and various interesting projects related to augmented reality. I am also grateful for having great colleagues elsewhere at VTT. In addition, I would like to thank the Jenny and Antti Wihuri Foundation for its contribution to financing this work.

I am happy to have had the opportunity to receive supervision from Professor Erkki Oja. His encouragement was invaluable to me during the most difficult moments of the process. I have enjoyed interesting discussions with my advisor Timo Tossavainen and I would like to thank him for his encouragement, support and coffee.

The postgraduate coffee meetings with Paula were a life-saver and an enabler of progress. Not to mention all the other creative activities and fun we had together. The Salsamania group made a great effort to teach me the right coordinates and rotations. The salsa dancing and the company of these wonderful people were of great benefit to my physical and mental wellbeing. I also give my heartfelt thanks to all my other close friends. I have been lucky enough to have so many great friends I cannot possibly mention all of them by name.

I am ever grateful for the presence of my mother Sirkka and my brother Konsta who persuaded me to study mathematics at high school, which eventually led me to my current career. My sister Sara has always been my greatest support. I am happy to have the best sister anyone could wish for.

My children Verneri, Heini and Aleksanteri are truly wonderful. They bring me back to everyday reality with their activity, immediacy and thoughtfulness. I am so happy they exist.

Most of all I want to thank my dear husband Antti who took care of all the practical, quotidian stuff while I was doing research. He has always been by my side and supported me; I could not have done this without him.

Contents

Abstract **3**

Tiivistelmä **4**

Preface **5**

List of acronyms and symbols **9**

1. Introduction **12**
- 1.1 Contribution 13
- 1.2 Structure of the work 14

2. Augmented reality **16**
- 2.1 Terminology 16
- 2.2 Simple augmented reality 19
- 2.3 Augmented reality as an emerging technology 21
- 2.4 Augmented reality applications 23
- 2.5 Multi-sensory augmented reality 32
  - 2.5.1 Audio in augmented reality 32
  - 2.5.2 Sense of smell and touch in mixed reality 34
- 2.6 Toolkits and libraries 35
- 2.7 Summation 37

3. Marker-based tracking **38**
- 3.1 Marker detection 40
  - 3.1.1 Marker detection procedure 40
  - 3.1.2 Pre-processing 41
  - 3.1.3 Fast acceptance/rejection tests for potential markers 44
- 3.2 Marker pose 47
  - 3.2.1 Camera transformation 49
  - 3.2.2 Camera calibration matrix and optical distortions 49
  - 3.2.3 Pose calculation 51
  - 3.2.4 Detection errors in pose calculation 53
  - 3.2.5 Continuous tracking and tracking stability 54
  - 3.2.6 Rendering with the pose 55

3.3 Multi-marker setups (marker fields) 57
3.3.1 Predefined multi-marker setups 58
3.3.2 Automatic reconstruction of multi-marker setups 59
3.3.3 Bundle adjustment 61
3.3.4 Dynamic multi-marker systems 62

4. Marker types and identification 64
4.1 Template markers 65
4.1.1 Template matching 66
4.2 2D barcode markers 68
4.2.1 Decoding binary data markers 70
4.2.2 Error detection and correction for binary markers 70
4.2.3 Data randomising and repetition 71
4.2.4 Barcode standards 72
4.2.5 Circular markers 73
4.3 Imperceptible markers 74
4.3.1 Image markers 74
4.3.2 Infrared markers 76
4.3.3 Miniature markers 80
4.4 Discussion on marker use 83
4.4.1 When to use marker-based tracking 83
4.4.2 How to speed up marker detection 87
4.4.3 How to select a marker type 88
4.4.4 Marker design 89
4.4.5 General marker detection application 90

5. Alternative visual tracking methods and hybrid tracking 92
5.1 Visual tracking in AR 93
5.1.1 Pose calculation in visual tracking methods 94
5.2 Feature-based tracking 94
5.2.1 Feature detection methods 96
5.2.2 Feature points and image patches 97
5.2.3 Optical flow tracking 98
5.2.4 Feature matching 98
5.2.5 Performance evaluation of feature descriptors 100
5.2.6 Feature maps 101
5.3 Hybrid tracking 101
5.3.1 Model-based tracking 102
5.3.2 Sensor tracking methods 102
5.3.3 Examples of hybrid tracking 104
5.4 Initialisation and recovery 105

6. Enhancing the augmented reality system 107
6.1 Enhancing visual perception 107
6.1.1 Non-photorealistic rendering 108
6.1.2 Photorealistic rendering 109

6.1.3 Illumination and shadows 109
6.1.4 Motion blur, out-of-focus and other image effects 112
6.2 Diminished reality 114
6.2.1 Image inpainting 114
6.2.2 Diminishing markers and other planar objects 116
6.2.3 Diminishing 3D objects 124
6.3 Relation with the real world 128
6.3.1 Occlusion handling 128
6.3.2 Collisions and shadows 132

7. Practical experiences in AR development 136
7.1 User interfaces 136
7.2 Avoiding physical contacts 141
7.3 Practical experiences with head-mounted displays 142
7.4 Authoring and dynamic content 143

8. AR applications and future visions 145
8.1 How to design an AR application 145
8.2 Technology adoption and acceptance 146
8.3 Where to use augmented reality 150
8.3.1 Guidance 151
8.3.2 Visualisation 151
8.3.3 Games, marketing, motivation and fun 151
8.3.4 Real-time special video effects 152
8.3.5 World browsers and location-based services 152
8.3.6 Other 153
8.4 Future of augmented reality 153
8.4.1 Technology enablers and future development 154
8.4.2 Avatars 159
8.4.3 Multi-sensory mixed reality 160

9. Conclusions and discussion 163
9.1 Main issues in AR application development 163
9.2 Closure 165

References 167

Appendices

Appendix A: Projective geometry
Appendix B: Camera model
Appendix C: Camera calibration and optimization methods

List of acronyms and symbols

Acronyms

AGPS	Assisted GPS (see also GPS)
API	Application Programming Interface
AR	Augmented Reality
B/W	Black and White
BA	Bundle Adjustment
BCI	Brain-Computer-Interface
BIM	Building Information Model
CAD	Computer Aided Design
CV	Computer Vision
DGPS	Differential GPS (see also GPS)
DLT	Direct Linear Transformation
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
GAFD	Gravity-Aligned Feature Descriptor
GPS	Global Positioning System
GREFD	Gravity-Rectified Feature Descriptor
HMD	Head-Mounted Display
HUD	Head-Up Display
ID	Identification (number)
IoT	Internet of Things
IR	Infra Red

KF	Kalman Filter
MAR	Mobile Augmented Reality
MBI	Machine-Brain-Interface
MMR	Mobile Mixed Reality
MR	Mixed Reality
NFC	Near Field Communication
NPR	Non-Photorealistic Rendering
OCR	Optical Character Recognition
PC	Personal Computer
PCA	Principal Component Analysis
PDA	Personal Digital Assistant
POI	Point of Interest
PSF	Point Spread Function
PTAM	Parallel Tracking And Mapping
PTZ	Pan-Tilt-Zoom (e.g. PTZ camera)
RFID	Radio Frequency Identification
RGB	Red, Green, Blue (RGB image consists of R, G and B channels)
RID	Retinal Imaging Display
SaaS	Software-as-a-Service
SfM	Structure from Motion (in literature also SFM)
SLAM	Simultaneous Localisation And Mapping
TOF	Time-Of-Flight
UI	User Interface
UMPC	Ultra Mobile PC
URL	Universal Resource Locator
UX	User Experience
VR	Virtual Reality
VRD	Virtual Retinal Display

Notations

T	**T**ransformation Matrix
T	**T**ranslation Matrix
A	**A**ffine Transformation Matrix
M	General **M**atrix
L	**L**inear Transformation Matrix
R	**R**otation Matrix
S	**S**caling Matrix
P	Perspective **P**rojection Matrix
K	Camera Matrix
C	Camera **C**alibration Matrix
t	Translation Vector

1. Introduction

Augmented reality (AR) is a field of computer science research that combines real world and digital data. It is on the edge of becoming a well-known and commonplace feature in consumer applications: AR advertisements appear in newspapers such as *Katso*, *Seura*, *Cosmopolitan*, *Esquire* and *Süddeutche Zeitung*. Printed books (e.g. *Dibitassut*) have additional AR content. As a technology, augmented reality is now on the top of the "technology hype curve". New augmented reality applications mushroom all the time. Even children's toys increasingly have AR links to digital content. For example, in 2010 Kinder launched chocolate eggs with toys linked to AR content if presented to a webcam.

Traditional AR systems, such as systems for augmenting lines and records in sport events on TV, used to be expensive and required special devices. In recent years, the processing capacity of the computational units has increased tremendously, along with transmission bandwidth and memory capacity and speed. This development of technology has enabled the transition of augmented reality onto portable, everyday and cheap off-the-shelf devices such as mobile phones. This in turn opens mass markets for augmented reality applications as the potential users already have the suitable platform for AR. Furthermore, cloud computing and cloud services enable the use of huge databases even on mobile devices. This development enables a new type of location-based services exploiting large city models, for example.

New mobile phones feature cameras as standard, most laptops have a built-in camera, and people use social media applications like MSN Messenger and Skype for video meetings and are accustomed to operating webcams. At a general level, consumers are ready for adapting augmented reality as one form of digital media.

Augmented reality benefits industrial applications where there is a need to enhance the user's visual perception. Augmented 3D information helps workers on assembly lines, or during maintenance work and repair, to carry out required tasks. This technology also enables visualisation of new building projects on real construction sites, which gives the viewer a better understanding of relations with the existing environment.

What is behind the term "augmented reality"? What is the technology and what are the algorithms that allow us to augment 3D content in reality? What are the limits and possibilities of the technology? This work answers these questions. We describe the pipeline of augmented reality applications. We explain algorithms and methods that enable us to create the illusion of an augmented coexistence of digital and real content. We discuss the best ways to manage interactions in AR systems. We also discuss the limits and possibilities of AR technology and its use.

1.1 Contribution

Over the last ten years, the author has worked in the Augmented Reality Team (formerly the Multimedia Team) at VTT Technical Research Centre of Finland. In this licentiate thesis, she gives an overview of the augmented reality field based on the knowledge gathered by working on numerous research projects in this area.

Often AR solutions are developed for lightweight mobile devices or common consumer devices. Therefore, the research focus is on single camera visual augmented reality. In many cases, non-expert users use the applications in unknown environments. User interfaces and user interactions have been developed from this viewpoint. In addition, marker-based systems have many advantages in such cases, as we justify later in this work. In consequence, the author's main contribution is in marker-based applications. Often, the ultimate goal is a mobile solution, even though the demonstration may run on a PC environment. Hence, the focus is on methods that require little processing capacity and little memory. Naturally, all development aims for real-time processing.

These goals guide all of the research presented in this work. However, we do give an overview of the state-of-the-art in augmented reality and refer to other possible solutions throughout the work.

The author has authored and co-authored 16 scientific publications [1–16]. She has also contributed to several project deliverables and technical reports [17, 18]. She has done algorithm and application development and contributed to software inventions and patent applications related to augmented reality. She has also contributed to the ALVAR (A Library for Virtual and Augmented Reality) software library [19].

This work capitalises on the author's contributions to these publications, but also contains unpublished material and practical knowledge related to AR application development. In the following, we describe the main contribution areas.

The author has developed marker-based AR in numerous research projects. In addition, she has been involved in designing and implementing an adaptive 2D-barcode system for user interaction on mobile phones. During this marker-related research, the author has developed methods for fast and robust marker detection, identification and tracking. In the publications [3, 8, 10, 11, 17] the author has focused on these issues of marker-based AR.

Besides marker-based tracking, the author has developed feature and hybrid tracking solutions and initialisation methods for AR. Some of this work has been published in [1, 4, 15].

During several application development projects, the author considered suitable user interaction methods and user interfaces for augmented reality and closely related fields. Several publications [2, 3, 5–7, 11, 12, 17] report the author's research in this field. In Chapter 7, we present previously unpublished knowledge and findings related to these issues.

The author has developed diminished reality, first for hiding markers in AR applications, but also for hiding real-time objects. Part of this work has been published in [10, 14]. Section 6.2 presents previously unpublished results regarding diminished reality research.

The author has contributed to several application fields. The first AR project was a virtual advertising customer project ten years ago, using an additional IR camera. The project results were confidential for five years, and so were not previously published. We refer to some experiences from this project in Section 4.3. The author has since contributed to several application areas. Two of the most substantial application areas are augmented assembly and interior design. Publications [2, 5–7] cover work related to augmented assembly. Publications [9, 12, 13, 16, 18] describe the author's work in the area of AR interior design applications. Many of the examples presented in this work arise from these application areas. For instance, in Chapter 6 we use our work on interior design applications as an example for realistic illumination in AR.

1.2 Structure of the work

The work is organised as follows: Chapter 2 provides a general overview of augmented reality and the current state-of-the-art in AR. It is aimed at readers who are more interested in the possibilities and applications of augmented reality than in the algorithms used in implementing AR solutions. We also assume that Chapters 6–9 are of interest to the wider audience.

Chapter 3 focuses on marker-based tracking. We concentrate on marker detection, pose calculation and multi-marker setups. Chapter 4 describes different marker type identification and includes a discussion on marker use.

In Chapter 5, we cover alternative visual tracking methods, hybrid tracking and general issues concerning tracking. We concentrate on the feature-based approach, but also briefly discuss model-based tracking and sensor tracking in the context of hybrid tracking.

We discuss ways to enhance augmented reality in Chapter 6. We consider this the most interesting part of the work. We concentrate on issues that greatly affect user experience: visual perception and the relation with the real world. We focus especially on diminished reality, which is used both to enhance the visual appearance and to handle relations with the real world.

We report our practical experiences in AR development in Chapter 7. We discuss user interfaces and other application issues in augmented reality.

In Chapter 7, we discuss technology adoption and acceptance in the development of AR. We summarize the main application areas in which AR is beneficial and, finally, speculate about the future of AR.

We end this work with conclusions and a discussion in Chapter 8. We revise the main issues of AR application development and design and make our final remarks.

Throughout the work, numerous examples and references are presented to give the reader a good understanding of the diversity and possibilities of augmented reality applications and of the state-of-the-art in the field.

The appendices present a theoretical background for those readers who are interested in the mathematical and algorithmic fundamentals used in augmented reality. Appendix A covers projective geometry, Appendix B focuses on camera models and Appendix C relates to camera calibration.

2. Augmented reality

Augmented reality (AR) combines real world and digital data. At present, most AR research uses live video images, which the system processes digitally to add computer-generated graphics. In other words, the system *augments* the image with digital data. Encyclopaedia Britannica [20] gives the following definition for AR: *"Augmented reality, in computer programming, a process of combining or 'augmenting' video or photographic displays by overlaying the images with useful computer-generated data."*

Augmented reality research combines the fields of computer vision and computer graphics. The research on computer vision as it applies to AR includes among others marker and feature detection and tracking, motion detection and tracking, image analysis, gesture recognition and the construction of controlled environments containing a number of different sensors. Computer graphics as it relates to AR includes for example photorealistic rendering and interactive animations.

Researchers commonly define augmented reality as a real-time system. However, we also consider augmented still images to be augmented reality as long as the system does the augmentation in 3D and there is some kind of interaction involved.

2.1 Terminology

Tom Caudell, a researcher at aircraft manufacturer Boeing coined the term *augmented reality* in 1992. He applied the term to a head-mounted digital display that guided workers in assembling large bundles of electrical wires for aircrafts [21]. This early definition of augmented reality was a system where virtual elements were blended into the real world *to enhance the user's perception*. Figure 1 presents Caudell's head-mounted augmented reality system.

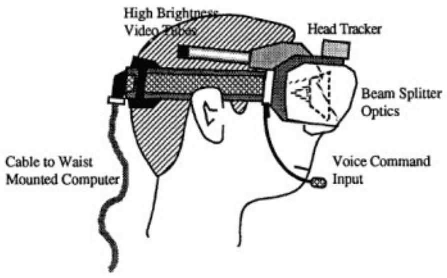


Figure 1. Early head-mounted system for AR, illustration from [21].

Later in 1994, Paul Milgram presented the *reality-virtuality continuum* [22], also called the mixed reality continuum. One end of the continuum contains the real environment, reality, and the other end features the virtual environment, virtuality. Everything in between is *mixed reality* (Figure 2). A Mixed Reality (MR) system merges the real world and virtual worlds to produce a new environment where *physical and digital objects co-exist and interact*. *Reality* here means the physical environment, in this context often the visible environment, as seen directly or through a video display.

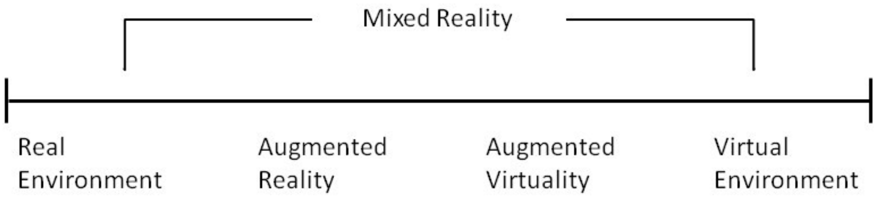


Figure 2. Milgram’s reality-virtuality continuum.

In 1997, Ronald Azuma published a comprehensive survey on augmented reality [23] and due to the rapid development in the area produced a new survey in 2001 [24]. He defines augmented reality as a system identified by three characteristics:

- it combines the real and the virtual
- it is interactive in real time
- it is registered in 3D.

Milgram and Azuma defined the taxonomy for adding content to reality or virtuality. However, a system can alter the environment in other ways as well; it can, for example, change content and remove or hide objects.

In 2002, Mann [25] added a second axis to Milgram’s virtuality-reality continuum to cover other forms of alteration as well. This two-dimensional reality-virtuality-mediality continuum defines *mediated reality* and *mediated virtuality* (see left illustration in Figure 3).

In *mediated reality,* a person's perception of reality is manipulated in one way or another. A system can change reality in different ways. It may add something (*augmented reality*), remove something (*diminished reality*) or alter it in some other way (*modulated reality*). Mann also presented the relationships of these areas in the Venn diagram (see right illustration in Figure 3). In diminished reality, we remove existing real components from the environment. Thus, diminished reality is in a way the opposite of augmented reality.

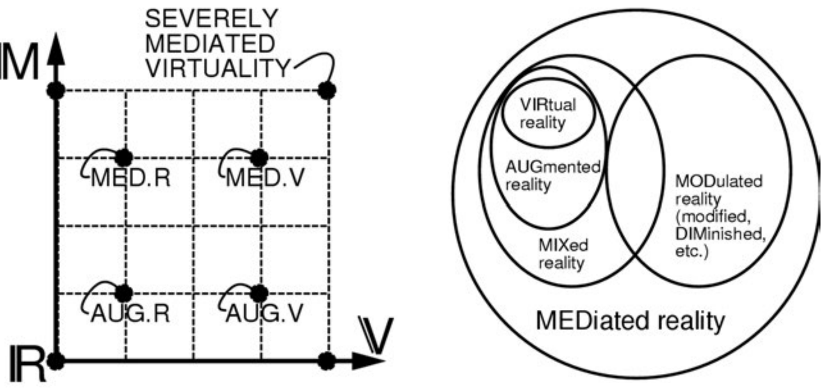


Figure 3. Mann's reality-virtuality-mediality continuum from [25].

Today most definitions of augmented reality and mixed reality are based on the definitions presented by Milgram, Azuma and Mann. However, the categorisation is imprecise and demarcation between different areas is often difficult or volatile, and sometimes even contradictory. For example, Mann defined virtual reality as a sub area of mixed reality, whereas Azuma completely separates total virtuality from mixed reality.

We define *virtual reality* (VR) as an immersive environment simulated by a computer. The simplest form of virtual reality is a 3D image that the user can explore interactively from a personal computer, usually by manipulating keys or the mouse. Sophisticated VR systems consist of wrap-around display screens, actual VR rooms, wearable computers, haptic devices, joysticks, etc. We can expand virtual reality to *augmented virtuality*, for instance, by adding real elements such as live video feeds to the virtual world.

Augmented reality applications mostly concentrate on *visual augmented reality* and to some extent on tactile sensations in the form of haptic feedback. This work also focuses on visual AR; other senses are covered briefly in Sections 2.5 Multi-sensory augmented reality and 8.4 Future of augmented reality.

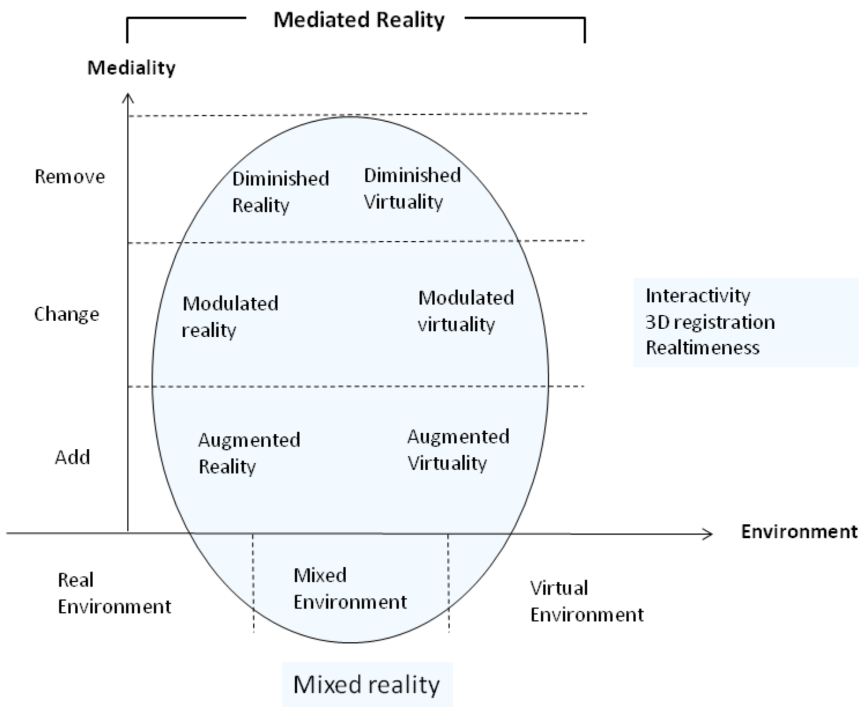


Figure 4. Mediated reality taxonomy.

We summarise the taxonomy for *mediated reality* in Figure 4. From left to right we have the reality–virtuality environment axis, the middle of which contains all combinations of the real and virtual, the *mixed environments*. The mediality axis is enumerable; we can add, remove or change its contents. *Mediated reality* consists of all types of mediality in mixed environments. The subgroup of mediated reality, which includes interaction, 3D registration and real-time components, is *mixed reality*.

Advertisers use mediated reality to enhance the attraction of their products and their brands in general. They manipulate face pictures in magazines by removing blemishes from the face, smoothing the skin, lengthening the eyelashes, etc. Editors adjust the colours, contrast and saturation. They change the proportions of objects and remove undesired objects from images. We consider this kind of offline image manipulation to be outside of the mixed or augmented reality concept.

2.2 Simple augmented reality

A simple augmented reality system consists of a camera, a computational unit and a display. The camera captures an image, and then the system augments virtual objects on top of the image and displays the result.

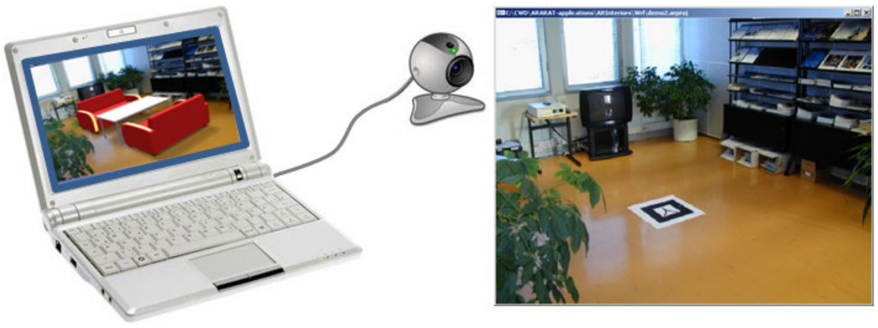


Figure 5. Example of a simple augmented reality system setup.

Figure 5 illustrates an example of a simple marker-based augmented reality system. The system captures an image of the environment, detects the marker and deduces the location and orientation of the camera, and then augments a virtual object on top of the image and displays it on the screen.

Figure 6 shows a flowchart for a simple augmented reality system. The capturing module captures the image from the camera. The tracking module calculates the correct location and orientation for virtual overlay. The rendering module combines the original image and the virtual components using the calculated pose and then renders the augmented image on the display.

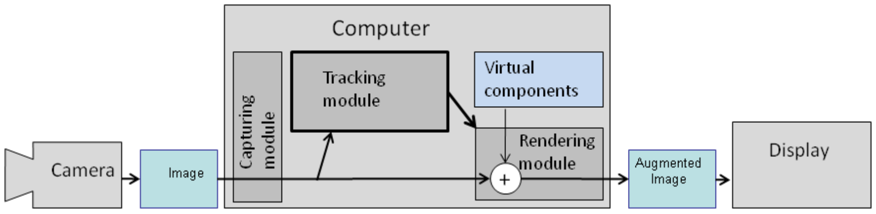


Figure 6. Flowchart for a simple AR system.

The tracking module is “the heart” of the augmented reality system; it calculates the relative pose of the camera in real time. The term *pose* means the six degrees of freedom (DOF) position, i.e. the 3D location and 3D orientation of an object. The tracking module enables the system to add virtual components as part of the real scene. The fundamental difference compared to other image processing tools is that in augmented reality virtual objects are moved and rotated in 3D coordinates instead of 2D image coordinates.

The simplest way to calculate the pose is to use markers. However, the mathematical model (projective geometry) behind other pose calculation methods is the same. Similar optimisation problems arise in different pose calculation methods and are solved with the same optimisation methods. We can consider markers to be a special type of features and thus it is natural to explain marker-based methods first and then move on to feature-based methods and hybrid tracking methods. We concentrate on marker-based augmented reality. We also give an overview of the projective geometry necessary in augmented reality in Appendix A. We discuss marker-based visual tracking in Chapter 3 and alternative visual tracking methods and hybrid tracking in Chapter 5.

Image acquisition is of minor interest in augmented reality. Normally a readily available video capturing library (e.g. DSVideoLib or HighGui) is used for the task. Augmented reality toolkits and libraries normally provide support for capturing as well.

The rendering module draws the virtual image on top of the camera image. In basic computer graphics, the virtual scene is projected on an image plane using a virtual camera and this projection is then rendered. The trick in augmented reality is to use a virtual camera identical to the system's real camera. This way the virtual objects in the scene are projected in the same way as real objects and the result is convincing. To be able to mimic the real camera, the system needs to know the optical characteristics of the camera. The process of identifying these characteristics is called camera calibration. Camera calibration can be part of the AR system or it can be a separate process. Many toolkits provide a calibration tool, e.g. ALVAR and ARToolKit have calibration functionality. A third party tool can also be used for calibration, e.g. Matlab and OpenCV have a calibration toolkit. Through this work, we assume that we have a correctly calibrated camera. For more detail about camera calibration, see Appendix C.

The variety of possible devices for an augmented reality system is huge. These systems can run on a PC, laptop, mini-PC, tablet PC, mobile phone or other computational unit. Depending on the application, they can use a digital camera, USB camera, FireWire Camera or the built-in camera of the computational unit. They can use a head-mounted display, see-through display, external display or the built-in display of the computational unit, or the system may project the augmentation onto the real world or use a stereo display. The appropriate setup depends on the application and environment. We will give more examples of different AR systems and applications in Section 2.4 and throughout this work.

2.3 Augmented reality as an emerging technology

ICT research and consulting company Gartner maintains *hype cycles* for various technologies. The hype cycle provides a cross-industry perspective on the technologies and trends for emerging technologies. Hype cycles show how and when technologies move beyond the hype, offer practical benefits and become widely accepted [26]. According to Gartner, hype cycles aim to separate the hype from the reality. A hype cycle has five stages (see Figure 7):

1. Technology trigger
2. Peak of inflated expectations
3. Trough of disillusionment

4. Slope of enlightenment
5. Plateau of productivity.

In Gartner's hype cycle for emerging technologies in 2011 [27] augmented reality has just passed the peak, but is still at stage *Peak of inflated expectations* (see Figure 7). Gartner's review predicts the time for mainstream adoption to be 5–10 years. Augmented reality is now on the hype curve in a position where *mass media hype begins.* Those who have been observing the development of augmented reality have noticed the tremendous increase in general interest in augmented reality. A few years ago, it was possible to follow blog writings about augmented reality. Today it is impossible. In October 2011, a Google search produced almost 90,000 hits for *"augmented reality blog".*

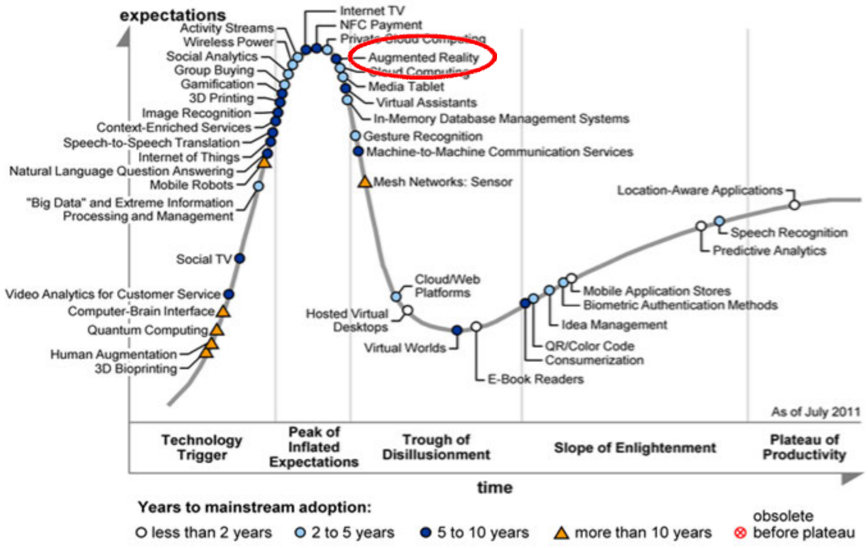


Figure 7. Gartner hype cycle for emerging technologies in 2011, with AR highlighted, image courtesy of Gartner.

Gartner treats the augmented reality field as one entity. However, there is variation among different application areas of augmented reality; they move at different velocities along the hype curve and some are still in the early stages whereas others are mature enough for exploitation.

Augmented reality is a hot topic especially in the mobile world. MIT (Massachusetts Institute of Technology) foresaw its impact on the mobile environment. In 2007 they predicted that Mobile Augmented Reality (MAR) would be one of the technologies *"most likely to alter industries, fields of research, and the way we live"* in their annual technology review [28]. The recent development of mobile platforms (e.g. iPhone, Android), services and cloud computing has really expanded *mobile augmented reality*. Gartner predicts MAR to be one of the key factors for next-generation location-aware services [29].

The New Media Consortium (NMC) [30] releases their analysis of the future of technology in a series called the *Horizon Report* every year. It identifies and describes emerging technologies likely to have a large impact on teaching, learning and research. The Horizon Report 2010 [31] predicts the time-to-adoption of augmented reality to be four to five years for educational use.

2.4 Augmented reality applications

Augmented reality technology is beneficial in several application areas. It is well suited for on-site visualisation both indoors and outdoors, for visual guidance in assembly, maintenance and training. Augmented reality enables interactive games and new forms of advertising. Several location-based services use augmented reality browsers. In printed media, augmented reality connects 3D graphics and videos with printed publications. In addition, augmented reality has been tested in medical applications and for multi-sensory purposes. The following presents a few examples of how visual AR has been used, and multi-sensory AR will be discussed later in Section 2.5.

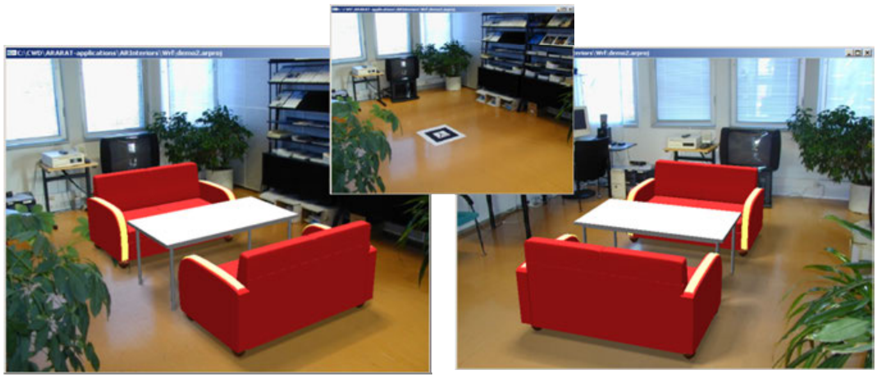


Figure 8. Augmented reality interior design (image: VTT Augmented Reality team).

In interior design, augmented reality enables users to *virtually test how a piece of furniture fits in their own living room.* Augmented reality interior design applications often use still images. However, the user interactions happen in real-time and the augmentation is in 3D. For example in our AR interior application [12], the user takes images of the room and uploads them onto a computer (see Figure 8). The user can then add furniture, and move and rotate it interactively. A more recent example of augmented reality interior design is VividPlatform AR+ [32]. Vivid Works presented it at the 2010 Stockholm Furniture Fair. VividPlatform AR+ also uses still images. Our experience is that users find still images convenient for interior design. However, interior design can use live video in PC environments or on mobile phones as well [33].

Outdoor visualisation systems normally use live video. Figure 9 shows an example of real-time augmented reality outdoor visualisation [34].

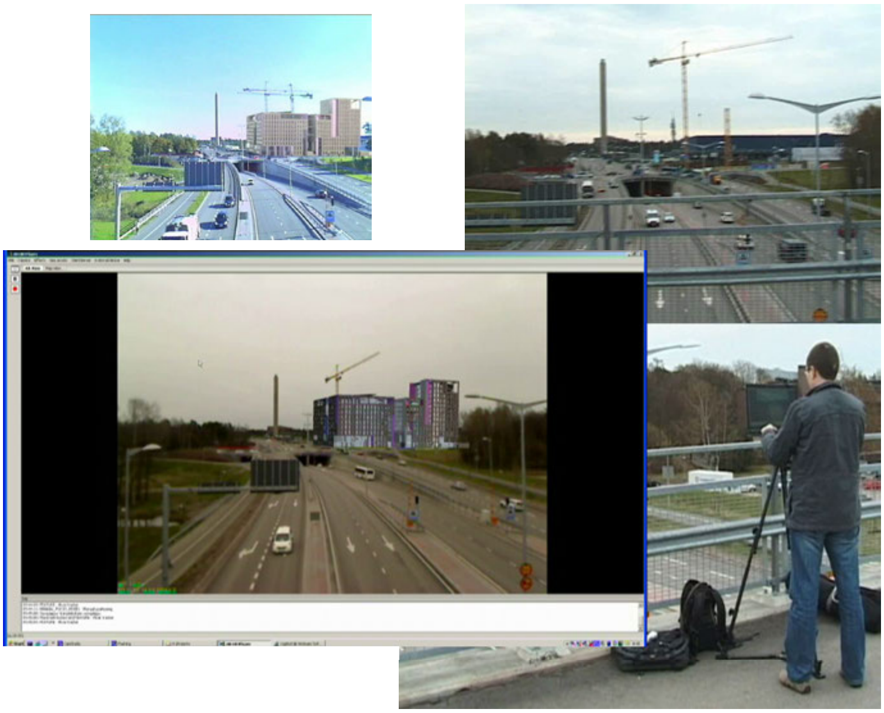


Figure 9. Outdoor visualisation: the user (bottom-right) sees the actual environment (top-right) and the augmented reality with the new building project through the display (bottom-left). The augmentation is adapted to the environment lighting (top-left). (Image: VTT Augmented Reality team).

Building projects can also be visualised using an augmented reality web camera. The augmented reality web camera can have several user interactions. Using a PTZ camera, the user can pan, tilt and zoom in on the view as in [9], for example. If the system has connection to the BIM (Building Information Model), the user can interact with materials and browse through the timeline of a construction project as we demonstrated in [1].

In assembly, augmented reality applications can show the instructions for the assembler at each stage. The system can display the instructions on a head-mounted display as e.g. in our assembly demonstration [2], on a mobile phone [35] or on a normal display (see Figure 10). The user can interact with an assembly system using voice commands, gestures or a keypad as we demonstrated in [7] and [6]. The benefits of augmented reality instructions compared to a printed manual are clear. The user can see instructions from all viewpoints and concentrate on assembly without having to scan through the paper manual.

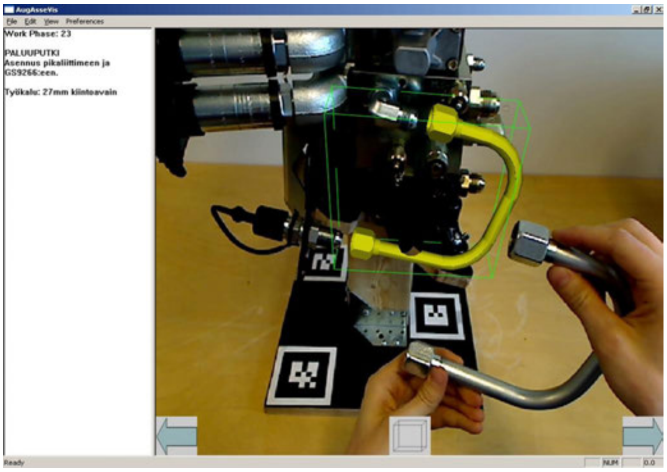

AugAsseVis
File Edit View Preferences
Work Phase: 23
PALUUPUTKI
Asennus pikaliittimeen ja GS9266:een.
Työkalu: 27mm kiintoavain
Ready
NUM
0.0

Figure 10. Augmented reality assembly instructions, figure from [36].

An AR system can aid maintenance work with augmented information, similarly to assembly. For instance, a mobile augmented reality system can provide maintenance workers relevant information from a database [37]. A mobile device is a good choice for displaying information in many cases. However, if the maintenance task is more of a hands-on assembly type of task, a head-mounted display is often a better choice. ARMAR is an augmented reality system for maintenance and repair developed at Columbia University [38, 39]. It uses a head-mounted display to show AR instructions for the maintenance worker, see Figure 11. The qualitative survey with ARMAR showed that the mechanics found the augmented reality condition intuitive and satisfying for the tested sequence of tasks [40].

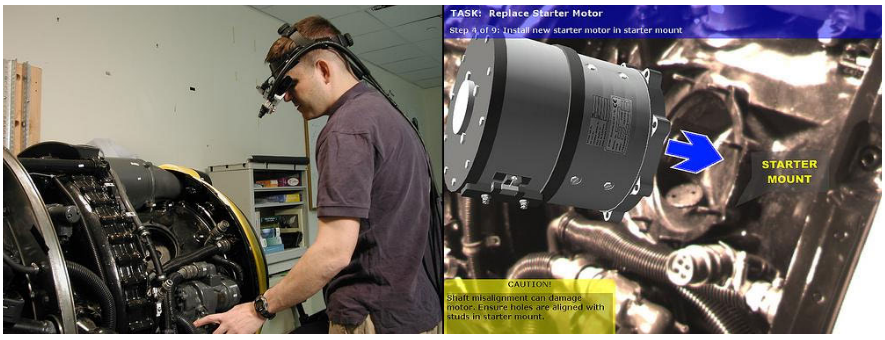

TASK: Replace Starter Motor
Step 4 of 9: Install new starter motor in starter mount
STARTER MOUNT
CAUTION!
Shaft misalignment can damage motor. Ensure holes are aligned with studs in starter mount.

Figure 11. ARMAR: augmented reality for maintenance and repair (image courtesy of Steven Feiner.

Besides assembly and engine maintenance, augmented reality is also used for teaching maintenance and repair, and for training purposes in several other fields as well.

In the game industry, AR has had a breakthrough. AR enables interaction with the user and the environment. Augmented reality can make games more attractive. For example, mobile game developer int13 [41] believes that *"Augmented Reality is a promising idea to enhance the player's gaming experience in providing exciting new ways to control his actions, through position and 3D moves."*

In addition, accuracy is less critical in games than in industrial or medical applications. Figure 12 is an example of a Kinder augmented reality game. A toy car found in a Kinder Surprise egg will launch an interactive game on a computer. The game detects the object (in this case the toy car) and then uses gesture detection. The user controls the race with hand gestures imitating steering wheel movements.

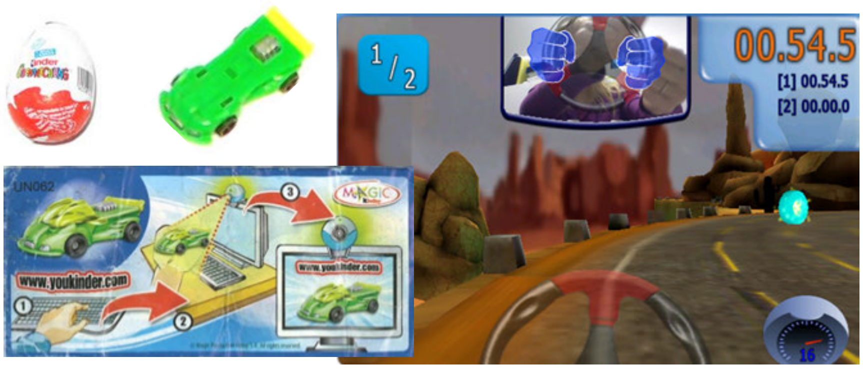


Figure 12. Kinder augmented reality game, November 2010.

Augmented reality mobile games are very popular; in November 2011, a quick search in the App Store resulted in about 200 mobile AR applications for the iPhone. Figure 13 shows one example of a mobile AR game, AR Defender (by int13, 2010), which works on iPhone and Samsung platforms, for example. It uses markers for camera registration, an example of which is shown in the lower right-hand corner of the left image in Figure 13.

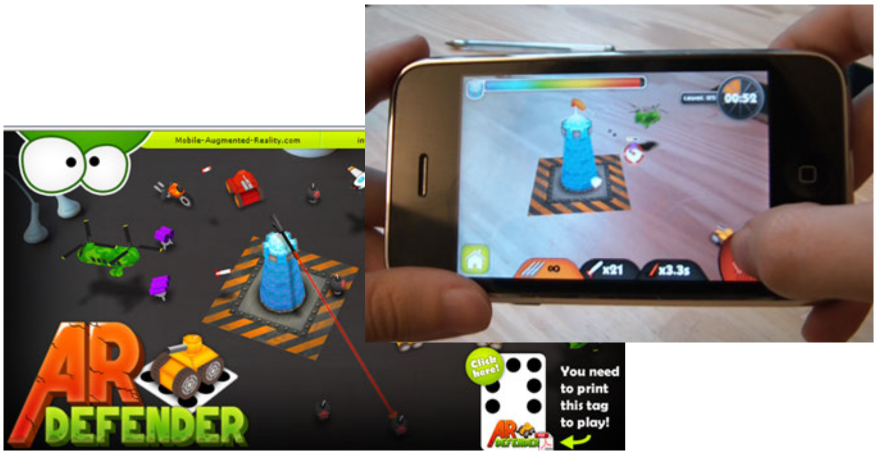

Mobile-Augmented-Reality.com
AR DEFENDER
Click here!
You need to print this tag to play!
00:52
x21
x3.3s

Figure 13. Mobile augmented reality game AR Defender uses markers for pose tracking (Images courtesy of Int13).

SpecTrek (Games4All, 2011) is another augmented reality game for Android phones. It uses GPS and a camera to guide the user to capture ghosts from the environment. In the map view, it shows the locations of the ghosts. In the camera view, it augments the ghosts in the view and allows the user to catch them (Figure 14).

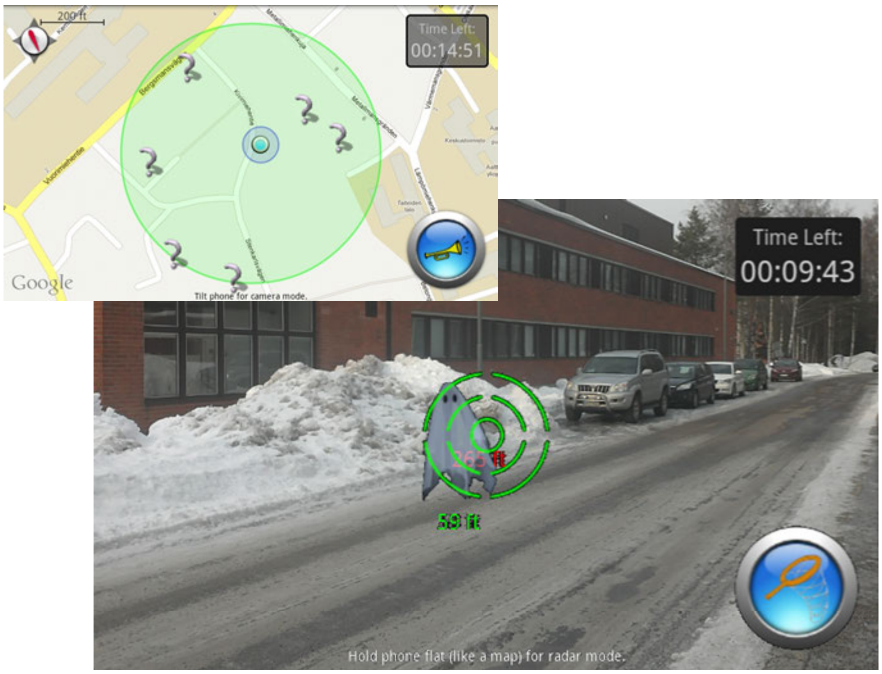

200 ft
Time Left:
00:14:51
Google
Tilt phone for camera mode.
Time Left:
00:09:43
59 ft
Hold phone flat (like a map) for radar mode.

Figure 14. Illustrations of screenshots from SpecTrek mobile AR game.

Besides games, location-based augmented reality services are popular on mobile platforms. One example is Wikitude World Browser, which uses GPS, a compass and the camera of the mobile device to augment location-based information for the user. It functions on several platforms (Symbian, Android and iPhone). Wikitude Drive also uses Navteq's maps to create augmented navigation instructions. Figure 15 shows examples of Wikitude World Browser. Currently several AR mobile browsers are on the market: Layar, Junaio Glue, Acrossair Browser, Yelp monocle, Robot Vision's Bing Local Search, PresseLite applications, etc.

AR browsers have two main approaches. The first approach is to have one browser and then different information environments, and the user can then choose which information the application augments. Layar, Junaio Glue and Wikitude use this approach. (In Junaio, the environments are called "channels", in Wikitude "worlds" and in Layar "layers"). The user can choose to see tourist information, for example. The other approach is to assign each information layer to a separate application. PresseLite uses this approach; Paris metro Guide and London Cycle Hire for the Tube are separate programs.

Yelp is a system used for sharing user reviews and recommendations on restaurants, shopping, nightlife, services, etc. Its Monocle add-on functionality bridges this social media with real world environments using augmented reality. It is probably the world's first social media browser. The user interface has motion detection; the user activates the monocle by shaking the phone.

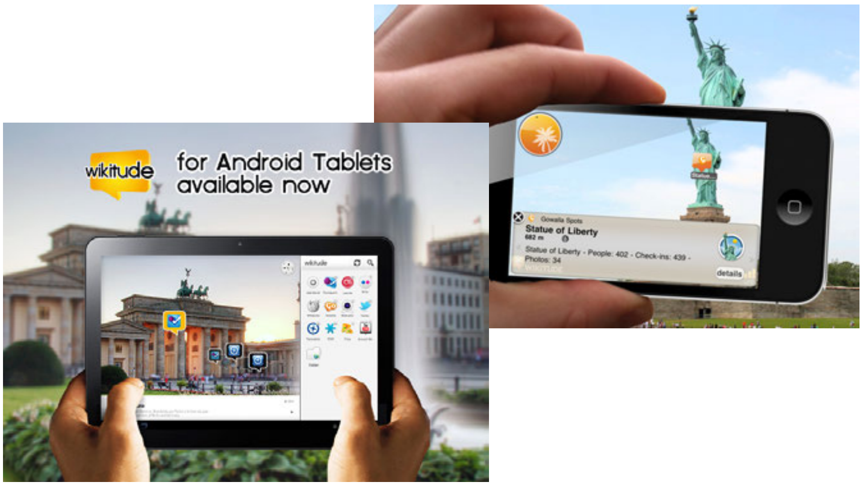


Figure 15. Wikitude World Browser (Images courtesy of Wikitude).

Augmented reality by its nature is well suited to advertising. In 2010, different companies launched advertising campaigns using AR. One of these is Benetton's campaign (2010) IT'S:MY:TIME. It connects their advertisements in journals, on billboards and in product catalogues with augmented reality. They use the same symbology in all of them (Figure 16). The small icons indicate that the user can use a webcam or download an application from the App Store. The AR application then augments videos on top of the marker, e.g. in the catalogue. The PC version uses Adobe Flash Player, which most users already have installed on the computer and thus do not need to download anything new.

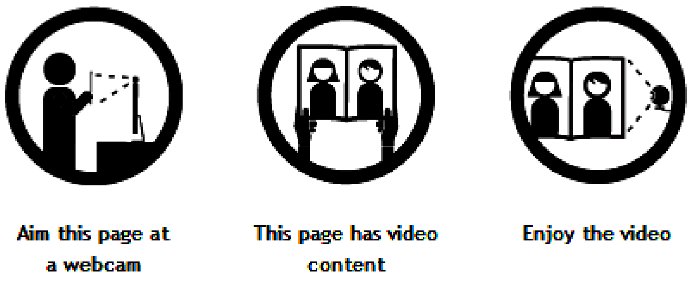


Figure 16. Benetton's AR campaign links user-created content (videos) and social media to catalogues, journals and billboards. It uses the same symbols everywhere to indicate the availability of virtual content.

Augmented reality technology is used to enrich printed media. *Esquire* magazine published an augmented reality issue in December 2009, *Süddeutche Zeitung* released their first issue with AR content in August 2010 (Figure 17). In *Esquire*'s case, users were able to see AR content when they showed the magazine to a PC webcam. In the case of *Süddeutche Zeitung*, users could see the content with a mobile phone after downloading the application. In Finland, *Katso* and *TVSeiska* magazines used AR in cooperation with VTT in advertising a new animated children series called *Dibitassut* in April 2010. Brazilian newspaper *O estado de Sao Paulo* has featured regular AR content since 2009.

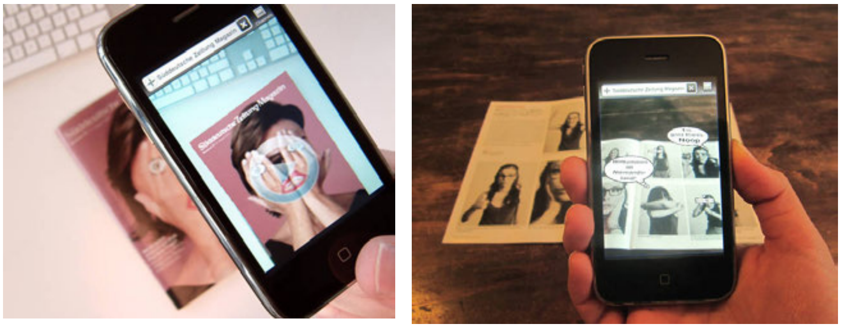


Figure 17. Example of augmented reality in magazines and newspapers: *Süddeutche Zeitung* (Images courtesy of Metaio).

The idea of an augmented reality book, "the magic book" is at least ten years old [42]. However, it took a while before the technology was robust enough for mass markets. *Aliens & UFOs* [43] was probably the first published book with AR content. In 2010, publishers released several AR books, e.g. *Dinosaurs Alive!* [44], *Fairyland Magic* [45], *Dibitassut* [46] and [47], and the trend continues. *Dibitassut* ("Dibidogs" in English) has a program made by VTT, which users can download and install on their own computers. Users can see augmented animations using a webcam (see Figure 18).

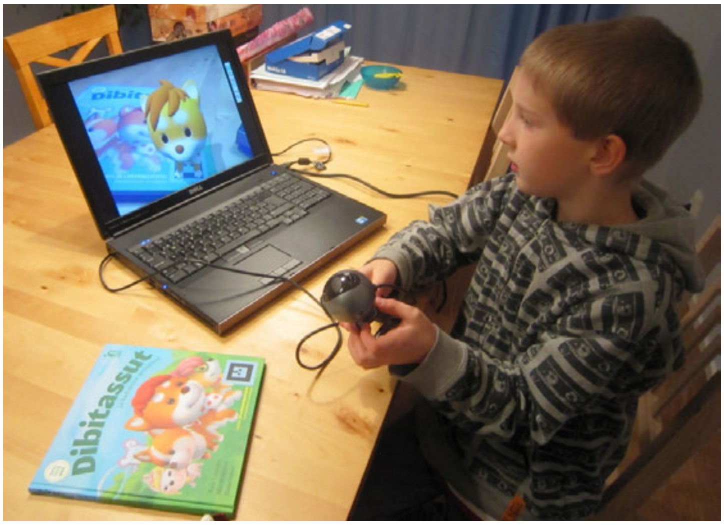


Figure 18. Dibidogs (*Dibitassut*) augmented reality book in use.

Medical applications require absolute reliability and a high degree of accuracy. Therefore, medical applications have more frequently been demonstrations than real applications.

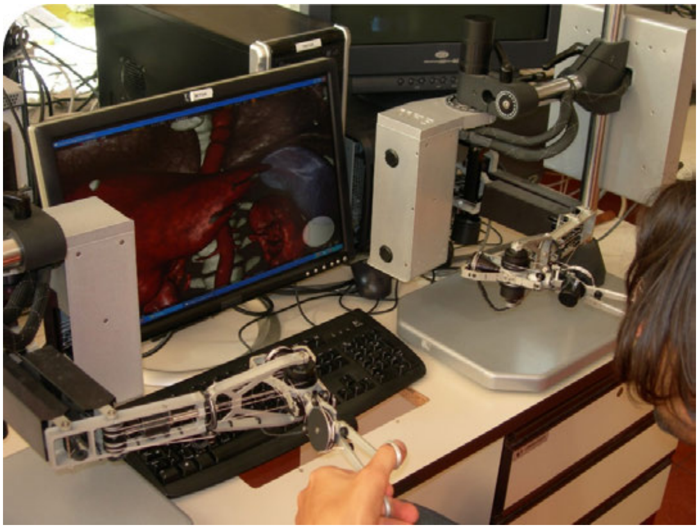


Figure 19. ALTAIR Robotics Lab's augmented reality surgical simulator (Image courtesy of ALTAIR [48]).

Researchers have proposed AR for laparoscopic surgery, for instance [49, 50]. In addition, augmented reality is used for medical and surgical training (Figure 19) and dental surgery training [51, 52].

2.5 Multi-sensory augmented reality

User experience (UX) is defined as *“a person's perceptions and responses that result from the use or anticipated use of a product, system or service”* ([53]). User experience is about how a person feels about using a system. The usability of the system is only one thing that affects user experience. The user and the context of the use influence UX as well. UX includes all the users' emotions, beliefs, preferences, perceptions, physical and psychological responses, behaviour and accomplishments that occur before, during and after use.

An AR system can expand the user experience by providing stimulus for other senses in addition to visual augmentation. A system can improve the immersivity of a mixed reality application with augmented 3D sound, scent, sense of touch, etc. In this section, we discuss the state-of-the-art of non-visual augmented reality and multi-sensory augmentation in mixed reality.

2.5.1 Audio in augmented reality

Audio has mainly been used in two different ways in augmented reality: as part of the user interface or for aural augmentation.

For example in our AR assembly demo [7], audio was used as one modality of the multimodal user interface. The user was able to give audio commands and the system gave feedback with audio signals (beeps). Interactive sound effects are used in the mobile phone version of the Dibidogs (*Dibitassut*) demo (mentioned in the previous section), for example, where the dog starts to growl if the user gets too close. This kind of non-directional audio is trivial from the technological point of view of audio processing. Yet even the simple use of audio brings a new dimension to mobile applications.

For the visually impaired augmented audio can give a better understanding of the environment. A good example is LookTel [54], which is a smartphone application for the visually impaired (Figure 20). Augmented audio and the audio interface are only parts of its functionality. The system uses optical character recognition (OCR) and computer vision techniques to detect objects and read texts. The user may point at objects with the device. The application then reads the information aloud.

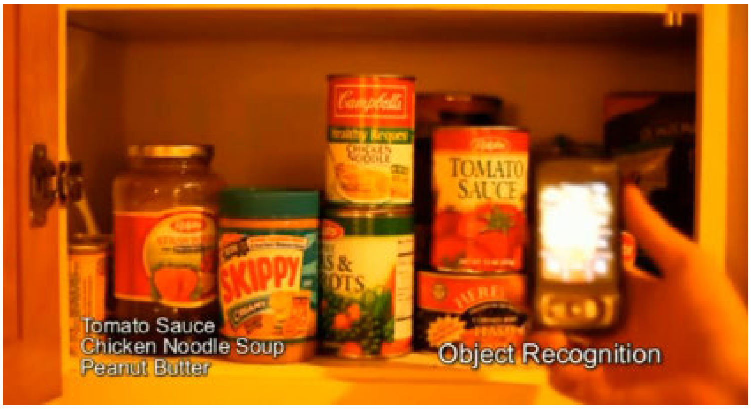


Figure 20. The LookTel smartphone application for the visually impaired recognises objects and characters, and reads aloud things at which the user points using the mobile phone (image courtesy of LookTel).

Similarly, the Hyperfit hybrid media application reads aloud nutritional information, which the system locates in a database [55]. Another similar application for the visually impaired is vOICe for Android [56], which adds sonic augmented reality overlay to the live camera view in real time. The vOICe technology is compatible with a head-mounted camera, in which case the system shares the view with the user. Sometimes the boundary between hybrid media and augmented reality is blurred. Hybrid media connects digital information with printed media or physical objects. Depending on how the connection is made and how the information is then presented it may be considered a sort of augmented reality.

Audio information can be geotagged in a similar way as any other information and then used for location-aware services such as Toozla [57]. Toozla can be described as an audio augmented reality browser. It works in a similar way to Layar's Wikitude in that it uses location services (GPS) and then gives users audio commentary on the subscribed channel (similarly to Wikitude's visual layers). Possible channels are e.g. the Touristic channel for information about nearby landmarks and the Service channel for promotions and information about shops and businesses, a Weather Channel and a Chat channel. Toozla works on several platforms and phone models.

3D sound is a research topic of its own, and numerous 3D sound recording, creation and playing systems exist. In movie and home theatre systems, 3D sounds are an ordinary feature. However, there is a fundamental difference between 3D sound in a film and that in augmented reality. In film, the desired relative position of the sound source is known beforehand and is fixed. In mixed reality applications, the user may be situated in any direction of the desired sound source position and in any orientation. This means in practice that the desired sound direction is known only after the user's pose is calculated for each time step.

3D sounds are more explored in the virtual reality end of the Milgram's mixed reality continuum than in augmented reality. Nevertheless, some studies cover the use of audio in AR, e.g. [58].

2.5.2 Sense of smell and touch in mixed reality

In closed-space environments, it is possible to control the environment and enrich the user experience by involving other senses such as the sense of smell, touch and warmth. Heilig invented the first multi-sensory simulator, called Sensorama, in 1962. Sensorama was a motorcycle simulator with visuals, sound, vibration and smell [59].

A more recent example of a multi-sensory environment is Pömpeli, a video space with multi-sensory user experience (see Figure 21) created at the Laurea University of Applied Sciences [60]. One setup is installed at Helsinki Airport, where tourists can look at videos of Finland. The visual experience is augmented with variety of scents, smells and wind blow that match with what is seen. In addition, temperature and a lighting atmosphere adapt to scenes and actions in the video [61].

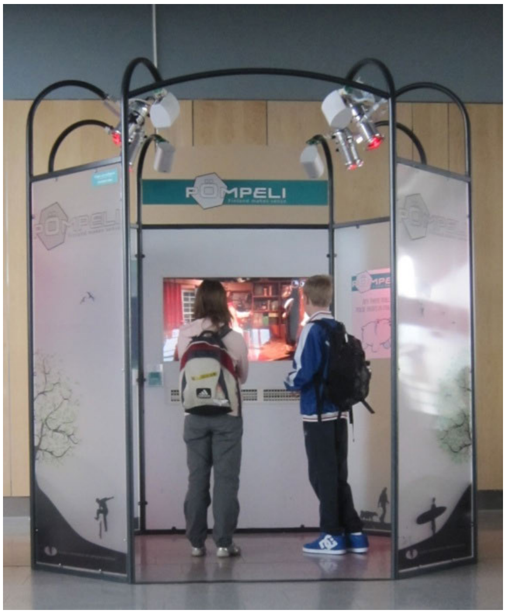


Figure 21. Pömpeli multi-sensory space with multi-touch video, audio, smell, wind and lights at Helsinki Airport.

These kinds of multi-sensory augmentations are more attractive for virtual environments than for mobile augmented reality, for example, where it is challenging to control the environment.

Building up a gustatory display is challenging because the perception of gustatory sensation is affected by other factors, such as vision, olfaction, thermal sensation and memories. However, people have also tested augmented flavours. Users can be tricked into tasting a non-existent flavour using visual and olfactory clues. An example of this kind of augmented flavour system is Meta Cookie [62], where users are given neutral tasting sugar cookies with marker decoration.

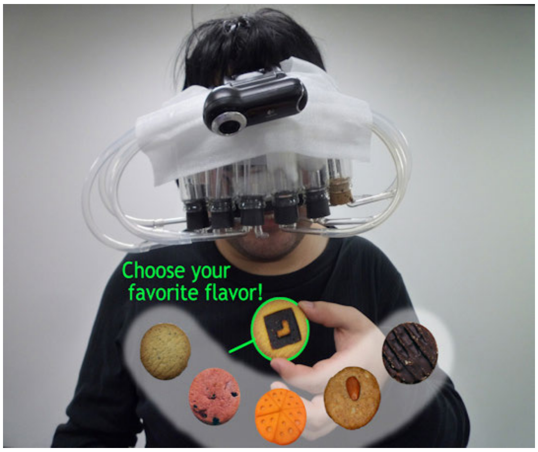


Figure 22. Meta Cookie setup: instead of neutral tasting sugar cookies, users see augmented cookies and are given corresponding smells image form [62].

The cookie is detected and virtually replaced with the flavour of cookie chosen by the user, e.g. chocolate, and the corresponding scent is emitted. The Meta Cookie system air pumps have seven kinds of scented air, which can be controlled in 127 increments. In addition, the system has the ability to emit fresh air.

Augmented flavours are still a future technology. Before smell becomes a feature in a larger scale in augmented reality, the user interface must be improved: the wearable scent producing system must be miniaturised for mobile applications (see Figure 22).

2.6 Toolkits and libraries

Researchers and developers have created a great number of *augmented reality tools* (software libraries, toolkits, SDKs, etc.) that are used for AR application development. They usually contain the methods for core augmented reality functionalities: *tracking, graphic adaptation and interaction*.

In the context of augmented reality, *authoring* means defining the content for an AR application and creating the rules for augmentation (e.g. animation paths), and an *authoring tool* is the implement for doing so. Some AR tools have components of both core AR functionalities and authoring, such as Artisan [63], which is a front end and management system for FLARToolkit and Papervision3D.

AR tools often use third party libraries for lower level tasks (external tools) and wrap them into the level needed for AR. They use OpenCV for computer vision and image processing, for example, and Eigen or LAPACK for linear algebra. In addition, they may provide an interface for existing tools for image acquisition (e.g. Highgui) and camera calibration (e.g. OpenCV), or provide their own utilities for these tasks. An AR application developer may naturally use any other software for image acquisition and calibration as well. Respectively, AR applications normally use existing graphics libraries and 3D engines for graphics and rendering (e.g. OpenGL, Open Scene Graph, OGRE, Papervision3D, etc.).

The first library for creating augmented reality applications was ARToolKit [64]. Together with its descendants, it is probably the best-known and most commonly used tool for creating augmented reality applications. Today the ARToolKit product family consists of libraries for creating stand-alone applications, web applications and mobile applications for several platforms, e.g. ARToolKitPro (C/C++ marker-based tracking library), FLARToolKit (the Flash version of ARToolKit), ARToolKit for iOS (the iPhone port of ARToolKit Pro) [65].

Augmented reality tools are difficult to compare, as some of them are specialised to one purpose (e.g. marker-based tracking or mobile environments), some support only certain platforms (e.g. Windows or iOS) and others support several platforms and are used for several purposes. For example, VTT's ALVAR [19] is a software library for creating virtual and augmented reality applications with support for several platforms, PC and mobile environments alike. It has both a marker-based and a feature-based tracking functionality. Furthermore, it has some support for diminished reality and rendering. The SMMT library (SLAM Multimarker Tracker for Symbian) [66] is an example of a very specialised AR tool. As its name suggests, it is suitable for multi-marker AR application development on Symbian and it uses the SLAM approach for tracking. On the other hand, some tools are more core AR tools such as the abovementioned ALVAR and SMMT libraries, and others are more authoring tools such as DART (The Designer's Augmented Reality Toolkit) [67].

We may classify AR tools based on the environments they use (mobile, PC, VR, etc.), the platforms they support (Windows, Linux, Symbian, iOS, Android, etc.), the language they use (C++, Java, etc.), the approach they use for tracking (marker, multi-marker, features), the algorithms they use for tracking (SLAM, PTAM etc.), or the functionalities they have (diminishing, interaction, etc.) Alternatively, we could have a more commercial viewpoint and compare the licensing and pricing issues as well.

In practice, people are often more interested in the performance of the applications created with the tools rather than the approach they use. However, the performance comparison is difficult due to the large diversity of abovementioned platforms, levels and functionalities, and because there is no standard for AR, not to mention a standard for benchmarking AR tools.

We can summarise that there is a large variety of tools available for AR application development and the best tool depends mostly on the application, which defines the environment, platform, functionalities needed, etc. Yet, developers have other aspects as well, e.g. how familiar they are with the tools, how easy they are to use and what third party libraries they require, etc.

2.7 Summation

The diversity of AR platforms, devices, tools and applications is stunning. Overall, augmented reality is a pronounced visualisation method, which is used in many application areas. It is especially advantageous in on-site real-time visualisations of database information and for purposes where there is a need to enhance the 3D perceptive skills of the user. Augmented reality enables natural interactions and is a good tool to create interactive games and enhance user experience in other areas as well. In this work, we aim to give a thorough overview of the whole field, whilst concentrating on the fundamental issues of single-camera visual augmented reality.

3. Marker-based tracking

Augmented reality presents information in a correct real world context. In order to do this, the system needs to know where the user is and what the user is looking at. Normally, the user explores the environment through a display that portrays the image of the camera together with augmented information. Thus in practice, the system needs to determine the location and orientation of the camera. With a calibrated camera, the system is then able to render virtual objects in the correct place.

The term *tracking* means calculating the relative *pose* (location and orientation) of a camera in real time. It is one of the fundamental components of augmented reality.

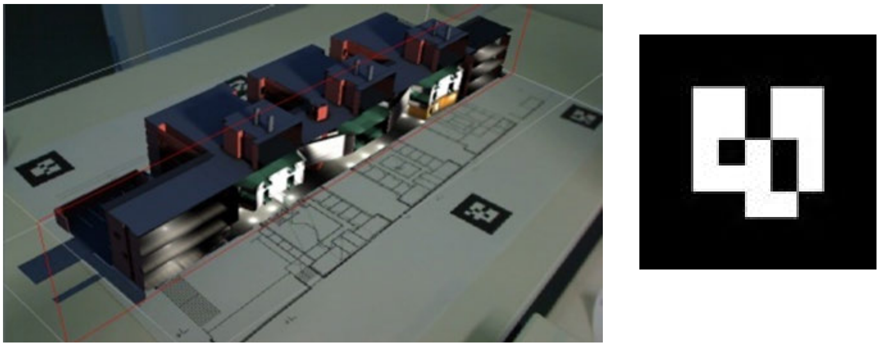


Figure 23. Left image: VTT's AR ScaleModel application augments a virtual model of a building on top of a floor plan in the correct scale and pose using marker detection. Right image: an example of a marker (ALVAR marker number 14). (Image: VTT Augmented Reality team).

Researchers in computer vision, robotics and photogrammetry have developed a considerable number of different tracking methods. People can divide these methods based on the equipment used in sensor tracking methods, visual tracking methods and hybrid methods. Since in most augmented reality setups the camera is already part of the system, visual tracking methods are of special interest in AR.

We will concentrate on them first and discuss sensor and hybrid tracking methods later in Chapter 5.

In visual tracking, the system deduces the pose of the camera based on observations of what it sees. In an unknown environment, this is challenging; it takes some time to collect enough data to be able to deduce the pose and then the calculated pose estimation easily drifts over time. As the environment is unknown to the system, the system selects the orientation of the coordinate axis at random, which may be inconvenient for the user. In addition, it is impossible to deduce the correct scale solely based on visual observations.
One solution to overcome these challenges is to add an easily detectable predefined sign in the environment and use computer vision techniques to detect it. A *marker* is such a sign or image that a computer system can detect from a video image using image processing, pattern recognition and computer vision techniques (e.g. right image in Figure 23). Once detected, it then defines both the correct scale and pose of the camera. This approach is called *marker-based tracking*, and it is widely used in AR.

Other approaches for visual tracking are *feature-based* and *model-based* methods, which we will discuss later in Chapter 5. In model-based tracking, the system has a model of the scene or part of the scene (e.g. a CAD model). It compares visual observations with the model and finds the best match, which then defines the pose. In feature-based tracking, the system detects optical features in the images and learns the environment based on observations of movements between frames.

Even though mainstream visual tracking research leans towards feature-based tracking, feature tracking and marker-based tracking are mutually non-exclusive. In fact, marker-based methods often outperform feature-based methods in certain occasions (as we will rationalise later in Section 4.4.1.), and marker-based systems are still widely used for visual tracking in augmented and mixed reality (e.g. [68–70].

The popularity of marker-based systems is also partly explained by the fact that they are easy to implement and that good and well-known marker-based toolkits are available (e.g. ARToolKit [64], ALVAR [19], ARTag [71]). Toolkits provide a good base for starting AR application development. Apart from that, markers provide the correct scale and convenient coordinate frames (Figure 23) as previously mentioned. In addition, they may encode information or at least have an identity. This enables a system to attach certain objects or interactions to the markers.

In marker-based tracking, the system needs to detect the marker, identify it and then calculate the pose. In this chapter, we focus on marker detection and pose estimation. We talk more about different marker types and identifying and decoding markers in Chapter 4.

We go through the pipeline of the marker detection procedure, considering commonly used black and white square markers. Section 3.1 gives an overview of the detection process and most of the steps and methods presented here serve for detecting other types of markers and other tracking methods as well.

In Section 3.2, we explain how camera pose is estimated using the four corner points of a marker. At the end of this chapter in Section 3.3, we discuss multi-marker setups that improve the tracking system.

3.1 Marker detection

A good marker is easily and reliably detectable under all circumstances. Differences in luminance (brightness) are more easily detected than differences in chrominance (colour) using machine vision techniques. [72] This is due to the poor automatic white balance of the cameras: the colours register incorrectly, and an object may change its colour in the image depending on what else is in the view, for example. Furthermore, the lighting changes the perceived colours of the objects and therefore colour detection is challenging. Naturally, the more contrast in the luminance the more easily objects are detected. In this sense, black and white markers are optimal.

The system should also be able to calculate the pose of the camera using the detected marker. Four known points are sufficient to calculate the pose of a camera uniquely [72] and the simplest shape to acquire them is a square. In addition, the locations of the corner points are relatively robust, as they can be estimated as intersections of edge lines.

Therefore, many of the marker systems use black and white square markers. This is also why we first go through the marker detection process considering such markers. We cover other marker types later in the Chapter 4.

3.1.1 Marker detection procedure

The first goal of a marker detection process is to find the outlines of potential markers, and then to deduce locations of marker's corners in the image. In addition, detection system needs to confirm that it really is a marker and decipher its identity. Finally, the system calculates the pose using the information from the detected marker location.

The basic marker detection procedure consists of the following steps:

0. Image acquisition
    - acquisition of an intensity image.
1. Preprocessing
    - low level image processing
    - undistortion
    - line detection/line fitting
    - detection of the corners of the marker.
2. Detection of potential markers and discard of obvious non-markers
    - fast rejection of obvious non-markers
    - fast acceptance test for potential markers.

3. Identification and decoding of markers
    - template matching (template markers)
    - decoding (data markers).
4. Calculation of the marker pose
    - estimation of marker pose
    - iterative pose calculation for accurate pose.

The image acquisition step is actually a separate process; it just provides the image for the marker detection process. Marker detection pipelines may differ from this template. The execution order of the steps may differ or the system may merge steps into the same algorithm. In particular, many implementations combine acceptance/rejection tests with other tasks; the system may reject a marker candidate at any stage of the detection process when it notices that the candidate cannot be a marker. However, the main concept is usually the same.

In the following, we discuss each step of the marker detection procedure in more detail, except for identifying and decoding markers, which depends on the marker type, and which we will cover in Chapter 4.

3.1.2 Pre-processing

Before the actual detection of the marker, the system needs to obtain an intensity image (a greyscale image). If the captured image format is something else, the system converts it, e.g. an RGB image is converted into an intensity image using a well-known technique (see [73], for example)[73]. From now on, we will assume that the marker detection system is operating with a greyscale image.

The first task of the marker detection process is to find the boundaries of the potential markers. Detection systems use two approaches: either they first threshold an image and search for markers from the binary image, or they detect edges from a greyscale image. These lower level image-processing tasks (thresholding, edge detection, line fitting, etc.) are well known and are therefore not discussed in detail here. For further information, consult [73, 74], for example.

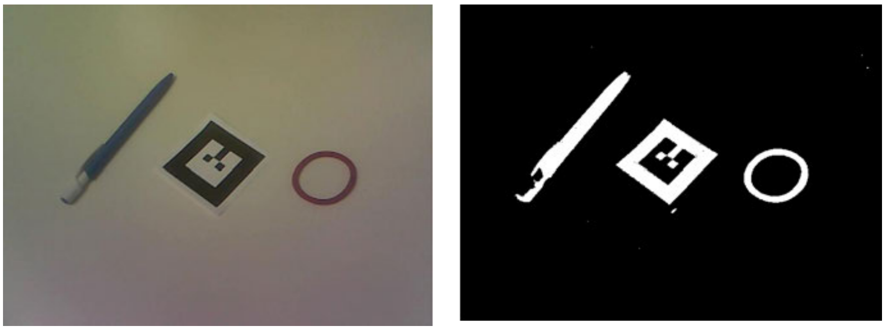


Figure 24. On the left, the original image; on the right, the image after adaptive thresholding (example produced with ALVAR).

Marker detection systems using the threshold approach normally use an adaptive thresholding method (e.g. [75]) to cope with local illumination changes (see Figure 24). After thresholding, the system has a binary image consisting of a background and objects. All objects are potential marker candidates at this stage. Habitually the next step is to label all of them or otherwise keep track of the objects. Suitable labelling algorithms are presented for example in [73]. During the labelling process, the system may reject objects that are too small or otherwise are clearly something other than markers. We discuss these fast acceptance/rejection tests more in Section 3.1.3. Finally, the edges of all potential markers are marked (Figure 25) and their locations are undistorted for line fitting (Figure 26). After line fitting, the system tests the potential markers again, checking whether they have exactly four straight lines and four corners. Finally, the system optimises the corner locations in sub-pixel accuracy, these are used in further calculations. Figure 27 shows the marker coordinate system and an augmentation on top of the marker.

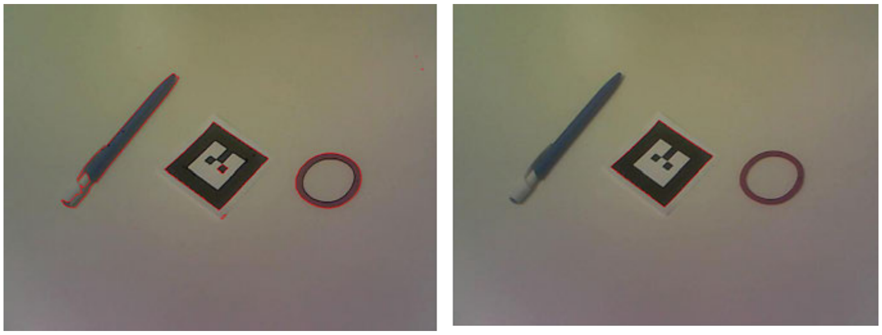


Figure 25. An example of edge detection using ALVAR. On the left: the edge contours detected on the threshold image (Figure 24) are superimposed onto the original image with red. On the right: the remaining edges after applying the four-corner test and size test.

Edge detection in greyscale images is time consuming, therefore marker detection systems using this approach generally use sub-sampling and detect edges only on a predefined grid [76]. As this approach results in separate line points, the marker detection system needs to link edge pixels into segments. Systems usually group these segments into longer lines using edge sorting. As the system samples the original points using a coarse grid, it needs to extend the lines to full length to find the exact corners of the marker. A common procedure is to use the gradient information of the original image to extend the edges to full length.

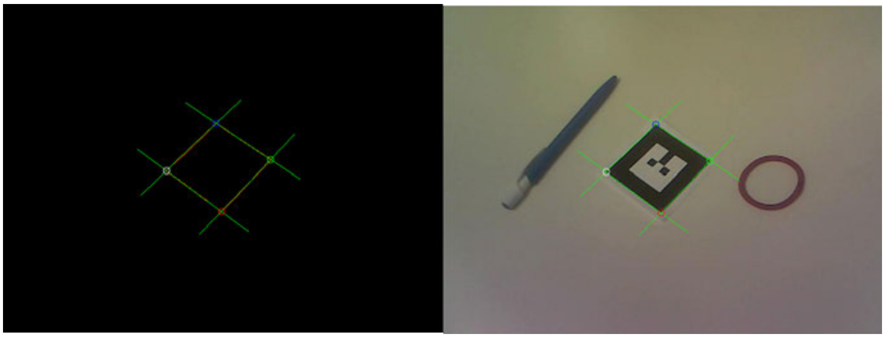


Figure 26. An example of line fitting using ALVAR. On the left, line fitting in undistorted coordinates; deduced corner point locations are marked with circles. On the right, detected lines over the original image.

Applications use several methods for line detection, line fitting and line sorting. In general, methods based on edge sorting (such as in ARTag) and the method presented in [76]) are robust against partial occlusion, but are computationally more expensive, which make them unsuitable for current mobile devices.

Traditionally in photogrammetry, the whole image is undistorted (commonly before preprocessing) using the inverse distortion function calculated during the camera calibration process. This is a suitable approach in off-line computer vision applications. In real-time augmented reality applications, systems typically undistort only the locations of feature points (e.g. the detected edges of a marker) to speed up the system. We discuss camera distortions a bit more in Section 3.2 and explain them in detail in the Appendix B.

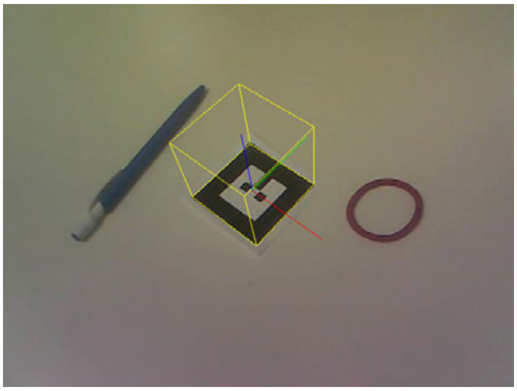


Figure 27. A cube augmented on top of a detected marker. The marker coordinate system (X,Y,Z) is rendered with (red, green, blue). Example produced using ALVAR.

Even small errors in detected 2D locations of edges and corners significantly affect the calculated pose of the camera [77–79]. Detection errors can be caused by a pixel quantisation error, incorrect threshold value, motion blur, noise, etc. These errors cause annoying jitter in an object's pose even if the camera hardly moves. In order to increase accuracy, detection systems optimise the locations after initial detection.

For example, if the system detects a marker from a threshold image, it may use the greyscale image to find edges or corners with higher accuracy. The system may also use the detected corners as an initial estimate for a more accurate corner detection method than initially used. Sometimes systems estimate motion blur from detected marker edges and decompensate motion blur for higher accuracy [80, 81]. Pixel quantisation errors are especially obtrusive if a marker edge coincides with the pixel coordinate axis. The whole edge may jump from one pixel line to another. In diagonal edges, errors occur in several directions and edge fitting stabilises these.

The number of edges and corners in the original image may be huge, and if all of them were analysed initially with high accuracy, the system would waste a lot of capacity in processing non-marker information. Therefore, these kinds of two-step approaches are common in real-time AR applications.

3.1.3 Fast acceptance/rejection tests for potential markers

As a rule, augmented reality applications aim for real-time processing and fast performance is essential. Systems cannot afford to waste time in processing non-markers. Therefore, many implementations use fast calculable acceptance/rejection criteria to distinguish real markers from objects that clearly are something else. In the following, we go through a number of these acceptance/rejection tests that are commonly used and/or that we have successfully used ourselves.

A system can reject areas consisting only of a few pixels. They are often something other than markers, and even if they were markers, their small size would mean that the marker is very far away from the camera. In this case, the pose of the marker would be very uncertain and therefore useless. Furthermore, if the marker shrinks to the size of a few pixels, the system cannot identify the marker, unless it keeps track of the history of each marker appearance, (we will discuss this in Section 3.2.5). The system needs to pay attention to areas inside of a marker and make sure that it does not remove parts or cells belonging to a marker.

The histogram of a black and white marker is bipolar, and the marker detection system may check the bipolarity as a fast acceptance/rejection criterion. However, the test must take into account reflections, which may create grey scale values.

Calculating the number of pixels belonging to the perimeter is a very fast process, and a system can carry it out as part of the labelling process, whereas calculating the exact size of the object is more complicated and time consuming. Therefore, it is rational to estimate the size of an object using the number of edge pixels, for example. Another quickly calculable criterion used to estimate the area is the biggest diagonal or span. Sometimes a system may also reject areas that are too large if it has some background knowledge to justify the assumption that they are something other than markers, e.g. dark edges due to vignetting (see next page). The optimal limiting value for the marker size depends on the application and the marker type.

For example, ARToolKit assumes that markers are within a reasonable distance of the camera. ARToolKit marker detection omits areas that are too small or too big after labelling. In other words, it keeps those areas where the number of pixels belonging to it is within the boundaries for further analysis [82].

Depending on the marker type, a system may know something about the overall appearance of the marker. For example, simple template markers have a black image on a white area surrounded by a black edge (see for example the left-hand image in Figure 28). In an ideal case, markers have a known (small) number of holes (white areas) inside the (black) marker area. Furthermore, completely black areas are not markers. Calculating the number of holes is fast, and thus a system may apply the number of holes as a rejection criterion during pre-processing. For example, the marker on the left in Figure 28 has one white hole inside the black border and the marker in the middle has three holes.

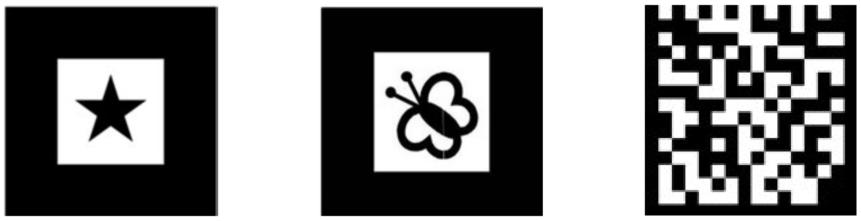


Figure 28. The marker on the left has exactly one hole (a white area) and the marker in the middle has three holes. The marker on right is an example of a binary marker with numerous strong edges. A system can use this kind of knowledge for fast acceptance/rejection tests.

2D barcode type binary markers (e.g. the marker on the right in Figure 28) have a number of strong edges inside the marker (edges between white and black cells). One heuristic method of rejecting obvious non-markers is to calculate the number of intensity changes in two perpendicular directions. If the number of changes is low, it cannot be a marker. We used this for example in our marker detection system [11] to avoid processing non-markers and to achieve real-time processing in a mobile application. The right-hand image in Figure 28 is an example of a marker used in our abovementioned work. This criterion is applicable only to marker types where marker encoding guaranties high variability and almost white or almost black markers do not exist.

Some cameras produce images that are darker towards image outer edges due to imperfect optics. This effect is called *vignetting*. The system may threshold these areas confusingly. In abovementioned work, we used the vignetting assumption to reject areas crossing image borders (see Figure 29).

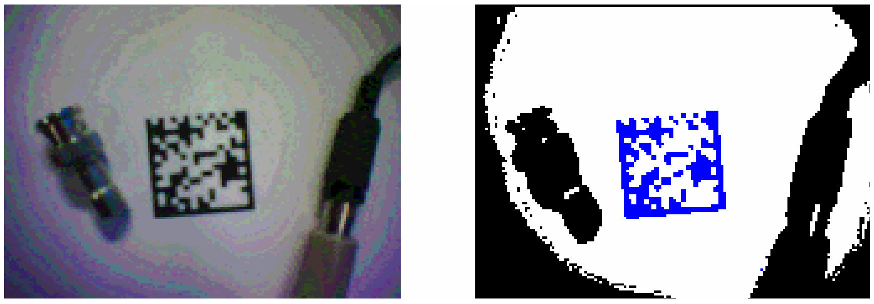


Figure 29. The left image shows how the image gets darker radial from the centre due to the camera's poor characteristics (vignetting). The right image shows how this affects the threshold. The detected marker is blue and the rest of the threshold areas are black. (Image: VTT Augmented Reality team).

A perspective image of a square is always quadrilateral. Therefore, it is convenient to apply a *quadrilateral test* for perspective images as the next criterion for square markers. A quadrilateral has exact four straight lines and four corners. The number of lines and corners are easy to calculate; therefore, marker detection implementations often use one of them as a fast approval/rejection criterion. For example, ARToolKit does line fitting for edge pixels to check the existence of exactly four lines on the perimeter [83]. Mobile or computationally inexpensive algorithms often use some heuristics, e.g. finding four corners using a simple algorithm of maximum distances (maximum diagonal) such as in [84] and [85]. In addition, the system described in [85] uses those corners as initial estimates for Harris corner detection. As a result, the system only needs to undistort the four detected corner points.

In addition, the system may apply other fast criteria. The suitable criterion naturally depends on marker type as well, e.g. for circular markers, a system would apply some sort of circularity test.

In real-time processing, the system must focus on processing relevant information thus these kinds of fast tests are essential for them. Systems typically carry out such fast tests during the whole marker detection and identification process.

3.2 Marker pose

The pose of an object refers to its location and orientation. The location can be expressed with three translation coordinates (x, y, z) and orientation as three rotation angles (α,β,γ) around the three coordinate axes (Figure 30). Thus, a pose has six degrees of freedom (6 DOF).

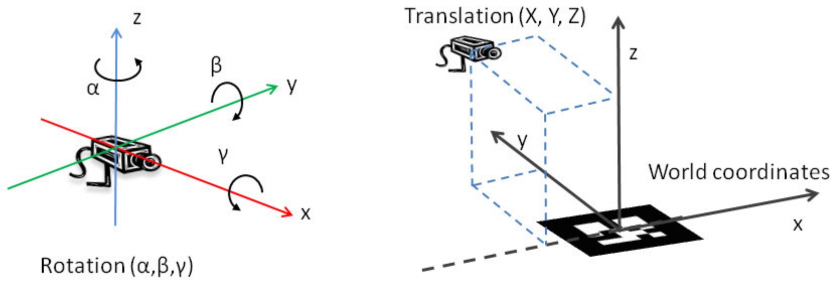


Figure 30. The camera's pose means its location and orientation in world coordinates. All orientations can be expressed with rotation angles (α,β,γ) around coordinate axes (on the left). Position (X,Y,Z) is defined by translations along each coordinate axis (on the right).

The pose of a calibrated camera can be uniquely determined from a minimum of four coplanar but non-collinear points [72]. Thus, a system can calculate a marker's pose (relative to camera) in 3D coordinates using the four corner points of the marker in image coordinates. In the following, we will go through this pose calculation procedure. First, we briefly revise projective geometry and camera calibration insomuch as is necessary to understand pose estimation. Readers may consult Appendices for further details on these subjects. At the end of this section, we explain how to increase the stability and robustness of a marker tracking system with continuous (frame-to-frame) marker tracking.

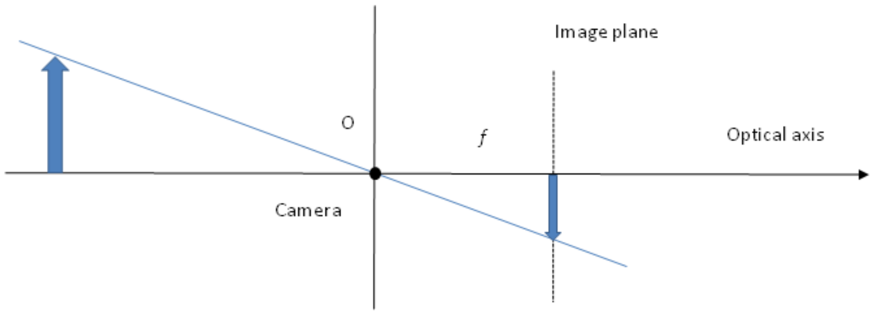


Figure 31. The geometry of a pinhole camera.

In an ideal pinhole camera model, all rays pass the infinitely small optical centre of a camera, and the object's image registers on an image plane. We call it an *ideal image*.

In digital cameras, the image registers on the image sensor and coordinates of its elements differ from ideal coordinates (Figure 33). The *camera image* depends on the camera's physical characteristics, e.g. focal length, image sensor orientation and size.

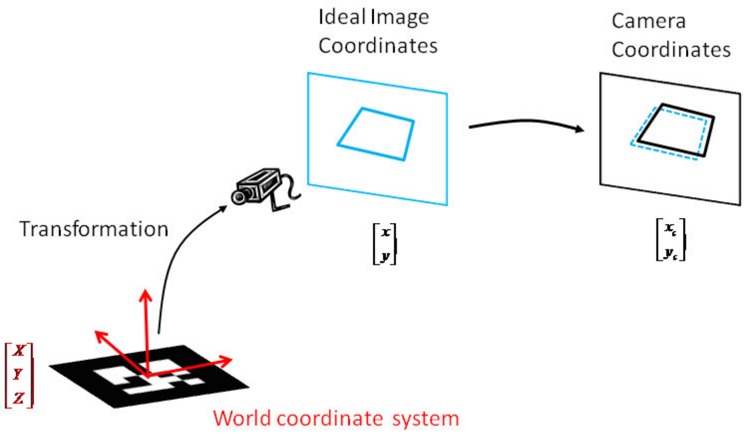

Ideal Image Coordinates
Camera Coordinates
Transformation
$\begin{bmatrix} x \\ y \end{bmatrix}$
$\begin{bmatrix} x_c \\ y_c \end{bmatrix}$
$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$
World coordinate system

Figure 32. The ideal pinhole camera model does not hold for a digital camera. Therefore, an additional conversion to the camera coordinate system is needed.

3.2.1 Camera transformation

The transformation $\mathbf{T}$ between a camera and a marker is

$$\mathbf{x} = \mathbf{TX},$$

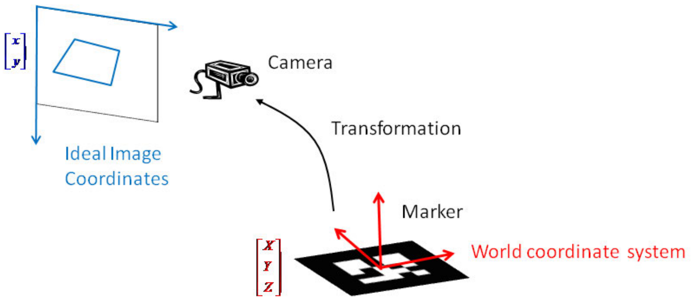
where $\mathbf{X}$ is a point in world coordinates, $\mathbf{x}$ is its projection in ideal image coordinates and $\mathbf{T}$ is the *camera transformation matrix* aka the *extrinsic camera matrix* or the *pose matrix* (see Figure 33). Transformation $\mathbf{T}$ consists of translation vector $\mathbf{t}$ and 3 x 3 rotation matrix $\mathbf{R}$, and can be expressed in matrix form

$$\mathbf{x} = \left[\mathbf{R} \mid \mathbf{t}\right]\mathbf{X},$$

which in homogeneous coordinates is

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

A rotation matrix has only three free parameters (α, β, γ) that define its nine elements (see Appendices). A translation vector has also three parameters, thus a pose matrix has six free parameters. A marker tracking system needs to solve this camera matrix for each frame when it detects a marker.



$\begin{bmatrix} x \\ y \end{bmatrix}$
Camera
Transformation
Ideal Image Coordinates
Marker
World coordinate system
$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$

Figure 33. Transformation matrix converts world coordinates to ideal camera coordinates.

3.2.2 Camera calibration matrix and optical distortions

The physical characteristics of a camera define how an image ultimately forms on the image sensor of the camera. In the pinhole camera, only the focal length of the camera has an effect on the image formation. The image of a pinhole camera forms in plane $z = f$, where f is the focal length of the camera. In actual cameras, the image sensor may be skewed, the pixel aspect ratio might be uneven and the centre of the image sensor element (p_x, p_y) may have an offset from the optical axis of the lens system.

A conventional way is to present camera model as an (*intrinsic) camera calibration matrix* **K,** or simply *camera matrix*. It is a mapping between ideal image and camera sensor coordinates (observed pixel coordinates), $\mathbf{x}_{pix} = \mathbf{K}\mathbf{x}_c$.

$$\begin{pmatrix} x_{pix} \\ y_{pix} \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & s & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix},$$

In addition, an actual camera may produce systematic geometrical errors, *distortions*, due to lens imperfections. Distortion occurs in the camera lens. Thus in theory, distortion needs to be taken into account before ideal coordinates can be converted into camera coordinates. However, in most present-day cameras, pixels are square and columns and rows are straight. This means that in the camera matrix $s = 0$, and $f_x = f_y$,

$$\mathbf{K} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

In this case, the distortion can be modelled *after* the camera's intrinsic transformation (see Figure 34). In this case, the camera matrix converts the ideal image coordinates into camera coordinates and the distortion function converts the camera coordinates into pixel coordinates,

$$\mathrm{D}\left(\begin{bmatrix} x_c \\ y_c \end{bmatrix}\right) = \begin{bmatrix} x_{pix} \\ y_{pix} \end{bmatrix}.$$

$\mathrm{D}^{-1}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} x_c \\ y_c \end{bmatrix}$. The inverse distortion function, *undistortion function* D^{-1}, undistorts the pixel coordinates into camera coordinates

Camera calibration is the identification of the camera's intrinsic parameters, that is the camera matrix, and the estimation of the undistortion function.

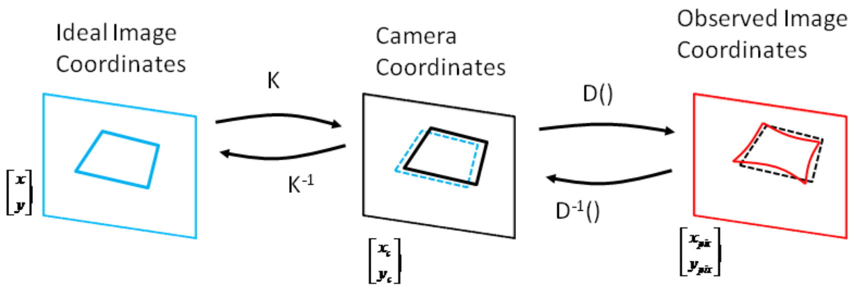

Ideal Image Coordinates
K
K-1
Camera Coordinates
D()
D-1()
Observed Image Coordinates
$\begin{bmatrix} x \\ y \end{bmatrix}$
$\begin{bmatrix} x_c \\ y_c \end{bmatrix}$
$\begin{bmatrix} x_{pix} \\ y_{pix} \end{bmatrix}$

Figure 34. Image registration on a real camera.

AR applications may use a separate calibration tool or otherwise carry out the calibration process separately from the actual AR application. Normally the calibration tool saves results in a file, which an AR application then reads. From now on we will assume that the undistortion function D^{-1} and camera calibration matrix K are known.

Altogether, a marker detection system can transform the observed image coordinates (pixel coordinates) of a calibrated camera into ideal screen coordinates. Pixel coordinates are first transformed into camera coordinates, then undistorted:

$$\mathbf{K}^{-1}\mathrm{D}^{-1}\left(\begin{bmatrix} x_{pix} \\ y_{pix} \end{bmatrix}\right) = \mathbf{K}^{-1}\left(\begin{bmatrix} x_c \\ y_c \end{bmatrix}\right) = \begin{bmatrix} x \\ y \end{bmatrix}.$$

On the other hand, ideal coordinates can be transformed to world coordinates and vice versa. Finally, we can represent the relation between world coordinates X and observed image coordinates x_{pix},

$$\begin{bmatrix} x_{pix} \\ y_{pix} \end{bmatrix} = \mathrm{D}\left(\mathbf{KT}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}\right).$$

3.2.3 Pose calculation

In the following, we assume that the distortion can be separated from the camera model. It is a safe assumption for most modern cameras. Points in *undistorted camera coordinates* are labelled with $\mathbf{x}$ and the corresponding points in *world coordinates* are labelled with $\mathbf{X}$.

The marker detection gives four corner points in image coordinates $\mathbf{x}_1, \ldots, \mathbf{x}_4$. For each corner point $\mathbf{x}_i$, $i = 1,2,3,4$, the following is true

$$\mathbf{x}_i = \mathbf{KTX}_i.$$

Written out, this is expressed as

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}.$$

If we multiply $\mathbf{KT} = \mathbf{M}$

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \\ m_5 & m_6 & m_7 & m_8 \\ m_9 & m_{10} & m_{11} & m_{12} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

The real location of each corner point in world coordinates is known. The system now has eight equations (one for each of the two coordinates of each of the four corners) and six free parameters. Thus, it is possible to solve (estimate) the transformation matrix.

A commonly used approach is to use some non-iterative methods to calculate an initial guess for the pose, e.g. *direct linear transformation* (DLT), and then use an iterative optimisation method to calculate the exact pose.

We reproject world $\mathbf{X}$ onto the image plane using the estimated transformation matrix $\widehat{\mathbf{M}}$, the reprojected point $\hat{\mathbf{x}}$. is

$$\hat{\mathbf{x}} = \widehat{\mathbf{M}}\mathbf{X}.$$

We can solve the transformation matrix by finding a matrix that minimizes the reprojection error $\left\| \mathbf{x} - \hat{\mathbf{x}} \right\|$, that is

$$err = \frac{1}{4} \sum_{i=1,2,3,4} \left\{ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right\}.$$

This is a non-linear estimation problem and the system can solve it analogously to camera calibration using an *iterative optimization method* such as *Levenberg-Marquard* (cf. Appendix C).

For a marker-based system, the world coordinates (i.e. the corners of the marker) are known. In general, we also need to estimate the world coordinates $\widehat{\mathbf{X}}$.

Optimisation of the reprojection error is fast and is used in ARToolKit, for example. In continuous tracking mode, ARToolKit also combines this with the tracking results of the previous frame [83]. A system can define the reprojection error in image space as above, for example, or in object space as in [86].

Reprojection errors usually have two local minima, and optimisation methods do not guarantee convergence to a global minimum. A robust planar pose (RPP) estimator [87] takes into account the pose ambiguities, i.e. the two local minima of the error function, and locates the optimal solution (the correct pose). It is computationally more expensive, but is able to avoid the local minimum. The RPP estimator first locates the local minimum of the reprojection error in object space using the algorithm of [86]. It transforms the coordinate system to parameterised space (of rotations around coordinate axes) and estimates the locations of all local minima. Then it uses all poses corresponding to these minima as the initial values for the same iterative optimisation algorithm that it used to find the first local minima. Finally, the global minimum is the correct pose.

The advantage of iterative methods is that they get an accurate pose estimate, yet the convergence depends on the initial guess and they are computationally complex and therefore unsuitable for low-capacity mobile devices. To overcome these limitations, researchers have proposed a non-iterative table based on the camera pose estimation method for square-shaped markers [78].

3.2.4 Detection errors in pose calculation

The detection of x and y translations is more reliable than the detection of z translation. The camera geometry explains this. If an object moves a certain distance in the z direction, the corresponding movement on the image plane is much smaller that if it moves the same distance in the x or y direction (Figure 35). Vice versa: small detection errors on the image plane have a greater effect on the z component of the translation vector than on x and y components. In addition, the pose of a marker seen from the front is more uncertain than a pose of a marker seen from an oblique angle.

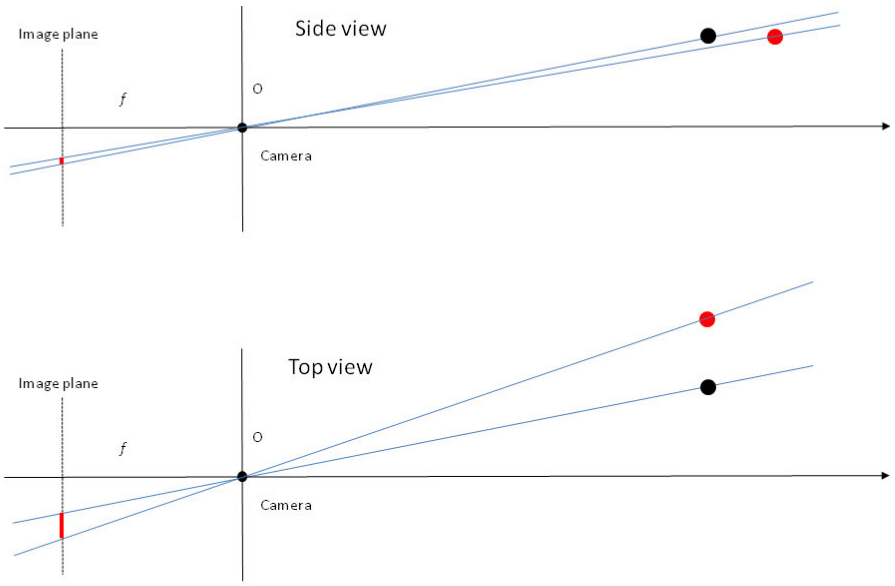


Figure 35. If an object (black dot) moves parallel to the optical axis (on the top), the movement in the image plane is small compared to the situation where it moves perpendicular to the optical axis (on the bottom). Therefore, small detection errors on the image plane have a great effect on the Z dimension.

3.2.5 Continuous tracking and tracking stability

Some implementations detect markers separately frame by frame. Applications can boost performance by keeping history information on marker appearance and tracking the markers continuously. Based on the information from the previous frames, the system can identify markers that otherwise would be too small to be identified, for example, or that are partially occluded. In other words, after decoding the marker data once, it is sufficient to detect the marker in the next frames without decoding its content again.

In addition, if an application keeps continuous track of the marker pose, it can filter the pose over time and thus detect outliers (flaws in marker detection) and handle inaccuracy in marker detection. Furthermore, it can use the previous pose as an initial guess for the iterative pose calculation method. With a high frame-rate, the camera normally moves only slightly between frames and this is a good initial guess. Should the frame rate be slow, the pose may change significantly between frames, and the system should calculate a new initial guess. The system may also predict camera movement based on several previous poses and use the predicted pose as an initial guess.

If marker detection fails for some reason and markers are detected frame by frame separately, the augmentation often "jumps" from the correct position to a random position, which is annoying for the user. A continuous tracking system is able to avoid this situation. Furthermore, in continuous marker tracking, markers are less likely to be confused with others as the system assumes that the marker is near its previous location. In addition, when the marker identity is difficult or impossible to detect because it is too small, for example, a continuous marker system is able to conjecture it based on the historical information.

Marker edges may be shifted due to a non-optimal threshold value or blurred image, for example. The corner positions are then respectively shifted, which then implies that the marker pose is inexact. This imprecision appears as a small oscillation in the augmentation. A continuous marker tracking system is able to reduce this oscillation with proper filtering. As always with filtering, filtering parameters must be chosen with care to find an optimal solution; if the pose filtering averages too strongly, the system is unable to react to fast movements; if it averages too little, the oscillation remains.

The amount of the data that a marker can encode depends on the amount of cells it contains. Typically, the more cells a marker has – the smaller is the size of the physical cells, if we keep the marker size fixed. However, bigger physical cell size makes it easier for the application to be able to detect and decode the marker from a farther distance away. Thus, having a large amount of data to encode (i.e. needing a large number of cells per marker) limits the maximum detection distance.

A continuous marker tracking system could overcome this limit using super-resolution images. Super-resolution images are high-resolution images integrated over time [88, 89]. The contents of a marker could be defined using a super-resolution image. This way a system could have a higher number of cells in a physically smaller marker and still be able to decode its content. An application using a continuous marker tracking system would need to create a super resolution image of each marker only once, and thereafter the system could concentrate on following each marker without decoding them again.

The stability of an augmented reality tracking system can be improved with several methods. For example, the Kalman filter (KF), extended Kalman filter (EKF) [90] and single constraint at a time (SCAAT) method [91] are used to predict marker or camera movements and to stabilise tracking results. For instance [92] uses a SCAAT algorithm to compute the estimated pose using infrared beacons in addition to three gate gyros and GPS sensor, and [93] uses SCAAT-Kalman filtering for real-time tracking with unsynchronised cameras.

3.2.6 Rendering with the pose

The main idea of augmented reality is to present virtual objects in a real environment as if they were part of it. The camera pose is used to render the virtual object in the right scale and perspective. The virtual camera of computer graphics is moved to same pose as the real camera and virtual objects are rendered on top of the real image (Figure 36).

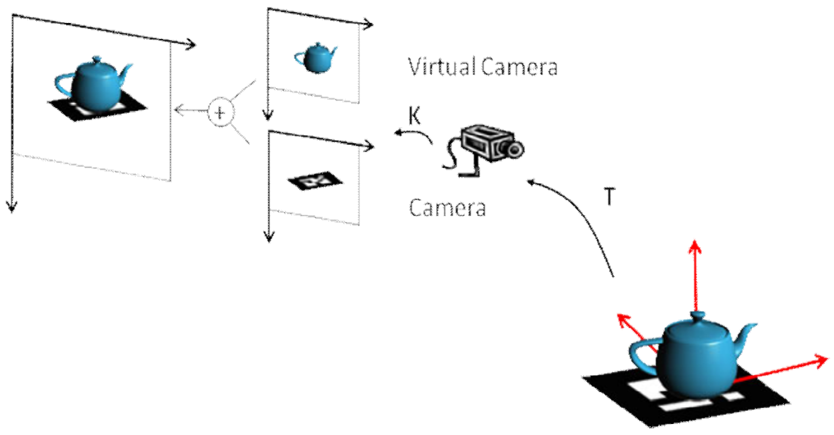


Figure 36. Augmentation in origin.

If a virtual object is rendered using camera transformation matrix T and camera matrix K, it appears on the origin, in the same orientation as the coordinate axes. If a system wants to augment an object in different pose, it needs to add object transformation $\mathbf{T}_{\text{object}}$ in the rendering pipeline (Figure 37).

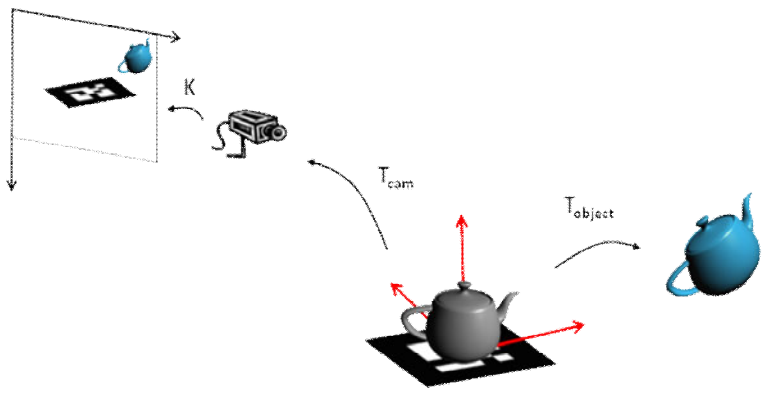


Figure 37. Augmenting in a general location and orientation.

If both the camera and marker are moving, the tracking system is able to derive the relative pose of the camera and the marker, but the absolute position (relative to the earth coordinates) is unknown. Sometimes it is convenient to use an additional device to orient the object, e.g. upright. The system can do this using an accelerometer, which is a built-in sensor in most new smartphones (e.g. iPhone 4G)). The accelerometer provides a gravitation vector in the phone's coordinate system. The task is to change it to the world coordinate system (change of coordinate basis) and then rotate the world z-axis parallel to the opposite gravitation vector. The application needs to add this rotation **R** in the rendering pipeline. Similarly, an application could orient augmentations according to coordinate data from a digital compass.

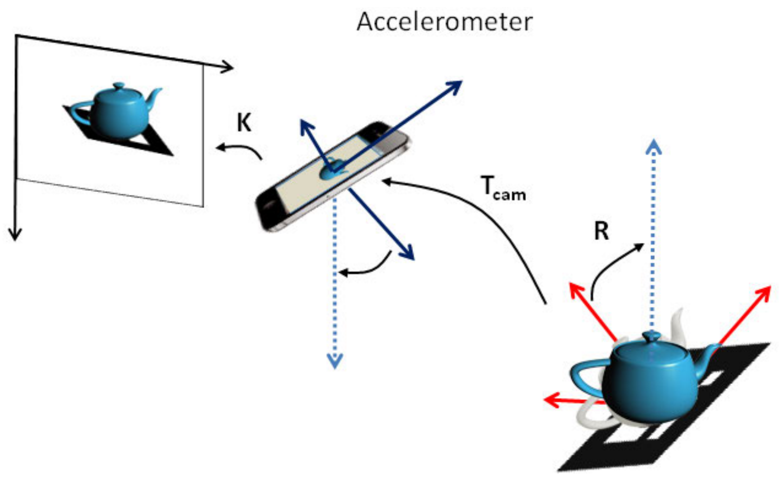


Figure 38. Augmenting upright pose using accelerometer, the gravitation vector and the inverse gravitation vector aremarked with dashed arrows.

3.3 Multi-marker setups (marker fields)

The four corners of a marker determine the pose of a camera as previously explained. However, additional points stabilise a tracking system, improve the accuracy and enable the system to discard outliers. Especially if there is noise, additional reference points improve robustness and accuracy [94, 95]. Three main approaches to increase the number of points used for pose detection are:

- to use more than four points per marker
- to use more than one marker
- to use natural features in addition to the marker.

The improvement in the stability of the first approach is small; the points still distribute on a physically narrow area and therefore they increase the stability of the pose close to nothing. A common problem in tracking is that the field-of-view of cameras is narrow, especial in mobile devices. If the user moves the camera, it soon loses the marker from view. A wider field-of-view does not help either if the user rotates the camera. Therefore, the use of a single marker tracking system restricts the permissible movements of the user, as the camera must see the marker all the time.

This is an unwanted limitation in many applications and therefore, the second and third options are commonly preferred. AR systems habitually use them to increase robustness and usability. In this section, we discuss the second option and describe how an AR system can define and track a multi-marker setup. The third approach will be covered in Section 5.3 Hybrid tracking.

A tracking system can cope with larger camera motion if the user distributes several markers in different directions. When the system detects and tracks each marker individually, the information related to each marker is lost as soon as the system is unable to detect the marker. *Multi-marker systems* (aka *marker fields*) combine the information from all markers, and therefore these systems are more robust and accurate. For example, multi-marker systems can handle partial occlusions and deduce the location of a marker even if it is invisible, as long as they detect some other markers belonging to the marker field.

A multi-marker setup or marker field is a system that uses several markers *jointly* to estimate the camera pose. A system where each marker is *individually* used to calculate its relative pose to the camera is not a multi-marker system even if several markers are used.

In order to deduce the location of a non-detected marker, a tracking system needs to know the relative position of the marker compared to the others. Either the relative location of the markers can be predefined, or the system can allow free distribution of the markers and deduce the configuration of markers as it detects them.

3.3.1 Predefined multi-marker setups

Predefined multi-marker setups are widely used and support for them is a standard feature in marker-based toolkits and libraries. For instance, ARToolKit [82], ARTag [71], ALVAR [19] and StudierStube Tracker [85, 96] offer support for a multi-marker setup. The planar multi-marker setup approach is the same as using a big marker with more than four points. Now the marker field is “a big marker” and markers in it are “sub features”.

A multi-marker system can use a non-planar predefined marker field as well, for example, markers may cover the sides of a cube, some of the markers are on the wall, etc. For instance, ARToolKit, ARTag and ALVAR all support non-planar multi-marker setups as well. Non-planar multi-marker setups provide tracking information for a larger scale environment than a single marker system. Non-planar multi-marker systems cope with larger camera movements than planar systems. Markers attached to 3D objects allow the system to recognise them from different angles, which is desirable with tangible user interfaces, for example.

The problem with non-planar setups is that in practice it is difficult to measure the physical position and orientation of each marker relative to each other. This calibration process is often time consuming and inaccurate if done by hand [8]. It is possible to use external aids for measuring the marker locations, for example a tachometer (as in [97]), but vision-based reconstruction approaches are more interesting from the viewpoint of the augmented reality system since an AR system contains a camera and a computational unit anyway.

In the ideal realisation of a multi-marker system, the user can place markers freely on site without any predefined constraints and then the system creates the marker field based on observations of the marker locations. This is called automatic reconstruction of multi-marker setups.

3.3.2 Automatic reconstruction of multi-marker setups

In automatic reconstruction of multi-marker setups, a system needs to determine the 3D coordinates of markers based on observations (2D images). This is a classical *structure from motion* (SfM) problem with the distinction to a general case that (some or all of) the features used for 3D reconstruction come from markers, not randomly from the environment. It is sufficient to model only the locations of the markers and leave the rest of the scene unmodelled.

Researchers have applied several visual methods successfully to the SfM problem. An AR application designer could apply any of these methods for marker field reconstruction. A good overview of the basic methods can be found in [72]. In the following, we discuss the most common approaches used in AR.

Since the SfM problem is computationally demanding, many of the algorithms work offline. A common approach is to create the 3D map in a separate process at the beginning of the application or implement the reconstruction process gradually.

Researchers have successfully applied the Kalman filter for SfM. One example is a recursive two-step method to recover structure and motion from image sequences based on Kalman filtering [98]. The algorithm consists of two major steps. The first step of this algorithm estimates the object's pose with an *extended Kalman filter* (EKF). In the second step, each feature point's 3D position is estimated with a separate EKF.

Simultaneous localisation and mapping (SLAM) is an approach where a map of the unknown environment is built simultaneously whilst tracking the camera pose. Researchers use it widely in mobile robotics and some have adopted it for augmented reality as well. Researchers have reported several SLAM implementations for AR. One of the first using it for AR was [99]. Developers may implement SLAM in a separate thread. This way, the reconstruction works in an incremental way. In the beginning, a coarse map is used, but within time, the accuracy improves.

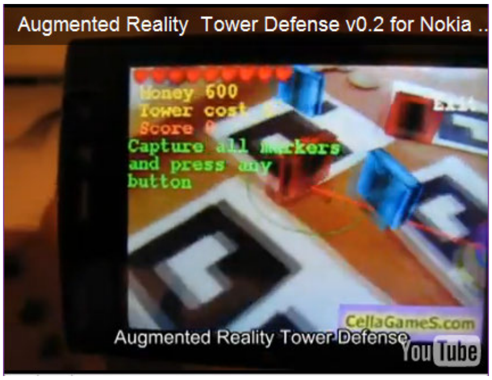

Augmented Reality Tower Defense v0.2 for Nokia ..
Money 600
Tower cost
Score
Capture all markers and press any button
CellaGameS.com
Augmented Reality Tower Defense
YouTube

Figure 39. Augmented reality game Tower Defence for Symbian smartphone.

The SLAM type approach is suitable even for mobile devices with limited computational capacity as the 3D map of the environment is built incrementally. [100] presents a real-time algorithm for mobile robotics that can recover the 3D trajectory of a monocular camera, moving rapidly through a previously unknown scene.

Mobile games have a lot of potential for the mass market, which makes mobile AR an interesting research area. In addition to games, companies and research groups provide several tools for mobile AR development. For example, CellaGames provides a marker-based tracking library for the Symbian platform [66, 66] for AR (game) development. This SMMT library (SLAM Multi-marker Tracker for Symbian) uses a SLAM type of approach (as its name suggests) to calculate the marker field at the beginning of application. The Tower Defence game demo (see Figure 39) is one application that uses it. For marker-based AR a SLAM type approach allows markers to be set up freely.

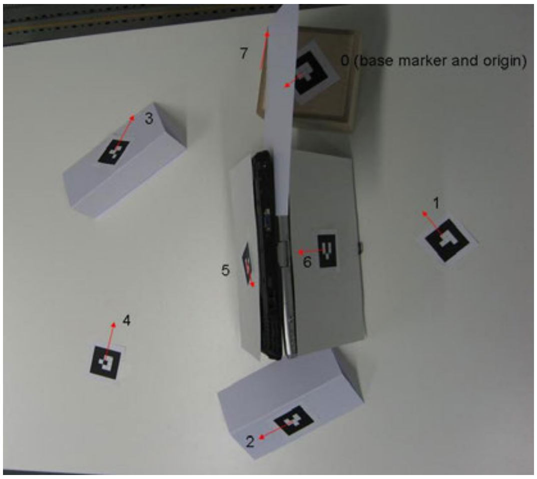


Figure 40. An example of a small-scale marker field test configuration from [8].

The idea of automatic real-time calibration without any preparation is old and researchers have carried out several implementations. For example, we presented in [8] a system where calibration is a real-time process and where the user can lay markers randomly on suitable places and start tracking immediately. This system allows the user to place markers in any 3D arrangement including even arbitrary angles and slanting planes. The accuracy of the system improves on the run as it updates the transformation matrices dynamically. In our system, we can implement the calibration of a marker field as a separate calibration stage as well. The user can save the results and use them later with another application. In our system, we created a graph of markers and used graph optimisation to create a marker field.

Our approach is well-suited to situations where the marker field as a whole cannot be seen but parts of it create chains of markers bridging one area to another. For example, it is suited to installations where marker fields extend from one room to another along a corridor or marker fields circulate around an object (Figure 40). In a situation where the whole set of markers is visible simultaneously a bundle adjustment would probably optimise the whole marker set better.

3.3.3 Bundle adjustment

Given a number of images taken from different viewpoints, *bundle adjustment* is defined as the solution of 3D coordinates describing the scene geometry, the relative motion of camera and the optical characteristics of the camera.

Let us assume that we have m images taken with one or more cameras. Let $\{\mathbf{X}_i \mid i = 1, \ldots, n\}$ be a set of 3D points. We mark the corresponding 2D coordinates of point $\mathbf{X}_i$ in image j with $\mathbf{x}_{ij}$. We denote the projection matrix associated with image j with $\mathbf{P}_j$. In an ideal case $\mathbf{x}_{ij} = \mathbf{P}_j \mathbf{X}_i$. However, the real measurements are subject to noise. Therefore, the problem is to find the maximum likelihood estimate for parameters $\{\mathbf{X}_i\}$ and $\{\mathbf{P}_j\}$ which minimise the reprojection error

$$\sum_{ij} d\left(\mathbf{P}_j \mathbf{X}_i, \mathbf{x}_{ij}\right)^2,$$

for all images where point *i* is visible. The $d(\mathbf{x}, \mathbf{y})$ is the geometric distance between points **x** and **y**. Finding a solution for this is a bundle adjustment problem.

The bundle adjustment problem can be formulated in most cases as a non-linear least squares problem and can be solved using the Levenberg-Marquard method, for example.

If a system has n points in *m* images and each camera has 11 degrees of freedom, then the system has $3n + 11m$ parameters. Thus, it needs to factorise (and sometimes even invert) the matrices of the size $(3n + 11m) \times (3n + 11m)$. If *n* and/or *m* increase, the processing time and capacity required for solving this increase polynomially. General solutions to this problem are [72]:

- data reduction,
- interleaving and
- sparse methods.

In *data reduction*, the system uses only some of the images (reduce *m*), or only some of the key points (reduce *n*). It may skip images with small parallax and reduce redundant data. The system includes only *key images* that best represent the data (based on a heuristic method). If a marker field is being created, it keeps the images containing several markers in different view angles, and retains enough images to cover the whole marker field. Generally, robust features covering the whole scene sparsely are used. Augmented reality systems can use data reduction easily. They reduce the number of features naturally, as they use only the corner points of markers for reconstruction.

Interleaving means minimising reprojection error by varying only one source at time. In interleaving, the system alternates minimising reprojection error by varying the cameras and minimising reprojection error by varying the points. It estimates each point independently assuming fixed cameras, and similarly it estimates each camera independently assuming fixed points. Thus, the biggest matrix in the minimising problem is the 11 x 11 camera matrix. Interleaving minimises the same cost function as the original problem, and thereby finds the same unique solution, if it exists. However, it takes a longer time to converge [72], which limits its use on real-time AR applications.

Sparse methods take into account the knowledge that the interaction between parameters is sparse (e.g. only some of the features exist in each image). In this case the matrices in the optimisation problem have large empty blocks (zeros). It has been shown that the use of the sparse variant of the Levenberg-Marquardt algorithm gains clear computational benefits compared to the standard version [101]. A widely used method for sparse bundle adjustment is the one described in [101].

In a general *bundle adjustment* problem, the internal camera parameters are unknown, e.g. 3D reconstruction based on historical images from unknown cameras. In AR applications, in most cases it is possible for the user to calibrate the camera and the system to gather the internal camera parameters beforehand, which simplifies the bundle adjustment problem. With a pre-calibrated camera, the number of free camera parameters is reduced from 11 to 6, the camera's translation (x, y, z) and rotation R(α, β, γ).

3.3.4 Dynamic multi-marker systems

The marker field may be dynamic; the user can move, remove or add markers during the application use. In this case, the system needs to be able to configure the marker field dynamically during the run time, (not only at initialisation phase). The Kalman filter approach is able to adapt to dynamic changes and is often used for dynamic marker fields.

However, a more common situation is that an application has two types of markers: static and dynamic. Static markers are used for camera tracking and dynamic ones to manipulate objects and for user interactions. In this case, the system creates a multi-marker setup with all static markers, and the camera pose is calculated relative to these markers. In addition, the system detects and tracks all dynamic markers and calculates their individual pose relative to camera.

An example of this kind of dynamic multi-marker system is an application where a static marker field defines a table surface, and users can use dynamic markers to pick, move and drop objects. Dynamic markers can also be used for defining interaction actions, e.g. proximity actions, deformations, etc.

4. Marker types and identification

A good marker is such that a computer vision system can robustly and reliably detect it and identify it unambiguously. In Section 3.1 (page 40), we discussed how black and white markers are favourable in the sense of reliable detection under different lighting conditions, and how square shapes support robust pose calculation. In addition, other marker characteristics affect the degree of successful detection. Markers are often seen from an oblique angle and a narrow border easily becomes discontinuous (Figure 41), causing the marker not to be detected. The system may process the image to connect line segments, but it requires more processing time. A thick border ensures fast and robust detection, and therefore AR markers commonly use them.

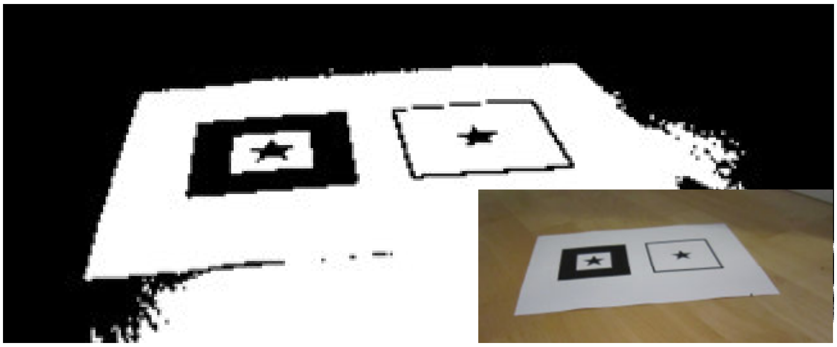


Figure 41. A thin border around the marker may become discontinuous when the system sees the marker under perspective transformation, whereas a thick border is more robust in such situations.

Besides the pose of a marker, it is useful to know the identity of the detected marker. With this information, the system can use several markers and associate different markers with different data or with different interactions etc. This is actually something that makes marker-based tracking systems versatile compared to feature-based systems. Thus, marker identification is a substantial part of marker-based tracking.

Augmented reality applications use a great number of different marker types. Two main categories are template markers and 2D barcode markers. In this section, we explain what we mean with these categories, give examples of other marker types and discuss invisible markers.

Marker identification techniques belong respectively to two main categories: matching (used for template markers) and decoding (used for 2D barcode markers). Matching algorithms only identify a marker, whereas decoding algorithms decipher the data encoded in the marker. We will explain these techniques in more detail in this chapter.

Matching techniques require a database of all possible markers and the system tests the marker under identification against all of them. Decoding algorithms do not need such a database; the content of a data marker can be unforeseeable (e.g. URL, text, etc.)

Furthermore, we discuss error detection and correction for binary markers. We present markers as having light backgrounds, i.e. all markers have black borders. However, all of them are suited to dark backgrounds as well; users may print markers in negative and the system thresholds and analyses the image in negative.

4.1 Template markers

Template markers are black and white markers that have a simple image inside a black border. The first ARToolKit markers were template markers. Figure 42 shows examples of template markers.

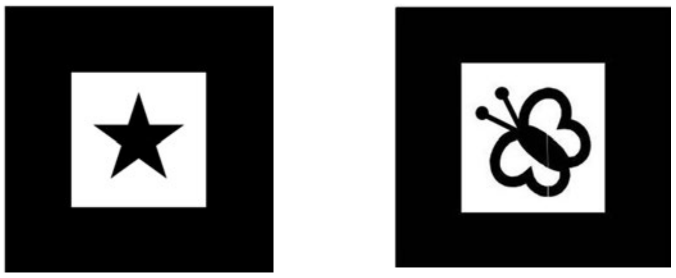


Figure 42. Examples of template markers.

Detection systems typically identify them by comparing their segmented images with marker *templates*. The marker templates are sample images of markers. During the identification process, the application matches a detected marker against each template and the best match defines its identity. Template markers have only a name or ID associated with each marker. If a system wants to link more data to a template marker, it needs to use a database. Next, we will discuss template matching in more detail.

4.1.1 Template matching

Marker identification is a simplified version of the general *template matching* problem: the marker detection process defines the matching area, whereas in a general matching problem, the location, size and orientation of the matching area are unknown.

A marker template is a sample image of the marker (Figure 43). In template matching, the detected marker is unwarped using the calculated camera pose, scaled to the same size as marker templates and compared in four different positions to all marker templates. The template that gives the highest similarity value (smallest dissimilarity value) is the correct marker. The orientation is the same in the best matching template. If all similarity values are lower than a threshold, the system rejects the marker. Figure 43 presents an example of a marker, its template and the detected marker with a sampling grid illustrated on top of it.

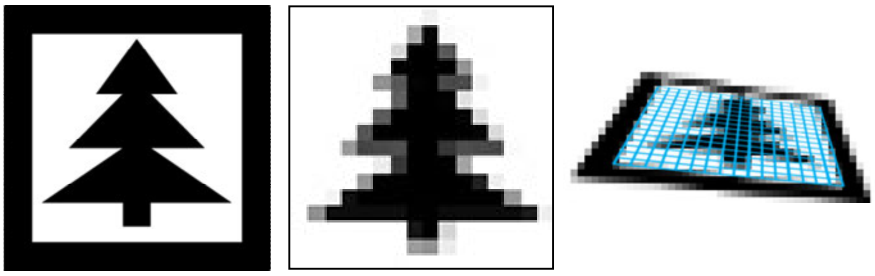


Figure 43. Example of a template marker: on the left, a marker, in the middle its template in size 16×16, and on the right, the marker as detected in an image with an overlay of 16×16 sampling grid.

Instead of unwarping the whole marker, the system can project the centres of all cells in the template into image coordinates using the calculated camera pose. Then it can sample the pixel values directly from the greyscale or threshold image. The value can be the value of the nearest pixel, the average or mean of N nearest pixels, the mean value of all pixels inside the sampling grid cell, etc.

Template matching is an especially suitable method for identifying *template markers*, as the name suggests. For example, ARToolKit uses template matching for identifying template markers. Before template matching a greyscale image is often normalised in such a way that the darkest areas are black and the lightest are white. The system may also have templates of different sizes.

The similarity can be based on SSD (the sum of squared differences) or cross-correlation, for example. In SSD the dissimilarity value between the marker and template is

$$D = \sum_i d(x_i, y_i)^2,$$

where *i* goes through all pixel positions, x_i and y_i are the pixel values of the marker's and template's ith pixel accordingly and *d* is difference of these values. For each template the minimum of dissimilarity values over all positions is selected $D_t = \min\{D_p \mid p = 1,2,3,4 \text{ (all positions)}\}$. Then the template with smallest dissimilarity value is selected

$$\min_t \{D_t \mid t = 1, \ldots \text{ number of templates}\},$$

if it is below a threshold. Usually Euclidean metrics is used, but another kind of metrics could be applied as well.

In normalised cross-correlation the dissimilarity value is

$$D = \frac{\sum_i \left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum_i \left(x_i - \bar{x}\right)^2}\sqrt{\sum_i \left(y_i - \bar{y}\right)^2}},$$

where $\bar{x}$ is mean of the marker's pixel values and $\bar{y}$ is the template's mean pixel value.

As a system needs to match the detected marker against each template four times, it is clear that the larger the set of markers the system uses, the longer the time it will need for marker identification. Therefore, template markers are inefficient in practice if the system requires a large set of markers.

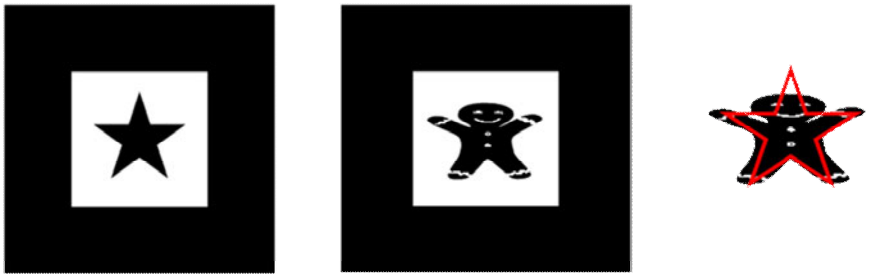


Figure 44. An example of undesired similarity between template markers. These markers differ clearly to the human eye, but a marker detector may confuse them as the image areas are almost overlapping.

Another limitation is that template markers can only be differentiated as long as they differ in sampling positions. Therefore, two markers may be confused even if they seem different to the human eye (see Figure 44). A system can overcome this problem by comparing templates beforehand and ensuring that they differ clearly from each other in all four orientations.

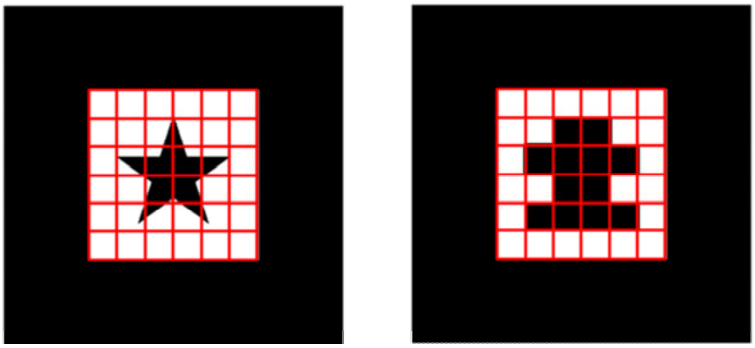


Figure 45. Example of a template marker and corresponding data marker.

If we look at a template marker consisting of an image (e.g. Figure 45), we notice that the figure cuts some of the sampling cells. Therefore, some of the cells in the template are grey instead of black or white (see Figure 43, for example), usually presenting the average value of the cell. For template matching, the system should also use the average of the pixels in the sampling cell accordingly. Nevertheless, in practice, the undistortion is imprecise and pixels may lie on the border of the cells. Therefore, only values in the centre of the sampling cell are normally used, and to speed up the detection process, it would be convenient to use only a few values (or even only one value).

On the other hand, if a system uses only few values, it may pick only black or only white pixels, even though the cell actually consists of both black and white cells. An imprecise sampling location, blur, unfavourable thresholding, noise and other factors make matching even more error-prone. What would then make matching more reliable? One obvious thing would be to have markers where each sampling cell is either completely black or white (e.g. the right-hand image in Figure 45). Then the blur and inaccurate thresholding caused by this only affects the borders of the cells instead of the centres of the cells. Thus, the system gets the correct value even from only few pixels and the value is correct even if the sampling position were slightly imprecise.

These kinds of markers consisting of binary cells have another advantage compared to image markers: the cells can be used as binary numbers to store data. Next, we will discuss this type of marker.

4.2 2D barcode markers

2D barcode markers are markers consisting of black and white data cells and possibly a border or other landmarks. Typically, in the marker detection process, the system samples the pixel values from the calculated centre of each cell, and then resolves the cell values using them.

We can categorise 2D barcode markers as those solely defining an identity (ID markers) and those containing more data (data markers). Next, we present examples of both. We also present some popular 2D barcode standards here that AR applications may use as well.

Binary *ID markers* consist of black and white squares that are interpreted as binary numbers. Simple ID markers contain little data, normally only an ID number. However, if a system wishes to link more information to such a marker, it can use a database, as with template markers. Figure 46 shows an example of a binary ID marker on the left and a binary data marker on the right.

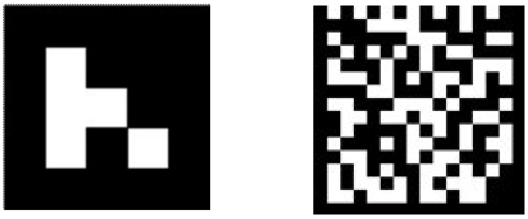


Figure 46. On the left, a simple binary marker (ID marker), and on the right, a binary data marker. The binary marker on the right is an example of our marker system presented in [11].

We call 2D barcode markers that contain more information than just an ID number *data markers*. Data markers seem similar to ID markers, yet they have more cells (and thus more data). Data markers usually also have built-in error detection and correction among the marker data. ARTag [102] is an example of this type of marker.

In [11] we presented a flexible multipurpose data-marker for pointing a user interface. We created a marker type with a flexible data size varying freely from 6 x 6 to 30 x 30. Thus the number of data cells (bits) varied from 36 to 900. Our application automatically detected the marker size; this enabled us to use markers of different sizes in the same application. One edge of our marker consists of varying black and white cells. The system can easily detect the varying edge, calculate the number of white cells and derive the number of data cells in the marker.

We divided the data in our marker system into actual data and metadata, the metadata indicating the type of information (e.g. URL or phone number). Thus, the system could read the type of data and act accordingly using the actual data. This approach enables a large variety of possibilities; AR was only one of the usages that we proposed.

In addition, our marker system has built-in error detection and correction. Figure 46 shows an example of this marker on the right.

4.2.1 Decoding binary data markers

For binary markers that have a fixed number of cells, we calculate the centre of each cell and eventually the cell size. The application can do this for an unwarped image as well, using the inverse perspective projection obtained from the marker pose.

The cell value may be the value of the pixel closest to the reading position, the bilinearly interpolated value of the four nearest pixels, the average of N nearest pixels, the median of N nearest values, etc. For each cell, the system gets a binary value and the whole data of the marker can be represented as a series of binary values or as one binary number. In the simplest binary data matrices, this (binary) number is the same as a marker ID (Figure 47).

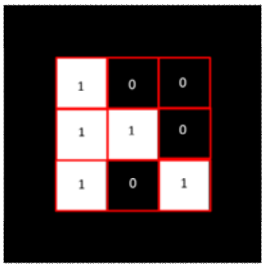


Figure 47. Example of simple marker decoding: marker ID 100110101 (= 309).

This kind of simple decoding system is less robust, and therefore marker systems usually have built-in error correction and detection.

4.2.2 Error detection and correction for binary markers

The advantage of data markers is that besides encoding information, a system can also use part of the bits for error detection and correction. This is impossible with image and template markers. We present here two main types of error detection and correction codes, namely Hamming codes and Reed-Solomon codes.

Hamming algorithm [103] is a widely used method for error detection and correction for its simplicity. The *Hamming code* is based on the use of parity bits. A parity bit tells whether the number of ones in a binary number is odd or even. *Even parity* equals one if the number of ones is odd and zero if the number of ones is even. *Odd parity* is respectively one if the number of ones is even, and zero if the number of ones is odd.

If a system adds a parity bit to data, it is able to detect if one bit is wrong, as the parity bit does not match the data. A single parity bit only reveals a single-bit error; it cannot detect a double-bit error, and it cannot tell which of the bits has changed. With more parity bits, a codec is able to detect the locations of detected errors and thus correct them. For example, the Hamming (7,4) code encodes 4 data bits into 7 bits by adding three parity bits. It can detect and correct single-bit errors.

With an addition of an overall parity bit to the Hamming (7,4) code, it becomes a Hamming (8,4) code that is able to detect (but not correct) any two-bit error, in addition to the ability to correct single-bit errors. The more parity bits a system uses, the more errors it is able to detect and correct. Usually, the data is divided into blocks (e.g. 8 bits) and each block is encoded using the desired Hamming code (e.g. Hamming (8,4)).

Hamming codes are easy to apply to 2D barcode markers. The system needs only to allocate some of the bits (cells) for parity checks and the rest of the bits for data encoding. We used Hamming codes in our marker system [11, 17] because Hamming codes have little overhead and thus are suitable for small binary data markers. Hamming codes are commonly used in AR marker systems, e.g. in ALVAR and ARTag [102].

Reed-Solomon codes [104] are block-based linear error correction codes; they add extra redundant bits for each block of digital data. The number and type of errors that can be corrected depends on the characteristics of the Reed-Solomon code [105]. The Reed-Solomon codes are named similarly to Hamming codes: an RS (n,k) code means that it encodes k data symbols with a total of n symbols, which means that there are n-k parity symbols, where each symbol consist of s bits. A Reed-Solomon decoder can correct up to t symbols in a codeword, where $2t = n-k$.

Reed-Solomon codes have more redundancy compared to Hamming codes, and therefore they are better suited to larger data sets that require a high degree of reliability. Reed-Solomon codes are used in CDs, DVDs and mobile communications, and in barcode standards such as Datamatrix. Augmented reality systems more often use simpler Hamming codes.

4.2.3 Data randomising and repetition

Error detection and correction methods are able to detect and correct only a certain amount of errors within a block. However, errors in other blocks do not affect them. Thus, a single error correction method is able to correct several errors if they are all in different blocks.

We can identify several error sources in marker detection. Reflections and shadows can lead to flawed thresholding. Inaccurate corner detection may shift the source locations of data samples. In both cases, the probability of error existence correlates between neighbouring cells. In *data randomizing*, the system scatters the bits of each block within the marker. This means that data cells belonging to a block are at different parts of the markers and not adjacent. This way, the probability of erroneous bits belong to different blocks is increased. Even small size binary markers can use data randomising, as it does not increase the number of data cells.

A system can improve the probability of getting correct data if it encodes the same data twice or even more times. This is called *data repetition*. Obviously, it improves the probability of correct decoding. Data repetition is unsuitable for small codes as it increases the amount of total data. However, large standards such as Datamatrix have redundant data.

4.2.4 Barcode standards

2D-barcode standards such as DataMatrix [331, 332], QR Code [333, 334] and PDF417 [335] were originally developed for logistics and tagging purposes but are also used for AR applications. We present these three here, though there are numerous other standards (e.g. MaxiCode, Aztec Code, SPARQCode, etc.) which might be used for tracking in some applications too.

The most popular application for DataMatrix is marking small items such as electronic components. The DataMatrix is scalable, with commercial applications as small as 300 micrometers and as large as one meter squared. Symbol sizes vary from 8 × 8 cells to 144 × 144 cells. It can encode up to 3116 characters from the entire ASCII character set (with extensions). The DataMatrix barcode is also used in mobile marketing under the name SemaCode [106].

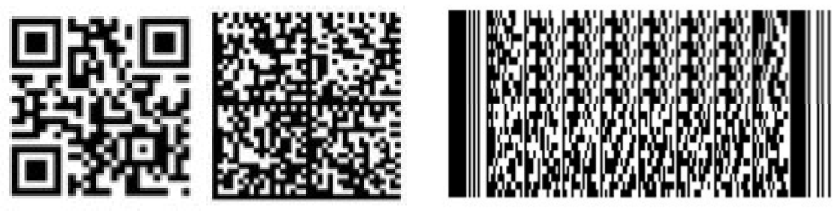


Figure 48. On the left, QR code; in the middle, DataMatrix and on the right, PDF417.

QR Code is a two-dimensional barcode created by the Japanese corporation Denso-Wave in 1994. QR is the abbreviation for Quick Response, as the code is intended for high-speed decoding. QR Code became popular for mobile tagging applications and is the de-facto standard in Japan. It has built-in support for Kanji encoding, which also explains its popularity in Japan. QR Code is also flexible and has large storage capacity. A single QR Code symbol can contain up to 7089 numeric characters, 4296 alphanumeric characters, 2953 bytes of binary data or 1817 Kanji characters [107]. QR codes are widely used in mobile AR applications.

PDF417 was developed in 1991 by Symbol (recently acquired by Motorola). A single PDF417 symbol can be considered multiple linear barcode rows stacked above each other. The ratio of the widths of the bars (or spaces) to each other encode the information in a PDF417 symbol. For that reason, the printing accuracy and a suitable printer resolution are important for high-quality PDF417 symbols. This also makes PDF417 the least suitable for AR applications where the marker is often under perspective transformation. A single PDF417 symbol can theoretically hold up to 1850 alphanumeric characters, 2710 digits or 1108 bytes. The exact data capacity depends on the structure of the data to be encoded; this is due to the internal data compression algorithms used during coding.

All these three standards DataMatrix, QRCode and PDF417 have built-in error correction and detection based on Reed-Solomon algorithms.

Many barcode readers can interpret several types of markers. For example, UpCode readers can also read DataMatrix and QR code in addition to its own UPCode markers [108].

4.2.5 Circular markers

Circles are projected as ellipses under perspective projection. The position of ellipse's centroid is more accurate that the position of the centre of a square [109]. This is due to the normal way of calculation. For a square, the centre is estimated using four corner points, whereas more pixels along the perimeter are used for fitting an ellipse and estimating its centre. This makes the centre of an ellipse statistically more stable and thus more accurate. The accuracy increases further if several circles within each other are used; the centre of all of them must coincide (as in VisTracker [109]).

In photogrammetry, circular markers are traditionally used. Photogrammetric applications often require high accuracy but allow offline processing. Circles are often identical (see Figure 49) and are mapped manually or semiautomatically from frame to frame. Most augmented reality applications are real-time applications and automatic camera pose estimation is a necessity. Therefore, basic identical circular markers are seldom used in AR. The nature of AR applications also often requires identifiable marker sets. Designing a large set of identifiable square markers is easier than designing a set of circular markers. Consequently, square markers are more popular in augmented reality.

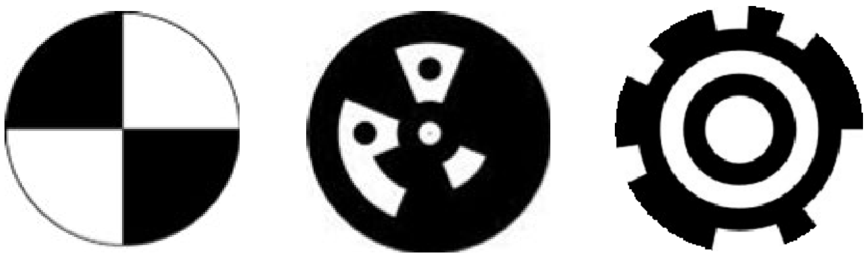


Figure 49. On the left, an example of a simple position marker used in photogrammetry; in the middle, a circular marker used by InterSense [109] and on the right, SpotCode [110].

A circular code gives a highly accurate position, but a single centre coordinate is not enough to derive camera pose, whereas a square marker has four corner points and defines the relative pose of the camera alone. Therefore, a system needs to detect at least four circular markers (or more if they a non-coplanar and non-collinear). The application can increase the accuracy of pose calculation using more markers (see Section 3.3 Multi-marker setups (marker fields)).

4.3 Imperceptible markers

Markers provide a convenient way for pose calculations, but in certain environments or situations visual markers are unwanted. Should the problem be more about "ugly" markers, people can use "nice-looking" image markers. If all visible patterns are undesired, the system can use markers that are invisible to the human eye but detectable by machine. One possibility is to use markers and detection devices operating on wavelengths other than visible light, e.g. in the infrared range. Another possibility is to use markers that are so small that the human eye cannot distinguish them. In this section, we will cover these three possibilities: image markers, invisible markers and miniature markers.

4.3.1 Image markers

A marker system can use natural (colour) images as markers. Image markers typically have a frame or other landmarks to aid detection and pose calculation, but these are not necessary. Image markers are typically identified using template or feature matching.

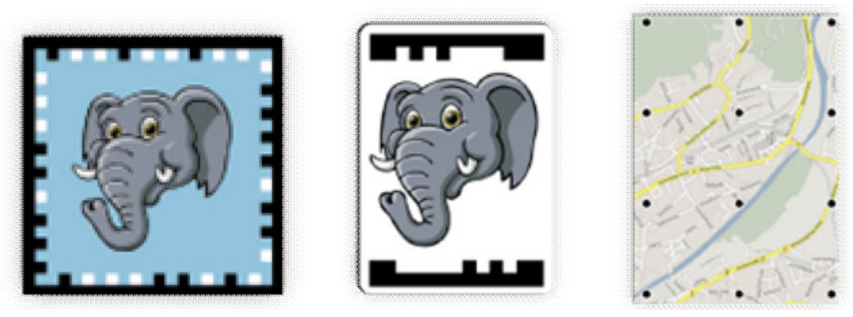


Figure 50. Examples of StudierStube image markers. From left to right: frame marker, split marker and dot marker (Images courtesy of Dieter Schmalstieg).

Developers often use framed markers in applications requiring a high degree of accuracy for pose and on mobile devices with a low processing capacity. For example, StudierStube tracker adds frames or dots to image markers (see Figure 50).

Implementations that detect images without frames have the advantage that an AR application can operate in an existing environment without changes to the environment itself. For example, an AR application may bring added value to a book without changing the book itself. AR application could use the images of an existing book to pop out extra 3D visualisation, animation, etc.

Figure 51 shows an example of how an existing image can be used as a marker. In this example developed by the VTT Augmented Reality Team, the application detects natural features and calculates the relative pose of the camera by matching features to the original reference image. An animated character is augmented on top of the book in the right pose. Natural images are in a way "imperceptible", as they can be built in the environment in such a way that they are part of the environment and do not distract the human eye.

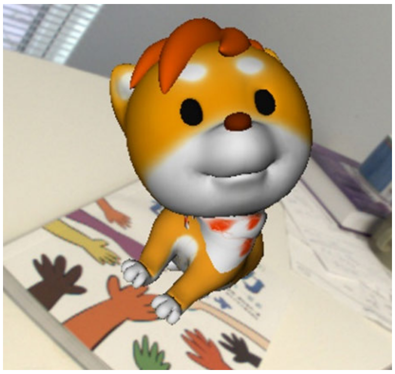


Figure 51. Example of natural image detection: An animated character is augmented on the top of a book. The detection is based on features in the cover image. (Image: VTT Augmented Reality team).

BazAR [111] is an example of a computer vision library based on feature point detection and matching. In particular, it is able to quickly detect and register known planar objects in images. It is suitable for augmented reality applications, and it can be used to create natural image-based AR. The computer vision and feature detection and matching of BazAR is based on [112] and [113].

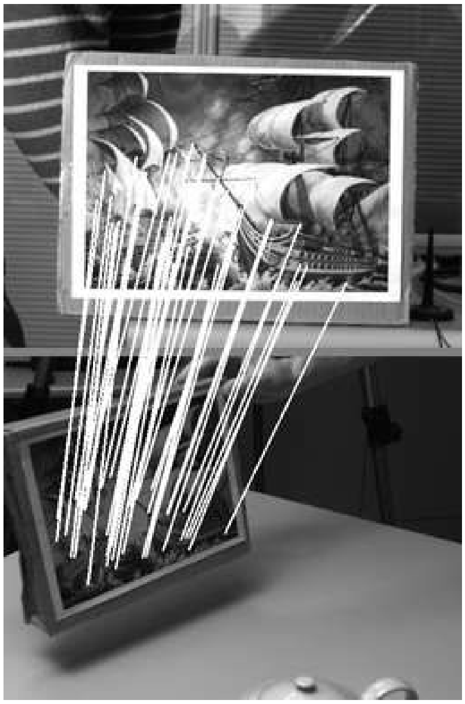


Figure 52. Feature matching with BazAR (Image courtesy of Vincent Lepetit).

4.3.2 Infrared markers

Infrared (IR) light has a wavelength in range (750–1 mm). It is greater than the visible light spectrum (380–770 nm) and is therefore invisible to the human eye. However, many cameras can perceive range adjacent to the visible spectrum (called near IR) and special IR cameras operate within a larger IR range. There are also special filters that limit the perceived light to a specific narrow IR band.

An IR marker system can either use a self-illuminated marker, retro-reflective material or IR spotlight. In addition, it can use an IR projector to create markers. The system can detect IR markers with a special IR camera or with a normal camera if the system uses the near infrared range.

Invisible marker systems have been much developed in terms of general barcodes, e.g. [114] and [115]. Researchers have also studied IR marker systems for augmented reality, for example in [70, 116]. In addition, invisible markers have been used for watermarking in augmented reality [117].

A common limitation of the IR markers is that they only work indoors, where no uncontrolled IR light source is present. Outdoors the sun emits IR light and disturbs the IR detection system.

A *self-illuminating marker* emits IR light. The marker itself may consist of IR LEDs, which the system detects using an IR camera, or the marker can be a binary marker, where white cells are transparent and black cells are opaque and the system has IR LEDs under it. The former approach is more common.

IR LEDs appear as dots in an IR camera image and as such, they just give a location, but no ID. Researchers have studied several approaches to attach an ID to them. For example, the SONY IC CAM system [118] used blinking LEDs with unique temporal codes to reveal their identity. The drawback is that the system needs to observe each LED for a certain period of time before being able to identify them. The system presented in [119] uses a heavy headset with two cameras, a scene camera pointing forwards, an IR tracking camera pointing upwards and LEDs installed on the ceilings.

The problem with these systems is that the LEDs need a power source and their installation is time consuming, complex and sometimes not even possible.

Retro-reflective material is material that reflects light back to its source with minimal scattering of the light. A marker made from IR retro-reflective material reflects IR light from an external light source. The marker itself does not need a power source. In this sense, retro-reflective markers are more convenient than self-illuminating IR markers.

Retro-reflective material reflects light towards its origin; therefore, the IR spotlights must be near the camera. Usually IR LEDs are around the camera's objective.

Most AR applications augment virtual objects on top of RGB images on displays, and a common approach is to copy the camera image, convert it to greyscale, do all computer vision processes and finally augment on top of the original colour image. Should an AR application use a see-through HMD, then the system only renders the augmentation on a see-through display and the application does not need the colour image in a visible spectrum. In this case, the system may rely solely on a camera operating in a non-visible spectrum (e.g. IR) and the system is able to use non-visible markers. An example of this kind of wearable augmented reality system is presented in [120]. In this case, the IR camera detects the retro-reflective markers illuminated with IR LEDs around the camera. The system computes the pose of the user normally and augments information on an optical see-through display.

If the application needs to augment on top of the colour image, it needs to have either a camera that can take normal and IR images in turns or two separate cameras. Both [121] and [122] present one camera AR system where the IR lights are synchronised with camera capture. The system takes every other image with the IR light on and every another image with the IR lights off. These images are subtracted to extract the retro-reflective markers. Thus, these two images are not exactly from the same moment of time, and this causes inaccuracy both for detection and augmentation. Therefore, this approach is unsuitable for an application with fast movements and high accuracy demands.

In the system presented in [123] the camera's ability to detect the IR range is used to detect printed halftone-based hidden markers. The augmentation is done over a (greyscale) IR image, and the lack of colours is distracting. An advantage of this approach is that it does not require any special devices or material.

The paper [116] presents an AR system fluorescent (IR) invisible with two cameras. The system has an IR camera and a visible camera, which are positioned on each side of a cold mirror (which reflects visible and transmits IR light) so that their optical centres coincide with each other. In this kind of setup both cameras can capture images exactly at the same moment of time and still share the same view. However, the physical setup makes movements difficult.

Sometimes it is impossible to place either visible or invisible markers in the environment and yet it would be nice to have a marker-based augmented reality system. An example of this kind of situation is an assembly line, where a new component replaces an old one after every few assembly tasks, and the environment itself is such that it is difficult to place markers beside the assembly area. One possible solution is to project the markers on the component and then detect them with an AR system. For example in [124] markers are projected with infrared projectors on the surface of the equipment to be maintained (see Figure 53). The system detects the projected markers with an IR tracking camera and augments on top of the colour image of the scene camera.

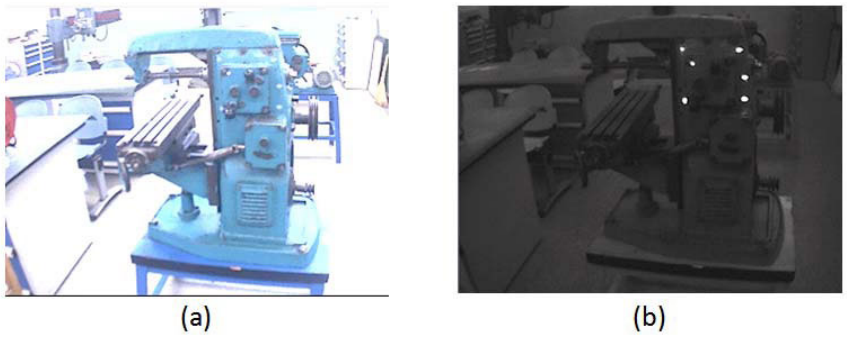


Figure 53. Projected IR markers for maintenance work: on the left, scene camera view and on the right, tracking camera (IR cam) view (image from [124]).

People have used the same basic idea for tagging purposes in [125]. This system uses an infrared (IR) projector to project temporally-coded (blinking) dots onto selected points in a scene. These tags are invisible to the human eye. A time-varying code is used and the system detects projected dots and identifies them using an IR photosensor. Each tag is associated with its 3D location and the identity of the object on which the tag is attached. This way the system can augment information for the object on the scene camera’s image.

Sometimes it is possible to skip the whole marker-detection procedure and project the augmentation directly on the environment. This kind of system where the system augments directly on the environment is called *projector-based augmented reality* or *projective AR*.

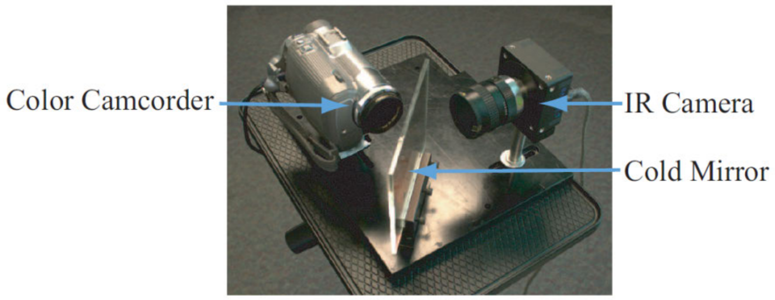


Figure 54. Two-camera setup with coinciding optical centres. Image from [125].

In the maintenance scenario, the projected markers can be visible and therefore do not require any special adaptation or devices. However, in some application environments it is vital that the markers are imperceptible and it must be ensured that they remain invisible even if the scene camera is able to capture the near infrared range.

This kind of situation would be an augmented reality TV-studio type application as suggested in [126]. It presents a mock-up of an AR TV studio with adaptive projected visual markers that are imperceptible (Figure 55). A coded image is temporally integrated into the projected image in such a way that it is invisible to the human eye but can be reconstructed by a synchronised camera.

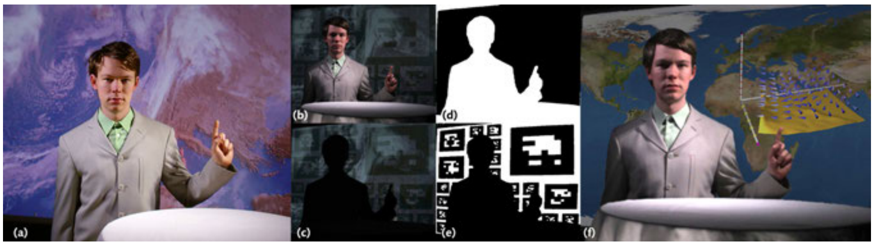


Figure 55. AR TV studio mock-up presented in [126] (a), images captured by a synchronised camera at 120 Hz (b, c), computed foreground matte from real-time flash keying (d), extracted multi-resolution marker pattern for in-shot camera pose estimation (e) and composite frame with virtual background and 3D augmentation (f).

We did some experiments with virtual advertising in 2001 in a customer project. With our system, we replaced real advertisements with virtual ones in real time. We used a similar set up as in Figure 54. The image was divided for a TV camera and an IR camera. The billboard had IR LEDs at the back. This way we were able to detect the area from the IR camera and augment the virtual ads onto the TV camera image. We were also able to detect occlusions from the IR camera, e.g. people walking by in front of the billboard. This way we were able to mask augmentation in a similar way as with the Kinect example on page 130. In our demonstration system, we used a separate TV camera and IR camera, but the intention was to use an embedded cold mirror to divide the same image to both cameras and to enable them to share the same optical centres similar to the system presented in Figure 54 in the final product.

We also proposed a new camera structure as a general solution for similar computer vision applications. Instead of a normal colour filter consisting of cells letting red, green or blue light pass into the image detector, the camera could have additional cells with an IR filter, thus it would be able to create a four-channel image $RGBI_R$. The colour image (RGB) and IR image (I_R) would then be aligned and synchronised and would share the same optical centre.

4.3.3 Miniature markers

Markers can also be so tiny that they are unnoticeable to the human eye. There are a few approaches to implementing a system that is able to detect miniature markers. Here we discuss two of them.
MIT Media Lab introduced a marker system called Bokode in [127]. The markers are inconspicuous to the human eye, but visible on common low-cost cameras from metres away.

The functionality of the markers is based on the defocus blur of a camera, to be more precise on the *bokeh effect* of the optical system. Bokeh occurs for parts of the scene that lie outside the depth of field and blurs those areas. The bokeh effect maps a cone of rays exiting from an out of focus scene point into a disc-shaped blur on the camera sensor.

The Bokode system consists of a tiny visual marker and lenslet placed a focal length away from the marker over it. The marker looks like a tiny dot to the human eye. The lenslet creates multiple directional beams (ray bundles) for each position in the barcode pattern. The camera lens focused at infinity captures a magnified version of the barcode pattern on the sensor. This way, the common off-the-shelf camera is able to capture features as small as 2.5 μm from a distance of four metres. The system is also able to compute the camera pose (distance and angle) with respect to Bokode within certain limits.

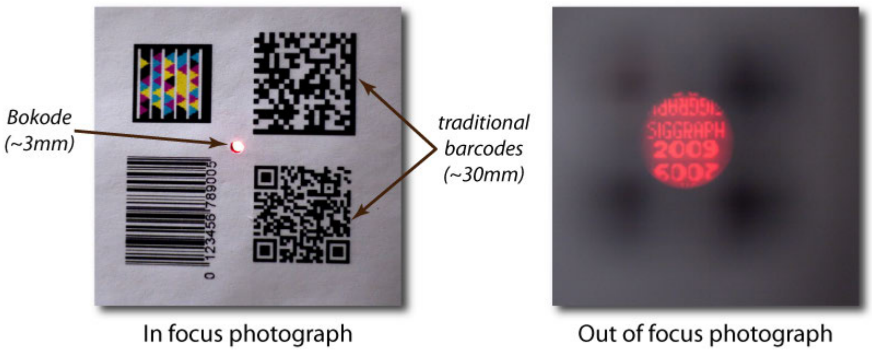


Figure 56. Bokode marker looks like a tiny dot to the human eye (on the left), but a camera is able to see the contents in out-of-focus mode (on the right), (images courtesy of Ankit Mohan).

Although the Bokode system performs well in marker detection, it is unsuitable for AR applications as such because the camera used for detection is out of focus. The system would normally need a second camera for base of augmented view unless a see-through display is used. Another limitation is Bokode's limited angular range, which is reported to be approximately 20° in the current implementation.

Besides Bokode, another approach for using tiny and thereby imperceptible markers is to use an add-on magnifying lens for detecting markers. For example [128, 129] describe an add-on magnifying lens for mobile phones, which is able to detect tiny markers.

We used a Nokia CC-49 add-on lens with a Nokia 3660 mobile phone for marker detection and encoding as described in [3]. With an add-on lens our system was able to detect and encode markers with a physical cell size as small as 0.06 mm^2 using a VGA resolution 640 x 480 image (Figure 57).

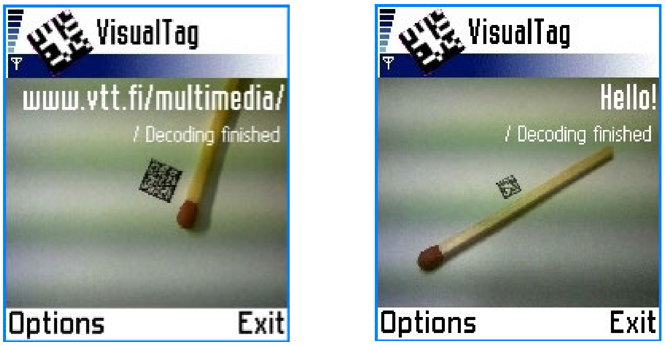


Figure 57. Screenshots of our marker detection system [11]. With the add-on lens, it is able to recognise tags with a cell size of only 0.06 mm^2. The marker in the left image contains the URL "www.vtt.fi/multimedia" and the marker in the right contains the word "Hello!"

In theory, a tracking system could combine this kind of tiny marker reading with feature tracking. The user could get the information from the marker by pointing at it with a mobile device at close range. Then as the user moves further away from it, the system could base the augmentation on feature tracking. Similarly, a Bokode type of system could first read the marker identity using the bokeh effect, but then switch to normal viewing mode and used feature tracking for augmentation.

However, most current mobile devices do not support this kind of camera actions. The add-on lens are either attached or not, and the system cannot switch between these two states on run time. Furthermore, in most devices, applications are not allowed to manipulate the camera's focus.

Nevertheless, the recent development of the camera software on mobile phones seems very promising for the future of mobile tagging applications. For example the Frankencamera API for Nokia N900, FCam [130, 131], enables easy and precise control of digital cameras. It enables the application to manipulate the camera's autofocus routine and to capture a burst of images with different parameters, for example.

On the other hand, current mobile devices with RFID (Radio Frequency Identification) readers could launch an AR application from an RFID tag and then use feature-based tracking for augmented reality.

Although the cameras on mobile phones do not have optical zoom, the development of the processing capacity and direct access to camera (e.g. with the abovementioned FCam API) enable the programmer to implement a digital zoom. An application can also interpolate images over time to acquire a super resolution image for detecting physically small markers. This approach is suitable for low quality cameras on mobile phones, where it benefits most. In good quality digital cameras, anti-aliasing filtering dilutes the super-resolution calculations.

4.4 Discussion on marker use

In many cases, the selection of tracking method (feature/marker-based) is unessential, but there are also situations where the selection matters. Sometimes it is beneficial to use markers. We explain these situations in the beginning of this section. Once an application developer has decided to use the marker-based method, he/she should also decide what type of marker to use. Because of the large variety of different markers, the decision is often difficult and there is no ground truth for the best marker type. However, we give some general guidelines for selecting marker types later in this section.

Augmented reality applications aim for real-time processing, therefore it is critical to speed up the marker detection process as much as possible. We report ways to quicken the detection process in Section 4.4.2, and after that, we discuss marker design and general marker detection application.

4.4.1 When to use marker-based tracking

Visual tracking does not require extra devices, as the camera is usually already part of the augmented reality system. Sensor tracking would require extra devices, which makes the system more complex and expensive. Thus, a developer may decide to use visual tracking because of the low costs or to keep the application and setup as simple and lightweight as possible. Model-based tracking would require a priori information, which limits its use. In practice, after the decision to use a visual tracking method is made, the choice is often between feature tracking and marker tracking methods (or a combination of these two).

Marker-based tracking often outperforms feature-based tracking in certain occasions and there are other reasons to prefer marker-based tracking. In the following, we list situations where a marker-based solution is a good choice.

1. Tracking in environments that are challenging for feature tracking

Environments with large uniform areas (e.g. large white walls) have almost no features and therefore feature tracking is impossible or at least very unreliable. However, if the user adds markers in such environment, tracking becomes possible.

Environments with repetitive textures (e.g. tiled walls) are extremely challenging for feature tracking due to the large number of similar features. In this kind of environment marker-based tracking is often more robust. In addition, environments with dynamic textures (e.g. trees moving in the wind) may be even more difficult than solely repetitive features as the locations of the features vary. If the user can attach markers in this kind of environment, a marker-based system is often a better choice. A tracking system can avoid some of these problems using prior 3D reconstruction of the static part of the scene with feature-based tracking, but it then requires an initialisation process.

An environment with reflective surfaces is also challenging for feature-based tracking systems, as the features seen in a physical location change due to reflections when the viewpoint changes. This kind of situation confuses a feature tracker. Again, a marker tracker may perform much better in this kind of environment, as a mirror image of a non-symmetric marker is distinguishable from the marker itself, whereas a feature tracker may collect confusing reflected features in its feature map.

There are also other challenging situations for feature tracking, where application developers might decide to use markers to increase the robustness of the system. For example, if the camera rotates wide angles between frames, features detected in one frame are invisible in the next, and the system is unable to deduce the relationship between the two views. Another difficult situation is a case where the camera stays in one location, because then the system is unable to get parallax between observations required to calculate features' distances. An adequate multi-marker configuration solves both problems.

2. Acquiring the correct scale and a convenient coordinate frame

A feature-based tracking system cannot deduce the scale from the images it sees. The environment could be a tiny scale model or a huge space; only the relative proportions can be derived from images (see Figure 58). The scale is fixed if the physical distance between two points is known. A marker tracking system knows the physical dimensions of markers and thus it knows the correct scale.

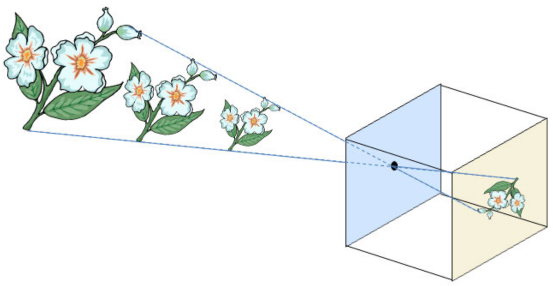


Figure 58. An optical tracking system can only derive relative proportions from camera images, not the scale. The camera image can be a projection of small object near camera or a projection of a big object far away. Based on the camera image, the system is unable to tell which of the three objects produced it.

A visual tracking system cannot deduce earth coordinates (i.e. which direction is "up", "down" or "horizontal") from what it sees without any additional clues. Therefore, the origin and coordinate directions of a feature tracking system are random. Markers lying on a floor, a table, a wall or on other known planar surface, as is often the case, define a reference frame for the world coordinates. Typically, the marker tracking system's origin is also relative to the marker origin.

An understandable coordinate origin and alignment of coordinate axes are important usability issues. A good example is an interior design application aimed at common users, where user-friendliness is one of the main issues. The user puts a marker on the floor and virtual furniture pop up on the floor plane in the application. Thereafter the user can move the virtual furniture on the floor plane. With a pure feature tracker, the user should do extra work to get the right scale and define the orientation floor plane (i.e. to align virtual and natural coordinate axis orientation). This might be all right in an application operated by experts, but it would probably prevent occasional end users from using this kind of the application.

In environments that are challenging for feature tracking, e.g. because of lack of features, the user could add posters (with suitable images) on the environment to enable feature-based tracking. However, as markers also provide the correct scale and convenient coordinate axis orientation, a better solution might be to use the posters as image markers and combine marker and feature tracking (see Point 5 Hybrid methods).

3. Environments with lots of moving objects and occlusions

Feature tracking may also fail in environments where a number of moving objects frequently occlude the background and objects themselves contain features. This happens for example when moving people or cars cover a major part of the scene. A feature tracking system often loses track in such situations and therefore a multi-marker system is often more robust. In such environments, a feature tracking system may also use markers for fast recovery (cf. initialisation in Point 5).

4. Need for extra information

Markers can maintain additional information, e.g. an ID, URL, text, etc.
This enables the system to associate data with markers and retrieve information. This is something that a feature-based tracking method is unable to do. Therefore, if an application needs the extra information that a marker can provide, a marker-based system is the natural choice, especially if the system needs to be able to read previously unknown information that cannot be stored in a database. Decoding a marker (see Section 4.2.1) is easier and faster than text recognition. Therefore, in a real-time application a marker-based system is more convenient than an OCR system, for example.

5. Hybrid methods

Hybrid methods aim to combine the advantages of different tracking methods. For instance, in an environment where the initialisation of a feature-based system is difficult for one reason or another, developers might find the use of markers for initialisation a good solution. In a completely new environment, the use of a marker is a good way to get the right scale, an understandable coordinate origin and alignment of the coordinate axis as explained in Point 2. This is especially important if end users are non-experts and cannot carry out an initialisation phase.

A common problem in feature-tracking methods is that they tend to drift over time. A hybrid tracking system is able to reset/adjust tracking each time a marker is visible and keep the system running correctly.

6. Efficiency

A marker-based system is typically computationally cheaper to implement. Marker-based tracking might be good for a proof-of-concept type of application where the emphasis is not yet on the tracking implementation but on easily demonstrating the application concept. Later, the real application can then use any tracking method (e.g. an off-the-shelf sensor tracking system).

7. Environment with existing markers

Should the environment already contain markers, the system could take advantage of them, e.g. an augmented reality application for a catalogue or journal with images could use those images as natural image markers (see Section 4.3.1 Image markers). In addition, if a system operates in an environment where some marker-like signs exist, the application developer could train the marker tracker to detect them and use them to achieve a more robust tracking system. This kind of environment could be a storehouse where each shelf is marked with an ID sign, for example, and the ID signs could function as markers.

8. Devices with limited computational capacity and memory

Marker-based systems need less processing power and memory compared to feature tracking. This is an important aspect in mobile augmented reality, for example, with lightweight mobile devices.

9. Interaction with the user

User interaction in certain types of applications is easy to implement with markers. For example, the user might move augmented objects by moving markers. Markers are tangible and even for an inexperienced user it is easy to understand how to move the objects. Users can also pick up and drop virtual objects with a marker, using it as a paddle [132], or the developer may attach markers to physical handles to create a tangible user interface [133].

10. Indication of the existence of virtual data

Markers also indicate the existence of virtual data for the user. Let us consider a magazine that has additional augmented content that the user can see with a camera phone. The system needs to indicate to the user somehow which pages do have virtual content. In practice, the magazine needs to use some sort of symbology (icons, markers) to catch the user's attraction. The system could utilise the same markers for tracking as well.

In VTT's self-funded TULARMAN project in 2010, we interviewed different players in printed media and advertising (printing houses, publishers, media houses, digital printing houses, advertising agencies, brand owners, etc.)

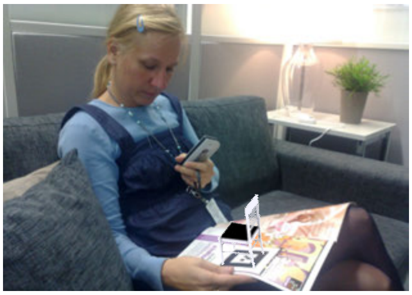


Figure 59. Concept of mobile AR advertising on printed media (image: VTT Augmented Reality team).

In this project, we wanted to clarify how to make it easy to add AR to printed media as an additional component. One thing that came up in many of the interviews was that although it is possible to use natural images for tracking, there is a need for an icon, tag or marker that indicates the existence of the digital content to the user. Otherwise, the user would be unsure which pages are linked to AR content and which are not. Should there already be a tag, there is no reason why it should not be used for initialising tracking as well.

4.4.2 How to speed up marker detection

Performance plays an important role in augmented reality. We have identified some general guidelines to speed up the marker detection and identification process.

- *Markers with frames are easier and faster to detect.* The system can easily detect lines and edges and thus limit e.g. template matching only for a few potential markers, instead of all possible locations and sides in an image. Frames also simplify corner detection and speed up pose calculation and undistortion of markers.

- *Fast acceptance/rejection tests in all phases of the process play a significant role in detection speed*. We went through several of them in the earlier chapter e.g. the quadrilateral test, number of holes, number of edges and marker's size in pixels.
- *It is wise to use as small a data size (number of cells) as possible for required data (e.g. 6 x 6, instead of 30 x 30), to avoid decoding redundant data.* This also minimises the number of source locations that the system needs to undistort. Naturally, the system could also use only part of the data cells, but small data size enables bigger physical size, which makes detection more robust. Thus, the small data size is advantageous.
- *For a large set of markers, decoding ID and data markers is faster than template matching*.

In special cases, an application developer has some deeper knowledge or some prior information for the most probable situations. The developer can use this knowledge to optimise application performance. For example in our pointing interface [11] the user typically pointed at a marker with the camera directly from the front. In addition, our marker system had built-in error correction, and therefore we were able to read pixels using a sampling grid without undistorting it.

4.4.3 How to select a marker type

The best marker type depends on the application. Table 1 gives an overview of system requirements and visual marker types that support them. There might be some other preferences as well; therefore, this table only provides suggestions.

Table 1. Suitable marker types depending on the system requirements: a table of recommendations for appropriate visual marker types.

System requirements	**Suitable marker types**
Application needs to be able to read unforeseen data (for example a URL to download a model).	• data markers
Each marker needs to contain a great amount of data	• template/ID markers with database • data markers
Application needs a large set of markers	• data markers
Visual appearance is an issue	• template markers with suitable design • image markers • invisible markers • watermarking markers • embedded markers to other content by design/new marker type designed (*)
An ID is enough	• image markers • ID markers • (small) data markers
Detection from long distance needed (**)	• template markers with clear design • ID markers with big physical cell size
Environment where visual markers are unwanted	• existing signs (***) • existing images(***) • invisible markers

(*) See Section 4.4.4.

(**) Use marker field approach with appropriate distribution and continuous tracking

(***) If possible, train system to detect existing signs/images as template/image markers

4.4.4 Marker design

Most of the marker types are visually unattractive, mainly because they are designed to support easy detection and thus are black and white, have strong straight edges, etc. Another reason is that engineers rather than professional designers or artists designed most of the markers.

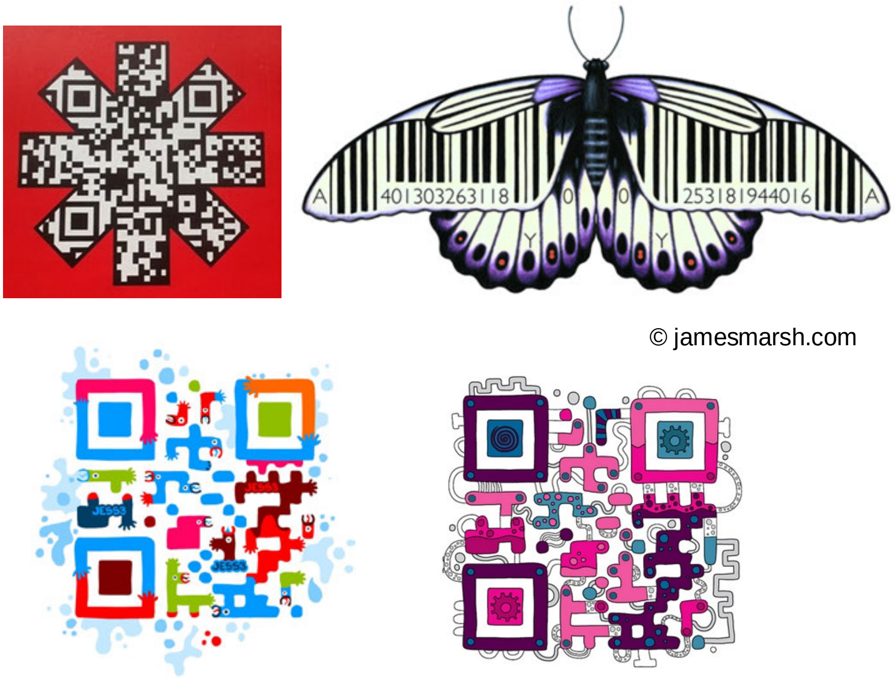


Figure 60. Examples of visual designs: On the upper left: QR code used in Red Hot Chilli Peppers' *'I'm with you'* album campaign. On the upper right: artwork by James Marsh, *'Barcode butterfly'* © jamesmarsh.com (Image courtesy of James Marsh). On the bottom: QR codes designed by JESS3 [134] (images courtesy of JESS3).

If the visual appearance plays a very high role in application, it might be a good idea to spend some effort on professional design (see Figure 60).

4.4.5 General marker detection application

Besides augmented reality, markers are widely used for other purposes as well. Our mobile phone application [3, 11, 17] automatically launched the appropriate application depending on the marker content. If a marker contained a phone number, it launched a phone application. Likewise, it launched a web browser if a marker contained a URL. We also proposed GPS as one type of data: users could get the GPS location from a marker. This would enable location-based AR browsers on mobile devices that are not equipped with GPS and at indoor locations where GPS is inoperative.

In a general AR browser scenario, we would like to attach hyperlinks to objects (cf. Internet of Things). However, the space of all possible objects is far too big to handle. We see several solutions to overcome this issue. For example, one could limit the browser to one type of data at a time. Current location-based browsers have different environments. In practice, the "bookstore application" would recognise the books in the store, the “music application” would detect the CD covers and the “magazine application” would recognise the pages of a magazine.

If the application is able to provide digital information for only some of the items, we should indicate to the user which books, CDs or pages that do have additional digital content. This we can do with a marker. Those products or pages that have additional content would have a marker. The application would recognise the type of data (current environment), retrieve information from internet and visualise the information using augmented reality.

5. Alternative visual tracking methods and hybrid tracking

Until now, we have discussed marker-based tracking and situations where a marker-based tracking system is a good choice. However, markers are undesirable or even impossible in some situations, for example in outdoor tracking, and in these situations the AR system needs to use some other tracking method. What are the other possibilities? One possibility is to use some *sensors* for tracking (e.g. GPS). Another possibility is to use some other *visual tracking* instead of the marker-based method. A visual tracking method deduces the camera's pose from what it sees; therefore, visual tracking is often called *camera(-based) tracking* or *optical tracking*.

Sometimes all tracking methods are insufficient in some aspect and a combination of several methods is beneficial. For example, while a GPS (Global Positioning System) sensor gives the application the global location, it is not able to reveal the orientation of the camera. Whereas a visual tracking method may be able to detect the orientation of the camera, if the system is unaware of its location, it might take an unreasonably long time to examine all possible locations. As a solution, the system may use an approach where GPS indicates the location and visual tracking is then able to deduce the orientation in real-time. This kind of approach that combines several tracking methods is called *hybrid tracking*.

Several visual tracking methods have been developed for virtual environments, motion capture and other purposes. For example, systems using retro-reflective spots and IR cameras with IR LEDs are popular in those areas. In theory, this kind of approach could be used for AR as well. The drawback of such a system, however, is that it requires additional devices, and besides an IR-based system only functions indoors. Therefore in practice, it is not feasible for most AR applications. In addition, we are more interested in visual tracking methods that can use the camera that is already part of the AR system setup and that require no additional equipment. We concentrate on such single camera approaches in this chapter. In addition, we examine how hybrid methods can enhance visual tracking and help in situations that would be too challenging for visual tracking otherwise. We consider hybrid tracking with only those sensors that are commonly integrated into potential AR platforms such as mobile devices.

Visual tracking can be based on detecting salient features in the images; this approach is called *feature-based tracking*. The system may also have a model of the scene or part of the scene and then tries to detect this model from the image and thus deduce the pose of the camera; this approach is *model-based tracking*.

This chapter is organised as follows: Section 5.1 covers general issues in visual tracking, such as the classification of visual tracking methods. Then in Section 5.2, we focus on feature tracking. We recite different features, explain how features are detected and matched and then explain different approaches to create feature maps. Later in Section 5.3 , we discuss model-based tracking and explain how common sensors are used in hybrid tracking. In addition, we give examples of hybrid tracking methods. Two challenging situations in tracking are initialisation and recovery. We consider these situations in the last Section 5.4. We explicate how a system can start tracking fluently and if it gets lost with tracking how it can best recover from the failure.

5.1 Visual tracking in AR

Visual tracking methods can be divided to methods requiring a priori knowledge (e.g. model-based tracking) and ad-hoc methods (e.g. feature tracking).

Ad-hoc methods can be further categorised based on the way they build the environment map into three categories: tracking-only methods, simultaneous localisation and mapping methods (SLAM) and extensible methods. A tracking system can also save an ad-hoc created map and use it the next time as a priori information, so the categorisation is non-exclusive.

Simple marker-based tracking belongs to ad-hoc methods, using a tracking-only, predefined marker field to a-priori methods and dynamic marker field setups to ad-hoc methods. Feature-based tracking methods belong mainly to ad-hoc methods, but they often still need some kind of initialisation for the scale. A feature tracking method can also use a previously learned feature map of the environment and thus belong to the a priori method category.

Visual tracking methods can be classified into three groups:

A priori methods
- model-based tracking
- a priori defined marker field setup
- methods using a priori learned feature map.

Ad-hoc feature tracking methods
- only tracking, no mapping (optical flow)
- simultaneous tracking and mapping (SLAM)
- extensible methods, parallel tracking and mapping (PTAM).

Ad-hoc marker tracking methods
- simple marker tracking
- dynamic marker fields.

Some implementations combine several approaches, and ad-hoc methods may save gathered information and use it next time as a priori knowledge.

5.1.1 Pose calculation in visual tracking methods

The basic idea of pose calculation is the same in all visual tracking methods. The system detects known “things” from the image (marker corners, features, edges, lines, model parts). It knows the real 3D physical relations between them (e.g. size of the marker, relative position of the features, 3D model) or it deduces the relations in the course of tracking.

The system detects these things from the camera image and gets correspondences $(\mathbf{x}_i, \mathbf{X}_i)$, where $\mathbf{x}_\mathrm{i}$ is the location in the image coordinates and $\mathbf{X}_\mathrm{i}$ is the corresponding location in the world coordinates of the ith detected “thing”. It then optimises the camera transformation matrix similarly as explained in Chapter 3 for marker-based tracking e.g. by minimising the reprojection error

$$\underset{\mathbf{T}}{\operatorname{argmin}} \sum_i \left\| \mathbf{T}\mathbf{X}_\mathrm{i} - \mathbf{x}_i \right\|.$$

Typically the number of correspondences is high in feature-based methods. Thus optimisation methods that decrease processing time and required processing capacity play an important role. We discussed such methods in Section 3.3.3. in bundle adjustment context.

5.2 Feature-based tracking

Feature detection and tracking algorithms are widely used for different purposes in computer vision applications. They are applied in motion detection, image matching, tracking, image mosaicing, panorama stitching, 3D modelling and object recognition, for example. In this case, tracking was considered as a means for detecting the relative pose of the camera, but the tracking methods can be applied to other abovementioned purposes as well.

We can divide localised features into three categories: *feature points* (e.g. corners), *feature descriptor*s (e.g. SIFT) and *edges*. A *feature point* (also called an *interest point* or *key point*) is a small area in an image, which has a clear definition and a well-defined position.

Term *feature descriptor* or *image descriptor* refers to the characteristics of an image region or a feature. In the literature, the terms feature and descriptor are often used to refer both to the feature point and to its characteristics.
Edges are often profiles or outlines of objects (e.g. the silhouette of a building), but they also appear in other regions (e.g. change of colour). Edges are matched based on their orientation and profile. In augmented reality applications edge detection and matching is often used in model-based tracking, which we discuss only cursively in Section 5.3.1. Another use of edge detection in AR is occlusion

handling. A system can deduce vanishing points using straight line segments and from them it can calculate camera parameters. In this work, we concentrate feature points and descriptors, and leave edge based tracking without further analysis. Figure 61 presents examples edges, lines and Harris corners.

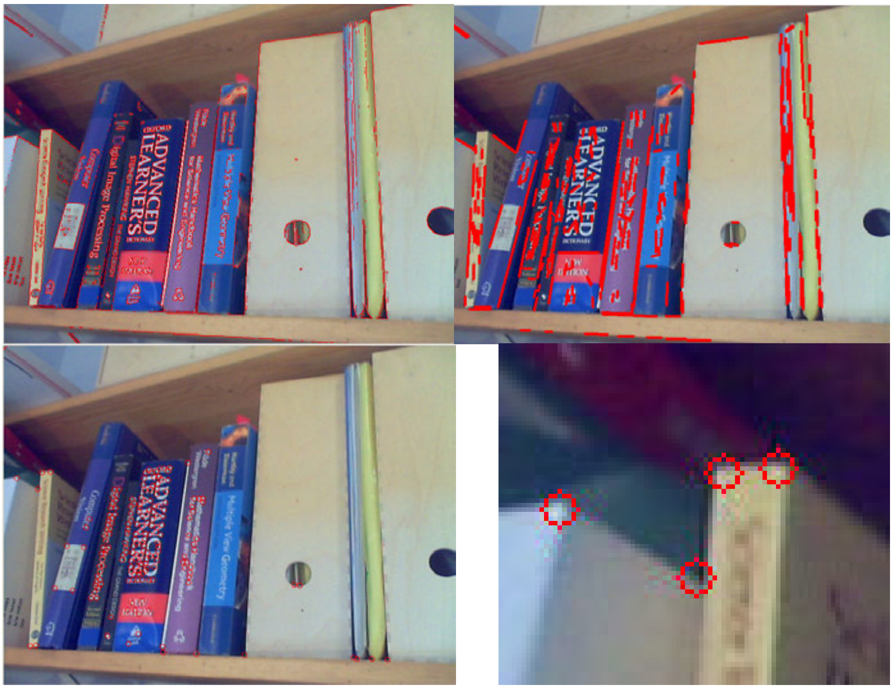


Figure 61. Examples of detected edges (upper left), lines (upper right), and Harris corners (bottom left) are marked with red. At the bottom right is a zoomed part of the image with the detected corners.

In general, a good feature has a clear and unambiguous definition, preferably a mathematical one. It has a well-defined position in image space and the local image structure around the feature is diverse and contains a lot of information. Furthermore, a good feature is invariant under perspective transformation, scale, rotation and translation. In addition, it should be invariant to changes in local and global illumination. For tracking purposes, a good feature should be such that it can be robustly detected at different times (visible in several frames).

Two approaches are commonly used to find feature points and their correspondences:

- Tracking only

  Selecting features that can be locally tracked.

- Detection + matching

  Detecting all features first and then matching them based on their local appearance.

A third approach is to combine these two:

- Detection + local matching / detection + tracking

  Detected features are matched only to locations near their previously detected location. In other words, detected features are tracked.

Tracking is generally faster than matching and therefore it is suitable for video processing. Tracking is only possible if camera movement between frames is small as the search is done in a local window. The tracking approach is suitable for real-time video analysis. If camera movement between images is large (e.g. a set of still images), tracking becomes impossible and matching is a more appropriate approach. This kind of situation is referred to as *wide baseline matching*. Matching is suitable for example in object recognition and image mosaicing. The third approach is a compromise between accuracy and speed. It is also widely used in AR applications.

Feature selection, feature detection and tracking or matching are interconnected and tracking systems often merge them in the same process. A system may also track features in real time and match features on backgrounds.

5.2.1 Feature detection methods

Feature detection methods are difficult to classify because of the large variety of approaches used in feature detection. We may classify them based on what kind features they detect: *edge detectors* (e.g. Canny), *corner detectors* (e.g. Shi&Thomasi), *blob detectors* (e.g. MSER) and *patch detectors* (e.g. [135]). This is, however, a non-exclusive division, e.g. SUSAN is based on both edge and corner detection, LoG detects both corners and blobs, and FASTER classifies image patches to find corner points. Furthermore, methods using feature classifiers can be trained to find any kind of features including different image patches, corners, blobs and edges.

Other possibility is to classify feature detectors based on the approach they use. Rosten et al. [136, 137] use following the categorisation: edge-based corner detectors, grey level derivative methods and direct grey level methods. They divide these further into several subcategories. This is again not an exclusive classification: some detection methods have characteristics from more than one class.

In practice, people are often more interested in tracker performance rather than their mathematical definition or the approach they use. For example, a real time mobile tracking application needs a fast method that uses little memory. The classification could also be based on their performance and/or their suitability for certain purposes: high accuracy offline tracking, the capability to detect certain type of features, memory consumption, processing time, etc. The selected method should be best suited to the task needed. For example, in mobile phones computational efficiency of the methods is essential, and therefore methods such as *Projection Shift Analysis* (PSA) [138] were used in [33, 139], for example.

In the following, we present an overview of feature detection and tracking methods starting from feature points, their detection and optical flow approach and then continuing with feature matching techniques and feature descriptors. At the end of this section, we discuss commonly used approaches for building feature maps (i.e. SLAM and extensible tracking approaches).

5.2.2 Feature points and image patches

Corner points and *blobs* are special types of feature points. A *corner point* is literally some kind of visual corner; it is an image area where two edges intersect. A *blob* or *blob feature* is an image area that is brighter or darker than its surroundings. The basic difference between a corner and blob feature is the scale. If we shrink a blob feature, it becomes sharper and similar to a corner feature. Thus, blob detectors can detect smoother interest points than corner detectors. However, a good blob feature has an exactly defined location, e.g. the centre of gravity or a local maximum. The distinction between corner detectors and blob detectors is vague as well as the distinction between corners and blobs. Blob features have the advantage that they are rotation invariant and thus of special interest. *Blob detectors* use either differential methods based on derivative expressions like *Laplacian of the Gaussian* (LoG) or methods based on local extrema in the intensity landscape such as *Difference of Gaussians* (DoG) In addition, *Determinant of the Hessians (*DoH) and invariant Hu-Moments [140] are used for blob detection.

The Harris corner detector [141] is widely used for feature detection. The Harris detector first identifies vertical and horizontal edges using a Sobel-type edge detector. To make the detection robust against noise, those edges are then blurred and the resulting edges are then combined together to form an energy map. The map contains peaks and valleys. The peaks correspond to corners in the image. Besides the original version of Harris corner detector, there are a variety of variations and derivatives based on it e.g. Harris-Laplace [142], multi-Scale Harris based on wavelets [143] and Harris-Affine [144]. The Harris corner detection method also forms the basis for many of the more complex methods, e.g. KLT tracker.

For example, maximally stable extremal regions (MSER) [145] are used as a method for blob detection. The two important properties of extremal regions are that they are invariant affine transformations (e.g. warping and skew) and they are invariant of the lightning (e.g. sunny vs. cloudy).

SUSAN (Smallest Univalue Segment Assimilating Nucleus) [146, 147] is a well-known corner detection method. The SUSAN algorithm generates a circular mask around a given pixel (nucleus of the mask) in an image. Then it compares the intensity of neighbouring pixels with it. The area with a similar intensity to the nucleus is called the USAN area. The procedure is repeated for each pixel in the image. This way it associates each point within an image with a local area of comparable brightness. The USAN area falls as an edge is approached (reaching a minimum at the exact position of the edge), and near corners it falls further, giving local minima in the USAN area at the exact positions of the image corners. This gives the method its name: SUSAN (Smallest USAN/Smallest Univalue Segment Assimilating Nucleus). The SUSAN algorithm uses no image derivatives, which explains its good performance even when there is noise [147].

Features from Accelerated Segment Test (FAST) [148] and [149] is yet another often used corner detection method. The FAST detector produces very stable features [150]. The FAST detector is available for several platforms including Windows, Linux, MacOS and iPhone [151], and it is widely applied in different applications. It is used for example in [135] for parallel tracking and mapping. In addition to original FAST, several variations have been developed e.g. FAST-ER [137] and FAST-10 [150]. FAST 10 corner detector is used for example in [135]. There it is used in pyramidal implementation to find blob-like clusters in corner regions.

In addition to aforementioned feature point detectors several other methods are used for detection (e.g. *Level curve curvature*, Curvature Scale Space (CSS), Extended CSS (ECSS) [152]) and for matching (e.g. *Normalized Cross Correlation* (NCC) [153]).

5.2.3 Optical flow tracking

The general motion estimation problem is to calculate the motion of each pixel between two consecutive frames. The motion of each pixel that is estimated independently from the movements of the other pixels is called *optical flow* (or *optic flow*). While a general optical flow method estimates the motion of each pixel, a more common approach is to limit the tracking to certain features only. Such methods contain two parts: feature selection and tracking. The tracking part finds the best matching location for each feature.

The *KLT tracker* (Kanade-Lucas-Tomasi tracker) [154] is a well-known and widely used optical flow method for feature detection and tracking. For example, OpenCV contains implementation of this method. KLT tracker forms the basis for many other methods. The optical flow part of the algorithm is often referred to as Lucas-Tomasi and the selection of the good features part as Shi-Tomasi [155]. However, the names are inconsistent; sometimes the names Kanade-Tomasi and Shi-Tomasi-Kanade are used.

5.2.4 Feature matching

After detecting the features, the system needs to *match* them, i.e. it needs to find corresponding features in different images. For feature matching, tracking systems use commonly two different approaches: they compare the small image areas around the features and find similar areas (template matching), or they calculate image characteristics around the features and compare them (descriptor matching). We will discuss both of them here.

Patches or *patch features* are small image areas. In the template matching approach, a feature detector matches image patches against regions in the image. This is also called the *patch matching* approach.

Most of the features are locally planar or can be approximated as locally planar. Naturally, this is an invalid assumption for features located on a sharp tip. Simple metrics (such as SSD and NCC) normally used for template matching assume pixel-wise correspondence. Template matching is the appropriate approach if the motion of the feature is mainly translational, which is the case for stereo pair images, for example.

In general, the appearance of a patch depends on the viewing angle. Therefore, many patch matching algorithms warp either the patch or image with affine mapping to simulate several viewpoints. Image patches are used in [156] and in [135] for parallel tracking and mapping. Template markers and image markers can be considered image patches, with the distinction that marker detection is a separate process and matching is used only for identification of the marker. Some researchers have treated feature or patch matching as a classification problem and used classifiers to feature detection, e.g. [112, 113, 157].

The appearance of the image patches around the feature normally deforms depending on the camera pose. The size varies and is warped due to the perspective view, it rotates, etc. Although, it is possible to compensate these to some extent by manipulating the patches before the matching, the matching often remains inaccurate. In addition, it requires a lot of processing time to compare all possible deformations.

To overcome these problems, the second matching approach is used. The system calculates image characteristics around the feature that describe it appearance distinctively. These calculated characteristics are called *feature descriptors*. A good feature descriptor is invariant under image deformations, it is scale and illumination invariant and is capable of distinguishing between different features. Next, we describe a couple of commonly used feature descriptors.

Scale-invariant feature transform (SIFT) is a widely used feature detection and tracking algorithm [158] and [159]. SIFT is based on feature descriptors. The SIFT algorithm computes a histogram of local oriented gradients around the interest point and stores the bins in a 128-dimensional vector (eight orientation bins for each of the 4×4 location bins). Researchers have proposed various enhancements to the basic SIFT algorithm.

PCA-SIFT [160] is a variation of a SIFT algorithm which is also based on the salient aspects of the image gradient in the feature point's neighbourhood. PCA-SIFT applies *principal components analysis* (PCA) to the normalised gradient patch image instead of using SIFT's smoothed weighted histogram. PCA-SIFT yields a 36-dimensional descriptor which is faster for matching, but has proved to be less distinctive than SIFT in the performance evaluation test [161].

The *Gradient Location and Orientation Histogram* (GLOH) is also a SIFT-like descriptor that considers more spatial regions for the histograms. GLOH uses principal components analysis like PCA-SIFT, but yields to a 64-dimensional descriptor [161].

Speeded Up Robust Features (SURF) is a scale and rotation-invariant feature point detector and descriptor for image matching and object recognition [162]. SURF is based on sums of 2D Haar wavelet responses and makes efficient use of integral images. As basic image features, it uses a Haar wavelet approximation of the determinant of Hessian blob detector. The standard version of SURF is faster than SIFT and more robust against different image transformations than SIFT [162].

A *Local Energy-based Shape Histogram* (LESH) is a robust front-end pose classification and estimation procedure originally developed for face recognition [163, 164]. It is a scale-invariant image descriptor, which can be used to get a description of the underlying shape. LESH features suit a variety of applications such as shape-based image retrieval, object detection, pose estimation, etc. LESH is based on a local energy model of feature perception. LESH accumulates the local energy of the underlying signal along several filter orientations, and several local histograms from different parts of the image patch are generated and concatenated together into a 128-dimensional compact spatial histogram.

5.2.5 Performance evaluation of feature descriptors

The evaluation of the feature detectors is difficult because of the large variety of the methods used for feature detection. Furthermore, feature detection is used for different purposes and therefore performance is evaluated in terms of some of the needs (e.g. location accuracy or speed). In addition, evaluations are usually made using only one implementation of each algorithm or one version (the *basic* version in many cases). However, many algorithms are subsequently modified after invention or fine-tuned for special purposes. The evaluations do normally not cover these numerous new variants. Nonetheless, performance evaluations give guidelines for performance and variations between different methods.

Mikolajczyk and Schmid have completed an extensive performance evaluation of feature descriptors [161], where they continue the work of the first evaluation [165]. The test was designed to evaluate the ability of matching and recognising the same object or scene. The evaluation demonstrates that region-based descriptors (SIFT, PCA-SIFT and GLOH) are best suited to feature matching for their robustness and distinctiveness compared to point-wise descriptors. Moreover, GLOH outperforms other methods in most of the tests. The implementations were not optimised regarding processing time; therefore processing time was not evaluated.

In a more recent publication [136, 137], describing the FAST-ER method, the speed and repeatability of several feature detection and tracking methods are compared against FAST and FAST-ER. In the evaluation FAST is generally much faster than the other methods tested (SUSAN, Harris, Shi-Tomasi and DoG). Furthermore, the learned version of FAST is approximately twice as fast as the traditional version of FAST. According to this evaluation, FAST and FAST-ER were the only ones suitable for real-time processing on a 3.0 GHz Pentium 4-D.

The evaluation [152] showed that the *enhanced curvature scale-space* (ECSS) method outperformed the Kitchen and Rosenfels [166], Plessey [167, 167] (later known as the Harris corner detector), SUSAN and CSS [168] corner detection methods regarding consistency and accuracy.

However, the abovementioned studies have not evaluated the most recent feature descriptors such as SURF. According to the authors of [162], SURF approximates or even outperforms several well-known detectors (e.g. DoG, SIFT and GLOH) with respect to repeatability, distinctiveness and robustness, yet it can be computed and compared much faster.

5.2.6 Feature maps

The systems normally build an environment map from detected features. We will next discuss two commonly used approaches for building feature maps.

Extensible tracking methods (PTAM methods)

Techniques where unknown scene elements are added to previously achieved or the initial scene map are called *extensible tracking* methods. For example [135] use the extensible tracking approach. Marker-based hybrid tracking methods are mostly extensible methods, where the initial scene map is based on markers and then is extended for instance with feature-based tracking. Extensible methods are also called the PTAM approach (Parallel Tracking And Mapping). Lately, research interest in the parallel tracking and mapping approach has focused on mobile phones [169] and the PTAM tracker has been ported to the iPhone, for example [170].

Simultaneous localisation and mapping (SLAM) methods

Simultaneous localisation and mapping (SLAM) is a family of techniques used to build up a map within an unknown environment while at the same time keeping track of the camera's position. SLAM techniques were originally used in robotics for navigating autonomous vehicles and robots. Finding solutions to the SLAM problem is considered one of the notable achievements of robotics research in the past decades [171].

Different methods for SLAM have been proposed, e.g. EKF-SLAM [90] and FAST-SLAM 2.0 [172] as well as real-time monocular SLAM [173] and [174, 175].

5.3 Hybrid tracking

Hybrid tracking means that the system combines two or more tracking methods. In this section, we shortly discuss model-based tracking and sensor tracking methods and then give examples of hybrid tracking methods for AR.

5.3.1 Model-based tracking

Model-based tracking in an AR context is a system that has a 3D model of the scene or part of the scene, detects correspondences with the model from the environment and then deduces the camera pose based on these correspondences. The model can be in several different formats. It can be a 3D graphical object, wire frame object or a 3D reconstructed object based on laser scanning, depth maps or visual reconstruction.

A common problem with the model is that its visual appearance differs from reality. The colours and textures are different, even very different from reality. The amount of details differs more or less from reality depending on the model format. Due to this, features and feature descriptors differ in reality and in the model and therefore matching cannot be based on feature points.

What are the invariants on which the system can base matching? The shapes of objects and lines remain the same, independent from textures and lighting conditions. Shape detection is used e.g. in object recognition, but it is often time consuming. For this reason, model-based tracking more commonly relies on line detection and matching in AR.

A model can be considered a 3D marker. It can be used in a similar way. If the user has a small object and a 3D model of it, the user can place the object in the scene and start tracking. The model defines the correct scale and pose and can be used to define an appropriate coordinate axis orientation as well. The tracking system can use a model for initialisation or combine model-based tracking with other tracking methods. We will give some examples of these in Sections 5.3.3 and 5.4.

5.3.2 Sensor tracking methods

Visual tracking is the most convenient for applications where a camera is already part of the system or where a camera can easily be added. Dedicated tracking equipment has been developed for other tracking purposes, e.g. gyroscopes, inertial trackers and GPS. Sometimes it is feasible to use some of these instead or in addition to visual tracking in augmented reality applications.

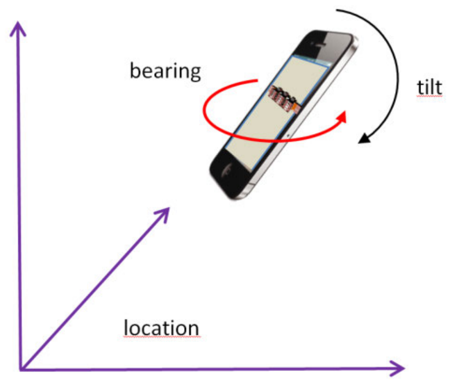


Figure 62. Different sensors define different properties of the object's pose. For example, GPS indicates the object's location, an accelerometer and magnetometer give the tilt (of all three axes) and a compass gives the object's bearing. All these together are enough to define the object's (6 DOF) pose.

The sensor tracking methods are divided into location (3 DOF), orientation (3 DOF) and pose tracking systems (6 DOF). Depending on the sensor type, the coordinates are local or global. A location tracking system returns only the user's 3D location, but does not tell the direction in which the user is looking. An orientation tracking system returns the 3D orientation of the user. Some sensors fix only one direction. For example, an accelerometer gives the direction of acceleration, which equals the direction of gravitation when the device is still. Together with magnetometer, it gives the direction of the gravitation vector even in movement that is the tilt of all three axes. A compass gives the object's bearing. Pose tracking systems usually combine different tracking methods to achieve a full 6 DOF pose (see Figure 62).

Global Position System (GPS) returns the location in earth coordinates (latitude, longitude and altitude). The GPS position is too inaccurate for some purposes and in this case some enhancement method can be used, e.g. *Differential GPS* (DGPS) or *Assisted Global Positioning System* (AGPS). On the other hand, the GPS errors correlate with nearby locations. Therefore, the relative error between the two nearby sensors (e.g. object and observer) is smaller [92].

GPS functions only outside when enough satellites are visible. The usage of GPS can be extended to cover indoor locations with *pseudolites*. Other possible positioning systems for getting a user's location are *infrared beacons* [176], *Bluetooth* and *WLAN positioning*, which give local position coordinates. *Inertial trackers*, gyroscopes and *compasses* give global/earth 3D orientation coordinates, whereas magnetic sensors return local coordinates. Visual tracking methods are able to solve local pose. In indoor tracking applications, local coordinates are often convenient, but outdoor tracking applications often utilise the global coordinate system.
Hybrid tracking methods combine visual and sensor tracking methods and are able to solve both the local and global pose.

5.3.3 Examples of hybrid tracking

The idea of combining a visual tracking system and inertial sensors is not new in augmented reality. An early example is InterSense's [109] hybrid system, which uses an inertial tracker together with the vision-based system. In this tracking system, the relative position and pose of the inertial tracker and camera is fixed. Using the information from the inertial tracker, the system predicts the position of the markers in the view and thus limits the search window that speeds up the image analysis part. Other examples of using magnetic and gyro sensors to stabilise the tracking systems are [177, 178, 179].

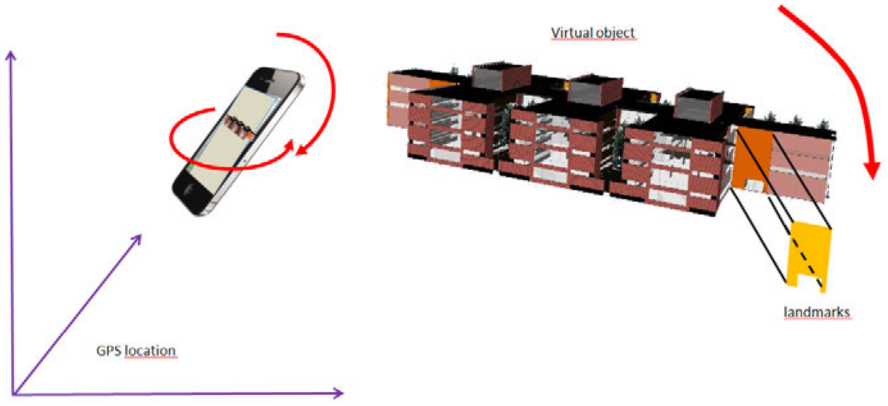


Figure 63. Principle of a hybrid tracking system using GPS and visual tracking: the location as well as the relative pose of the virtual object is known. The user needs to rotate and tilt the camera until the landmarks match.

At present the processing capacity and memory of mobile devices are still too limited for advanced visual tracking methods. However, many models have built-in additional sensors (e.g. gyroscopes and GPS) and people use them with visual tracking in many mobile augmented reality applications.

Marker tracking systems often combine it together with feature-based tracking to benefit the advantages of both tracking methods.

GPS positioning is insufficient alone for augmented reality purposes, therefore it is often combined with a visual tracking method as for example in [180]. The basic idea in these methods is that GPS gives the overall position (the location on earth) and this information is used for initialising the visual tracking system, which then in turn gives the user's local pose, e.g. view direction. We used visual tracking and GPS for outdoor building visualisation in [4]. The user placed virtual models on Google Earth. Later the system retrieved and visualised them based on the user's GPS location.

A system that combines GPS with visual tracking knows the location of the user and is able to augment the correct scene. However, the orientation of the device is unknown, and therefore the user needs to rotate to correct the bearing and tilt the camera to correct the pose. Commonly in this kind of application, the user rotates the camera until fixed landmarks match as for example in our abovementioned outdoor visualisation (see Figure 63). Alternatively, the user can interactively point at landmarks in the image and the system matches the augmentation with them. Thereafter, the system uses a visual tracking system to keep the augmentation in the correct place even if user moves the camera.

Additional sensors can be used for initialising the tracking, e.g. GPS for rough location, and then subsequently the system relies on visual tracking. Additional sensors can also be continuously used for tracking.

In *sensor fusion*, the data from several sensors are combined together to get better information than the sensors are able to provide individually. Traditionally AR applications use sensor fusion to e.g. combine visual information with GPS and inertial sensors for pose calculation [181]. The data fusion is often implemented with a Kalman filter.

Sensor fusion is not the only way to use information from several sensors. Researchers have used inertial sensors to change the way in which computer vision tasks are carried out. For example, the system can use the information from the accelerometer to rectify horizontal and vertical image patches [182] and gyroscope to warp the whole image to support feature tracking.

Accelerometer measure accelerations applied to the device. It is consistent with the gravitation vector only when the device is still. While combining the accelerometer data with gyroscopes it is possible to deduce the gravitation vector even if the device is moving. For example, Apple’s Core Motion Framework (for iPhone) provides the combined gravitation vector. Knowledge of the gravitation direction is beneficial in several ways.

The gravitation vector can be used to align local feature descriptors with the gravity in vertical surfaces. Such gravity-aligned feature descriptors (GAFD) increase descriptor distinctiveness [183].

A feature seems different when seen from a different angle. One solution to detect features or (image patches) from different viewpoints is to warp image patches to produce different viewpoints as discussed in Section 5.2.2. Another possible solution for horizontal surfaces is to use gravity information to rectify the image as proposed in [184] and use gravity-rectified feature descriptors (GRFD).

Both GAFD and GRFD are based on the prior assumption of the surface’s orientation, either horizontal or vertical. We could deduce the orientation of any 3D reconstructed surface without any pre-assumption, as for a marker (cf. Figure 38, page 57).

5.4 Initialisation and recovery

Initialisation of the tracking system is non-trivial; usually the system needs some estimate for the camera pose to start tracking. Without any clues, calculation may require a remarkable amount of time. Once the pose is known, the system can use previous poses to get a new estimate for the next pose.

Visual methods are able to create a map of the environment only up to a scale factor. This means that if a feature-based tracking method starts from a scratch, the system is unable to derive the correct scale for augmentation. It needs some kind of initialisation phase or prior knowledge of environment, or both. Common approaches for initialisation are based on using a marker, a model, image registration (known features) or user interaction.

Tracking may fail in the course of the application. It may happen due to many reasons, e.g. an object moving in front of the camera, sudden changes in lighting or fast camera movements. After a failure, it is important that the system is able to recover from it as smoothly as possible.

Initialisation and recovery are important parts of the tracking system. Both of them benefit from hybrid tracking. Next, we will give examples of handling these situations in AR.

A model of the scene or part of the scene for initialisation as in [185] and [186]. Initialisation can also be based on line correspondences instead of a model. A line based approach is used for example in [187] and in [188], which combines textured 3D models and appearance-based line detection for tracking.

In addition, image registration can be used for initialisation, for example in [188] the key frames are saved and if tracking fails, it can be reinitialised using these images. The other approach is to use a pre-learned feature map. For example, the work presented in [186] uses semi-automatic initialisation by assuming an initial pose for the camera and learned features. It requires the system to always start from the same pose that a user needs to know. In SLAM methods a scene object of known size is commonly used to initialise tracking [189].

These approaches need prior knowledge, which limits the use for predefined locations and prevents ad-hoc installation. Marker-based systems can be used without any prior requirements. Therefore, people use markers or other objects [189] of a known size for initialising a tracking system.

The user can also initialise the tracking system with simple interaction. For example, if GPS gives the location, the user can fix the location by pointing at a landmark, e.g. [34].

Feature-based tracking systems match detected features with a previously learned feature map. However, if the map is large and the user can be at any part map the search time increases far beyond real time. On the other hand, WLAN-based indoor positioning systems are inaccurate. For example, the WLAN positioning system can detect the user's location within a few metres, but not the orientation. However, combining these two techniques gives good results; using the WLAN positioning system for the initial guess and restricting the search area, the feature matching becomes feasible. For example, an AR guidance application in a shopping mall could use kind of hybrid tracking.

6. Enhancing the augmented reality system

In the earlier chapters, we have concentrated on tracking issues, i.e. techniques aligning virtual objects in the correct pose and scale. We have also discussed other technological issues related to tracking, e.g. how to define a convenient coordinate axis and how to decode information from the markers. The methods that we have discussed in previous chapters enable the system to augment correct information (e.g. model associated with a marker) in the correct pose. They define *what to augment* and *where to augment it*.

In this chapter, we discuss the non-tracking related issues of augmented reality. These things define *how to augment.* Augmented reality systems may enhance the human perception in different ways, and the best rendering technique depends on the purpose of the application. We discuss these perception issues and rendering techniques supporting different purposes. In an augmented reality system, virtual and real objects coexist and interact. We will also consider these interaction issues. These are all issues that affect greatly user experience.

This chapter is organised as follows. We first focus on enhancing visual perception and ways to adapt augmentation to increase realism. Then we consider diminished reality, which is used both to improve the visual appearance of the augmented reality system (e.g. hiding the marker) and to interact with real world objects (e.g. removing existing objects). At the end of this chapter, we cover other aspects of interaction between real and virtual objects: occlusions, collisions and ways to handle them in an AR environment.

6.1 Enhancing visual perception

AR systems aim to enhance the human perception in several ways: they steer human attention, aid understanding of 3D space and dimensions or visualise information in the correct environment.

The best visualisation technique depends on the purpose of the application. For example, the system can best catch human attention with *non-photorealistic rendering* (NPR), where the augmentation is bright and distinctive from the background. In contrast, *photorealistic rendering*, where virtual elements are indistinguishable from real ones, enhances visual perception in applications where high quality visualisation is required.

In this section, we explain methods for enhancing visual perception and increasing realism, and situations where these methods should be used. We concentrate on photorealistic rendering, adapting illumination and shadows and adapting to other image effects.

6.1.1 Non-photorealistic rendering

Non-photorealistic rendering (NPR) is a rendering style that is not aiming for realism. In AR, non-photorealism is often used to emphasise augmentation with bright colours and clear borders similar to scientific visualisations in general.

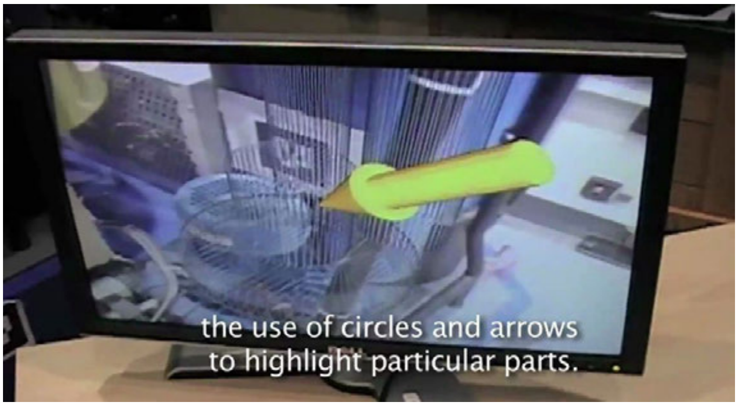


Figure 64. In visualizing a maintenance task, NPR rendering is appropriate: the highlighted arrow draws the maintenance worker's attention to a particular part (image from [15]).

Non-photorealistic rendering is preferable in situations in which augmented objects are used to give instructions and the augmentation is supposed to draw the attention: it allows for more efficient visualization as the user can focus on the information to be conveyed [190]. Examples of such situations are augmented assembly and maintenance support (see Figure 64).

Besides catching human attention, NPR also supports the understanding of 3D shapes and dimensions; it is well suited to maintenance, repair and the visualisation of information from a database. According to our experiences, non-photorealistic rendering of building models gives a good understanding of architectural modelling, even without textures. For example, a few years ago when we visualised the Helsinki Music Centre building project for city authorities and the Helsinki Music Centre planning committee, they gave positive feedback concerning the understanding of the architectural modelling.

Image editing tools commonly have functionalities to create artistic or cartoon-like effects. Some researchers have brought such an effect to real-time AR, e.g. [190, 191]. This kind of NPR supports AR art, for example, and provides a different kind of user experience. Altogether, people may use NPR in any application where photorealism is not required. For example, AR games may use NPR for the animated characters to distinguish them from the real environment. Other such applications are location-based services and AR browsers.

6.1.2 Photorealistic rendering

Many applications aim to embed the virtual objects in the real environment in such a way that the user cannot tell the difference between real and virtual objects. Virtual interior design and virtual dwelling renovation are examples of such applications [18].

In computer graphics, *photorealistic rendering* refers to rendering techniques that produce high quality graphics that look like real photos. In an ideal case, the user is unable to distinguish between real and virtual objects.

Although researchers have developed various techniques for photorealistic rendering in computer graphics for decades, photorealism is difficult to apply to augmented reality; many of those techniques are computationally demanding and suitable only for offline processing (e.g. ray tracing). Furthermore, many known computer vision algorithms for analysing the captured image require more memory than available in mobile platforms (e.g. 3D reconstruction). Therefore, in augmented reality people often face the trade-off between quality and processing time, or between quality and memory consumption.

Several attributes affect how realistically the user perceives the augmentation. We introduce methods that one can apply in AR applications for improving the visual realism regarding these aspects. We ignore computer graphics techniques that are independent from the real environment (such as bump maps and mipmaps). Instead, we concentrate on adapting the rendering to the real environment.

Realistic lighting and shadows affect the user's perception and perceived realism. In the next section, we focus on adapting rendering to those. Sometimes worsening the quality improves the realism; if the captured image is of inferior quality (e.g. noisy), the realism is achieved by degrading the quality of augmentation (i.e. adding noise). We discuss these a bit later in Section 6.1.4.

6.1.3 Illumination and shadows

When an AR application renders virtual models on the top of a picture without any processing, the result is unrealistic; the virtual objects seem to hang in the air and draw attention with unreal brightness (as for example in the left image of Figure 65).

Virtual shadows increase the feeling that the virtual objects are on the ground, whereas virtual objects without shadows seem to hovering in the air. Moreover, adjusting the direction of the virtual lights based on physical light sources increases realism. In the best case, virtual lights imitate the natural lights of the scene in such a way that the virtual objects are illuminated naturally. Figure 65 from our augmented reality interior design application [12] shows an example of how virtual shadows and lights affect the perceived realism.

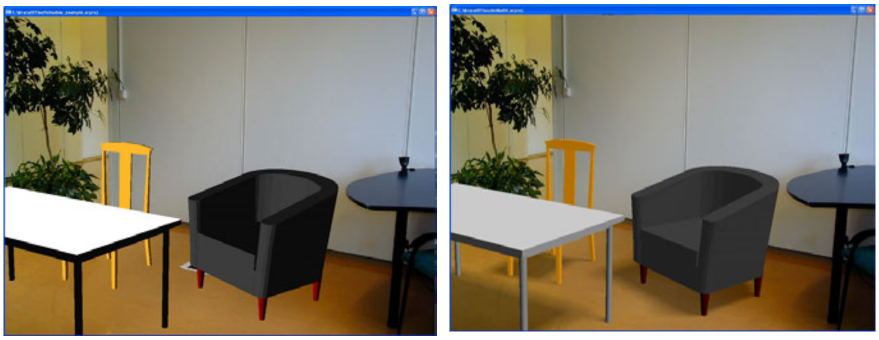


Figure 65. Example of realistic virtual lights and shadows from our interior design application [12]. On the right, the user has manually adjusted the virtual light sources based on real ones and the soft shadows of the virtual objects are similar to real shadows. We have also removed the marker from the resulting image for a better visual appearance.

In our interior design application [12], the user was able to create new light sources and move them around in 3D. A light's location and floor plane was visualised for the user to aid the positioning of virtual lights (see Figure 66). These visualisations were useful and helped the user to understand the 3D locations of the virtual lights.

In practice, users had mutual information about real world light sources (e.g. pendant and windows) and were able to move virtual lights to the same or very close locations. Besides the virtual light sources, the users were able to adjust ambient lighting with sliders. According to our experiences, in many cases adding a default virtual light source on the ceiling already gives nice results.

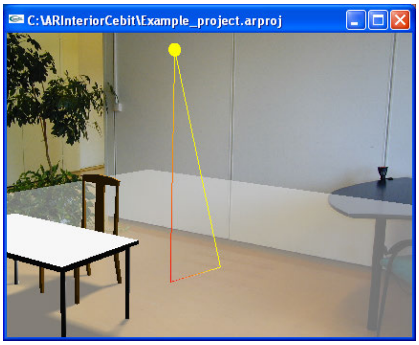


Figure 66. In light source mode, our implementation illustrates the current light source with a yellow sphere and its location is clarified with vectors. A (yellow) vector connects the light source to the coordinate origin, and its projections to the floor plane and perpendicular to the floor plane are also shown for the user. In addition, we visualise the floor plane as a semitransparent plane (in later versions with a grid).

Several approaches can be used for adapting the illumination of the virtual objects to the real environment. For example, the user can adapt virtual lights manually as described earlier. The advantage of this approach is that it does not require any special action or extra devices when taking images.

The adaptation can also be based on measured real lighting. The user can carry out the measurements using special devices such as a photometer or the application can use computer vision methods with prior knowledge. Lighting conditions can be detected using a white object, e.g. a ball. After which, the inverse lighting method can be used to adapt the augmentation on the lighting as proposed in [192]. Figure 67 shows the difference in the rendering result when this measurement is used with the inverse lighting method for adaptation and with basic OpenGL rendering. This approach gives impressive results, but requires special devices and places demands on the hardware.

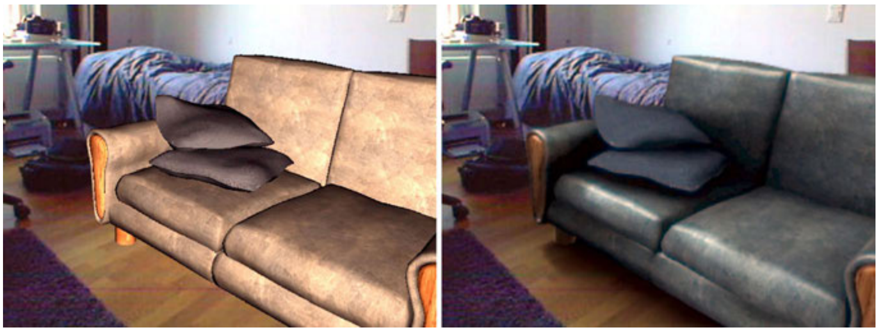


Figure 67. Rendering with OpenGL (on the left) and using adaptive lighting with the inverse lighting model (on the right). Image from [192].

An application can also calculate the lighting model from the image if it has a reference image with neutral illumination. This approach is used for example in [193], where the lighting model of a surface is used to cast shadows over the augmented object and adapt it to the background illumination.

Even if a reference image of the background does not exist, it is possible to detect changes in illumination and adapt augmentation to them as for example in our mobile implementation [14]. In our solution, we first generate a texture for marker hiding and detect the colour and intensity of the reference points. Thus, we can divide the texture into an illumination independent texture and a small resolution illumination texture. Then when we compare the source values of the same reference points, we can update the illumination texture and adapt the real texture with it. We used this to hide a marker in real time with adaptive realistic illumination on a lightweight mobile device, and we will discuss its use in diminished reality more in Section 6.2 Diminished reality. Besides adapting a texture, an AR system could use a similar approach to adapt a virtual object. Furthermore, in marker-based methods, the white parts of the marker could be used to estimate the real lighting conditions in the space, similar as a white ball in the abovementioned work presented in [192].

In addition to these methods, people have used several other approaches to improve the realism of lighting in augmented reality, e.g. tracing [194] as well as detecting light sources and real illumination with a sphere mirror [195, 196].

6.1.4 Motion blur, out-of-focus and other image effects

Rendered objects should have a similar appearance and quality to the captured image to achieve a credible combination of the real and virtual imagery. Thus, besides the illumination, a system needs to adapt the focus and the motion blur of an augmented object to those of the captured image. In addition, an application may adapt rendering to noise, lens distortions and other artefacts appearing in camera images in aspiring to ultimate realism.

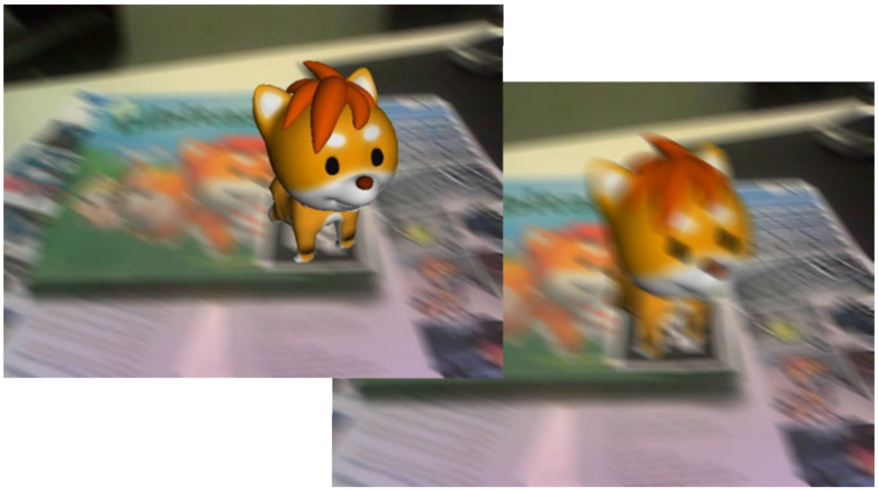


Figure 68. An example of motion blur adaptation: on the left augmentation without blur adaptation, on the right with adaptation.

Traditional rendering methods render augmented objects sharp and in focus. Thus, they produce an unrealistic looking combination of a sharp augmentation and a fuzzy image, in case the captured image is out-of-focus or has motion blur (see Figure 68).

Adapting the augmented image to the artefacts on captured image consists of two parts: detecting the attributes of the captured image and rendering the augmentation with those attributes in mind.

Image processing and computer vision researchers have presented several methods for detecting the motion blur and defocusing. People often model the blur effect in a captured image as a convolution of ideal image and a point spread function (PSF). The motion blur and defocusing usually appear together in the images, therefore it is often convenient to use PSF that expresses both of them as in [80, 197]. In special cases, only defocus blur needs to be taken into account and a different type of PSF is preferable like in the projector-camera system in [198].

On the other hand, computer graphics researchers already presented realistic rendering methods simulating a real camera focus and blur a decade ago [199, 200]. Today animated films use this kind of camera simulation techniques and render motion blur, out-of-focus, noise and other artefacts to create results that are more realistic. However, most of these methods are time consuming and often better suited to offline processing rather than real-time AR.

Nevertheless, researchers have developed suitable methods for detecting and adapting to defocusing and motion blur in augmented reality applications [80, 81, 197, 201, 202]. Once the defocusing or motion blur has been estimated it can be used to correct the detected marker or feature positions as well, which improves the accuracy of estimated camera pose, as demonstrated in [80, 81], for example.

Particularly in small low-cost cameras, e.g. web cameras and embedded cameras in mobile devices, the captured image may have other artefacts such as noise, distortions, chromatic aberrations, layer masking, antialiasing, etc.. Likewise, in the case of lighting and blurring, most methods for detecting and correcting these artefacts typically require some amount of processing capacity and are therefore more appropriate for offline processing and inapplicable in AR applications as such. However, some AR implementations adapt to these artefacts to some extent, e.g. those presented in [201, 202].

6.2 Diminished reality

Augmented reality applications often face a situation where existing objects disturb the augmentation, yet removing them physically away in the real world is impossible. One possible solution is to remove them virtually from the images. Virtual removal of existing objects from reality is called *diminished reality*.

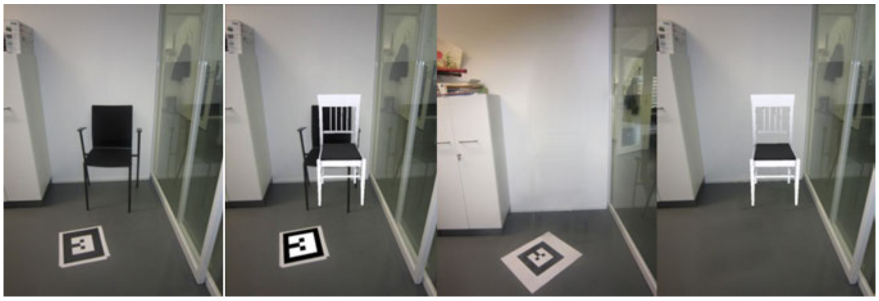


Figure 69. Example of diminished reality, images from left to right: original image, augmentation over existing object, diminished image, augmentation over diminished image.

Diminished reality is in a way an opposite function to augmented reality where virtual objects are added to reality. Strictly speaking, diminished reality and augmented reality are separate sub-areas of mixed reality. However, people often use the term “augmented reality” to refer to applications that have both augmented reality and diminished reality components, as we do in this work, unless we want to emphasise the diminishing functionality.

6.2.1 Image inpainting

In image processing, *image inpainting* means filling in image regions in such a way that the regions merge with the rest of the image and the inpainted regions are as inconspicuous as possible. In the context of augmented reality, people more often use the terms *diminishing*, *object hiding* and *object removal* rather than inpainting, although they all refer to same function.

Traditional examples of situations where image inpainting takes place are correcting damaged images, removing objects from images and filling in missing blocks of transmitted images (aka error concealment). Image inpainting methods have been substantially developed for processing still images and for offline use. The processing time is often counted in minutes like in the methods presented in [203–205] or in tens of seconds like in [206, 207] or at least in several seconds [208, 209]. Although the processing capacity has improved since many of these methods were originally presented, most of the inpainting methods are still unsuitable for real-time applications.

In offline processing, an application can use any image inpainting methods, e.g. some of the abovementioned or other sophisticated methods such as those presented in [210–212].

However, most augmented reality applications require real-time diminishing. Thus, in the rest of this chapter we concentrate on real-time approaches applicable to AR applications.

Diminishing methods belong to two main categories:

- Methods with unknown background.
- Methods using background information.

Traditional inpainting methods belong to the first category; these methods restore areas, where the original content of the image area is unknown or hidden. Methods using background information are similar to inpainting methods used in video processing, where the system may have knowledge of the inpainting area based on previous frames.

Single-camera AR systems seldom use background information, whereas multicamera systems typically have information about the background and use it for generating an inpainting texture. As a whole, methods not relying on the background information are more common in AR.

On the other hand, diminishing methods use two different approaches:

- Methods operating on a 2D image plane.
- Methods relying on 3D structures and information.

Usually methods that do not need any background information operate on a 2D image plane and are very fast. In contrast, methods using background information commonly collect and process knowledge about the 3D structures of the scene, which they then take into account. These methods typically achieve good visual quality but require more processing capacity (both memory and computational power).

6.2.2 Diminishing markers and other planar objects

A visible marker is distracting in AR applications aiming for realistic high-quality visualisations. One solution to achieve natural visualisation is to get rid of the markers completely, and use some markerless tracking method. Another solution is to use invisible markers. A third solution is to use a marker-based system and then remove markers from augmented view. Marker removal or marker hiding is a special case of diminished reality: the object to be diminished is planar and in addition, the system knows the area to be hidden as a result of the marker detection process. In the following, we consider marker hiding and then hiding other planar objects and finally removal of 3D objects in Section 6.2.3.

The simplest way of hiding a marker is to augment a plate or render a picture over it. AR games commonly use this approach. In games, objects are often allowed to stand out. For example, AR Defender game augments a plate in addition to the actual object (tower) over the marker (Figure 70). In this case, the plate serves as foundation of the graphical tower object, and the visual appearance is pleasant, even though it is clear that the virtual object does not belong to the real environment. AR Defender game uses markers such as the one presented in Figure 70, on the left. On the right, one can see it on the table, and in addition the game view on mobile phone screen, where augmentation hides the marker.

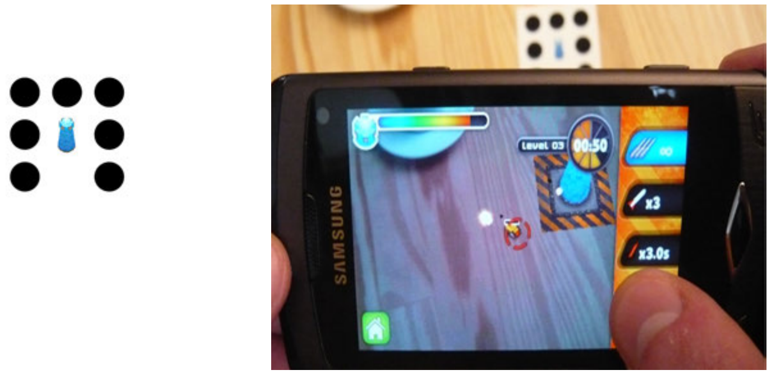


Figure 70. AR Defender game augments a plate in addition to the actual object (tower) over the marker (images courtesy of Int13).

As noted earlier, augmenting a predefined image or plane over a marker separates the augmentation from the background. Therefore, applications that require realistic visualisation cannot use this approach. Often, people want to make the augmentation to look like the virtual object would be on a real surface. For example in garden design, the application should be able to illustrate plants growing from the ground.

The visual appearance is different if the augmentation seems to be directly on the ground, or if the augmentation seems to be on a separate plate, as easily happens with a predefined plate (see Figure 71). The visual appearance is especially poor if the texture or the colour of the plate differs strongly from the background, as on the rightmost image in the Figure 71. Garden plants commonly grow in the middle of grass, and a greenish plate should be a good choice, but in this example it fails to be realistic.

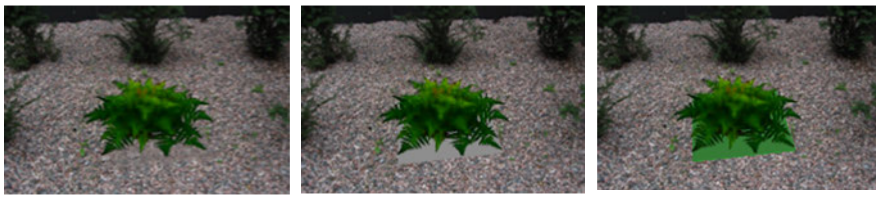


Figure 71. The perception of realism is different depending on the marker hiding method. On the left, the marker is hidden using our marker hiding method [10]. In the middle, a grey colour plate is used, and on the right, a green colour plate is used.

Realistic visualisation usually requires some sort of dynamic adaptation as it is impossible to know all environmental parameters beforehand. Best method depends on the application and its main requirements. In the following, we present several methods for marker hiding and discuss their pros and cons.

Bilinear interpolation is suitable method for hiding planar objects on uniform background (see Figure 72). A simple improvement to bilinear interpolation is to use more values along the edges for interpolation (see left image in Figure 73). Interpolation methods are very fast and simple, and require little memory and processing capacity. Therefore, they are suitable for mobile applications. An interpolation method is also a good choice for an application, where the augmentation largely covers a marker, and where application can allocate little processing capacity for marker hiding due to other tasks. The drawback of simplicity is that, the quality is non-optimal: the hidden area is often distinguishable (as in Figure 72).

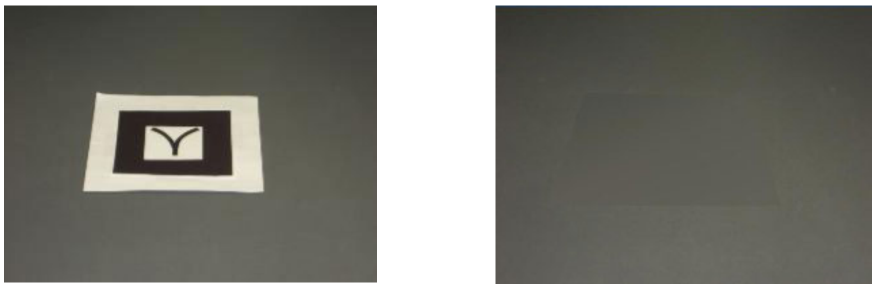


Figure 72. Hiding a marker using bilinear interpolation (with four corner values).

On a textured background, the distinction between the hidden area and the background is clearly visible if the system uses a simple interpolation method (see right image in Figure 73).

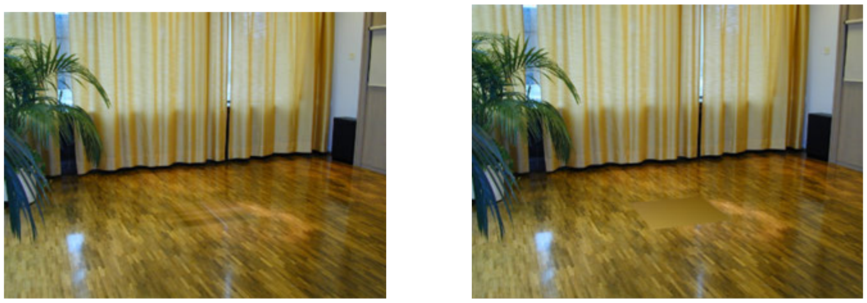


Figure 73. On the left: marker hiding using all edge values for interpolation. On the right: marker hiding with bilinear interpolation using only four corner values.

Traditional inpainting methods require offline processing, and the visual quality of simple interpolation methods is insufficient. Therefore, we proposed a fast texture generation method for AR applications, which combines interpolation and texture generation [10]. We have modified the method slightly later, and we explain the new implementation next.

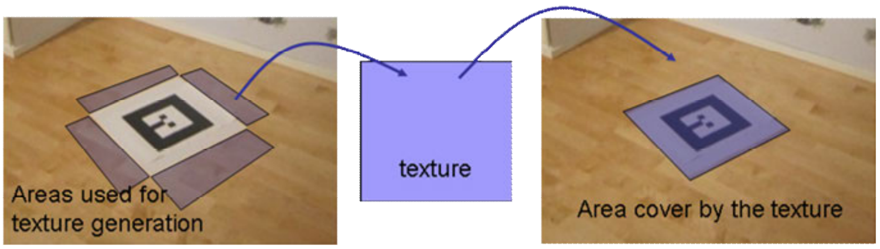


Figure 74. Marker hiding is a special case of inpainting problem: the texture generation and rendering can be done in 3D marker coordinates. On the left: only the area around marker is used for texture generation. On the right: the rendering is also done in 3D coordinates.

Marker hiding is a special case of inpainting problem: the system knows the 3D coordinates of the inpainted area. Our implementation operates in world coordinates (see Figure 74 and Figure 75). This way the texture mimics the real environment even if the diminished area is under strong perspective distortion and yet it is fast. This is an advantage compared to inpainting methods operating on a 2D image plane.

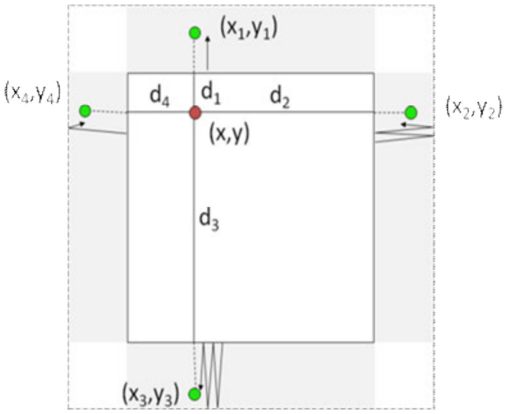


Figure 75. Our texture generation for marker hiding in marker plane. The white square in the middle is the area over which the texture is rendered. Grey areas are source areas used for texture generation. The texture value at location (x,y) is a weighted average of the four source values (x_i, y_i). The location of each source value depends on the corresponding border's distance d_i from (x,y). The source locations oscillate in the border area. The distances d_i are used as weights for the average.

The texture pixel's value is a weighted average of the four source values (see Figure 75). Let the values of the source pixels to be a_i and the corresponding distances from the image border to the target pixel d_i. Thus, the value of target pixels would be

$$a = (d_3/l)a_1 + (d_4/l)a_2 + (d_1/l)a_3 + (d_2/l)a_4,$$

where

$$l = 2(a_1 + a_2) = 2(a_3 + a_4).$$

The source locations are mirrored against the border, but oscillated in a small band around the area.

Our method is suitable for hiding objects from textured background (see Figure 76). This method interpolates values from outside of texture area, but alternates the source locations on a narrow strip. This way it is able to repeat textures and match colours and intensity on borders. The ALVAR library [19] contains a real-time implementation of this marker hiding method.

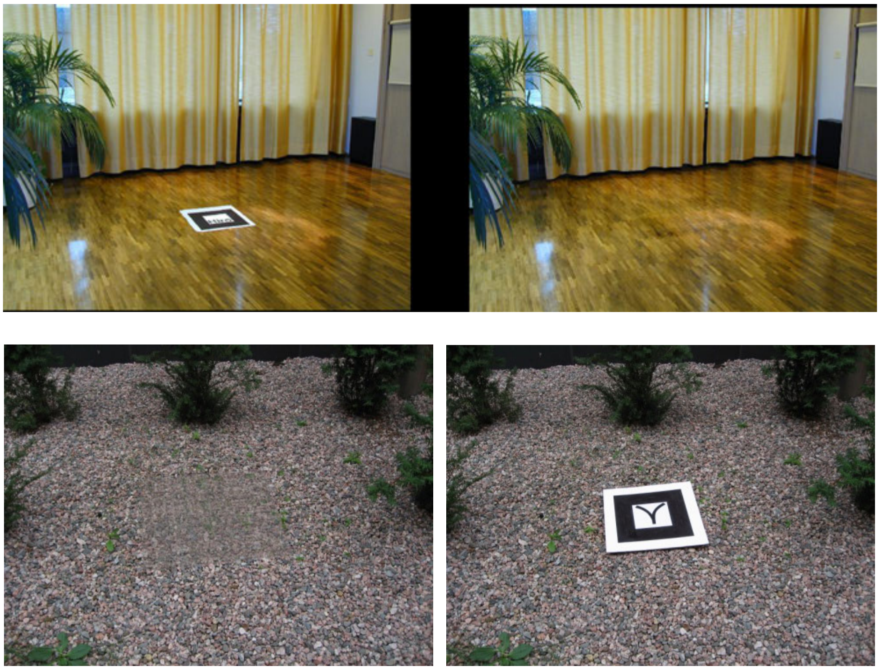


Figure 76. Examples of our simple texture generation method for marker hiding presented in [10].

Our method is a compromise between quality and speed. The processing time is critical in real-time AR application, real-time here meaning 25 fps, which implies that within 40 ms, the application needs to capture an image, detect a marker, hide a marker, render the augmentations and possibly do other things as well. Thus, the application cannot allocate much processing time for marker hiding. Our method takes only few milliseconds to generate a texture, which makes it usable in real-time applications. Even though the diminished area is still noticeable, the visual quality of our method is better than other real-time methods, e.g. interpolation, using a predefined texture or the inpainting Navier-Stokes and Telea methods implemented in the OpenCV library (see Figure 77).

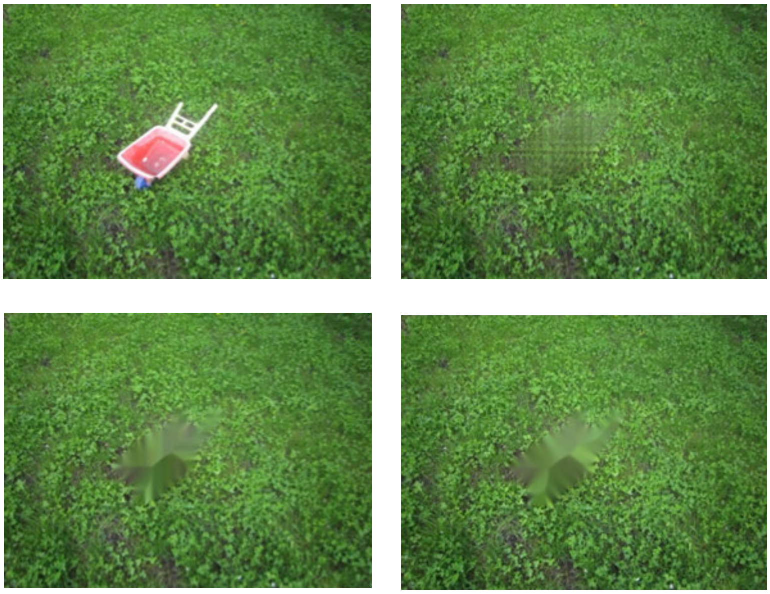


Figure 77. Examples of real-time inpainting methods: Our texture generation method applied to a free-shape object (top-left), and for comparison OpenCV inpainting methods: Navier-Stokes (bottom-left) and Telea (bottom right). Original picture on top-left.

Furthermore, the visual appearance is much better than if the marker is left visible (see Figure 78 and Figure 71). In many cases, part of the marker is behind the augmentation, and the visual quality is completely adequate as in the right image of Figure 78.

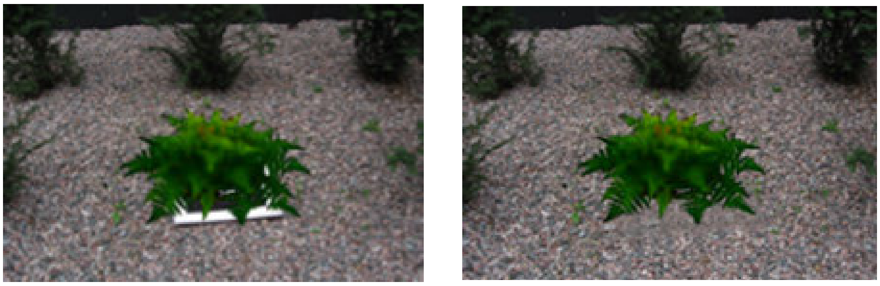


Figure 78. Visible marker is distracting, even if it is only partly visible under the augmentation (on the left), whereas the hidden area blends nicely with the rest of the image (on the right).

The good thing with interpolation methods and with the simple texture generation method mentioned above is that they are quick and simple to implement, and the visual results are often satisfactory and always less eye-catching than showing the outstanding black and white marker. Moreover, these approaches are computationally modest and therefore well suited to mobile environments, where markers are often favoured due to the limited capacity.

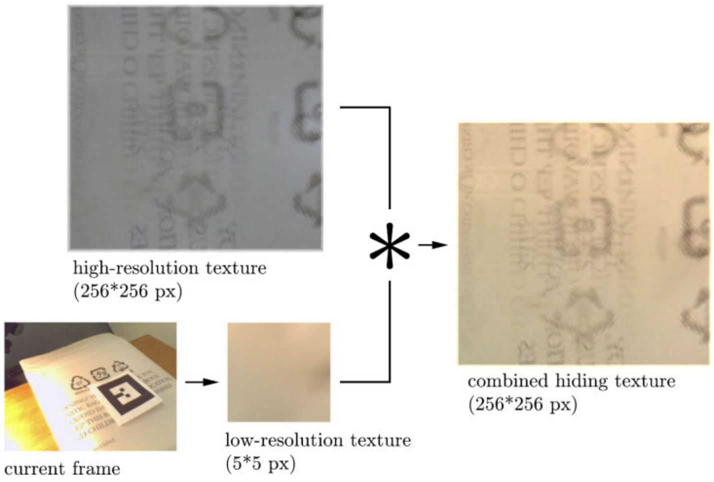


Figure 79. The principle of our fast marker hiding method [14]: At the first frame we create a colour-independent high-resolution texture. Then for each following frame we only update the colour and lighting-dependent low-resolution texture. The final texture is a convolution of these.

In certain mobile applications, even interpolating methods consume too much processing capacity. We presented a solution for this where we generate a high-resolution texture once, and then update only the illumination of the texture for each frame. We use a lower resolution illumination component, which we update for each frame and then convolve it with the texture component [14]. This approach is much faster than updating the whole texture (Figures 79 and 80).

In this case, it is even possible to use a more time-consuming texture generation algorithm as the texture is generated only once. Analogously, an application could use this approach to create a more detailed texture with any sophisticated time-consuming image inpainting method in a PC environment as well.

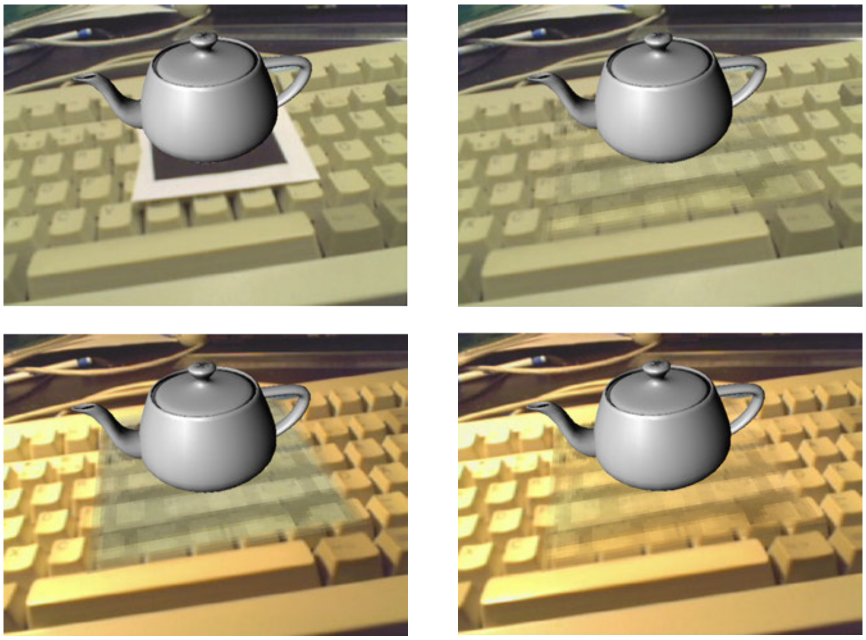


Figure 80. Our fast marker hiding method [14] in action. Top-left: original image with marker. Top-right: marker hidden with a texture created from original image. Bottom-left: for comparison same texture is used without adaptation after lighting conditions change. Bottom-right: Same situation with adaptation.

Another possible approach would be to use a client-server system with remote computing facilities to implement time consuming inpainting. For example, [213] suggest using exemplar-based inpainting method [214] with remote computing farm in wireless mobile or wearable solutions.

Diminishing other planar objects is very similar to marker hiding. The same constraints apply to it and the same approaches and methods can be used as for marker hiding. The only difference is the shape of the inpainting area and the way of defining the inpainting area. For a marker, the marker detection defines the inpainting area, but for other objects, the area is initially unknown. Usually the user needs to indicate the diminished area manually in some way. For example, the user can outline the area with the mouse on a keyframe. For objects lying on the marker plane or on another known plane, the projection of a planar area to the image plane becomes a plane-to-plane homography and the application can calculate it easily for each frame once the area is defined. A practical example where the application needs to remove planar objects on known planes is for instance removing a painting from a wall in an interior design application. On the other hand, if a planar object, such as a flat railing, occurs on an unknown plane or non-existent plane, then the application should treat it like a 3D-object.

Unlike traditional inpainting methods that can operate only on the image plane, AR applications have information of world coordinates, e.g. the marker plane. Thus, they can diminish planar objects on the 3D coordinates and this way take into account the perspective distortion. This yields a better visual result in some situations.

6.2.3 Diminishing 3D objects

Conflicts between virtual and real objects is a challenge that comes up frequently in augmented reality. People seldom use AR applications in an empty space. They often use them in environments that have several real objects, which may then overlap with the augmentation. For example, in interior design the user may want to test a virtual couch on a spot where a physical couch is. Besides, the user may want to see an augmentation from a viewpoint where something comes partly in front of a virtual object (as e.g. backrest of a chair in Figure 82). These kinds of situations where real objects overlap with virtual objects are problematic in augmented reality. One solution is to remove disturbing existing objects virtually. Furthermore, the whole purpose of the application might be to visualise changes in environment, including the possibility to virtually remove existing structures or objects.

The diminished area for hiding an object is object's projection into image plane. Diminishing a 3D object differs from diminishing a planar object in one aspect; for a planar object it is straightforward to calculate its projection into the image plane (it is a plane-to-plane homography). For a 3D object, the projection depends on the shape of the object and viewing direction. In addition, the neighbourhood of the object in image plane changes depending on the 3D structures of the background. Defining the object and the diminished area is an essential issue in generic object hiding.

In the following, we first describe methods defining the diminished area for 3D objects and then discuss further inpainting methods for real-time augmented reality. Later, in Section 6.3 we discuss more handling relations between real and virtual objects.

Multi-camera systems have been used in diminished reality, e.g. [215, 216]. These two approaches, like most multi-camera systems, use background information for texturing the diminished area. However, single-camera systems are more common in AR. Thus we focus in this work on single-camera AR and discuss multi-camera approaches only briefly here.

A typical situation in augmented reality is that the operator uses the application in an unforeseen environment and the items in the environment are unknown. Therefore, a conventional approach is to use user interaction to define the object instead of object recognition methods [217].

A straightforward and commonly used approach is that the user draws a polygon around the object with a mouse. Usually this is done in several key-frames and these are used to build a 3D volume roughly defining the object as proposed in [218]. In diminished reality, it is sufficient that the reconstruction approximates the shape of the object as long as the object is inside the volume.

The user may also indicate the object by drawing a loop around it. Defining a volume based on round-shaped loops is complex as well as calculating projections of free shape. Therefore, free-shape loops are seldom used together with the volume reconstruction approach in real-time AR applications. In contrast, construction of a 3D polygon and calculating projection of a polygon mesh is feasible. However, the user can indicate the object circling it with a mouse if some other approach is used.

For example, the method presented in [219] tracks the object on an image plane after the user has once indicated it. It tracks the boundaries of the objects using active contour algorithm [220]. This approach uses the assumption that the object differs clearly enough from background and that the appearance of the object in successive frames is almost the same, and it may fail if there are strong boundaries in the background. This implementation increases the area of an object in previous frame for following frame, and then the object boundary is again searched with the active contour algorithm. The diminished area is always selected a bit larger than the object to ensure that the object is totally covered.

We propose a new approach as one possible solution: the user can select predefined 3D volumes and cover objects with them. In our implementation, the user can select some predefined volume, e.g. a cube. The volume appears then on the scene as a wireframe object (red cube in Figure 81), which the user can scale and move around in the application and position to cover the desired object. For each frame, the projection of the volume on the image plane is calculated (blue polygon in Figure 81). The projection area is then diminished (right image in Figure 81). Our approach is fast and well-suited to real-time applications.

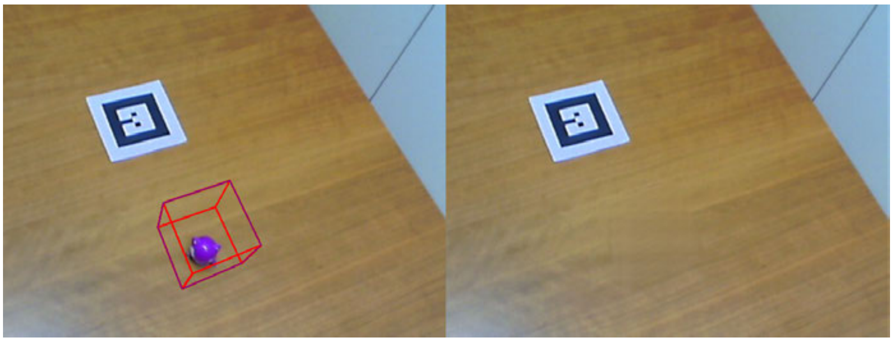


Figure 81. Images of our diminished reality implementation using predefined 3D volume. On the left: red wireframe illustrated cube, blue polygon illustrated the diminished area. On the right: object removed.

Besides the volumes to be removed, an application can add volumes defining “holes” inside the volume to be removed. This allows it to define simple nonconvex volumes and volumes with holes easily. Our current implementation supports partially this functionality. For example, in Figure 82, we diminish the backrest of the chair in such a way that only the parts belonging to actual chair are manipulated and the holes are left untouched.

Figure 82 also illustrates the occlusion problem: a virtual object is rendered on top of a real object (in the middle), and on the right, the same situation with the exception that the chair in front of the image is first removed virtually. The middle image illustrates how the visual illusion is disturbed if a background object is rendered partially in front of foreground object.

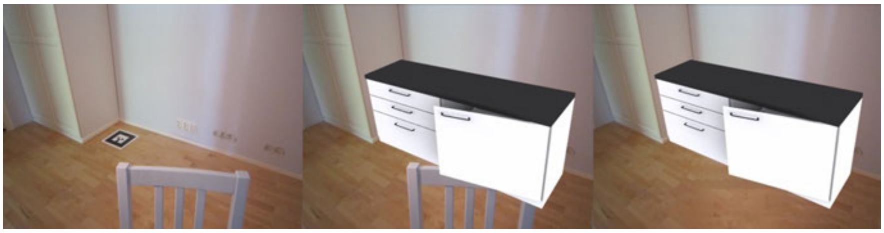


Figure 82. Example of our occlusion handling with diminished reality.

An augmented reality system with haptic feedback is a special case; the system knows the visually disturbing object (the haptic device) beforehand, the pose of the object is known, and users can even influence the appearance of the object. In this kind of situation, special approaches such as optical camouflage are possible. For example, users can paint the device with retro reflective paint, detect it and use a projector to project a background image on top of it as proposed in [221], for example.

A different approach for avoiding visual obtrusion is to define the volume of the haptic device based on the knowledge of its location, its posture and physical shape. This approach is used in [222], for example, where the haptic device is covered with a combination of boxes and a sphere. These volumes are rendered in stencil buffer to form a mask of the diminished area. Then pre-recorded background images with associated camera positions and rough geometric approximation of the background are used for texture generation.

Methods using information of the background structures are usually computationally demanding or require extra equipment (e.g. additional video cameras or depth cameras). Therefore, these methods are unsuitable for lightweight solutions. Besides, they often require some sort of initialisation, which in turn requires more or less expertise. This limits their use in consumer applications.

Furthermore, fast methods (e.g. texture interpolation) usually blend the colours and textures, which creates unwanted blur on structural edges, see the rightmost image in Figure 84, for example.

Yet with a simple modification, we can improve the visual result significantly. If we take the divisions on the background (e.g. the junction between floor and wall) into account, we can divide the diminishing area into sub areas and achieve a more natural result, as can be seen in our example in lower image of Figure 84.

We identify several means for doing this. First, should we have some prior knowledge of the environment and the 3D-structure we could use that. For example in interior design, the floor plan indicates the boundaries of the room, i.e. the location of the floor-wall intersection. Secondly, we can use image processing and computer vision algorithms to detect the lines between image segments and use them. Figure 83 shows an example of our automatic line detection implementation. Our real-time method finds dominant lines in the area around the diminished area, finds their counterparts from the opposite boundary and interpolates textures on these directions.

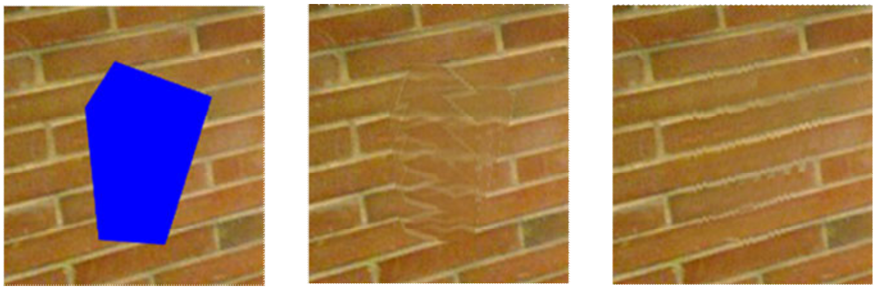


Figure 83. An example of automatic line detection. On the left: the blue area shows the area to be diminished. In the middle: interpolation without line detection. On the right: our texture interpolation method with line detection.

A third possibility is to use simple user interaction to define these kinds of lines, as was the case in our example shown in Figure 84. Our application also supports predefined edge locations as well.

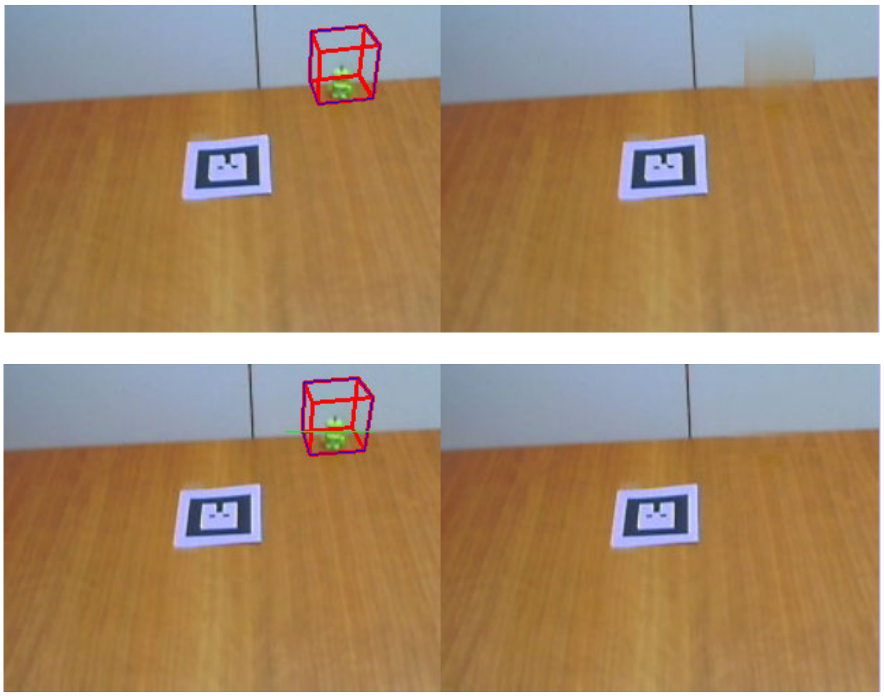


Figure 84. Lower images: our fast volume hiding, which takes 3D structures into account. Upper images: same method without taking 3D structures into account.

Should the object be moving, the system should track the movements of the diminished object. Only in special cases, as e.g. in the abovementioned case of haptic device, the system knows the movement. Object tracking can rely on motion detection and tracking, image segmentation, feature detection and tracking or any other real-time object tracking method. Our abovementioned object hiding using a 3D volume can be combined with feature tracking to remove moving objects.

6.3 Relation with the real world

Besides the overall relation with the real world coordinate system defined by pose and scale, the virtual objects have relations with individual real world objects as well. Virtual objects interact with real objects: they may collide, occlude or overlap each other. In this section, we discuss how to detect and handle these occurrences adequately in AR applications.

6.3.1 Occlusion handling

Occlusion means a situation where part of a scene is invisible because something is in front of it. In the context of augmented reality, this means that something is between the camera and the 3D location of virtual elements. In practice, an AR application needs to pay special attention to situations where a real object occludes an augmented one and it needs to augment something behind an existing object.

The main alternatives to handle occlusions are:

- Foreground masking: the occluding object is masked and only visible parts of the virtual object are augmented.
- Diminished reality: the occluding object is removed virtually and the whole virtual object is augmented.
- (Simple) transparency: occluding or occluded objects are rendered as transparent.
- X-ray vision: the real environment and the augmentation are blended in a manner that creates an illusion of seeing through or inside a real object.

We already covered ways to remove objects in Section 6.2 Diminished reality. In the following, we will first discuss how an AR system can detect occlusion and methods for masking occluding objects, and then we will address transparency and X-ray vision.

Either a system can rely on user interaction similarly as discussed earlier in the context of a diminished object or it can detect occlusion automatically. The best way for detecting occluding objects depends on the situation. Therefore, researchers have applied a number of different approaches. For example if the camera is relatively static and there are moving objects between the user and augmented object, the system can detect the occluding moving objects based on the motion and background detection methods as in [223], for example.

The method presented in [224] segments images into foreground and background using depth estimation and motion detection with a spherical camera. The method divides foreground objects into actual objects and shadows. Objects in front of augmentation are then masked, and virtual objects are rendered behind them. Furthermore, a virtual shadow caster is used to cast shadows of real foreground objects on virtual ones.

A new trend in augmented reality is to use an additional time-of-flight (TOF) cameras, which create distance maps. This approach is well-suited to detecting occluding objects from the foreground. A couple of hardware solutions have integrated combination of TOF and video cameras; a well-known example is Microsoft Kinect. The first AR game for the Kinect Xbox 360 was Fantastic Pets by THQ, released in March 2011. The Kinect is predicted to boost the number of AR games.

Figure 85 shows an example of using the Kinect for diminishing foreground objects. In the Kinect, the TOF camera and RGB camera are side by side and therefore their view is slightly different, and the mask produced using the depth information is inaccurate for masking out foreground objects (see bottom-left image). However, we can easily use it for diminishing purposes. In this example, we dilated the mask (small images in top-left image) to ensure that it covers the whole object. We used our real-time texture interpolation for diminishing foreground object defined by the dilated mask (small image in top-right image), and then we the augmented a dresser (bottom-right image).

We noticed in our early work with virtual advertising that for masking foreground objects, the difference in view angles (TV camera + IR camera) causes such a difference in mask-image correspondence that the result is unpleasant for the viewer, even if the object is far away and difference is small.

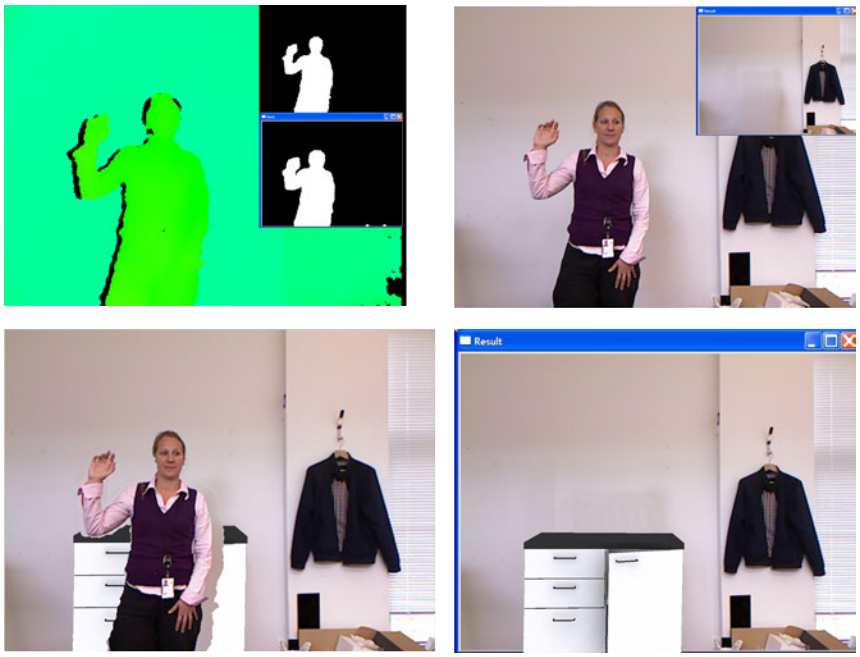


Figure 85. Example of using the Kinect for diminishing a foreground object. The image at the top-left is the false-colour depth map from the Kinect TOF camera, its thresholded mask image and dilated mask. The top-right image is the colour camera image and its diminished version. The bottom-left mask is used to mark out foreground object, and a dresser is augmented behind it. The bottom-left image shows augmentation on a diminished image.

Although the TOF camera enables new functionalities for AR, the use of TOF cameras is unfeasible in most AR applications in practice, as it requires a special device, which restricts its use, especially in mobile set-ups. The same holds for stereo and multi-camera systems. They are well-suited to detecting foreground objects, but complicate the overall system setup.

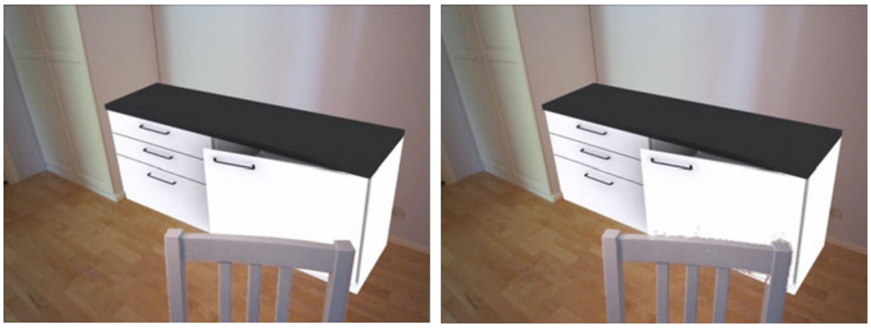


Figure 86. Example of masking the foreground object and augmenting behind it. On the left: accurate mask, and on the right: inaccurate mask. Compare the images with diminished reality in Figure 82.

A mask used for diminished reality can be approximate as long as it totally covers the diminished object [225], whereas a mask used to define a foreground object needs to be accurate. If a foreground mask is inaccurate, defects are visible in final image as on the right image in the Figure 86. On the left image we use user defined correct mask for comparison.

Although several solutions have been demonstrated for occlusion handling in augmented reality, it is still problematic. The existing solutions require additional or special devices which limits their use. As for simple equipment systems, an AR application developer needs to find a compromise between quality and speed. Most of the methods that researchers have presented for foreground masking are inapplicable in real-time implementations, for example [224] performs 2–5 fps, [193] needs 6–7 minutes for calculations, and part of the calculations in [223] require offline processing etc.

If foreground occluding objects are contextually unimportant, the application can use diminished reality for handling occlusion. This can be done in real time. For example our software implementation for ad-hoc texture generation and object removal (presented in Figure 81, Figure 82 and Figure 84) performs at 30–50 fps (Dell latitude D620 Laptop, Intel Core2 Duo processor), even though the code is not optimised. Moreover, we have the confidence that performance can be increased significantly using an optimised code and the graphics processing unit (GPU) for texture generation.

Sometimes an AR application needs to visualise something inside or behind a real object in its real context. This kind of augmentation is called X-ray vision. X-ray vision in an effective and useful method for outdoor inspection and maintenance tasks of underground infrastructure such as voltage bands and gas pipes [226, 227].

Simple transparency means that occluding objects are made transparent. It gives a hint of occlusion but the order of objects is confusing [228, 229]. In X-ray vision, the application provides additional visual cues to create a visually correct spatial relationship. Simple transparency is used in low capacity mobile applications, for example.

In X-ray vision the augmented results consist of actual visualised information (aka focus) and its spatial context. Therefore, this problem is identified as Focus and Context (F+C) rendering [230].

The perception of distance and depth order of objects is improved with tunnel frustum cut-out and rendering edges of the occluding object [231, 232]. However, users tend to underestimate the distance to the occluded in X-ray vision even with depth cues [233]. Besides edges, other salient features such as hue, luminosity and motion are maintained to provide richer content for the occluding object [234].

In addition to the abovementioned application areas, X-ray vision AR is appropriate for visualising medical imaging on a patient. It is proposed for example for laparoscopic surgery applications, where computer tomography images are visualised on patients as in [50].

In navigation and location-based services type of applications, a system needs to visualise points of interest (POI) for a user. This is quite straightforward if the POI is visible in the image; the system then just needs to highlight it. Should the POI be outsize the field of view or occluded, some other approach is needed.

For POIs outside of field of view, distortions (e.g. radial distortion) are used, for example, to enlarge the field of view [235] or virtual pop-ups [236] and virtual radar [237].

For occluded POIs, a system can use X-ray vision, diminished reality, transparent highlights, or any other focus and content visualisation approach. In addition to these, a paper [235] proposes an approach suitable for navigational purposes. The approach that they call Melt is in a way an intermediate form of diminished reality and X-ray vision. The basic idea is that the occluding part of the video image is distorted in a way as if the distorting object would have melted on a ground plane, which creates space for augmenting the object of interest behind the occluding one. In addition to the ability to see behind objects, the Melt application provides a functionality where a user can use virtual zoom on objects of interest while the rest of the image remains as it was before.

6.3.2 Collisions and shadows

Collision detection is routine in computer graphics and virtual reality. The bounding volumes of objects are checked against each other, and the application prevents objects from overlapping. Collision detection between a real and a virtual object in augmented reality is more complex than between virtual ones. The application does not know the boundaries of real objects. Instead, the system needs first to use some 3D reconstruction or computer vision method to detect and analyse the real object. This is challenging and time consuming, and therefore AR applications seldom have any mechanism for collision prevention and users may for example move virtual objects through real ones.

Occlusion detection methods, which we discussed earlier, are usually limited to detecting the order of the objects' facades. This information is insufficient for collision detection.

Augmented reality systems that are capable of detecting and preventing collisions with static environment often require offline processing for 3D reconstruction. It is also possible to use a model-based approach as proposed in [238]. The model-based approach is able to handle both occlusion (masking) and collisions. However, it requires a priori information of the environment (models) and registration technique to align the model and the corresponding real object.

Systems that are able to detect and prevent collisions with moving physical objects typically have complex hardware systems (e.g. multi-camera systems [239], TOF cameras or a complex stationary setup [240]). To our best knowledge, there is no lightweight AR system capable of handling collisions. However, there are application areas where collision prevention is an important issue and there is a clear need for development and further research in this area.

Preventing physical collisions is a big issue in robot operating, which AR can help in programming collision free robot movements. For instance, an interactive AR system is used for planning collision free paths for maintenance robots [241].

Interior design is another area where collision detection would enhance the AR system. In our ongoing work [16] we interviewed professional interior designers, consumers and interior design bloggers among other players in relation to AR interior design. Our interviewees considered the lack of collision prevention in current AR interior design applications as one of the bottlenecks preventing serious or professional use of them.

A typical question in interior design is whether a specific piece of furniture will fit in the space available. However, current VR/AR systems let the user move virtual furniture through physical walls and furniture. Interior design plans are supposed to illustrate by default how furniture will fit in a room. Therefore, to be able to use the application for making an interior design plan, the application should ensure that objects do not overlap, and that the user is not able to move them through the wall accidentally. Naturally, the user can visualise furniture with the current systems and use them for daydreaming, as our interviewees mentioned.

Marker-based AR is able to get the floor plane coordinates from a marker position and thus keep objects on ground level. In straightforward manner, markers can be used to mark walls and prevent collision with them, i.e. to keep furniture inside a room. In our earlier work [18], we learned that automatic detection of walls or simple user interaction to point out walls is an essential function in a renovation and interior design application. Once the application detects the walls, it can also change the wall coatings virtually.

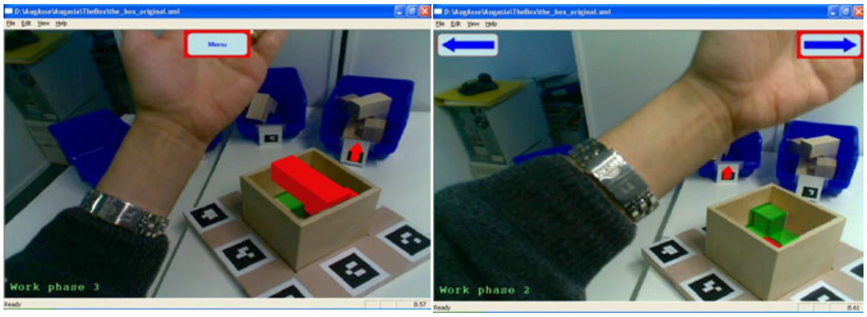


Figure 87. Our gesture detection user interface in action [7].

In some games and applications using gesture interface, collision detection reduces to 2D. In this kind of selection, it is sufficient to detect when objects (e.g. a hand and an icon) overlap on an image plane. For example, in our augmented assembly demonstration with multimodal user interface [7], the gesture recognition works this way. The user selects a menu item by moving a hand on desired icon, then forward and backward icons appear, and the user selects the desired action (see Figure 87). In some games the user may hit virtual objects, which then bounce from the player's hand. This kind of collision detection and prevention can be implemented for example using the 2D sphere-to-sphere collision estimation presented in [242].

Another approach for collision prevention is marking collision-free areas. Typically, this is done by sweeping collision-free areas with a marker (as in [241]) or other detectable object. This normally requires user interaction and is usually done in separate initialisation phase.

A simple robot equipped with a camera could be used for creating a 3D model of a space. Modern robot-hoovers such as Samsung VCR8845 have an integrated camera and are capable for 3D reconstruction of the free space. Some robot-hoover manufacturers (e.g. iRobot for Roomba) provide an Application Programming Interface (API) for making individual enhancements. This indicates that in the near future, feasible (i.e. cheap and easy to use) "scanning robots" will be available for defining collision-free areas and other 3D construction tasks, for example.

In Section 6.1.3, we considered shadows that virtual objects cast on real environment. However, to perceive seamless integration of the virtual and real world a system should also consider real shadows on virtual objects. Couple of research papers have addresses this issue and proposed methods for recasting real shadows on virtual surfaces.

For example, [193] uses an approach similar to template matching for detecting predefined surfaces and its lighting conditions in the presence of partial occlusion. The surface texture is then replaced with a virtual one. This paper proposes a visibility map based on the work of [243], but with the difference of using very local similarity measures instead of global ones. Thus, the method is able to distinguish between occluding objects and shadows. Occluding objects are then masked to foreground and shadows are recasted on top of augmentation.

As mentioned earlier, the method presented in [224] recasts shadows of real foreground objects on virtual ones. This paper uses simple spherical geometry together with spherical vision camera for estimating the foreground depth, height and directional light (shadows are caused by the sun), which gives enough information to recast shadows on virtual objects.

7. Practical experiences in AR development

Until now, we have discussed technical and visual perception aspects affecting usability: the accuracy and speed of the camera tracking, interaction functionalities and different rendering paradigms and adaptive rendering, etc. However, a good technical implementation and effective algorithms are not the only conditions required for a successful AR application. In the end, the popularity of an application depends on its perceived usefulness and user experience. Several factors affect the usability and user experience of an AR application. In this chapter, we share our experiences of those things.

One of the key factors affecting the successfulness is the user interface (UI). We start this chapter by telling about our observations related to UIs. A high-end tracking algorithm is worth nothing if the user finds the application difficult to use, or if the use of application is inconvenient or even unsafe. People tend to concentrate on application and the user may collide with physical objects and harm oneself or equipment. Therefore, a good design system prevents unwanted physical collisions. We will discuss this kind of physical safety issue later in this chapter. In addition, we say few words about head-mounted displays, and their effect on user experience. At the end of this chapter, we discuss the importance of authoring for AR.

7.1 User interfaces

Augmented reality is by its nature well-suited to novel and natural user interfaces, e.g. gesture-based and tangible interfaces. On the other hand, mobile AR applications benefit from pointing and motion interfaces. Our publication [3] provides a more detailed survey on user interaction and user interfaces for mobile devices for those more interested in the subject. Next, we report some of our experiences regarding user interfaces in AR. We cover pointing and motion user interfaces, multimodal interfaces and feedback from the system.

Pointing user interfaces

The potential of smartphones to become the default physical user interface for ubiquitous mobile multimedia applications has already been realised a couple of years ago [244]. However, the conventional tiny keypad of a mobile phone is unsuitable for many situations; for example typing a simple URL such as "http://www.google.com" might require over 70 key presses [11].

An alternative approach is a pointing interface. Pointing is a natural way of communicating. Children in all cultures use pointing inherently. In a pointing paradigm the user points at a tag to launch a desired action. Pointing user interfaces can rely on RFID, NFC (Near Field Communication), Bluetooth or visual tags to activate an event as we demonstrated in [17]. RFID, NFC and Bluetooth all share a common problem; a user cannot see the tag or beam, which makes pointing difficult. NFC has a very narrow beam, which means the user needs to aim carefully without knowing where to aim. Bluetooth on the other hand covers a large area, thus selecting the right Bluetooth device can be problematic. Typically, some kind of visual mark is used with these to indicate available tag for a user.

Visual 2D barcodes both indicate the availability of a link for the user and contain the information for the application. The use of the visual markers is reasonable for example in printed media as there is no additional cost. In our work [11], we implemented a natural user interface for camera phones using visual tags (VisualTag). The usability of our VisualTag user interface was tested successfully with a group of elderly people who normally have great difficulties using the tiny keypads of mobile phones. The group used VisualTag for making phone calls and for sending short messages to relatives, for example. The usability results were very good [17]. In the tests, users had photos of people and a marker at bottom of each photo, and they were able to phone each person by pointing the picture (the marker at the bottom of it). Our technical implementation was at the level where it was sufficient that a user just pointed the marker. Our system detected the marker even if the hand was shaking. Our marker system (see Chapter 4) was a gateway to different applications. Instead of a phone number, it could contain an ULR and launch web browser as well. AR was one of the suggested end applications.

A pointing paradigm is widely used in AR browsers and location-based applications. For example Nokia's Point & Find [245] and GeoVector's World Surfer [246] let users explore information and services on the internet simply by pointing their mobile phone cameras at real-life objects. With Google Goggles [247] users can retrieve information on landmarks, books, logos, etc. just by pointing at them.

Our experience regarding this kind of pointing interface is that it should be smooth and easy for the user: users do not want to aim for a long time. Furthermore, an application should also indicate when it is processing something. In addition, the user should know where additional information is available. The user experience is destroyed if the user points at several objects without any response. Without any response, the user is unsure whether the system failed to detect the objects or the object does not contain any information. The system should clearly indicate that it is processing information and then give a result.

If the pointing paradigm is used together with GPS or other location information (as in Layar [237]), user sees all available information and there is no confusion about the availability of information. Nonetheless, it is important that the aiming, selection and processing are fluent.

Motion user interface

User interaction based on detecting the motion of the mobile device is suitable for replacing a mouse or a joystick in applications where they are traditionally used to move something, e.g. the cursor or a game object. For example, SymBall [248] is a virtual table tennis game where the user moves the racket naturally by moving the phone.

The motion detection can rely on mobile device's camera and use computer vision to detect camera movements or it can take advantage of the additional sensors such as accelerometers and compass, as for example the Fairy Trails mobile AR game we mentioned in Section 2.4.

Besides the mobile application itself, this kind of motion-based user interface can be used to control other devices using for instance a Bluetooth connection. We used this approach in our PhoneMouse [249] application, where a camera phone works as an optical mouse for a PC. The user can move the cursor on the PC screen by moving the phone in the air.

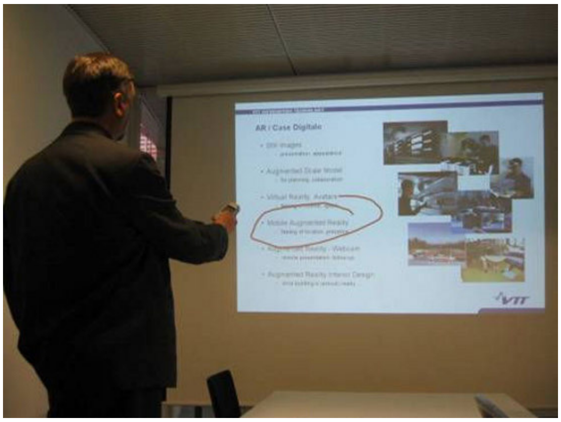


Figure 88. A mobile camera phone acts as an optical mouse for a PC in the PhoneMouse application (Image: VTT Augmented Reality team).

Motion detection of a mobile phone enables a very natural interaction for the user. It was widely used before touchscreens were common. Nowadays similar natural interactions can be achieved with touchscreens.

Tangible user interfaces are widely used in virtual reality and in video games. In a tangible interface, a physical object represents a virtual object and the user interacts very naturally with it. For example, users may have a stick or just a handle, which represents a golf stick in a simulated golf game. The mobile device serves at the same time as tangible user interface replacing the mouse or racket in the applications such as the abovementioned PhoneMouse and Symball.

It is possible to extend the natural interaction approach to pure virtual interface and use natural gestures for interacting with virtual object in similar manner as in real physical interaction. For example, the user can use virtual grabbing to grab and move virtual objects and then an opening hand to drop them, using a pushing gesture to move big objects, etc. The methodology for continuous natural user interfaces is studied in [250].

Multimodal user interface

Multimodal interfaces allow the user to interact with a computer using more than one input and/or output modes. Multiple modalities offer additional flexibility and make machines readily accessible to non-expert users. In addition, appropriately designed multimodal interfaces that exploit synergies among modalities can improve efficiency as well as the naturalness of interaction [251].

Augmented assembly is an example of a hand-busy, eye-busy interaction where the use of tactile input, e.g. a keyboard, to command and control the application is both unnatural and inefficient [252].

In our work [7], we studied a multimodal user interface with gesture and speech input for augmented assembly task. All users preferred the multimodal user interface compared to different single modalities. They also found this kind of application useful for practical assembly tasks (e.g. installing a digital TV box, assembling furniture, etc.)

In the experiments, we used a very lightweight video display that could be attached to (safety) glasses and a camera attached to the middle of the glasses (see Figure 89). Our gesture control was implemented with a head-up display (HUD) like a virtual menu. It consisted of icons that the user could select by moving a hand over them (see Figure 87).

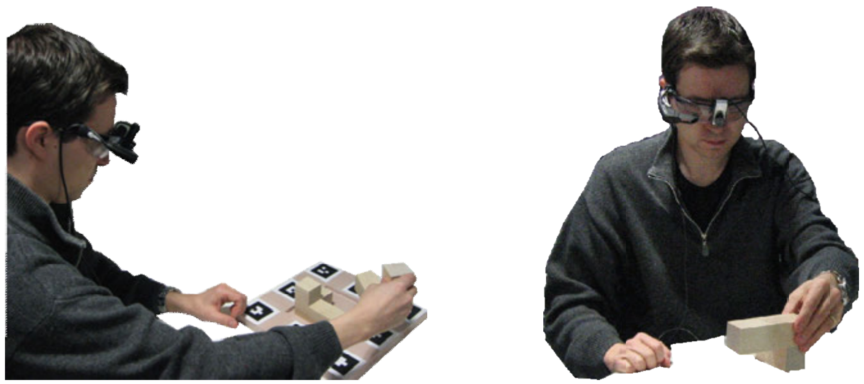


Figure 89. User assembling a 3D puzzle.

In practice, people tend to look at their hands while working on an assembly task. We attached the camera in the middle of the safety glasses facing forwards. This way the hands appeared most of the time in the centre of the image. Placing the menu icons on the top of the image, we were able to prevent unintentional selections of them, which is critical concerning the user experience.

In conclusion, we can say that people have different use preferences and "one size does not fit all". A good multimodal interface can provide all users with a pleasant user experience. In addition, we see that it is important for the user to be able to easily adapt/teach a system that he or she will use frequently or over a longer period of time. For instance, the speech recognition should learn the user's way of pronouncing commands, the gesture recognition should adapt to lighting and skin colour, etc.

Feedback

Visualising each task with AR is easy, whereas confirming whether a user has performed a task is challenging. Suppose that the user adds a small sunk screw, which is the same colour as the part where it belongs. It would require technology beyond the state of the art to be able to follow when user has completed this kind of task. Therefore, AR systems often need user interaction to move from one phase to another. In our multimodal system the user was able to move forward and backward with audio commands, using gesture-based virtual menus or a keyboard.

According to our experiences and our user tests [7], an augmented reality guidance system should clearly indicate when it moves on to the next step and provide feedback for the user. In our case, users wanted a confirmation that the application had interpreted their gesture or audio command correctly. Users also hoped to be able to follow the progress of the assembly work and remaining work. The system could provide this information with a progress bar, for example. Our test users found the feedback from the system very important. Furthermore, the phase number or arrow visualising movement forward or backward could blink, for example. Users also wished for audio feedback (a beep).

Users wanted the system automatically to detect when the user had assembled a part and whether it had been correctly assembled. Recently, depth cameras have been used to create dense 3D surfaces (e.g. [253]). In future, a combination of depth-cameras and computer vision might be able to detect the actual progress in assembly, give feedback of performance and automatically move to next phase.

Our use experience with pointing interfaces was also in line with these findings of multimodal user interfaces. Users want to be sure that the pointing was successful. The system should indicate when it starts processing an image and get the results quickly. Furthermore, we learned that users wanted to keep control of the actions. Thus, we had a confirmation phase. For example after detecting a phone number, our application asked, "Call this number?" and the user selected "Yes/No".

7.2 Avoiding physical contacts

In AR systems the user normally moves, turns his or her head or moves some handheld devices and this movement is transferred to the virtual or augmented environment. We have noticed that AR applications are often mentally immersive, which might lead to dangerous situations. For example, a user may wear a head-mounted video display in an application where the system tracks the user's location and head movements and then displays a correct view of the virtual or augmented environment. Alternatively, users move a camera phone to catch the same virtual objects in an AR game. In both cases, the user's focus is on the application and little attention is paid to the environment.

Other researchers have similar experiences. For example in [254] researchers reported that in user tests users concentrated on the AR game and discovered their surroundings through the mobile display in a way related to tunnel vision. This led to the danger of colliding with cars entering and leaving a parking lot, even when the test users were adults.

In addition, the user's ability to perceive depth is reduced in some AR applications [228, 232, 255] which may lead to unwanted collisions.

Our own experiences are in line with these findings. People tend to concentrate on the AR application and its display. For example in a mobile AR application the user looks at the environment through the narrow display and may not notice obstacles on the ground or nearby. With video see-through displays this effect is even worse: people do not even have a glimpse of the environment outside of the field of view. We have noted in addition that users pay little attention if at all to their surroundings when using an application with a motion-based user interface. Users may hit something accidentally if there are physical objects nearby.

Therefore, it is very important that the system ensures that physical collision with the environment and other users is avoided. One way to do this is to use virtual metrics that differ from real metrics and can be adapted to each person separately.

For instance the movements of the user can be extracted so that the actual physical movement is smaller than its counterpart in the AR environment. This way the application can keep the user at a safe distance from walls and other obstacles.

A multi-user application may guide users sharing the same view further away from each other in reality than in virtuality. This way the application can avoid users colliding or blocking each other's views. In [256] this approach is called redirected motion.

In practice, this means for example that two people manipulating the same virtual object and acting at the same virtual location are in different physical locations at a safe distance depending on the application. Furthermore, the metrics may change during the use of the application. The important thing is that when user is approaching a physical obstacle or another user, the application shrinks or stretches the user's movements to avoid unwanted contact.

7.3 Practical experiences with head-mounted displays

Over the years, we have tested and used several devices and different device setups for several augmented reality applications and demonstrations. Our experience is that the terminal equipment affects the user experience significantly. For example, the head-mounted displays (HMD) serve well for creating cool demos, but most of them have proven immature for real-life use. Next, we tell a little bit about our experiences with head-mounted displays, but it should be noted that this is by no means an extensive survey of them.

With immersive video glasses, the user sees the environment through a narrow-angle low-resolution camera image, often even non-stereo. This causes a danger of colliding with real-world obstacles as previously mentioned. HMDs are often heavy to use for any long period, and some people experience nausea when using them.

Most of the head-mounted displays have wires, which limits the user's movements in a disruptive way. With one type of glasses that we tested, the wires were so short that they even prevented the use of a simple application. Naturally, users can extend the wires, but then there is a risk of tripping on them.

Our experiences with projector-based see-through glasses are even worse; many people who wear glasses are unable to adjust the image into focus at all. The image is visible only if the see-through display is exactly at the correct position. Therefore, the mounting band around the user's head is often heavy and needs to be very tight, which is unpleasant for the user and easily causes headache. In addition, sometimes even slight head movement may cause them to move out of the focus.

Moreover, the augmentation in optical see-through devices is transparent due to limited light intensity as can be seen in Figure 90. In practice, we often had to turn out most of the lights to be able to see the augmentation.

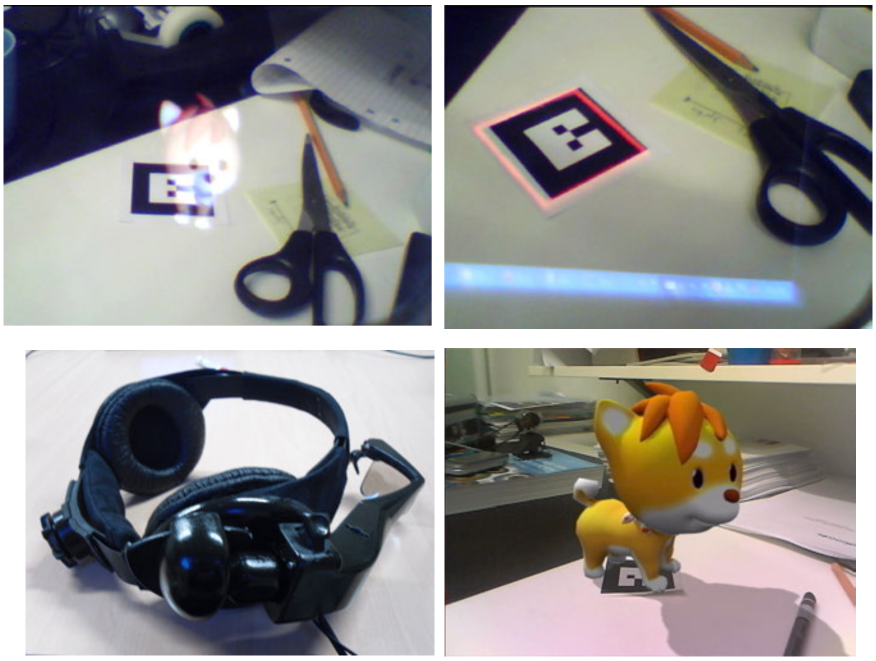


Figure 90. On the top-left: view of a see-through head-mounted device, as the user sees it. On the top-right: the taskbar is visible at the bottom of the projected area. On the bottom-left: see-through HMD used in these examples. On the bottom-right: augmentation on a normal display (rendering on top of video image) for reference.

Based on our practical experience, immersive video glasses are suitable for AR applications that are used for short periods at time, preferably while sitting down.

The best head-mounted display that the author has used was a MicroOptical SV-3 PC Viewer, which is no longer on the market. We used it for example in augmented assembly demonstrations [7]. It was very lightweight (less than 40 g) and was easy to attach to glasses. The user saw a video display at the side of the view and could glance at it when needed (see Figure 89). The user also saw the environment as normal. This kind of setup is suitable for assembly work. The user can see normally with both eyes and stereo view is preserved, and the assembly instructions can be viewed any time as needed, without disturbing the ongoing assembly work.

7.4 Authoring and dynamic content

Augmented reality concerns visualising relevant information on site. Application needs to know what, where and when to augment, and sometimes how to augment it as well. The process of defining these relations for the application is called *authoring*. Good authoring tools are essential for wide use of AR in assembly, maintenance and repair. In addition, content needs to be easily available for authoring.

Augmented reality systems for assembly instructions usually have a database with 3D models of the parts, their interaction (e.g. animation paths), their relative locations, the assembly order, etc. The actual assembly task can be described with an XML file, for example, as for instance in our AR assembly demonstrations [2, 6, 7]. It is important that the authoring is automated as much as possible and that the user can define new assembly tasks easily. If authoring requires a considerable amount of manual work, the benefit of using AR decreases

Authoring tools bring AR application development to a higher level, where the user can build AR applications without a deeper knowledge of theoretical background or programming experience. For example BuildAR [257] is an augmented reality authoring tool for non-programmers. ARMedia is another tool for creating AR content and ARPlayer is a free application to see AR content created with ARMedia [258].

Besides a system-specific database, some AR applications may load content dynamically, for example from Google Earth. The user can select models as e.g. in ARSights [258], which is a desktop AR application that visualises the models on top of markers. Alternatively, a system can automatically download models based on the user's GPS location as in our outdoor visualisation demonstration [4].

Sometimes an AR application needs to show animations, for example in assembly a part that moves to the correct position; in maintenance, a system that shows how to remove or adjust a part, etc. Defining animation paths in code is time consuming and often requires a "trial and error" approach. Authoring tools could record these animation paths; the user would move a marker (or an instrument with an attached marker) on the desired path and the system would record it. Similarly, an authoring tool could record the permitted driving area of a virtual car for an AR car racing game, for example.

8. AR applications and future visions

As we have discussed earlier in this work, augmented reality application development is often about making compromises: visual quality vs. speed, speed vs. accuracy, performance vs. system complexity, devices vs. costs, number of features vs. performance, etc. Although individual methods run on real time in demonstrations, the combination of several of them is no longer able to perform on the required frame rate. Therefore, there is no template for the absolute best combination of methods and functionalities for AR: *the best combination depends on the purpose of the application and the target user.*

In this chapter, we discuss the most important issues of AR application development and the application areas with most potential. We also consider technological enablers and other things affecting the development of the area and speculate about the future of AR.

8.1 How to design an AR application

As the processing capacity still limits the functionalities of AR applications, the development should focus on the main issues arising from the purpose of the application and the target user. For example, if an AR application is used to measure distances and physically fit something somewhere, accuracy is a high priority. This enables the main thing that is *"measuring"*. If the operators are experts, they can be trained to use the system and do some initialising (e.g. to perform a calibration phase).

Whereas, if the system is used to visualise the 3D shape of an object, and it is used by occasional persons, the starting point is totally different. The user should be able to operate the system without any training. In such a case, the accuracy of the pose would not be the major concern, as long as it is consistent. The main thing *"visualising a 3D shape"* is independent of the object's location and the correct scale. Thus, the major concern might be usability and user interactions with the object, instead of the high accuracy of pose or scale. In fact, users prefer an inaccurate but stable pose over an unstable pose [139], if processing capacity or some other circumstance forces them to make a compromise. This is in line with our practical experiences. As AR applications mainly aim for real time processing, optimisation is very important. For example in marker detection, the fast acceptance/rejection tests are essential as we explained in Section 3.1.3.

Feature-based tracking is often considered to be more advanced technology than marker-based tracking. However, this does not imply that it is always the better choice for all purposes. Feature-based tracking alone does not tell the correct scale, and the origin and bearing of coordinate axes are arbitrary. Some kind of user interaction is often needed to fix the scale and the coordinates. This is fine with expert users, but it does not suit all situations. Some users may find it more convenient to place a marker on the floor as the application then automatically knows the scale and orientation of the floor plane (e.g. in interior design consumer applications). This way, the application can automatically align coordinate axes to a vertical and two horizontal directions, which is intuitive for humans.

In AR, due to limitations of devices and their capacity and the human factors, *the best technology is not always the best technical solution for the purpose in question.* Naturally, in former example, there is no need to limit the tracking to use only markers. Markers have proved a good tool for initialisation; the application could rely on feature tracking, but use markers for user-friendly initialisation as discussed in Sections 4.4.1 and 5.3. Markers are beneficial in hybrid tracking: they stabilise the system, help in recovery, define the correct scale and natural axis orientation and serve as a trigger so that people are aware of the existence of virtual data (as explained in Sections 4.4.1, 4.4.5 and 5.3). Furthermore, the system can associate markers with different interactions, and retrieve data from them.

If different types of users use the same application, it might be good idea to think about the possibility to have several different user interfaces. In our ongoing work concerning augmented reality interior design, we learned that professional users (interior designers) would be willing to attend training session to learn to use an AR interior design application with necessary amount of features, whereas, consumers would prefer a simpler user interface with less features.

Users also have other preferences and a good adaptive multimodal interface can provide all users with a pleasant experience. In addition, adequate feedback from the AR system improves the user experience as we reported in Section 7.1.

There are different ways to present the virtual content. The visualisation (or auralisation), should support the task and purpose of the applications. In Chapter 6, we discussed ways to enhance the AR application with a proper visualisation mode, and gave examples of photo-realistic rendering, NPR and X-ray vision and their use. In Section 8.3, we will review the different application types which visualisation techniques typically support them best.

8.2 Technology adoption and acceptance

Several factors affect adoption of a new technology in general. First of all, personal characteristics affect individual adoption. People belong to five categories of the technology adoption lifecycle depending on how easily they adopt new technology.

Innovators and early adopters are the first to try out a new technology, then come the early majority and late majority, and at the end the laggards [259] (see Figure 91).

Furthermore, the characteristics of the technology and use situation influence the adoption as well. Key factors of adoption are relative advantage compared to other technologies, compatibility and relevance, complexity or simplicity, trialability (how easily the user can test it) and observability (peers and social networks) [260, 261].

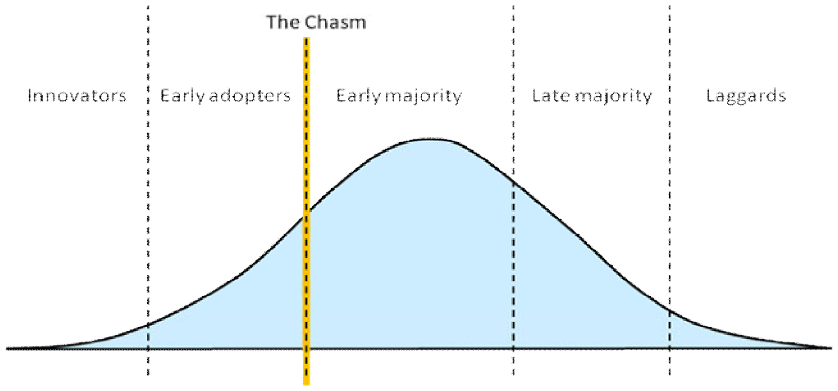


Figure 91. The technology adoption life cycle. For discontinuous or disruptive innovations, there is a critical chasm between visionaries (early adopters) and pragmatist (early majority) [262]. The horizontal axis represents time and the vertical axis the number of individuals.

Besides usability, the assumed benefit and attitude towards use affect how users come to accept and use a technology. Other factors affecting the adoption and acceptance of an individual are: voluntariness of use, experience, output quality, demonstrable results, perceived ease of use and social acceptance [263, 264]. A commonly referred technology acceptance model is presented in Figure 92.

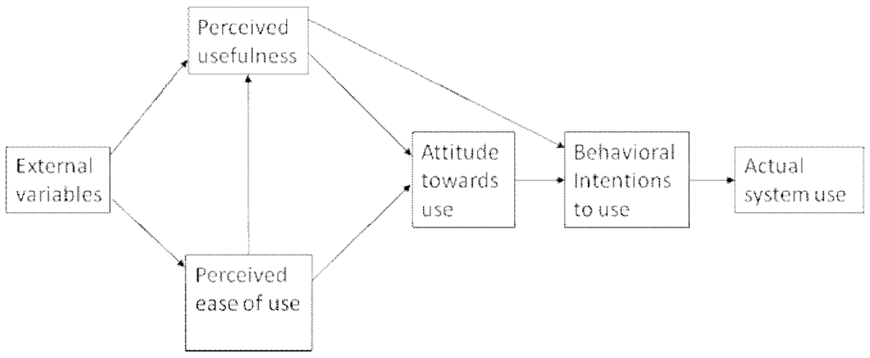


Figure 92. Technology acceptance model [263].

Several factors affect adoption and acceptance of augmented reality technology. A few years back a person talking alone presumably to himself was considered to be insane and as such the first people to use hands-free devices with their mobile phones raised a few eyebrows on the street. After hands-free devices and head-sets became more common, "talking alone" became a socially acceptable behaviour. Although pointing is a natural interaction method (see Section 7.1), some studies show that people are unwilling to point around *constantly* with their mobile phones (unless taking pictures or video). People may even feel embarrassed for doing so. Besides, looking at an environment through a display leads to tunnel vision effect and the risk of physical collisions. Considering AR browsing applications, people find it more natural to point at something just once with the phone and then manage the information in a more discreet manner, e.g. keeping the phone near their body. Will constant pointing become a normal behaviour? Will see-through data glasses become a normal accessory in the future? Social acceptance will have an influence on the future development, and the development will change which behaviours become socially acceptable.

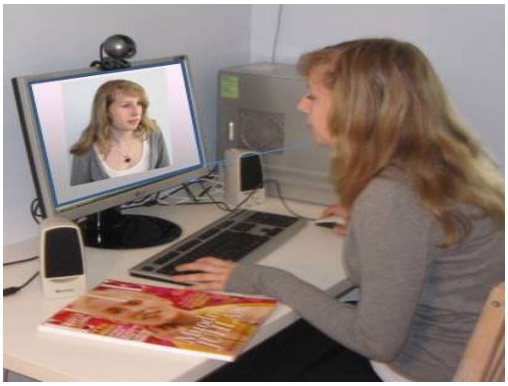


Figure 93. People are unwilling to download applications, yet they are willing to try out web browser-based applications. Image illustrating our web-based jewel testing demo.

People are unwilling to download applications unless they are certain that it is worth the trouble and sometimes not even then. Innovators and early adopters are more likely to test new technology and make the effort of downloading, but for the majority it might be "the chasm". Therefore, AR advertising, for instance (in a PC environment), is often implemented as a web browser application using the PC's webcam and e.g. Adobe Flash Player. Our AR Jewel demo [265] demonstrates this concept (see Figure 94). The idea is that a post-it type of marker comes with an ad in a magazine with instructions to go to certain webpage. The user can then put the marker on his/her neck and see an augmented jewel in the AR mirror application. This demonstration uses 2D images of the jewel as no 3D models are available for hand-made jewellery and the user usually looks at them directly from the front as if looking at themselves in a mirror. Naturally, we could use 3D models if they were available. We can easily list a number of applications that could use this kind of virtual mirror concept: selecting glasses, virtual fitting rooms for trying on clothes, testing haircuts, testing hats, etc. Altogether, all applications where the user wants to try something on and would normally look at themselves in a mirror.

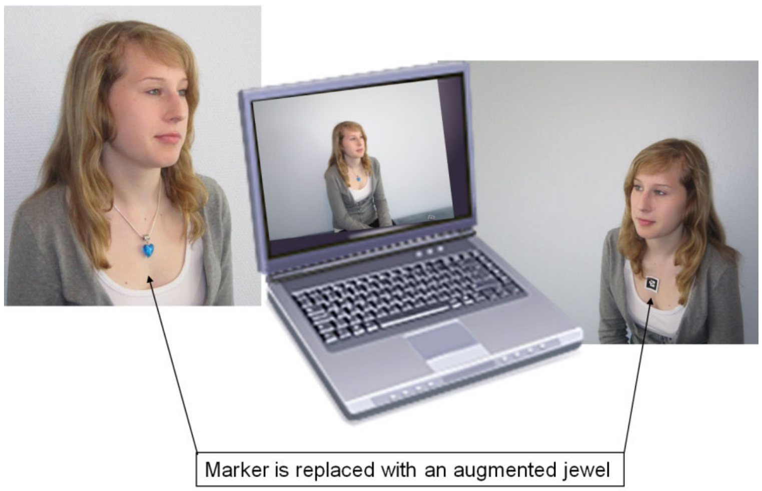


Figure 94. Concept of our web-based jewel demo (www.vtt.fi/multimedia). The user puts a marker on her neck and sees a jewel augmented in place of the marker.

Mobile application stores (e.g. iTunes App Store, Google Android Market, Nokia Ovi Store, BlackBerry App World, Palm App Catalog and Windows Marketplace for Mobile) have changed mobile application markets a lot. Earlier, consumers found it difficult to download, pay and even find applications. Application stores have wrapped mobile application retail in an understandable and secure package. Today people are accustomed to buying mobile applications from these stores and the purchasing threshold is far lower than it used to be. However, they still have some practical problems. For example, in current mobile AR browsers, each information environment (called e.g. a “layer”, “channel” or “world” depending on developer) needs to be separately downloaded via the application market. Although the threshold for downloading has decreased, people do not want to do it constantly.

A virtual retinal display (VRD) is a system that renders the image directly onto the retina of user’s eye. A VRD enables a “Terminator-vision” type of augmenting a user’s view, as in James Cameron’s film The Terminator (1984). People see a lot of potential in AR applications using HMDs including VRDs. VRD has an advantage compared to any setup including a display. In a VRD, the image is formed only on user’s retina and therefore no passer-by is able to see the contents of the display. This might be an important security aspect in some applications.

When it comes to new cutting-edge devices such as virtual retinal displays human factors play a significant role in how well and how quickly these technologies are adopted. The idea of projecting laser beam onto one's eye might frighten the large majority of people. VRD technologies have been developed over 20 years by now and research suggests that the intensity of the light is safe.

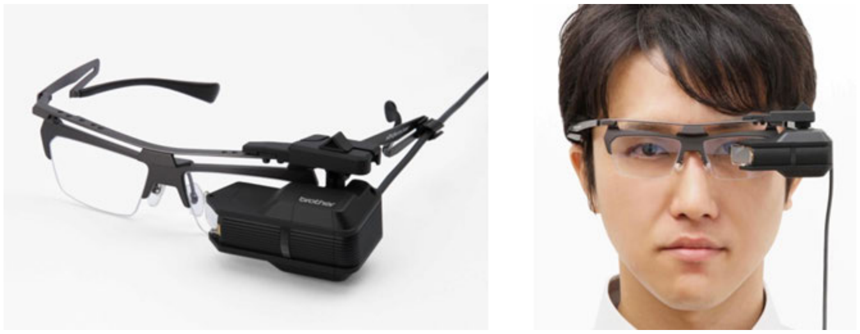


Figure 95. A commercially available see-through head-mounted display 'AiR-Scouter' by Brother Industries Ltd. (Images courtesy of Brother Industries).

One great thing in VRDs is that the laser beam bypasses many of the eye's optical and retinal defects, and therefore they can be used to enhance a low vision or even blindness [266]. Even if they would never become mainstream products, there is a part of the population that will clearly benefit from using them, and who would therefore be some of the first to adopt them.

A Japanese company, Brother Industries, presented a prototype of a Retinal Imaging Display (RID) in 2010. Its prototype projects fast-moving light directly onto the user's retina and it appears to the viewer as a 16-inch display floating transparently at a distance of about three feet away. However, VRDs have not gained commercial success yet. For example, Brother Industries adopted LC (Liquid Crystal) methods as a light source instead of laser in commercial products. In its AirScouter see-through display the light passes through the lens and reflects against a half mirror and projects to the eye (see Figure 95).

Until now most of the augmented reality research papers have been on enabling technologies (tracking and devices), or they concerned prototypes or short-term pilots. Little research has been carried out on user evaluation of AR interfaces [267].

In future, augmented reality research should concentrate more on user experience, usability, the effects of long-term use, interactions and object manipulation.

8.3 Where to use augmented reality

A few years ago it was still cool just to have augmented reality; a number of applications based solely on the wow effect. There is still some hype around AR, but the development is towards applications where the use of AR brings added value. This is an important issue: *an application designer should always consider whether augmented reality really is the best technique for the matter at hand and use AR only in those applications where it makes sense to do so*. Furthermore, *the features that support the purpose of the application should be the main concern*.

Next, we summarise the main AR application types and key technologies and aspects in their development.

8.3.1 Guidance

Augmented reality has proved to be useful for inspection, maintenance and assembly tasks and training in many areas (e.g. surgery), etc. These are all areas where information related to performing a task is visualised for human operators in such a way that it gives the user a better understanding of the performance of the task. In traditional training, an apprentice follows how the master performs the tasks in this kind of situation.

In these application areas, a focus and content type NPR visualisation is often beneficial, as discussed in Sections 6.1.1 and 6.3. One of the main concerns is natural interaction with the system, e.g. multimodal interface, which we discussed in Section 7.1. Others are content creation and authoring (see Section 7.4). These kinds of applications often have task dependent requirements and functionalities, and thus require some amount of customisation, in comparison to AR browsers where new information environments can easily be added without modifications to the AR browser application.

8.3.2 Visualisation

AR is suitable for visualisation; it is used for visualising in interior design, planning, industrial design and prototyping. Here the (photo)-realistic rendering is often required and visual quality is habitually important. AR is suitable for interactive design, where people make real-time modifications to the model. It is also suitable for testing something on the user using a virtual mirror paradigm (haircut, clothes, glasses, etc.). Yet another important visualization area is building and construction, where on-site AR visualization enhances human’s understanding of construction projects.

8.3.3 Games, marketing, motivation and fun

A third type of use is leisure and fun: games, motivating learning, advertising, etc. The application development concentrates on the user interface, user interactions, enjoyment, playfulness, smoothness of the use, etc. Technology is often used in a creative way in these applications, and typically, user experience is a major priority.

8.3.4 Real-time special video effects

In digital printing personalisation is a technique where the printed material is created automatically from a digital source with personalised parts e.g. the user's name "*Dear Mrs Siltanen,…*" Augmented reality enables marketing with personalised videos. A good example is the campaign for paying radio and TV fee launched at 2010 by Swedish state-owned company Radiotjänst i Kiruna. The campaign utilises web-based augmented reality in an impressive way. It takes a picture of a user, and then uses it to augment a high quality film. The user appears as the country's hero in the film (see Figure 96).

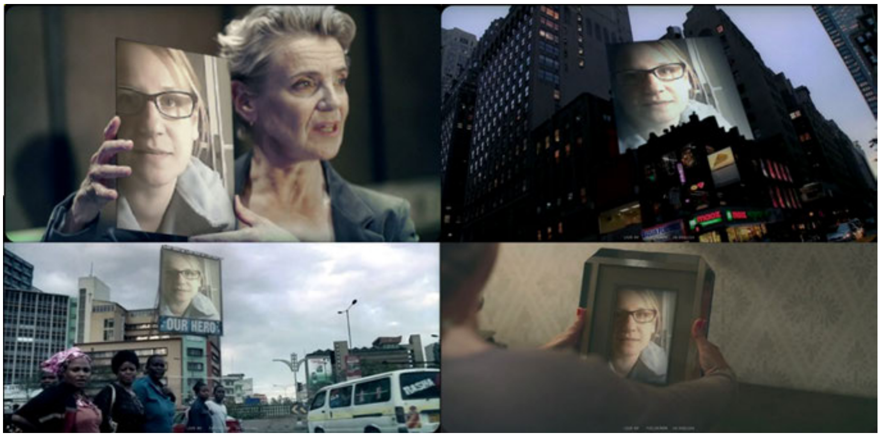


Figure 96. Frames captured from the personalised video film (Radiotjänst i Kiruna, "Hjälten" campaign). The system takes a picture of a user (using a webcam or user-defined image) and uses it to augment a film. As a result, the user appears as the hero within the video (Images presented with the permission of Radiotjänst i Kiruna).

Similarly, augmented reality can be used for product placement and advertising in the TV and film industry. The same product may appear under a different brand name in different countries. With augmented reality it can be replaced with the correct product in each occasion. Augmented reality product placement changes the marketing logic. A certain product in a film can appear under a different brand name in different showings. We will discuss the possibilities of AR in TV production a bit more in Section 8.4.1.

8.3.5 World browsers and location-based services

AR provides a means to link information to real world objects and certain locations. It is used in AR browsers and other location-based services for visualising variety data. These applications use AR to show nearby restaurants (e.g. Wikitude), metro stations, shops, cafes, offers and museums. Location-based AR applications can provide more information on buildings, history, bus timetables and restaurant menus, and help with navigation. They can visualise weather (e.g. Weather Reality), satellites (e.g. Satellite AR), star charts (e.g. Star Chart) and twitter tweets (e.g. Mixare).

AR location-based applications let the user define tags and points-of-interests and link them to Facebook, Twitter, Foursquare and other social media networks (e.g. Tagwhat and Sekai Camera). Users can share photos and see geo-tagged photos from other media such as Panoramio on-site (e.g. Photos Around).

Above application examples from Android market are just the tip of the iceberg. It is impossible to list all of them. What they share in common is that they use GPS, camera and other mobile device sensors to locate the user, and to present information relevant to the current location or object. The function of applications is more than just AR, for instance social media/geo-local service. For example, Yelp’s monocle [268] is an application that visualises nearby restaurants and provides user reviews. From the user’s point of view it is a location-based (social media) restaurant guide, where AR is the means for bringing information to the user.

8.3.6 Other

Augmented reality visualisation is used to support other tasks in the area of robotics, e.g. for collision-free robot programming as mentioned in Section 6.3.2 and for improving robotic operator performance [269, 270]. It also benefits many medical applications e.g. laparoscopic surgery. AR benefits all tasks, where real-time 3D visualisation of information on-site helps the human operator.

8.4 Future of augmented reality

There are some restrictions in AR. For example, a system is able to show the augmented view only from those viewpoints from where it has a real image. For example, the user can see a virtual building from ground level looking through a display, but is unable to see the scene from a bird's eye view. In order to provide such visualisations, applications often complement AR with virtual reality mode.

Other limitations are due to restricted capacity of devices: their power consumption is too high and their processing, telecommunication and memory capacities are too low, the resolution of cameras is too low, etc. Engineers develop new and better devices, and the capacity of devices increases, they have more built-in sensors, therefore future devices will solve many of current obstacles. Cloud services will in turn help with computationally intensive tasks in future.

8.4.1 Technology enablers and future development

From the application developers' point of view, the diversity of platforms is problematic: they need to port the application to different platforms, as a lot of the code is platform-dependent. HTML5 will be a step towards device-independent applications. It is supported by a large number of mobile devices. It enables presenting videos and audio on web pages as easily as presenting text and images is now, without a need for any plug-ins [271]. HTML5 offers a number of useful properties such as the user's locations based on GPS or WLAN, canvas technology for dynamic graphics, etc. Although HTML5 is already available for application developers, the standard will be finished probably only on 2014 in W3C.

Integration on global databases such as Google Earth, with GPS and local information for example from security cameras together with cloud computing, could lead to real applications similar to the one presented in [272], where the AR navigator is able to visualise cars coming out of the driver's view (e.g. behind buildings). Another new type of application is security guard guidance system that is able to visualise people behind or inside buildings using AR as means for visualising information from security cameras.

The usability of the see-through-devices needs to be improved before they are suitable for mass-market applications, especially the head-mounted see-through devices, which are clumsy as we discussed in Section 7.3. In future, the see-through portable devices, such as the See-Through Laptop Samsung presented in 2010, might provide a better platform for mobile AR applications (see Figure 97).

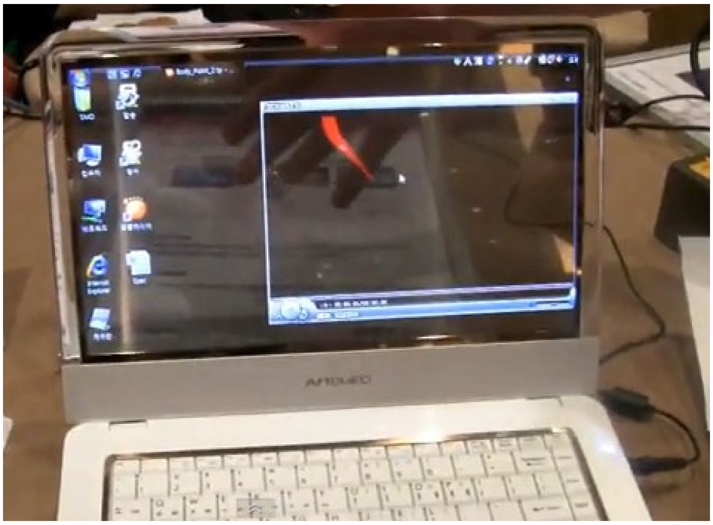


Figure 97. Samsung's See-Through Laptop presented in 2010 (Image courtesy of Engadget [273]).

Virtual retinal displays that render images with a laser directly on the user's retina may become more common (see Section 8.2). Another alternative is the use of augmented reality contact lenses as for example envisioned in [274, 275]

In future users may use devices intuitively by bending, twisting and squeezing, e.g. the *Nokia Kinetic* phone presented at Nokia World 2011. This kind of user interface works even if the user wears gloves in cold weather, where touchscreens are unpractical. The displays may be flexible and foldable, such as Polymer Vision's *Readius* presented in 2008. Samsung has announced that their new mobile device line-up will feature flexible screens starting in 2012. The whole surface of a mobile device could be touchscreen as in Nokia's *GEM phone concept* [276].

While waiting for reasonably sized foldable mobile device with interactive surface (and long-lasting batteries), people might find tablets to be a nice platform for AR applications: they have bigger screens than phones for visualisations, they are lighter than laptops, and they have built-in cameras and sensors.

In the early days of virtual reality, Ivan Sutherland visioned *"The ultimate display would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal."* [277].

It is hard to believe that researchers will ever build a virtual environment that will fulfil the last sentence. One of the exact benefits of virtual and augmented reality is that they enable safe environment for training and experiencing situations that would be too dangerous in reality.

On the other hand, researchers have developed such physically altering environments that Sutherland visioned. Physically rendered environment is an environment where the system can alter the physical environment and manipulate grids of "moving physical pixels" ("moxels"). The system can raise and lower moxels and render physical shapes. It is a world-size version of the 3D pin art table where the user presses an object on the pins that will raise and create a 3D replica of the object's shape.

Today the quality of such systems is poor and their density is low. Furthermore, most current systems are able to model only vertical shapes, e.g. the Holodec presented in [278].

3D holograms, furthermore, are able to visually render arbitrary 3D shapes, but without interaction possibilities. The University of Arizona presented a dynamic 3D hologram, which allows the three-dimensional projection, without the need for special eyewear. The resolution and speed (refreshes every two seconds) leave space for improvement, however [279–281].

In future we will probably see interactive large-scale 3D augmentations, virtual or physical. This will bring telepresence to a different level and enable new possibilities for mixed reality environments, e.g. telesurgery. Besides large 3D mixed environments, there is a trend towards mobile handheld AR with projective and 3D reconstruction capabilities.

Small projectors have also been demonstrated for augmented reality [282]. The integration of small projectors to mobile devices will give a boost for hand-held projective AR.

As the development of the mobile devices continues (e.g. processing capacity and battery life), it will open way to applications that are currently computationally too demanding, such as robust feature tracking on mobile phones. In addition, 3D reconstruction will become feasible on mobile devices. Researchers have already demonstrated it using a mobile phone [283, 284]. The first mobile devices with a 3D display are already on the market; we may assume that they will become more common in the future.

Easy mobile 3D reconstruction enables a new type of interaction with virtual worlds: people can scan real objects and bring them to virtual world (Second Life, Habbo Hotel, etc).

The development of augmented reality enables a new type of interactive TV production; one of the first examples was BBC's Bamzooki, an augmented reality TV game show that aired in 2009, where participants on the show shout instructions to control virtual autonomous game creatures called Zooks. It is easy to imagine that this kind of TV production will become more common in future. Current technology enables also real-time virtual staging (augmented virtuality), where the whole environment is virtual with real people acting in it. For example, Aalto University Media Centre Lume has a studio that enables this kind of production (see Figure 98). Future AR will bring film effects to live TV broadcasts.

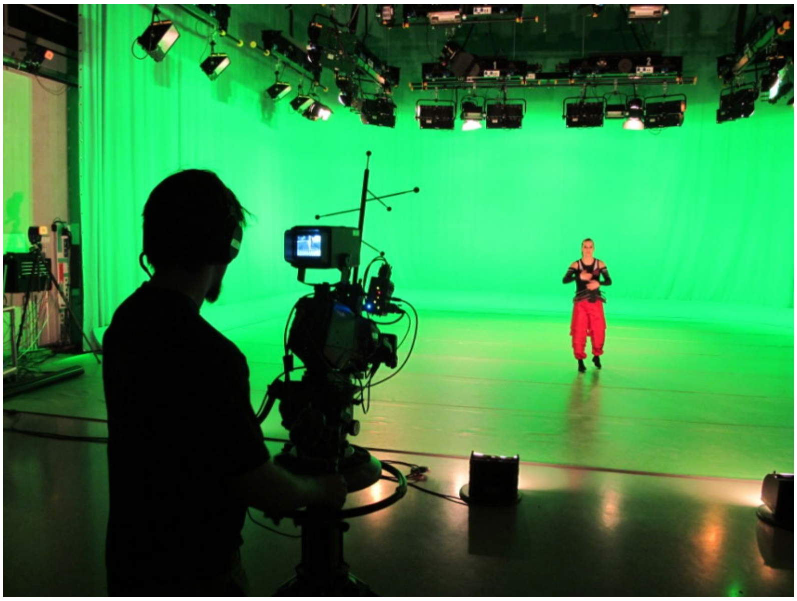


Figure 98. Aalto University Media Centre Lume Tv-studio virtual studio system (Image: Toni Tolin).

In future ubiquitous environments, any surface can be made into a touchscreen; the Nokia research lab even converted an ice cube into a touchscreen (see Figure 99). Researchers have used several approaches in converting ordinary surfaces into touchscreens, e.g. multiple cameras [285], computer vision and projectors as in Sixth sense [286, 287] (see Figure 100) and depth cameras for surface computing as in Microsoft Lightspace [288].

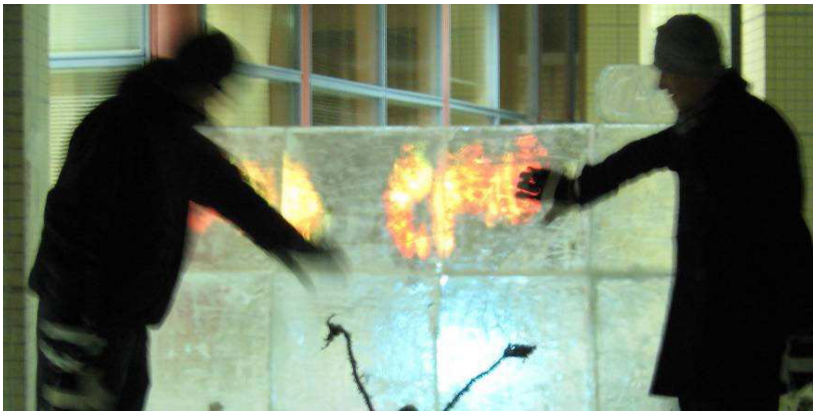


Figure 99. Researchers at Nokia converted an ice cube into a touch-screen (image from [289]).

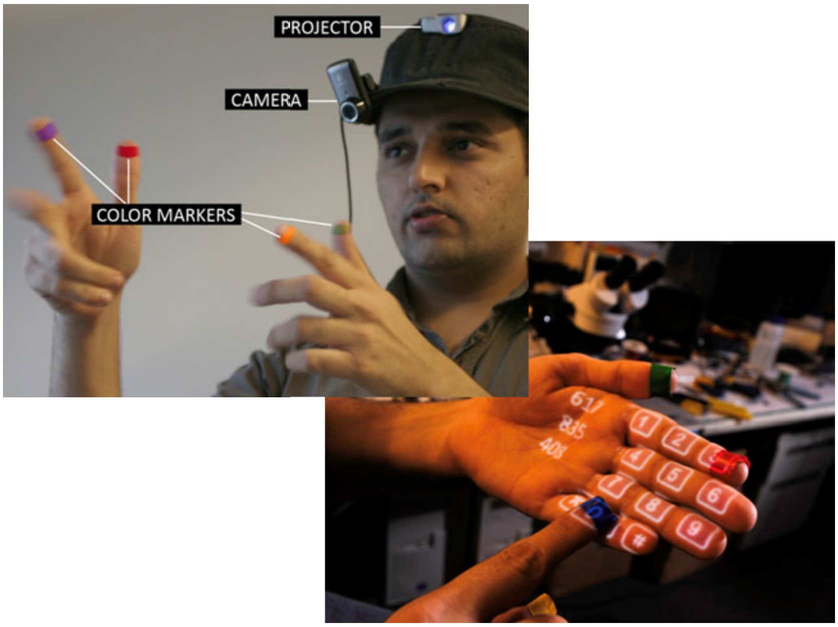


Figure 100. On the left: Sixth sense gesture interface. On the right, Sixth sense virtual keypad (Images courtesy of Pranav Mistry).

Existing mobile object/image recognition services such as Google Goggles [247], Kooaba [290] and Snaptell [291] give us only a hint of all the future possibilities. One problem with current solutions is the long lag, typically tens of seconds, between snapping the picture and receiving the response to the query [292]. It is possible to speed up detection using low bit-rate local descriptors and data compression [292], but a better transmission rate and better indexing of data also enable faster queries.

The business model of search software is also altering. The SmartAds service that Kooaba offers is an example of the Software-as-a-Service (SaaS) model. The user is charged for access and use of the search and recognition engine. It allows customers to turn their printed products into virtual hyperlinks to additional information [293].

Another new type of AR application is CrowdOptic [294], which received the Frost & Sullivan 2011 Annual Innovation Award for live-event technology. It is an AR application bound to certain events, e.g. football matches, concerts, etc. It lets users point their smartphones at an athlete or performer and see additional information about the target in real time. Users obtain coaching insights and the stats of the target, and they receive exclusive invitations, ticket discounts and other material through the system. In addition, the event organisers get information on crowd behaviour. The system detects where the attention of people is at any given moment, using GPS data, triangulations and analysing what is being photographed or videoed. Organisers can immediately consider the crowd's interests in the production. We could call this real-time crowdsourcing or a crowd behaviour analysis tool.

The trend in mobile AR browsers is to link information with other social media, use other technologies such as object recognition and face detection, etc. They take advantage of additional sensors and use remote computing and data storage facilities.

Future mixed reality concept probably connects location-based services, user-created content, social media, etc. with the physical world (magazines, billboards, buildings, places, etc.) It allows users to comment, share and link ideas, as well as attach bookmarks, tag physical objects and get information related to them. It provides a platform for interact with the concept of *internet of things*, where all objects are networked with information and services.

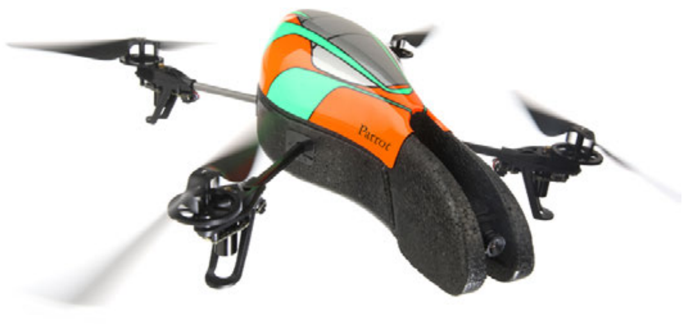


Figure 101. ARDrone by Parrot (Image courtesy of Parrot).

Future augmented reality development interconnects with robotics in several levels. ARDrone (see Figure 101) is a flying iPhone accessory (helicopter) equipped with a multitude of sensors and intelligence. It is controlled via iPhone using simple upper-level commands such as forward, rotate, up, hover, land, take-off, etc. It turns commands into signals for the four on-board motors using information from various sensors such as accelerometer and gyros. Several ARDrones operate in the same environment and users can play augmented reality games with them. It is easy to invent a useful application for such small robots. As we mentioned earlier (in Section 8.3.6) augmented reality has proved useful as an aid to robot operators. One straightforward idea is to use small remote-operated maintenance robots, which are able to augment maintenance instructions and other information for the human operator. Another idea is to use small autonomous robots capable of object recognition to search for missing items.

8.4.2 Avatars

In James Cameron's film *Avatar* (2009), humans are mining a valuable mineral on Pandora, an Earth-like moon with an atmosphere poisonous to humans. The venue is in Alpha Centauri star system in far future. Pandora is inhabited by the Na'vi, three-metre-tall, blue-skinned humanoids. In order to explore Pandora scientists create Na'vi-Human hybrid bodies, which a genetically matched human can mentally operate. The human operating a hybrid body has an illusion of being inside the avatar body, although lying in an avatar link tank at the base.

This kind of avatar technology is far in the future, but perhaps not as far as we might think. Researchers have demonstrated that it is possible to create a perceptual illusion of body swapping, i.e. being in a body other than one's own [295]. The illusion of being in a different body is possible to achieve even with an artificial body of extreme size. Test persons experienced being Barbie (30 cm) as well as a large doll (4 m) in the experiments reported in [296]. The key factor affecting the perception of being in another body is synchronous multisensory input: the user needs to feel touch when she or he sees the artificial body touched.

In the abovementioned Barbie/doll experiment, the test person was lying on a table with a head-mounted video display. On the other table was a doll (of a different size). The camera providing the video images for the test person was mounted on a tripod on the place where the dolls head would have been, facing the body. This way the user had the feeling of looking at the doll's body from a first-persons view.

The multi-sensory input was created by touching the doll's body (in view) and simultaneously touching the participant's body (out-of view) at the corresponding location.

Based on these findings it would be possible to create an illusion of being inside a virtual avatar as well. A human would explore the world through a virtual avatar's eyes (and see the virtual body from a first-person perspective). With a haptic suit, it would be possible to "feel" virtual collisions that the avatar experiences. This could be the future of Second Life.

Furthermore, in the field of Brain-Computer Interface (BCI) and Brain-Machine Interface (BMI), people have been able to create direct neural interfaces to operate artificial limbs, for example (e.g. [297, 298]). Studies with monkeys (which have neural systems that are considered to be very similar to those of humans) demonstrate the ability to control a computer or a robotic arm with their thoughts [299]. In addition, modern humanoid robots such as ASIMO [300] are able to mimic human movements: walk, run, climb stairs, grab with a hand, etc.

Creating a real avatar experience becomes a matter of cross-disciplinary cooperation in the future. In principle, people could merge all these technologies and use direct neural interface to operate a humanoid robot. The robot would have cameras on head and these cameras would provide the view for the user (rendered using a retinal display). The robot's microphones would record audio, which is played to the user's headphones. The humanoid robot would naturally be equipped with a number of sensors, and their input would be transferred to the user using a multi-sensory (haptic, thermal, etc.) suit.

8.4.3 Multi-sensory mixed reality

Embodied and tangible haptic input/output devices enable transferring sense of touch between the virtual and real world. The person using such a device is able to sense physical interaction with a remote or virtual person.

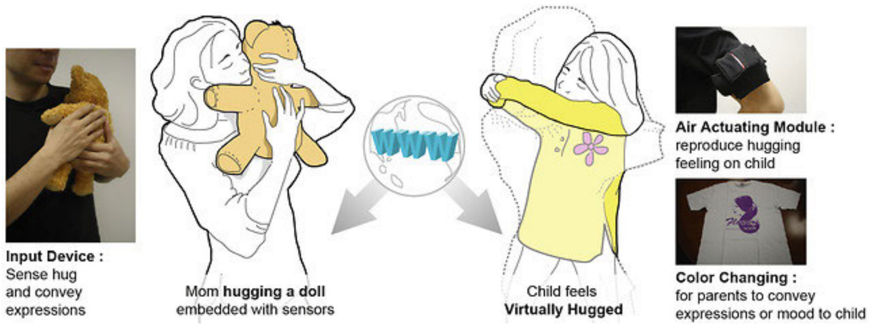

Input Device :
Sense hug and convey expressions
Mom hugging a doll embedded with sensors
WWW
Child feels Virtually Hugged
Air Actuating Module : reproduce hugging feeling on child
Color Changing : for parents to convey expressions or mood to child

Figure 102. Huggy Pajama concept transfers a hug over the internet (Image courtesy of Adrian Cheok).

Multi-sensory environments are a substantial research area. For example, researchers in CUTE centre (National University of Singapore and Keio University) and MXR Lab in Singapore have done a lot of research in the area of multi-sensory environments. *Huggy pajama* [301] is one of their research projects (see Figure 102), where the system is able to transfer a hug over the internet. The user at the other end touches an input device embedded with sensors. The information is transferred to the other end, where the other user receives the same touch from an output device equipped with a haptic interface. *Kissenger* (aka Kiss Messenger) is a similar system; with special input/output devices, it is able to transfer a kiss over the internet [302] (see Figure 103).

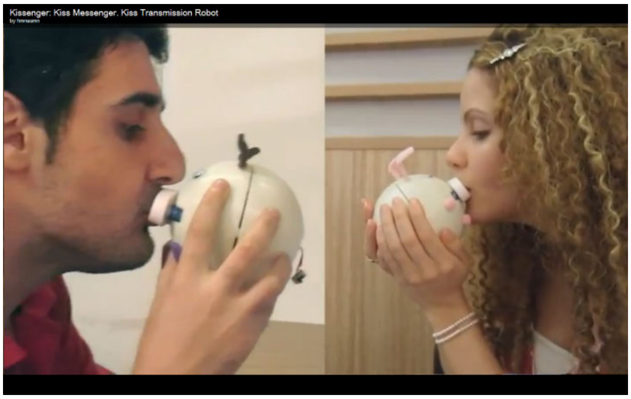


Figure 103. Kissenger: Kiss transmission robot by Lovotics [303] (Image courtesy of Adrian Cheok).

Augmented reality systems most commonly employ haptic, visual and audio sensory feedback. The immersive feeling increases if even more senses receive input from the system.

Sense of taste is composed of five basic tastes (sweet, salty, bitter, sour, umami), which are relatively easy to produce by blending the appropriate chemicals. Nonetheless, the actual gustation is a combination of taste, smell and food texture, plus some other factors. Therefore, it is possible to bluff sense of taste to some extent with scent and visuals as in the Meta Cookie demonstration we discussed in Section 2.5.2. However, humans can recognise thousands of different smells and are able to detect smells even in infinitesimal quantities. Therefore, it is impossible to produce a set of primary smells to produce all possible smells for the virtual environment (unlike primary colours (red, green, blue) for sense of sight). Sensation is ultimately formed in the human brain when the brain analyses electrical pulses coming from sense organs. Therefore senses can be provoked digitally, by feeding electrical pulses to sensory nerves.

The digital taste interface presented in [304] produced sense of taste by actuating the tongue through electrical and thermal stimulations. The experimental results suggested that sourness and saltiness are the main sensations that could be evoked while there is evidence of sweet and bitter sensations too. In medical science, devices enabling digitally produced vision and audition have been used for over a decade (e.g. cochlear implants for deaf).

Today's multi-sensory interface devices are still clumsy and it will take a while before the input/output devices are mature enough to feel natural. In the future, digital gustatory devices become more accurate, and perhaps researchers will be able to produce a substantial amount of digital odours as well support a wide variety of virtual tastes and odours. The haptic devices will improve, immersive display systems become feasible, etc. The way users experience virtual environments is going to change.

In future multi-sensory mixed reality environments people will be able to sense temperature, touch, taste, smell, etc., and the whole of the atmosphere. Such environments will support immersive vision and sound systems, and people will be able to sense physical interaction with virtual characters. Today people share their experiences in social media; they send multimedia messages; they use mobile video conferencing to show what they see (the "see-what-I-see" paradigm). The future mixed reality aims to enable a "sense-what-I-sense" paradigm. Telepresence is brought to a new level.

People accept the idea that they could love or be loved by a robot [305], and people fall in love with celebrities they have never personally met. It is not far-fetched to imagine an intimate relationship with a virtual character. The future multi-sensory mixed reality environment will be able to provide a "multi-sensory-second-life" where people can interact, communicate and live together with avatars mastered by other humans, robots or computer. Naturally, this development will provoke some ethical issues, which we leave open in this discussion.

Furthermore, researchers have been able to reconstruct images that people have seen from brain activity using functional magnetic resonance imaging (fMRI) [306]. The future brain interface could be bidirectional; the system reads from the user's brain what she/he senses and simulates the other user's brain accordingly. This would really be sharing experiences.

9. Conclusions and discussion

In this work, we have presented a thorough overview of the theory and applications of augmented reality. We have concentrated on marker-based tracking and lightweight single-camera approaches, but also gave an overview of alternative tracking methods and referred to how additional cameras, sensors and other devices are used in different types of AR applications. We discussed the ways in which basic AR applications can be enhanced and the ways in which interactions between real and virtual objects can be handled. In addition, the appendices give a comprehensive review of theoretical background of methods and algorithms used in augmented reality.

We have also presented how the author has contributed to different issues in AR application development. In addition, we have reported practical experiences in AR application development and usability issues. Furthermore, we reported our research results in many areas of augmented reality.

In the previous chapter, we discussed AR application development and application areas in which the use of AR is beneficial, and finally we had a glance at future possibilities of AR.

In the following, we summarize the main issues of AR application development and design.

9.1 Main issues in AR application development

In conclusion, the augmented reality application developer needs to take into consideration several different issues: technical, application and other issues affecting the user experience. The main technological issues relate directly to the definition of augmented reality (real-time, interactive, 3D, combining real and virtual). Application issues arise from the ease of creating AR applications. Other important issues relate to user experience.

The main technological issues in augmented reality are

- performance
- interaction
- alignment.

The main application issues are

- content creation
- authoring.

Other important issues affecting the user experience are

- visual perception
- user interface
- devices
- power consumption.

Next, we review what we mean by these issues and how they affect the usability and user experience of an AR application.

An augmented reality system needs to be able to perform in real-time. Otherwise, the system may augment old or flawed information, or the augmentation may not correspond to the current state of the environment. Performance issues are characteristic to all AR algorithm and application development. Research results from other fields (e.g. image processing) are not directly applicable to AR. For instance, traditional image inpainting methods do not fulfil the real-time requirement, and therefore they cannot be used for diminished reality as such (see Section 6.2). Performance is an issue especially in mobile environment where the processing power and memory are limited.

The user should be able to interact with the system naturally. The usability and the user experience are disturbed if the interaction is unnatural. The interaction needs to be natural in the user interface level as we discussed in the Section 7.1. The same holds true at the application level; the interaction between the real world objects and virtual objects needs to be smooth as well. Application needs to adapt virtual elements according to real scene, as for example in our interior design application where the user was able to adjust virtual lights easily according to real ones (see Section 6.1.3). At times, the application needs to remove existing objects virtually to be able to augment virtual objects on the same place. We discussed in Section 6.2 how to handle this kind of interaction with diminished reality.

The camera calibration needs to be correct and the tracking needs to be accurate. Otherwise, the augmented data is shifted in the real environment: the virtual overlay is in the wrong place or it flutters. People find this alignment error annoying. In Chapter 3, we concentrated on marker-based approaches for accurate tracking, and in Chapter 4, on alternative tracking methods, mainly feature-based tracking and hybrid tracking methods. In addition, Appendix C gives an overview of camera calibration.

The content creation is also an important aspect of application development. An application can visualise information from a database (e.g. in augmented assembly) or provide textual information (e.g. in AR browsers). Sometimes the information in database is in unsuitable format and format conversion is needed. In addition, when no database is available someone needs to create the content. Furthermore, if nice graphics are required, they need to be created to the appropriate degree of accuracy and in the right format. The same content does not suit both mobile environments and high quality visualisation.

Besides content creation, authoring is a big application issue as we discussed in Section 7.4. Creation of AR applications should be brought to a non-expert non-programming level, where users can combine objects, interactions and events at a conceptual level.

Visual perception should support the purpose of the application as we discussed in Chapter 6. Some applications require (photo-)realistic rendering, other applications benefit from focus and content -type highlighting of augmented objects. The user should be able to concentrate on the task, and the visual perception should sustain the task, without distracting the user.

The user interface should be, as always, easy to use and intuitive. It should support the task at hand and make the user experience smooth as discussed in Section 7.1.

The AR application should run on the appropriate device; mobile applications on lightweight devices, high-end visualisations on larger good-quality monitors. Furthermore, the terminal device should be taken into account already at the application design stage. There is no point in implementing computationally intensive methods on mobile phones if the application would then run on a slow frame rate.

Devices often play very important role in the development process. The diversity of mobile platforms is perhaps the main obstacle for wider use of mobile AR applications. Applications need to be ported mostly to each platform separately, which deprives resources from application development. Furthermore, mobile devices are an ideal platform for consumer applications; they are equipped with cameras and new models with various additional sensors; people carry them with them all the time. Likewise, in special applications where an expert operates the system, it is feasible to invest in special devices such as HMDs, 3D displays, additional sensors, etc. if they support the task.

One more aspect that significantly affects user experience is power consumption. Many applications require the user to be able to move freely, and thus wireless devices are optimal and then battery life plays a big role. A mobile application that discharges the battery in 15 minutes is unrealistic. We once tested a HMD where the camera ran out of batteries in less than two hours. The user had to change the batteries often, which was annoying especially as the camera and projector were wired to a computer anyway. It is hard to imagine this kind of setup in practical use, e.g. in a factory.

In conclusion, *the most important issue of augmented reality application development is the user experience*, which is affected by all technological, application and other issues.

9.2 Closure

Augmented reality is an efficient visualisation technique for on-site 3D visualisation and location-based services. It is beneficial in situations where the perception *skills of a human need to be enhanced.* AR is applied in the fields of maintenance, assembly, building and construction, interior design, etc. Due to its “magical” nature, it is also suited to marketing and advertising purposes.

The universal development of the processing capacity and battery life of mobile devices, and the development of display devices together with cloud computing, will enable use of computationally demanding methods on mobile devices. Furthermore, customised solutions with special devices will provide high-end visualisations and instructions on specific tasks.

The full potential of the technology is still on the way; it largely relies on general technological development and the development of devices supporting AR. There are many great ideas waiting for someone to make those to become true. Jules Verne (1828–1905), French author and father of science fiction, once said:

“Anything one man can imagine, other men can make real.”

References

1. Kähkönen, K., Hyväkkä, J., Porkka, J., Siltanen, S. & Woodward, C. Integrating building product models with live video stream. Proceedings of 7th International Conference on Construction Applications of Virtual Reality. Penn State University, USA, 22–23 October 2007. Pp. 176–188. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.90.7713 [15 May 2012].

2. Sääski, J., Salonen, T., Hakkarainen, M., Siltanen, S., Woodward, C. & Lempiäinen, J. Integration of Design and Assembly Using Augmented Reality. In: Micro-Assembly Technologies and Applications. Vol. 260. IFIP International Federation for Information Processing. Springer Boston, 2008. Pp. 395–404. http://dx.doi.org/10.1007/978-0-387-77405-3_39 [15 May 2012].

3. Siltanen, S., Woodward, C., Valli, S., Honkamaa, P. & Rauber, A. User Interaction for Mobile Devices. In: Multimodal Processing and Interaction – Audio, Video, Text. Multimedia Systems and Applications, Vol. 33. Springer, 2008. http://www.springer.com/computer/information+systems+and+applications/book/978-0-387-76315-6 [15 May 2012].

4. Honkamaa, P., Siltanen, S., Jäppinen, J., Woodward, C. & Korkalo, O. Interactive outdoor mobile augmentation using markerless tracking and GPS. Laval, France, 2007. http://virtual.vtt.fi/multimedia/publications/aronsite-vric2007.pdf [25 May 2012].

5. Sääski, J., Salonen, T., Siltanen, S., Hakkarainen, M. & Woodward, C. Augmented reality based technologies for supporting assembly work. In: Zupanèiè, B., Karba, R. & Blažiè, S. (eds.). Proceedings of the 6th EUROSIM Congress on Modelling and Simulation. Ljubljana, Slovenia, 9–13 September 2007.

6. Salonen, T., Sääski, J., Hakkarainen, M., Kannetis, T., Perakakis, M., Siltanen, S., Potamianos, A., Korkalo, O. & Woodward, C. Demonstration of assembly work using augmented reality. Proceedings of the 6th ACM international conference on Image and video retrieval (CIVR). Amsterdam, The Netherlands, New York, NY, USA: ACM, 2007. P. 120. http://doi.acm.org/10.1145/1282280.1282301 [15 May, 2012].

7. Siltanen, S., Hakkarainen, M., Korkalo, O., Salonen, T., Sääski, J., Woodward, C., Kannetis, T., Perakakis, M. & Potamianos, A. Multimodal User Interface for Augmented Assembly. Multimedia Signal Processing (MMSP). IEEE 9th Workshop on 1–3 October 2007. P. 78.

8. Siltanen, S., Hakkarainen, M. & Honkamaa, P. Automatic Marker Field Calibration. Virtual Reality International Conference (VRIC). Laval, France, 18–20 April 2007. P. 261.

9. Woodward, C., Lahti, J., Rönkkö, J., Honkamaa, P., Hakkarainen, M., Siltanen, S. & Hyväkkä, J. Case Digitalo – A range of virtual and augmented reality solutions in construction application. Proceedings of 24th CIB W78 Conference, 26–29 June 2007. P. 26.

10. Siltanen, S. Texture generation over the marker area. IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR). 22–25 October 2006. P. 253.

11. Siltanen, S. & Hyväkkä, J. Implementing a natural user interface for camera phones using visual tags. In: Piekarski, W. (ed.). 7th Australasian User Interface Conference (AUIC); CRPIT. Vol. 50. ACS, 2006. P. 113.

12. Siltanen, S. & Woodward, C. Augmented interiors with digital camera images. In: Piekarski, W. (ed.). 7th Australasian User Interface Conference (AUIC); CRPIT. Vol. 50. ACS, 2006. P. 33.

13. Woodward, C., Lahti J., R., J., Honkamaa, P., Hakkarainen, M., Jäppinen, J., Rainio, K., Siltanen, S. & Hyväkkä, J. Virtual and augmented reality in the Digitalo building project. International Journal of Design Sciences and Technology 2007, Vol. 14, No. 1, pp. 23–40.

14. Korkalo, O., Aittala, M. & Siltanen, S. Light-weight marker hiding for augmented reality. 9th IEEE International Symposium on Mixed and Augmented Reality (ISMAR). Seoul, Korea, 13–16 October 2010. Pp. 247–248

15. Azpiazu, J., Siltanen, S., Multanen, P., Mäkiranta, A., Barrena, N., Díez, A., Agirre, J. & Smith, T. Remote support for maintenance tasks by the use of Augmented Reality: the ManuVAR project. 9th Congress on Virtual Reality Applications (CARVI). Vitoria-Gasteiz, Spain, 10–11 November 2011.

16. Oksman, V., Kulju, M. & Siltanen, S. Designing Creative Tools: A User-Centered Approach to Ambient Home Design. Amsterdam, Netherlands, 16 November 2011.

17. Ailisto, H., Korhonen, I., Tuomisto, T., Siltanen, S., Strömmer, E., Pohjanheimo, L., Hyväkkä, J., Välkkynen, P., Ylisaukko-oja, A. & Keränen, H. Physical Browsing for Ambient Intelligence (PB-AmI). Espoo, Finland: VTT, 2007.

18. Pinto Seppä, I., Porkka, J., Valli, S., Siltanen, S., Lehtinen, E. & Teerimo, S. Dwelling Renovation Studio. Interactive tool for private residences renovation service. Research report No. VTTR0145807. Espoo, Finland: VTT, 2007.

19. ALVAR – A Library for Virtual and Augmented Reality. www.vtt.fi/multimedia/ alvar.html [27 April 2012].

20. Augmented reality in: Encyclopædia Britannica 2010. http://www.britannica.com/ EBchecked/topic/1196641/augmented-reality [15 May 2012].

21. Caudell, T.P. & Mizell, D.W. Augmented reality: an application of heads-up display technology to manual manufacturing processes., 1992. Proceedings of the 25th Hawaii International Conference on System Sciences. Vol. ii. 1992. P. 659.

22. Milgram, P., Takemura, H., Utsumi, A. & Kishino, F. Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum. Proceedings of SPIE, Vol. 2351, Telemanipulator and Telepresence Technologies, Hari Das; Ed. 1994. Pp. 282–292.

23. Azuma, R. A Survey of augmented reality. Presence: Teleoperators and Virtual Environments 6, 1997, pp. 355–385.

24. Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S. & MacIntyre, B. Recent advances in augmented reality. IEEE Computer Graphics and Applications 2001, Vol. 21, No. 6, pp. 34–47.

25. Mann, S. Mediated Reality with implementations for everyday life. Presence 2002, Teleoperators and Virtual Environments.

26. Gartner Research Methodologies – Hype Cycles. Gartner 2010. http://www. gartner.com/it/products/research/methodologies/research_hype.jsp.[27 April 2012].

27. Fenn, J. & LeHong, H. Hype Cycle for Emerging Technologies, 2011. Publication Date: 28 July 2011; ID Number: G00215650. Gartner, 2011. http://community.mis.temple.edu/mis2501sec001f11/files/2011/08/Hype-Cycle-for-Emerging-Technologies-2011.pdf. [27 April 2012].

28. 10 Emerging Technologies 2007. Technology Review. MIT 2007. http://www. technologyreview.com/specialreports/specialreport.aspx?id=19. [15 May 2012].

29. Fenn, J. 2010 Emerging Technologies Hype Cycle is Here – a blog writing. http://blogs.gartner.com/hypecyclebook/2010/09/07/2010-emerging-technologies-hype-cycle-is-here [27 April 2012].

30. The New Media Consortium. http://www.nmc.org/about [27 April 2012].

31. Johnson, L., Smith, R., Levine, J. & Haywood, K. 2010 Horizon Report: K-12 Edition. 2010. Austin, Texas: The New Media Consortium.

32. VividWorks Vivid Works – Vivid Platform. http://www.vividworks.fi [27 April 2012].

33. Honkamaa, P., Jäppinen, J. & Woodward, C. A lightweight approach for augmented reality on camera phones using 2D images to simulate 3D. Proceedings of the 6th international conference on Mobile and ubiquitous multimedia (MUM). Oulu, Finland; New York, NY, USA: ACM, 2007. Pp. 155–159.

34. Woodward, C., Hakkarainen, M., Korkalo, O., Kantonen, T., Aittala, M., Rainio, K. & Kahkonen, K. Mixed reality for mobile construction site visualization and communication. In: Makanae, Yabuki & Kashiyama (eds.). Proceedings of the 10th International Conference on Construction Applications of Virtual Reality (CONVR). Sendai, Japan, 4–5 November 2010. Pp. 35–44.

35. Billinghurst, M., Hakkarainen, M. & Woodward, C. Augmented assembly using a mobile phone. Proceedings of the 7th International Conference on Mobile and Ubiquitous Multimedia (MUM). Umeå, Sweden; New York, NY, USA: ACM, 2008. Pp. 84–87.

36. Salonen, T., Sääski, J., Woodward, C., Korkalo, O., Marstio, I. & Rainio, K. Data pipeline from CAD to AR based assembly instructions. ASME Conference Proceedings, Vol. 2009, No. 43376, pp. 165–168.

37. Savioja, P., Järvinen, P., Karhela, T., Siltanen, P. & Woodward, C. Developing a mobile, service-based augmented reality tool for modern maintenance work. Proceedings of the 2nd international conference on Virtual reality (ICVR). Beijing, China, Berlin, Heidelberg: Springer-Verlag, 2007. Pp. 554–563.

38. Henderson, S. & Feiner, S. Exploring the Benefits of Augmented Reality Documentation for Maintenance and Repair. IEEE Transactions on Visualization and Computer Graphics, Vol. 17, No. 10, Pp. 1355–1368.

39. Henderson, S. & Feiner, S. Augmented reality in the psychomotor phase of a procedural task. Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2011. Pp. 191–200.

40. Henderson, S.J. & Feiner, S. Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret. Mixed and Augmented Reality, IEEE/ACM International Symposium on 2009, pp. 135–144.

41. Mobile smartphone games company. Mobile Augmented Reality 2012. http://www.mobile-augmented-reality.com [27 April 2012].

42. Billinghurst, M., Kato, H. & Poupyrev, I. The MagicBook – Moving Seamlessly between reality and virtuality. IEEE Comput. Graph. Appl. 2001, Vol. 21, No. 3, pp. 6–8.

43. Evans, C. Aliens and UFOs. Carlton books, 2008.

44. Mash, R. Dinosaurs Alive! (Augmented Reality Book). Carlton Books, 2010.

45. Moffet, P. Fairyland Magic (Augmented Reality Book). Carlton Books, 2010.

46. Lehtinen, T., Mäkijärvi, M. & Inkeri, E. Dibitassut ja kadonnut prinsessa. Kustannusosakeyhtiö Paasilinna, 2010.

47. Green, R. The Magic of Christmas by Santa. Carlton Books, 2010.

48. ALTAIR Robotics Lab (A Laboratory for Teleoperation and Autonomous Intelligent Robots). http://metropolis.sci.univr.it [27 April 2012].

49. Bichlmeier, C., Heining, S.M., Rustaee, M. & Navab, N. Laparoscopic virtual mirror for understanding vessel structure evaluation study by twelve surgeons. Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Nara-Ken New Public Hall, Nara, Japan, 13–16 November 2007. Pp. 125–128.

50. Shekhar, R., Dandekar, O., Bhat, V., Philip, M., Lei, P., Godinez, C., Sutton, E., George, I., Kavic, S., Mezrich, R. & Park, A. Live augmented reality: a new visualization method for laparoscopic surgery using continuous volumetric computed tomography. Surgical endoscopy 2010, Vol. 24, No. 8, Pp. 1976–1985.

51. Rhienmora, P., Gajananan, K., Haddawy, P., Dailey, M.N. & Suebnukarn, S. Augmented reality haptics system for dental surgical skills training. Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology (VRST). Hong Kong; New York, NY, USA: ACM, 2010. Pp. 97–98.

52. Kim, K. & Park, J. Virtual bone drilling for dental implant surgery training. Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology (VRST). Kyoto, Japan; New York, NY, USA: ACM, 2009. Pp. 91–94. http://doi.acm.org/10.1145/1643928.1643950 [15 May 2012].

53. International Organization for Standardization: ISO FDIS 9241-210:2009. Ergonomics of human system interaction – Part 210: Human-centered design for interactive systems (formerly known as 13407). Switzerland: ISO 2009.

54. LookTel: electronic assistant for visually impaired and blind. http://www.looktel.com [27 April 2012].

55. Järvinen, P., Järvinen, T.H., Lähteenmäki, L. & Södergård, C. HyperFit: Hybrid media in personal nutrition and exercise management. Second International Conference on Pervasive Computing Technologies for Healthcare, 2008. Pervasive Health 2008.. 2008. Pp. 222–226.

56. Meijer, P.B.L. The vOICe for Android – see with sound. http://www.seeingwithsound.com [27 April 2012].

57. Toozla – An Audio Augmneted Reality Browser. http://toozla.com [27 April 2012].

58. Tikander, M. Modeling the attenuation of a loosely-fit insert headphone for augmented reality audio. Audio Engineering Society Conference: 30th International Conference: Intelligent Audio Environments. Saariselkä, Finland, 15–17 March 2007.

59. Heilig, M.L. Sensorama simulator. Aug. 1962, No. 3050870. http://www.freepatentsonline.com/3050870.html [27 April 2012].

60. Isacsson, A., Alakoski, L. & Bäck, A. Using Multiple Senses in Tourism Marketing: The Helsinki Expert, Eckero Line and Linnanmäki Amusement Park Cases. Germany: University Library of Munich, 2009. An International Multidisciplinary Journal of Tourism 15 November 2009, Vol. 4, No. 3, pp. 167–184.

61. Pömpeli – A multiple sensory space. 2010. http://www.pompeli.fi/enindex.html [2010].

62. Narumi, T., Kajinami, T., Tanikawa, T. & Hirose, M. Meta cookie. ACM SIGGRAPH, Emerging Technologies. Los Angeles, CA; New York, NY, USA: ACM, 2010. P. 1.

63. OneZeroThrice ARtisan – Augmented Reality Solution for FLARToolkit and Papervision3D. 2010. http://onezerothrice.com/artisan [2011].

64. ARToolKit website. 2006, 4/20. http://www.hitl.washington.edu/artoolkit [27 April 2012].

65. ARtoolworks ARToolworks – ARToolKit product family. 2010. http://www.artool works.com/products [15 May 2012].

66. CellaGameS CellaGameS: SMMT library – SLAM Multimarker Tracker for Symbian. 2010. Available: http://cellagames.com/smmt.html [1 October 2010].

67. Georgia Tech The Designer's Augmented Reality Toolkit (DART). http://www. cc.gatech.edu/dart/aboutdart.htm [27 April 2012].

68. Celozzi, C., Paravati, G., Sanna, A. & Lamberti, F. A 6-DOF ARTag-based tracking system. Consumer Electronics, IEEE Transactions on 2010, Vol. 56, No. 1, pp. 203–210.

69. Jun, J., Yue, Q. & Qing, Z. An Extended Marker-Based Tracking System for Augmented Reality. International Conference on Modeling, Simulation and Visualization Methods, 2010, pp. 94–97. http://doi.ieeecomputer society.org/10.1109/WMSVM.2010.52 [15 May 2012].

70. Santos, P., Stork, A., Buaes, A., Pereira, C. & Jorge, J. A real-time low-cost marker-based multiple camera tracking solution for virtual reality applications. Journal of Real-Time Image Processing 2010, Vol. 5, No. 2, pp. 121–128. http://dx.doi.org/10.1007/s11554-009-0138-9 [15 May 2012].

71. ARTag Augmeneted Reality system. http://www.artag.net [27 April 2012].

72. Hartley, R.I. & Zisserman, A. Multiple View Geometry in Computer Vision. Cambridge University Press, 2000.

73. Gonzalez, R.C. & Woods, R.E. Digital image processing. Boston, MA, USA: Addison-Wesley Longman, 2001.

74. Szeliski, R. Computer Vision: Algorithms and Applications. New York, NY, USA: Springer-Verlag, 2010. http://research.microsoft.com/en-us/um/ people/szeliski/book/drafts/szelski_20080330am_draft.pdf. [4 January 2012].

75. Pintaric, T. An adaptive thresholding algorithm for the augmented reality toolkit. 2003, in Proc. of the 2nd IEEE International Augmented Reality Toolkit Workshop (ART), 2003.

76. Hirzer, M. Marker detection for augmented reality applications. Austria: Inst. for Computer Graphics and Vision, Graz University of Technology, 2008.

77. Uematsu, Y. & Saito, H. Improvement of accuracy for 2D marker-based tracking using particle filter. Proceedings of the 17th International Conference on Artificial Reality and Telexistence (ICAT). Washington, DC, USA: IEEE Computer Society, 2007. Pp. 183–189.

78. Li, S. & Xu, C. Efficient lookup table based camera pose estimation for augmented reality. Computer Animation and Virtual Worlds 2011, Vol. 22, No. 1, pp. 47–58. http://dx.doi.org/10.1002/cav.385 [15 May 2012].

79. Freeman, R.M., Juliet, S.J. & Steed, A.J. A method for predicting marker tracking error. Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Nara-Ken New Public Hall, Nara, Japan, 13–16 Nov. 2007. Pp. 157–160.

80. Okumura, B., Kanbara, M. & Yokoya, N. Precise geometric registration by blur estimation for vision-based augmented reality. Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2007. Pp. 1–4. http://dx.doi.org/10.1109/ISMAR.2007.4538851.

81. Park, Y., Lepetit, V. & Woo, W. ESM-Blur: Handling & rendering blur in 3D tracking and augmentation. Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2009. Pp. 163. http://dx.doi.org/10.1109/ISMAR.2009.5336480 [15 May 2012].

82. ARToolKit API Documentation. http://artoolkit.sourceforge.net/apidoc [1 October 2010].

83. Kato, H. Inside ARToolKit. http://www.hitl.washington.edu/artoolkit/Papers/ART02-Tutorial.pdf [1 October 2010].

84. Wagner, D. Handheld Augmented Reality. Dissertation. Institute for Computer Graphics and Vision, Graz University of Technology, 2007.

85. Schmalstieg, D. & Wagner, D. Experiences with Handheld Augmented Reality. Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Nara-Ken New Public Hall, Nara, Japan, 13–16 November 2007. Pp. 3–18.

86. Lu, C.-P., Hager, G.D. & Mjolsness, E. Fast and globally convergent pose estimation from video images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, Vol. 22, No. 6, pp. 610–622.

87. Schweighofer, G. Robust Pose Estimation from a Planar Target. IEEE Transactions on Pattern Analysis and Machine Intelligence 2006, Vol. 28, No. 12, pp. 2024–2030. http://dx.doi.org/10.1109/TPAMI.2006.252 [15 May 2012].

88. Irani, M. & Peleg, S. Super resolution from image sequences. Proceedings of the 10th International Conference on Pattern Recognition, 1990. Vol. 2. Pp. 115–120.

89. Park, S.C., Park, M.K. & Kang, M.G. Super-Resolution Image Reconstruction: A Technical Overview. IEEE Signal Processing Magzine 2003, Vol. 20, No. 3. Pp. 21–36.

90. Smith, R.C. & Cheeseman, P. On the representation and estimation of spatial uncertainly. Int. J. Rob. Res. 1987, Vol. 5, No. 4, pp. 56–68. http://dx.doi.org/10.1177/027836498600500404 [25 May 2012].

91. Welch, G. & Bishop, G. SCAAT: Incremental tracking with incomplete information. Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH). New York, NY, USA: ACM Press/Addison-Wesley, 1997. Pp. 333–344. http://doi.acm.org/10.1145/258734.258876 [15 May 2012].

92. Azuma, R. Performance analysis of an outdoor augmented reality tracking system that relies upon a few mobile beacons. IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR), Santa Barbara, USA, 2006. Pp. 101–104.

93. Rasmussen, N.T., Störring, M., Moeslund, T.B. & Granum, E. Real-time tracking for virtual environments using scaat kalman filtering and unsynchronised cameras. Proceedings of International Conference on Computer Vision Theory and Applications. Setúbal, Portugal, 25–28 February 2006.

94. Lepetit, V., Moreno-Noguer, F. & Fua, P. EPnP: An Accurate O(n) Solution to the PnP Problem. Int. J. Comput.Vision 2009, Vol. 81, No. 2, pp. 155–166.

95. Ansar, A. & Daniilidis, K. Linear Pose Estimation from Points or Lines. IEEE Transactions on Pattern Analysis and Machine Intelligence 2003, Vol. 25, pp. 282–296.

96. StudierStube Tracker. http://studierstube.icg.tugraz.at/handheld_ar/stbtracker.php [27 April 2012].

97. Schall, G., Fraundorfer, F., Newman, J. & Schmalstieg, D. Construction and Maintenance of Augmented Reality Environments Using a Mixture of Autonomous and Manual Surveying Techniques. 7th Conference on Optical 3-D Measurement Techniques. Vienna, Austria, 3–5 October 2005.

98. Ying Kin Yu, Kin Hong Wong & Chang, M.M.Y. Recursive three-dimensional model reconstruction based on Kalman filtering. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 2005, Vol. 35, No. 3, pp. 587–592.

99. Davison, A.J. Real-Time Simultaneous Localisation and Mapping with a Single Camera. Proc. International Conference on Computer Vision (ICCV). Nice, France, 13–16 October 2003.

100. Davison, A.J., Reid, I.D., Molton, N.D. & Stasse, O. MonoSLAM: Real-Time Single Camera SLAM. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 29, No. 6, pp. 1052–1067.

101. Lourakis, M.I.A. & Argyros, A.A. SBA: A software package for generic sparse bundle adjustment. ACM Trans.Math.Softw. 2009, Vol. 36, No. 1, pp. 1–30. http://doi.acm.org/10.1145/1486525.1486527 [15 May 2012].

102. Fiala, M. ARTag, a Fiducial Marker System Using Digital Techniques. Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 2. Washington, DC, USA 2005. Pp. 590–596.

103. Hamming, R. Error detecting and error correcting codes. Bell Systems Technology Journal 1950, Vol. 29, pp. 147–160.

104. Reed, I.S. & Solomon, G. Polynomial Codes Over Certain Finite Fields. Journal of the Society for Industrial and Applied Mathematics 1960, Vol. 8, No. 2, pp. 300–304.

105. Welch, L.R. The Original View of Reed-Solomon Codes. Lecture Notes, 1997. http://csi.usc.edu/PDF/RSoriginal.pdf [27 April 2012].

106. TEC-IT 2D Barcode: Data Matrix ECC200. http://www.tec-it.com/en/support/knowbase/symbologies/datamatrix/Default.aspx [27 April 2012].

107. TEC-IT 2D Barcode: QR-Code. http://www.tec-it.com/en/support/knowbase/symbologies/qrcode/Default.aspx [15 May 2012].

108. UpCode Mobile Solutions. http://www.upcode.com [26 April 2012].

109. Naimark, L. & Foxlin, E. Circular Data Matrix Fiducial System and Robust Image Processing for a Wearable Vision-Inertial Self-Tracker. Proceedings of the 1st International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2002. Pp. 27–36.

110. Madhavapeddy, A., Scott, D., Sharp, R. & Upton, E. Using Camera-Phones to Enhance Human-Computer Interaction. 6th International Conference on Ubiquitous Computing (Adjunct Proceedings: Demos). Nottingham, UK, 7–10 September 2004.

111. CVLab EFPL BazAR: A vision based fast detection library. http://cvlab.epfl.ch/software/bazar/index.php [27 April 2012].

112. Lepetit, V., Lagger, P. & Fua, P. Randomized Trees for Real-Time Keypoint Recognition. Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 2. Washington, DC, USA: IEEE Computer Society. San Diego, CA, USA, 20–26 June 2005. Pp. 775–781.

113. Lepetit, V. & Fua, P. Keypoint recognition using randomized trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 28, No. 9, 2006, pp. 1465–1479.

114. Kamijo, K., Minami, M. & Morikawa, H. An invisible hyperlink marker. Signal Processing and Communication Systems (ICSPCS). 3rd International Conference. Omaha, NE, USA, 28–30 September 2009. Pp. 1–10.

115. Kodak Introduces Virtually Invisible Ink for the NEXPRESS Production Presses, Opening Up New Security and Inventory Management Applications. Kodak News, 13 May 2010 http://graphics.kodak.com/US/en/About_GCG/News/2010/100513a.htm [27 April 2012].

116. Park, H. & Park, J. Invisible marker based augmented reality system. Visual communications and image processing. Beijing, Chine, 12–15 July 2005, Vol. 5960.

117. Lee, H.R., Shin, J.S. & Hwang, C.J. Invisible Marker Tracking System Using Image Watermarking for Augmented Reality. IEEE International Conference on Consumer Electronics (ICCE). Digest of Technical Papers. Las Vegas, NV, USA. 12–14 January 2007. Pp. 1–2.

118. Matsushita, N., Hihara, D., Ushiro, T., Yoshimura, S., Rekimoto, J. & Yamamoto, Y. ID CAM: A Smart Camera for Scene Capturing and ID Recognition. Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR). Tokyo, Japan, 7–10 October 2003. Washington, DC, USA: IEEE Computer Society. P. 227.

119. Dongdong Weng, Yue Liu, Yongtian Wang & Lun Wu Study on an Indoor Tracking System Based on Primary and Assistant Infrared Markers. 10th IEEE International Conference on Computer-Aided Design and Computer Graphics, 2007. Beijing, China. 15–18 October 2007. Pp. 377–382.

120. Nakazato, Y., Kanbara, M. & Yokoya, N. Wearable augmented reality system using invisible visual markers and an IR camera. 9th IEEE International Symposium on Wearable Computers. 2005. Pp. 198–199.

121. Nakazato, Y., Kanbara, M. & Yokoya, N. A Localization System Using Invisible Retro-reflective Markers. Proc. IAPR Conference on Machine Vision Applications (MVA). 2005. Pp. 140–143.

122. Nota, Y. & Kono, Y. Augmenting Real-world Objects by Detecting “Invisible” Visual Markers. 21st ACM Symposium on User Interface Software and Technology (UIST). Monterey, CA, USA, 19–22 October 2008. Pp. 39–40.

123. Wang, H., Liu, W., Chang, C. & Chen, Y. Design of Halftone-Based AR Markers under Infrared Detection. Proceedings of the International Conference on Computer Science and Software Engineering (CSSE). Washington, DC, USA: IEEE Computer Society, 2008. Pp. 97–100.

124. Wang, T., Liu, Y. & Wang, Y. Infrared Marker Based Augmented Reality System for Equipment Maintenance. Proceedings of the 2008 International Conference on Computer Science and Software Engineering (CSSE), Vol. 5. Washington, DC, USA: IEEE Computer Society, 2008. Pp. 816–819.

125. Li Zhang, Subramaniam, N., Lin, R., Raskar, R. & Shree Nayar Capturing Images with Sparse Informational Pixels using Projected 3D Tags. Virtual Reality Conference (VR). IEEE, 2008. Pp. 11–18.

126. Grundhofer, A., Seeger, M., Hantsch, F. & Bimber, O. Dynamic Adaptation of Projected Imperceptible Codes. Mixed and Augmented Reality, IEEE/ACM International Symposium on 2007, Vol. 0, pp. 1–10.

127. Mohan, A., Woo, G., Hiura, S., Smithwick, Q. & Raskar, R. Bokode: imperceptible visual tags for camera based interaction from a distance. ACM Trans.Graph. 2009, July, Vol. 28, No. 3, pp. 98:1–98:8.

128. Mäkinen, J., Keränen, K., Hakkarainen, J., Silvennoinen, M., Salmi, T., Syrjälä, S., Ojapalo, A., Schorpp, M., Hoskio, P. & Karioja, P. Inmould integration of a microscope add-on system to a 1.3 Mpix camera phone. Optical Sensing Technology and Applications. In: F. Baldini, J. Homola, R.A. Lieberman & M. Miler (eds.). Optical Sensing Technology and Applications Conference, Prague, Czech Republic, 16–19 April 2007. SPIE Proceedings 2007, Vol. 6585. P. 658507.

129. Mäkinen, J.-T., Niemelä, K., Vasama, H., Mattila, R., Aikio, M., Aikio, S. & Aikio, J. Add-on laser reading device for a camera phone. In: Building European OLED Infrastructure. Edited by Pearsall, Thomas P.; Halls, Jonathan. Proceedings of the SPIE, Vol. 5962. 2005. Pp. 685–695. http://adsabs.harvard.edu/abs/2005SPIE.5962..685M [16 May 2012].

130. Adams, A., Talvala, E., Park, S.H., Jacobs, D.E., Ajdin, B., Gelfand, N., Dolson, J., Vaquero, D., Baek, J., Tico, M., Lensch, H.P.A., Matusik, W., Pulli, K., Horowitz, M. & Levoy, M. The Frankencamera: an experimental platform for computational photography. ACM Trans.Graph. 2010, July, Vol. 29, No. 4, pp. 29:1–12.

131. Adams, A., Talvala, E., Park, S.H., Jacobs, D.E., Ajdin, B., Gelfand, N., Dolson, J., Vaquero, D., Baek, J., Tico, M., Lensch, H.P.A., Matusik, W., Pulli, K., Horowitz, M. & Levoy, M. The Frankencamera: an experimental platform for computational photography. (ACM SIGGRAPH) papers. Los Angeles, California, New York, NY, USA: ACM, 2010. Pp. 29:1–12.

132. Dünser, A., Looser, J., Grasset, R., Seichter, H. & Billinghurst, M. Evaluation of Tangible User Interfaces for Desktop AR. International Symposium on Ubiquitous Virtual Reality 2010. Gwangju, Korea, 7–10 July 2010. Pp. 36–39.

133. Looser, J., Grasset, R. & Billinghurst, M. A 3D Flexible and Tangible Magic Lens in Augmented Reality. Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Pp. 51–54.

134. JESS3 Data visualization agency JESS3. Available: http://jess3.com [27 April 2012].

135. Klein, G. & Murray, D. Parallel Tracking and Mapping for Small AR Work-spaces. Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Nara-Ken New Public Hall, Nara, Japan, 13–16 November 2007. Pp. 225–234.

136. Rosten, E., Porter, R. & Drummond, T. Faster and Better: A Machine Learning Approach to Corner Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 2010, Vol. 32, pp. 105–119.

137. Rosten, E., Porter, R. & Drummond, T. Faster and better: A machine learning approach to corner detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 2009.

138. Drab, S.A. & Artner, N.M. Motion Detection as Interaction Technique for Games & Applications on Mobile Devices. Pervasive Mobile Interaction Devices (PERMID) Workshop at the Pervasive, Munich, Germany, 11 May 2005. Pp. 48–51.

139. Keitler, P., Pankratz, F., Schwerdtfeger, B., Pustka, D., Rodiger, W., Klinker, G., Rauch, C., Chathoth, A., Collomosse, J. & Song, Y. Mobile augmented reality based 3D snapshots. Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2009. Pp. 199–200.

140. Hu, M. Visual pattern recognition by moment invariants. IEEE Transactions on Information Theory, Vol. 8, No. 2, February 1962, pp. 179–187.

141. Harris, C. & Stephens, M. A combined corner and edge detector. 4th Alvey Vision Conference, 1988. Pp. 147–151.

142. Mikolajczyk, K. & Schmid, C. Scale and Affine Invariant Interest Point Detectors. Int.J.Comput.Vision 2004, Vol. 60, No. 1, pp. 63–86.

143. Liu, M., Wu, C. & Zhang, Y. Multi-resolution optical flow tracking algorithm based on multi-scale Harris corner points feature. Control and Decision Conference (CCDC), July 2008. Pp. 5287–5291.

144. Mikolajczyk, K. & Schmid, C. An affine invariant interest point detector. Proceedings of the European Conf. Computer Vision. Springer Verlag, 2002. Pp. 128–142.

145. Matas, J., Chum, O., Urban, M. & Pajdla, T. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In: British Machine Vision Conference. 2002. Pp. 384–393.

146. Method for digitally processing images to determine the position of edges and/or corners therein for guidance of unmanned vehicle. (Smith, S.M. & Brady, J.M.) Patent No: UK Patent 2272285. 1997.

147. Smith, S.M. & Brady, J.M. SUSAN – A new approach to low level image processing. International Journal of Computer Vision 1997, Vol. 23, No. 1, pp. 45–78.

148. Rosten, E. & Drummond, T. Machine learning for high-speed corner detection. European Conference on Computer Vision, Vol. 1, May 2006. Pp. 430–443.

149. Rosten, E. & Drummond, T. Fusing points and lines for high performance tracking. IEEE International Conference on Computer Vision, Vol. 2. October 2005. Pp. 1508–1511.

150. Rosten, E. & Drummond, T. Machine learning for high-speed corner detection. In European Conference on Computer Vision. Springer, 2006. Pp. 430–443.

151. Rosten, E. FAST Corner Detection (10 November 2010 – last update). http://svr-www.eng.cam.ac.uk/~er258/work/fast.html.

152. Mokhtarian, F. & Mohanna, F. Performance evaluation of corner detectors using consistency and accuracy measures. Computer Vision and Image Understanding 2006, 4, Vol. 102, No. 1, pp. 81–94.

153. Di Stefano, L., Mattoccia, S. & Tombari, F. Speeding-up NCC-Based Template Matching Using Parallel Multimedia Instructions. Proceedings of the 7th International Workshop on Computer Architecture for Machine Perception (CAMP). Washington, DC, USA: IEEE Computer Society, 2005. Pp. 193–197.

154. Lucas, B.D. & Kanade, T. An Iterative Image Registration Technique with an Application to Stereo Vision. Proceedings of the 7th international joint conference on Artificial intelligence (IJCAI), Vol. 2, 1981. Pp. 674–679.

155. Bouguet, J.Y. Pyramidal implementation of the Lucas Kanade feature tracker. Intel Corporation, Microprocessor Research Labs, 1999.

156. Ozuysal, M., Fua, P. & Lepetit, V. Fast keypoint recognition in ten lines of code. IEEE Conference on Computer Vision and Pattern Recognition, (CVPR). 2007. Pp. 1–8.

157. Calonder, M., Lepetit, V. & Fua, P. Keypoint Signatures for Fast Learning and Recognition. Proceedings of the 10th European Conference on Computer Vision (ECCV), Part I. Marseille, France, Berlin, Heidelberg: Springer-Verlag, 2008. Pp. 58–71.

158. Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. Int.J.Comput.Vision 2004, Vol. 60, No. 2, pp. 91–110. Hingham, MA, USA: Kluwer Academic Publishers.

159. Lowe, D.G. Object Recognition from Local Scale-Invariant Features. Proceedings of the International Conference on Computer Vision (ICCV), Vol. 2. Washington, DC, USA: IEEE Computer Society, 1999. Pp. 1150–1157.

160. Ke, Y. & Sukthankar, R. PCA-SIFT: a more distinctive representation for local image descriptors. 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2. 2004. Pp. 506–513. http://dx.doi.org/10.1109/CVPR.2004.1315206 [16 May 2012].

161. Mikolajczyk, K. & Schmid, C. A Performance Evaluation of Local Descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence 2005, Vol. 27, No. 10, pp. 1615–1630.

162. Bay, H., Tuytelaars, T. & Van Gool, L. SURF: Speeded Up Robust Features. In: Anonymous. Computer Vision, ECCV 2006. Pp. 404–417.

163. Sarfraz, M.S. & Hellwich, O. On Head Pose Estimation in Face Recognition. In: Anonymous Computer Vision and Computer Graphics. Theory and Applications. Vol. 24. Springer Berlin Heidelberg, 2009. Pp. 162–175. (Communications in Computer and Information Science).

164. Sarfraz, M.S. & Hellwich, O. Head Pose Estimation in Face Recognition Across Pose Scenarios. Int. Conference on Computer Vision Theory and Applications (VISAPP), Vol. 1, January 2008. Pp. 235–242.

165. Mikolajczyk, K. & Schmid, C. A performance evaluation of local descriptors. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003, Vol. 2. Pp. 257–263.

166. Kitchen, L. & Rosenfeld, A. Gray-level corner detection. Pattern Recognition Letters 1982, 12, Vol. 1, No. 2, pp. 95–102.

167. Harris, C.G. Determination of ego-motion from matched points. Alvey Vision Conference, Cambridge, UK, 1987.

168. Mokhtarian, F. & Suomela, R. Robust Image Corner Detection Through Curvature Scale Space. IEEE Transactions on Pattern Analysis and Machine Intelligence 1998, Vol. 20, pp. 1376–1381.

169. Klein, G. & Murray, D. Parallel Tracking and Mapping on a camera phone. Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2009. Pp. 83–86..

170. Klein, G. Parallel Tracking and Mapping for Small AR Workspaces. 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, (ISMAR). Pp. 225–234. http://www.robots.ox.ac.uk/~gk/PTAM [27 April 2012].

171. Durrant-Whyte, H. & Bailey, T. Simultaneous Localisation and Mapping (SLAM): Part I, the essential algorithms. Robotics and Automation Magazine 2006, Vol. 13, No. 2, pp. 99–100.

172. Montemerlo, M., Thrun, S., Koller, D. & Wegbreit, B. FastSLAM2.0: An improved particle filtering algorithm for simultaneous localization and mapping that probably converges. International Joint Conference on Artificial Intelligence 2003. Pp. 1151–1156.

173. Davison, A.J. Real-Time Simultaneous Localisation and Mapping with a Single Camera. Proc Internation Conference on Computer Vision (ICCV). Nice, France, 13–16 October 2003. P. 1403.

174. Eade, E. & Drummond, T.W. Edge landmarks in monocular SLAM. The 17th British Machine Vision Conference (BMVC) August 2006. Pp. 469–476

175. Eade, E. & Drummond, T. Scalable Monocular SLAM. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. June 2006, Vol. 1, pp. 469–476.

176. Mori, S., Hida, K. & Sawada, K. Advanced car positioning method using infrared beacon. 8th International Conference on ITS Telecommunications (ITST), October 2008. Pp. 45–50.

177. Ribo, M., Lang, P., Ganster, H., Brandner, M., Stock, C. & Pinz, A. Hybrid tracking for outdoor augmented reality applications. Computer Graphics and Applications, IEEE, 2002, Vol. 22, No. 6, pp. 54–63. 178. In: Uchiyama, S., Takemoto, K., Satoh, K., Yamamoto, H. & Tamura, H. MR Platform: A Basic Body on Which Mixed Reality Applications Are Built. ISMAR 2002, Vol. 00. P. 246.

179. Jiang, B., Neumann, U. & Suya Y. A robust hybrid tracking system for outdoor augmented reality. Proceedings of Virtual Reality, March 2004. Pp. 3–275.

180. Reitmayr, G. & Drummond, T.W. Initialisation for Visual Tracking in Urban Environments. 2007. Pp. 161–172.

181. Hol, J.D., Schon, T.B., Gustafsson, F. & Slycke, P.J. Sensor Fusion for Augmented Reality. 9th International Conference on Information Fusion, July 2006. Pp. 1–6.

182. Wonwoo Lee, Youngmin Park, Lepetit, V. & Woontack Woo Point-and-shoot for ubiquitous tagging on mobile phones. 9th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), October 2010. Pp. 57–64.

183. Kurz, D. & Ben Himane, S. Inertial sensor-aligned visual feature descriptors. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2011. Pp. 161–166.

184. Kurz, D. & BenHimane, S. Gravity-Aware Handheld Augmented Reality. Proceedings IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Basel, Switzerland, 26–29 October 2011. Pp. 111–120.

185. Comport, A.I., Marchand, E. & Chaumette, F. A real-time tracker for markerless augmented reality. Proceedings of the 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2003. Pp. 36–45.

186. Bleser, G., Wuest, H. & Stricker, D. Online camera pose estimation in partially known and dynamic scenes. IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR), October 2006. Pp. 56–65.

187. Kotake, D., Satoh, K., Uchiyama, S. & Yamamoto, H. A Fast Initialization Method for Edge-based Registration Using an Inclination Constraint. 2007. Pp. 239–248.

188. Reitmayr, G. & Drummond, T.W. Going out: robust model-based tracking for outdoor augmented reality. October 2006. Vol. Mixed and Augmented Reality. IEEE/ACM, 2006. Pp. 109–118.

189. Civera, J., Davison, A.J. & Montiel, J.M. Dimensionless Monocular SLAM. Proceedings of the 3rd Iberian conference on Pattern Recognition and Image Analysis (IbPRIA), Part II. Girona, Spain. Berlin, Heidelberg: Springer-Verlag, 2007. Pp. 412–419.

190. Fischer, J., Haller, M. & Thomas, B. Stylized Depiction in Mixed Reality. International Journal of Virtual Reality 2008, December, Vol. 7, No. 4, pp. 71–79.

191. Fischer, J., Bartz, D. & Stra\sser, W. Reality Tooning: Fast Non-Photorealism for Augmented Video Streams (poster). Proceedings of IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Vienna, Austria, October 2005. Pp. 186–187.

192. Aittala, M. Inverse lighting and photorealistic rendering for augmented reality. The Visual Computer 2010, Vol. 26, No. 6, pp. 669–678.

193. Pilet, J., Lepetit, V. & Fua, P. Retexturing in the Presence of Complex Illumination and Occlusions. Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2007. Pp. 1–8.

194. Scheer, F., Abert, O. & Müller, S. Towards Using Realistic Ray Tracing in Augmented Reality Applications with Natural Lighting. Proceedings of the 4th workshop virtual and augmented reality of the GI-group (VR/AR) Weimar, 2007.

195. Supan, P. & Stuppacher, I. Interactive Image Based Lighting in Augmented Reality. 10th Central European Seminar on Computer Graphics (CESCG), 2006.

196. Pessoa, S., Moura, G., Lima, J., Teichrieb, V. & Kelner, J. Photorealistic rendering for Augmented Reality: A global illumination and BRDF solution. Virtual Reality Conference (VR), March 2010. IEEE.Pp. 3–10.

197. Okumura, B., Kanbara, M. & Yokoya, N. Augmented reality based on estimation of defocusing and motion blurring from captured images. Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR), Washington, DC, USA: IEEE Computer Society. Pp. 219–225.

198. Oyamada, Y. & Saito, H. Defocus Blur Correcting Projector-Camera System. Proceedings of the 10th International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS). Juan-les-Pins, France. Berlin, Heidelberg: Springer-Verlag, 2008. Pp. 453–464.

199. Kolb, C., Mitchell, D. & Hanrahan, P. A realistic camera model for computer graphics. Proceedings of the 22nd annual conference on Computer graphics and interactive techniques (SIGGRAPH). New York, NY, USA: ACM, 1995. Pp. 317–324.

200. Asada, N. & Baba, M. A Unified Camera Model of Zoom, Focus and Iris Parameters for Camera-Calibrated Computer Graphics. International Conference on Computer Graphics and Imaging. 2000. Pp. 101–106.

201. Fischer, J., Bartz, D. & Strasser, W. Enhanced visual realism by incorporating camera image effects. Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2006. Pp. 205–208.

202. Klein, G. & Murray, D. Compositing for small cameras. 7th IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR). Cambridge UK, 15–18 September 2008. Pp. 57–60.

203. Bertalmio, M., Sapiro, G., Caselles, V. & Ballester, C. Image inpainting. Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH). ACM Press/Addison-Wesley Publishing Co., 2000. Pp. 417–424.

204. Yamauchi, H., Haber, J. & Seidel, H. Image Restoration using Multiresolution Texture Synthesis and Image Inpainting. Computer Graphics International Conference 2003, Vol. 0, pp. 120–125.

205. Allene, C. & Paragios, N. Image Renaissance Using Discrete Optimization. International Conference on Pattern Recognition, 2006, Vol. 3, pp. 631–634.

206. Kokaram, A. Parametric texture synthesis for filling holes in pictures. Image Processing on International Conference (ICIP). Vol. 1. 2002. Pp. 325–328.

207. Ting, H., Chen, S., Liu, J. & Tang, X. Image inpainting by global structure and texture propagation. Proceedings of the 15th international conference on Multimedia. Augsburg, Germany, New York, NY, USA: ACM, 2007. Pp. 517–520.

208. Criminisi, A., Perez, P. & Toyama, K. Region filling and object removal by exemplar-based image inpainting. IEEE Transactions on Image Processing, 2004, Vol. 13, No. 9, pp. 1200–1212.

209. Bornemann, F. & März, T. Fast Image Inpainting Based on Coherence Transport. J.Math.Imaging Vis. 2007, July, Vol. 28, No. 3, pp. 259–278.

210. Wang, Z. Image Affine Inpainting. Proceedings of the 5th international conference on Image Analysis and Recognition (ICIAR). Póvoa de Varzim, Portugal. Berlin, Heidelberg: Springer-Verlag, 2008. Pp. 1061–1070.

211. Li, H., Wang, S., Zhang, W. & Wu, M. Image inpainting based on scene transform and color transfer. Pattern Recogn. Lett. May 2010, Vol. 31, No. 7, pp. 582–592.

212. Sen-Ching S. Cheung, Jian Zhao & Venkatesh, M.V. Efficient Object-Based Video Inpainting. Image Processing, IEEE International Conference on. 2006. Pp. 705–708.

213. Lee, I. Towards Wireless Augmented Reality A Review of its Enabling Technologies. Proceedings of the 2009 International Symposium on Ubiquitous Virtual Reality. Washington, DC, USA: IEEE Computer Society, 2009. Pp. 25–28.

214. Criminisi, A., Perez, P. & Toyama, K. Object removal by exemplar-based inpainting. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003, Vol. 2.

215. Enomoto, A. & Saito, H. Diminished Reality using Multiple Handheld Cameras. ACCV 2007 Workshop on Multi-dimensional and Multi-view Image Processing. 2007. Pp. 130–135.

216. Jarusirisawad, S., Hosokawa, T. & Saito, H. Diminished reality using plane-sweep algorithm with weakly-calibrated cameras. Progress in Informatics 2010, Vol. 7, pp. 11–20.

217. Zokai, S., Esteve, J., Genc, Y. & Navab, N. Multiview Paraperspective Projection Model for Diminished Reality. Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2003. Pp. 217–226.

218. Lepetit, V. & Berger, M. An Intuitive Tool for Outlining Objects in Video Sequences: Applications to Augmented and Diminished Reality. Interna-

tional Symposium of Mixed Reality. Proceedings of the International Symposium on Mixed Reality, Yokohama, Japan. 2001. Pp. 159–160.

219. Herling, J. & Broll, W. Advanced self-contained object removal for realizing real-time Diminished Reality in unconstrained environments. 9th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2010. Pp. 207–212.

220. Kass, M., Witkin, A. & Terzopoulos, D. Snakes: Active contour models. International Journal Of Computer Vision 1988, Vol. 1, No. 4, pp. 321–331.

221. Inami, M., Kawakami, N. & Tachi, S. Optical Camouflage Using Retro-Reflective Projection Technology. Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality. (ISMAR). Washington, DC, USA: IEEE Computer Society, 2003. Pp. 348–349.

222. Cosco, F.I., Garre, C., Bruno, F., Muzzupappa, M. & Otaduy, M.A. Augmented touch without visual obtrusion. Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2009. Washington, DC, USA: IEEE Computer Society, 2009. Pp. 99–102.

223. Chen, J., Granier, X., Lin, N. & Peng, Q. On-line visualization of underground structures using context features. Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology (VRST). Hong Kong, New York, NY, USA: ACM, 2010. Pp. 167–170.

224. Boun Vinh Lu, Kakuta, T., Kawakami, R., Oishi, T. & Ikeuchi, K. Foreground and shadow occlusion handling for outdoor augmented reality. 9th IEEE International Symposium on Mixed and Augmented Reality (ISMAR). Seoul, Korea, 13–16 October 2010. Pp. 109–118.

225. Avery, B., Piekarski, W. & Thomas, B.H. Visualizing Occluded Physical Objects in Unfamiliar Outdoor Augmented Reality Environments. Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2007. Pp. 1–2.

226. Schall, G., Mendez, E. & Schmalstieg, D. Virtual redlining for civil engineering in real environments. 7th IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR), 15–18 September 2008. Pp. 95–98.

227. Schall, G., Mendez, E., Kruijff, E., Veas, E., Junghanns, S., Reitinger, B. & Schmalstieg, D. Handheld Augmented Reality for underground infrastructure

visualization. Personal Ubiquitous Comput., May 2009, Vol. 13, No. 4, pp. 281–291.

228. Livingston, M.A., Swan II, J.E., Gabbard, J.L., Höllerer, T.H., Hix, D., Julier, S.J., Baillot, Y. & Brown, D. Resolving Multiple Occluded Layers in Augmented Reality. Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2003. Pp. 56–65.

229. Furmanski, C., Azuma, R. & Daily, M. Augmented-Reality Visualizations Guided by Cognition: Perceptual Heuristics for Combining Visible and Obscured Information. Proceedings of the 1st International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2002. Pp. 215–224.

230. Kalkofen, D., Mendez, E. & Schmalstieg, D. Interactive Focus and Context Visualization for Augmented Reality. Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2007. Pp. 1–10.

231. Bane, R. & Höllerer, T. Interactive Tools for Virtual X-Ray Vision in Mobile Augmented Reality. Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2004. Pp. 231–239.

232. Avery, B., Sandor, C. & Thomas, B.H. Improving Spatial Perception for Augmented Reality X-Ray Vision. 2009, IEEE Virtual Reality Conference (VR). Lafayette, Louisiana, USA, 14–18 March 2009. Pp. 79–82.

233. Dey, A., Cunningham, A. & Sandor, C. Evaluating depth perception of photorealistic mixed reality visualizations for occluded objects in outdoor environments. Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology (VRST). Hong Kong, New York, NY, USA: ACM, 2010. Pp. 211–218.

234. Sandor, C., Cunningham, A., Dey, A. & Mattila, V.-. An Augmented Reality X-Ray system based on visual saliency. 9th IEEE International Symposium on Mixed and Augmented Reality (ISMAR). Seoul, Korea, 13–16 October 2010. Pp. 27–36.

235. Sandor, C., Dey, A., Cunningham, A., Barbier, S., Eck, U., Urquhart, D., Marner, M.R., Jarvis, G. & Sang Rhee. Egocentric space-distorting visu-

alizations for rapid environment exploration in mobile mixed reality. Virtual Reality Conference (VR), 2010. IEEE. Pp. 47–50.

236. Viewlity – Augmented Reality Engine. http://curs.pub.ro/index.php/android-projects/4-viewlity-augmented-reality-engine [6 July 2011] .

237. Layar. http://www.layar.com [26 April 2012].

238. Breen, D.E., Whitaker, R.T., Rose, E. & Tuceryan, M. Interactive Occlusion and Automatic Object Placement for Augmented Reality. Computer Graphics Forum 1996, Vol. 15, No. 3, pp. 11–22.

239. Yoon, T. & Jung, K. Fast 3D Collision Detection Algorithm using 2D Intersection Area. Proceedings of the International Conference on Computer, Electrical, and Systems Science, and Engineering (CESSE), World Academy of Science, Engineering and Technology, No. 60, December, 2009. Pp. 719–722,.

240. Murase, K., Ogi, T., Saito, K., Koyama, T. & Biru, K.S. Correct Occlusion Effect in the Optical See-through Immersive Augmented Reality Display System. ICAT 2008. Pp. 12–19.

241. Chong, J.W.S., Ong, S.K., Nee, A.Y.C. & Youcef-Youmi, K. Robot programming using augmented reality: An interactive method for planning collision-free paths. Robotics and Computer-Integrated Manufacturing 2009, 6, Vol. 25, No. 3, pp. 689–701.

242. Lee, D., Lee, S.G., Kim, W.M. & Lee, Y.J. Sphere-to-sphere collision estimation of virtual objects to arbitrarily-shaped real objects for augmented reality. Electronics Letters 2010, Vol. 46, No. 13, pp. 915–916.

243. Fransens, R., Strecha, C. & Van Gool, L. A Mean Field EM-algorithm for Coherent Occlusion Handling in MAP-Estimation Prob. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006. Vol. 1. 2006. P. 300.

244. Ballagas, R., Borchers, J., Rohs, M. & Sheridan, J.G. The smart phone: a ubiquitous input device. Pervasive Computing, IEEE 2006, Vol. 5, No. 1, pp. 70–77.

245. Nokia Point & Find, 2011. http://pointandfind.nokia.com [26 April 2012].

246. GeoVector World Surfer, 2011. http://www.geovector.com/applications/world-surfer [18 June 2011] .

247. Google Goggles, 2011. http://www.google.com/mobile/goggles [26 April 2012].

248. Hakkarainen, M. & Woodward, C. SymBall: camera driven table tennis for mobile phones. Proceedings of ACM SIGCHI International Conference on Advances in computer entertainment technology (ACE). Valencia, Spain. New York, NY, USA: ACM, 2005. Pp. 391–392.

249. VTT Phonemouse. http://www.vtt.fi/multimedia/index_pm.html [26 April 2012].

250. Petersen, N. & Stricker, D. Continuous natural user interface: Reducing the gap between real and digital world. Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2009. Pp. 23–26.

251. Potamianos, A. & Perakakis, M. IDesign Principles for Multimodal Spoken Dialogue Systems. In: Anonymous Multimodal Processing and Interaction. Vol. 33. Springer US, 2008. Pp. 1–18. (Multimedia Systems and Applications.)

252. Bolt, R.A. “Put-that-there”: Voice and gesture at the graphics interface. Proceedings of the 7th annual conference on Computer graphics and interactive techniques (SIGGRAPH). Seattle, Washington, United States, New York, NY, USA: ACM, 1980. Pp. 262–270.

253. Newcombe, R., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A., Kohli, P., Shotton, J., Hodges, S. & Fitzgibbon, A. KinectFusion: Real-Time Dense Surface Mapping and Tracking. Proceedings IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR), 2011. Basel, Switzerland, 26–29 October 2011. Pp. 127–136.

254. Specht, M., Ternier, S. & Greller, W. Dimensions of Mobile Augmented Reality for Learning: A First Inventory. Journal of the Research Center for Educational Technology 2011, Vol. 7, No. 1, pp. 117–127.

255. Singh, G., Swan II, J.E., Jones, J.A. & Ellis, S.R. Depth judgment measures and occluding surfaces in near-field augmented reality. Proceedings of the 7th Symposium on Applied Perception in Graphics and Visualization (APGV). Los Angeles, California, New York, NY, USA: ACM, 2010. Pp. 149–156.

256. Oda, O. & Feiner, S. Interference avoidance in multi-user hand-held augmented reality. Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2009. Pp. 13–22.

257. HITLabNZ BuildAR. http://www.buildar.co.nz [26 April 2012].

258. Inglobe Technologies ARSights. http://www.arsights.com [26 April 2012].

259. Bohlen, J.M. & Beal, G.M. The Diffusion Process. 1957, May. Special Report No. 18 (Agriculture Extension Service, Iowa State College), pp. 56–77.

260. Rogers, E. Diffusion of Innovation. 5th Edition. New York, USA: The Free Press, 2003.

261. Compeau, D.R. & Higgins, C.A. Application of Social Cognitive Theory to Training for Computer Skills. Information Systems Research 1995, Vol. 6, No. 2, pp. 118–143.

262. Moore, G.A. Crossing the Chasm. Revised ed. Harper Paperbacks, 2002.

263. Davis, F.D. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly 1989, Vol. 13, No. 3, pp. 319–340.

264. Venkatesh, V. & Davis, F.D. A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies. Management Science 2000, Vol. 46, No. 2, pp. 186–204.

265. VTT Augmented Reality team. www.vtt.fi/multimedia [26 April 2012].

266. Lin, S.V., Seibel, E.J. & III, T.A.F. Testing Visual Search Performance Using Retinal Light Scanning as a Future Wearable Low Vision Aid. Int.J.Hum.Comput.Interaction 2003, Vol. 15, No. 2, pp. 245–263.

267. Dünser, A., Grasset, R., Seichter, H. & Billinghurst, M. Applying HCI Principles in AR Systems Design. 2nd International Workshop on Mixed Reality User Interfaces (MRUI): Specification, Authoring, Adaptation. 11 March 2007. Pp. 37–42.

268. Yelp Official Blog “You're Gonna Want to ‘Check-Out’ Yelp for iPhone v.4”, 2010. http://officialblog.yelp.com/2010/01/youre-gonna-want-to-checkout-yelp-for-iphone-v4.html [16 May 2012].

269. Maida, J., Bowen, C. & Pace, J. Improving Robotic Operator Performance Using Augmented Reality. Vol. Human Factors And Ergonomics Society, 51st Annual Meeting. 2007. Pp. 1635–1639.

270. Stone, R., Bisantz, A., Llinas, J. & Paquet, V. Improving Tele-robotic Landmine Detection through Augmented Reality Devices. Vol. 52. 2008. Pp. 206–210.

271. Rogers, R. Augmented reality with HTML5 Linux Journal 2011, Vol. March 2011, No. 203.

272. Barnum, P., Sheikh, Y., Datta, A. & Kanade, T. Dynamic seethroughs: Synthesizing hidden views of moving objects. Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR). Washington, DC, USA: IEEE Computer Society, 2009. Pp. 111–114.

273. Engadget Engadget web magazine. http://www.engadget.com [2 November 2011]

274. Davies, H. Reality through the invisible interface. IEEE International Symposium On Mixed and Augmented Reality – Arts, Media, and Humanities (ISMAR-AMH), 2010. Pp. 63–64.

275. Parviz, B.A. Augmented Reality in a Contact Lens. IEEE 2009. http://spectrum.ieee.org/biomedical/bionics/augmented-reality-in-a-contact-lens/0# [26 April 2012]

276. Nokia GEM phone concept. 2011.http://research.nokia.com/news/12133 [26 April 2012].

277. Sutherland, I.E. The Ultimate Display. Proceedings of the IFIP Congress. New York, Vol. 2. 1965. Pp. 506–508.

278. Krunic, V. & Han, R. Towards Cyber-Physical Holodeck Systems Via Physically Rendered Environments (PRE's). 28th International Conference on Distributed Computing Systems Workshops (ICDCS), 2008. Pp. 507–512.

279. Graydon, O. Holography: Polymer yields 3D video. Nature Photonics 2010, Vol. 4, No. 12, p. 811. doi: 10.1038/nphoton.2010.282.

280. Thomas, J., Christenson, C.W., Blanche, P., Yamamoto, M., Norwood, R.A. & Peyghambarian, N. Photoconducting Polymers for Photorefractive 3D Display Applications. Chemistry of Materials, 8 February 2011, Vol. 23, No. 3, pp. 416–429.

281. Stolte, D. UA News – Moving Holograms: From Science Fiction to Reality, 2010. http://www.uanews.org/node/35220 [30 June 2011].

282. Linder, N. & Maes, P. LuminAR: portable robotic augmented reality interface design and prototype. Adjunct proceedings of the 23nd annual ACM symposium on User interface software and technology (UIST). New York, New York, USA, New York, NY, USA: ACM, 2010. Pp. 395–396.

283. Hartl, A., Gruber, L., Arth, C., Hauswiesner, S. & Schmalstieg, D. Rapid reconstruction of small objects on mobile phones. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2011. Pp. 20–27.

284. Pan, Q., Arth, C., Rosten, E. & Reitmayr, G. Towards Rapid 3d Reconstruction on Mobile Phones from Wide-Field-of-View Images. Augmented Reality Supermodels Workshop (held in conjunction with ISMAR), 2010. Pp. 1–8.

285. Korkalo, O. & Honkamaa, P. Construction and evaluation of multi-touch screens using multiple cameras located on the side of the display. ACM International Conference on Interactive Tabletops and Surfaces (ITS). Saarbrücken, Germany, New York, NY, USA: ACM, 2010. Pp. 83–90.

286. Mistry, P. & Maes, P. SixthSense: a wearable gestural interface. ACM SIGGRAPH ASIA, 2009 Sketches. Yokohama, Japan, New York, NY, USA: ACM, 2009.

287. Mistry, P. & Maes, P. SixthSense: a wearable gestural interface. ACM SIGGRAPH ASIA 2009, Art Gallery & Emerging Technologies: Adaptation. Yokohama, Japan, New York, NY, USA: ACM, 2009. P. 85.

288. Wilson, A.D. & Benko, H. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. Proceedings of the 23nd annual ACM symposium on User interface software and technology (UIST), 2010. New York, NY, USA: ACM, 2010. Pp. 273–282.

289. Virolainen, A., Puikkonen, A., Kärkkäinen, T. & Häkkilä, J. Cool interaction with calm technologies: experimenting with ice as a multitouch surface. ACM International Conference on Interactive Tabletops and Surfaces (ITS). Saarbrücken, Germany, New York, NY, USA: ACM, 2010. Pp. 15–18.

290. Kooaba. http://www.kooaba.com [2011].

291. Snaptell. http://www.snaptell.com [2011].

292. Tsai, S.S., Chen, D., Chandrasekhar, V., Takacs, G., Cheung, N., Vedantham, R., Grzeszczuk, R. & Girod, B. Mobile product recognition.

Proceedings of the international conference on Multimedia. Firenze, Italy, New York, NY, USA: ACM, 2010. Pp. 1587–1590.

293. Nikolopoulos, S., Nikolov, S.G. & Kompatsiaris, I. Study on Mobile Image Search. NEM Summit. Torino, Italy, 29 September 2011.

294. CrowdOptic. The Connected Fan. http://www.crowdoptic.com [26 April 2012].

295. Petkova, V.I. & Ehrsson, H.H. If I Were You: Perceptual Illusion of Body Swapping. PLoS ONE, 3 December 2008, Vol. 3, No. 12, p. e3832.

296. van, d.H., Guterstam, A. & Ehrsson, H.H. Being Barbie: The Size of One's Own Body Determines the Perceived Size of the World. Public Library of Science, 2011, p. e20195.

297. Micera, S., Navarro, X., Carpaneto, J., Citi, L., Tonet, O., Rossini, P.M., Carrozza, M.C., Hoffmann, K.P., Vivo, M., Yoshida, K. & Dario, P. On the Use of Longitudinal Intrafascicular Peripheral Interfaces for the Control of Cybernetic Hand Prostheses in Amputees. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2008, Vol. 16, No. 5, pp. 453–472.

298. Widge, A.S., Moritz, C.T. & Matsuoka, Y. Direct Neural Control of Anatomically Correct Robotic Hands. In: Anonymous Brain-Computer Interfaces. Vol. 0. London, UK: Springer, 2010. Pp. 105–119. (Human-Computer Interaction Series.)

299. Velliste, M., Perel, S., Spalding, M.C., Whitford, A.S. & Schwartz, A.B. Cortical control of a prosthetic arm for self-feeding. Nature, 19 June 2008, Vol. 453, No. 7198, pp. 1098–1101.

300. Honda ASIMO – The World's Most Advanced Humanoid Robot. http://asimo.honda.com; http://world.honda.com/ASIMO [26 April 2012].

301. Teh, J.K.S., Cheok, A., Choi, Y., Fernando, C., Peiris, R. & Newton Fernando, O. Huggy pajama: a parent and child hugging communication system 2009, Vol. 1, No. Interaction Design and Children, 2009, pp. 290–291.

302. Cheok, A.D., Godagey, I.S., Samaniz, H.A., Narangodax, M.R. & Newton Fernado, O.N. Intimate Physical Interaction with Robots , Human – Robot Kiss. 2nd International conference on Human-robot personal relationship (HRPR), 2009.

303. Lovotics KISSENGER: Kiss Messenger Robot. http://kissenger.lovotics.com [2 December 2011].

304. Ranasinghe, N., Cheok, A.D., Nii, H., Fernando, O.N.N. & Ponnampalam, G. Digital taste interface. Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology (UIST). Santa Barbara, California, USA, New York, NY, USA: ACM, 2011. Adjunct. Pp. 11–12.

305. Aghaebrahimi Samani, H., Cheok, A.D., Tharakan, M.J., Koh, J. & Fernando, N. A Design Process for Lovotics. In: Human-Robot Personal Relationships. Vol. 59. Berlin, Heidelberg, Germany: Springer, 2011. Pp. 118–125. (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering.)

306. Nishimoto, S., Vu, A., Naselaris, T., Benjamini, Y., Yu, B. & Gallant, J. Reconstructing Visual Experiences from Brain Activity Evoked by Natural Movies. Current Biology, 11 October 2011, Vol. 21, No. 19, pp. 1641–1646.

307. Faugeras, O. Three-dimensional computer vision: a geometric viewpoint. Cambridge, MA, USA: MIT Press, 1993.

308. Weng, J., Cohen, P. & Herniou, M. Camera Calibration with Distortion Models and Accuracy Evaluation. IEEE Trans.Pattern Anal.Mach.Intell. 1992, Vol. 14, No. 10, pp. 965–980.

309. Mikhail, E.M., Bethel, James, S, & McGlone, C., J. Introduction to modern photogrammetry. New York: Wiley, 2001.

310. Willson, R.G. & Sharer, S.A. What is the Center of the Image? Journal of the Optical Society of America A. 1993. Pp. 670–671.

311. Pers, J. & Kovacic, S. Nonparametric, Model-Based Radial Lens Distortion Correction Using Tilted Camera Assumption. Proceedings of the Computer Vision Winter Workshop 2002. Pp. 286–295.

312. Fitzgibbon, A.W. Simultaneous linear estimation of multiple view geometry and lens distortion. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) 2001. Vol. 1. Pp. 125–132.

313. Optical Metrology Centre (OMC). OMC Technical Brief – Camera Calibration. 2001.

314. Wang, J., Shi, F., Zhang, J. & Liu, Y. A new calibration model of camera lens distortion. Pattern Recogn. 2008, Vol. 41, No. 2, pp. 607–615.

315. Nocedal, J. & Wright, S.J. Numerical optimization. Springer, 2006. 316. Sanz-Ablanedo, E., Rodríguez-Pérez, J.R., Arias-Sánchez, P. & Armesto, J. Metric Potential of a 3D Measurement System Based on Digital Compact Cameras. Sensors 2009, Vol. 9, No. 6, pp. 4178–4194.

317. Rieke-Zapp, D.H. & Peipe, J. Performance evaluation of a 33 megapixel alpa 12 medium format camera for digital close range photogrammetry. IEVM, 2006.

318. Brown, D. A strategy for multi-camera on-the-job self-calibration. Festschrift Friedrich Ackermann zum 60. Geburtstag. Vol. Schriftenreihe 14. Stuttgart: Institut für Photogrammetrie der Universität, 1989. Pp. 9–21.

319. Willson, R.G. & Shafer, S.A. A Perspective Projection Camera Model for Zoom Lenses. Proceedings of the 2nd Conference on Optical 3-D Measurement Techniques. 1993.

320. Zhang, Z. A flexible new technique for camera calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence, November 2000, Vol. 22, No. 11, pp. 1330–1334.

321. Abdel-Aziz, Y.I. & Karara, H.M. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. Proceedings of the Symposium on Close-Range Photogrammetry. Falls Church, American Society of Photogrammetry, 1971. Pp. 1–18.

322. Heikkila, J. & Silven, O. A Four-step Camera Calibration Procedure with Implicit Image Correction. Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR). Washington, DC, USA: IEEE Computer Society, 1997. Pp. 1106–1112.

323. Faugeras, O. & Toscani, G. Camera Calibration for 3D Computer Vision. In: Anonymous 1987. Pp. 240–247.

324. Gill, P.R., Murray, W. & and Wright, M.H. The Levenberg-Marquardt Method. In: Anonymous. Practical Optimization. London: Academic Press, 1981. Pp. 136–137.

325. Levenberg, K. A method for the solution of certain problems in least squares. Quart.Applied Math. 1944, Vol. 2, pp. 164–168.

326. Marquardt, D.W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. SIAM Journal on Applied Mathematics 1963, Vol. 11, No. 2, pp. 431–441.

327. Bates, D.M. & Watts, D.G. Nonlinear regression analysis and its applications. New York: Wiley, 1988.

328. Hartley, R.I. In defence of the 8-point algorithm. Proceedings of the 5th International Conference on Computer Vision (ICCV). Washington, DC, USA: IEEE Computer Society, 1995. P. 1064.

329. Griewank, A. & Walther, A. Evaluating Derivatives, 2nd Edition. SIAM, 2008.

330. Golub, G.H. & Loan, C.F.V. Matrix computations. 3rd Edition. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

331. Information technology – Automatic identification and data capture techniques – Data Matrix bar code symbology specification. ISO/IEC 16022:2000. International Organization for Standardization. 2000.

332. Information technology – Automatic identification and data capture techniques – Data Matrix bar code symbology specification. ISO/IEC 24720:2006. International Organization for Standardization. 2006.

333. Quick Responce Code aka QR Code. JIS X 0510. Japanese Industrial Standards. Denso Wave. 1999.

334. Information technology – Automatic identification and data capture techniques – QR Code 2005 bar code symbology specification. ISO/IEC 18004. International Organization for Standardization. 2006.

335. Information technology – Automatic identification and data capture techniques – PDF417 bar code symbology specification. ISO/IEC 15438:2006. International Organization for Standardization. 2006.

Appendix A: Projective geometry

Projective geometry is a branch of mathematics that deals with the relationships between geometric objects and their projections. While *Euclidean geometry* describes the world as it is, projective geometry describes the world *as it appears* or *as it is seen*. Euclidean geometry deals with properties of objects that are invariant under rigid motion, e.g. lengths, angles and parallelisms, whereas projective geometry deals with the properties that are invariant under perspective projections, e.g. incidences and cross-ratios.

If we were to close one eye and draw the world on a glass plate as we see it, we would notice that the distances between the points on a drawing differ from the true distances in the world, and the angles between the lines differ from the angles in the real world. Euclidean geometry is not able describe this, whereas projective geometry is.

In this appendix, we focus on the aspects of projective geometry that are essential in many application areas of computer vision, including augmented reality (AR). Projective geometry is discussed widely in computer vision; good reference books for further reading include [72, 307] and [307].

Homogeneous coordinates

With *Cartesian coordinates*, we can conveniently present transformations in Euclidean space with matrix operations. For example, we can calculate a rotation of a rigid object by multiplying the Cartesian coordinates with an adequate matrix. Cartesian coordinates, however, are unable to perform transformations of projective space with matrix operations. For instance, it is impossible to present a translation in matrix form using Cartesian coordinates.

Homogeneous coordinates make matrix calculations possible in projective space just as Cartesian coordinates do in Euclidean space. Homogeneous coordinates simplify calculations, as all the necessary transformations (and series of transformations) and their actions on points are presentable using matrices and matrix multiplications. This enables the computational efficiency required for real-time computer graphics systems such as OpenGL and DirectX.

Homogeneous coordinate vectors of a point $\mathbf{x} = \left(x_1, x_2, x_3, ..., x_n\right)$ in Cartesian coordinates are all vectors $\mathbf{X} = (x_1', x_2', ..., x_n', w)$ such that

$$\mathbf{x} = \left({}^{x_1'}\!/\!{}_{w}, {}^{x_2'}\!/\!{}_{w}, \ldots, {}^{x_n'}\!/\!{}_{w} \right),$$

and vice versa, each point in projective space $\mathbf{X} \neq \mathbf{0}, \mathbf{X} = (x_1, x_2, ..., x_n, w)$ has a corresponding projection point in Euclidean space

$$\mathbf{x} = \left({}^{x_1}\!/_{w}, {}^{x_2}\!/_{w}, {}^{x_3}\!/_{w}, ..., {}^{x_n}\!/_{w} \right).$$

In projective space, scaling is unimportant and therefore in homogeneous coordinates

$$a\mathbf{X} \equiv \mathbf{X}, \forall a \neq 0.$$

Thus, a point in homogeneous coordinates is always equivalent to a representation with the last coordinate equal to one.

$$\begin{aligned}\mathbf{X} = (x_1', x_2', ..., x_n', x_{n+1}') &\equiv \frac{1}{x_{n+1}}\mathbf{X} = ({}^{x_1'}\!/_{x_{n+1}'}, {}^{x_2'}\!/_{x_{n+1}'}, ..., {}^{x_n'}\!/_{x_{n+1}'}, {}^{x_{n+1}'}\!/_{x_{n+1}'})\\ &= (x_1, x_2, ..., x_n, 1).\end{aligned}$$

Furthermore, the point $\mathbf{X} = \left(X_1, X_2, ..., X_n, 0\right)$ corresponds to a *point at infinity* in the direction of the line passing through $\mathbf{0} \in \mathbb{R}^n$ and $\left(x_1, x_2, x_3, ..., x_n\right)$. The zero vector $\mathbf{0} = (0, 0, ...0) \in \mathbb{R}^{n+1}$ is undefined in projective space.

For consistency, the representations with the last coordinate equal to one are often used for homogeneous coordinates. The division operation to get the last coordinate equal to one is called *homogeneous divide.* It maps the vector to the real plane.

For simplicity, the projection to a lower dimension (e.g. perspective projection) is often first calculated in the original dimension. The dimensions are then reduced one by one by performing successive homogeneous divisions. For example, mapping the perspective projection into the image space is the projection $\mathbb{R}^4 \to \mathbb{R}^2, \mathbf{X} = (X, Y, Z, 1) \to \mathbf{x} = (x, y).$

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \to \begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} \equiv \begin{pmatrix} {}^{x'}\!/_{w'} \\ {}^{y'}\!/_{w'} \\ {}^{z'}\!/_{w'} \\ {}^{w'}\!/_{w'} \end{pmatrix} \text{ and further } \begin{pmatrix} {}^{x'}\!/_{w'} \\ {}^{y'}\!/_{w'} \\ {}^{z'}\!/_{w'} \end{pmatrix} \equiv \begin{pmatrix} {}^{x'}\!/_{z'} \\ {}^{y'}\!/_{z'} \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \text{which equals to} \begin{pmatrix} x \\ y \end{pmatrix}.$$

Geometric transformations

In AR as in 3D computer vision in general, perspective projections play an essential role. To be able to analyse a camera image, we need to understand how world coordinates are transformed into image coordinates, and this can be explained using projective geometry.

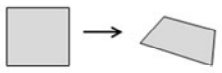
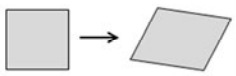
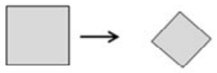
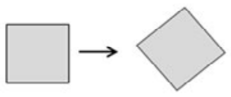
Geometric transformations form a hierarchy of subsets

Projective $\supset$ Affine $\supset$ Similarity $\supset$ Linear (Euclidean geometry),

where the transformation groups become smaller and less general and the corresponding spatial structures become more rigid and have more invariants as we go down the hierarchy. Homogeneous coordinates provide a framework for geometric operations in projective space. Euclidean geometry is a special case of projective geometry with more restrictions. Thus, it is possible to use homogeneous presentation in Euclidean geometry as well, if the operations are restricted to Euclidean ones. Accordingly, homogeneous presentation can also be used in affine and similarity transformations. Thus, all geometric transformations and their combinations can be presented with matrix multiplications using homogeneous coordinates.

In a projective transform, only collinearity, cross-ratios and incidences remain invariant. Affine transformations also preserve parallelism and the ratios of areas. Similarity transforms preserve angles and length ratios. Euclidean transformations preserve angles, lengths and areas (see Table 2).

Table 2. Properties of different transformation spaces.

Transform	DOF	Invariants	
Perspective (Projective)	8DOF	Collinearity, Cross-ratios, Incidences	
Affine	6DOF	Parallelism, Ratios of areas	
Similarity	4DOF	Angles, Length ratios	
Linear (Euclidean)	3DOF	Angles, Lengths, Areas	

We start the discussion with linear transformations and propagate to affine and projective transformations. In the following, we also define specific transformation types, such as rotations, translations and scaling, which are commonly needed in augmented reality.

Linear transformation

A mapping $L : R^n \rightarrow R^n$ is called *linear transformation* if

$$\mathrm{L}(u + v) = \mathrm{L}(u) + \mathrm{L}(v) \text{ and } \mathrm{L}(cu) = c\,\mathrm{L}(u).$$

Linear transformation $\mathbf{x}' = L(\mathbf{x})$ can be presented in matrix form

$$\mathbf{x}' = \mathbf{L}\mathbf{x},$$

where $\mathbf{L}$ is an $n \times n$ matrix.

Matrix operations are easily expanded to homogeneous space. Any linear transformation presented by matrix $\mathbf{A}$ can be expanded to homogeneous form

$$\begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix}, \text{ where } \mathbf{A} \in \mathbb{R}^{N\times N} \text{ and } \mathbf{0} \text{ is zero vector, } \mathbf{0} \in \mathbb{R}^{N}.$$

For example, rotation, scaling and shear are linear transformations.

Rotation with an angle α around the x-axis is presented by a rotation matrix

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$\mathbf{x}' = \mathbf{R}_x\mathbf{x}.$$

Similarly, rotations with angle ß around the y-axis and with angle γ around the z-axes are jkj

$$\mathbf{R}_y = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \text{ and } \mathbf{R}_z = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Rotation around an arbitrary axis can be split into rotations around coordinate axes, and rotations around coordinate axes can be concatenated to present an arbitrary rotation. The rotation matrix (and the angles) depend on the rotation order and, for example, $\mathbf{R} = \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x$ is

$$\mathbf{R} = \begin{bmatrix} \cos\beta\cos\gamma & \cos\gamma\sin\alpha\sin\beta - \cos\gamma\sin\alpha & \cos\gamma\sin\beta\cos\alpha + \sin\alpha\sin\gamma \\ \cos\beta\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\gamma\sin\beta\cos\alpha - \cos\gamma\sin\alpha \\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\beta \end{bmatrix}$$

when written out. The written out forms are seldom used; it is more convenient to calculate separate rotations and multiply them.

Sometimes people use the terms *yaw*, *pitch* and *roll* for rotations. The yaw is a right-handed rotation about the *z*-axis. The pitch is a right-handed rotation about the new (once rotated) *y*-axis. The roll is a right-handed rotation about the new (twice rotated) *x*-axis. Let φ, θ and ψ be the yaw, pitch and roll angles respectively, then the total rotation is

$$\mathbf{R} = \mathbf{R}_x(\varphi)\mathbf{R}_y(-\theta)\mathbf{R}_z(-\psi).$$

Occasionally, we may find it more convenient to represent the rotation by giving an arbitrary rotation axis and rotation angle around it, instead of dividing it into rotations around the coordinate axes. The rotation around an axis represented by a unit vector $\mathbf{u} = (u_1, u_2, u_3)$ by an angle ω is

$$\mathbf{R} = \begin{bmatrix} u_1^2 + (1-u_1^2)\cos\omega & u_1u_2(1-\cos\omega) - u_3\sin\omega & u_1u_3(1-\cos\omega) + u_2\sin\omega \\ u_1u_2(1-\cos\omega) + u_3\sin\omega & u_2^2 + (1-u_2^2)\cos\omega & u_2u_3(1-\cos\omega) - u_1\sin\omega \\ u_1u_3(1-\cos\omega) + u_2\sin\omega & u_2u_3(1-\cos\omega) + u_1\sin\omega & u_3^2 + (1-u_3^2)\cos\omega \end{bmatrix}.$$

The rotation here is clockwise about the axis defined by $\mathbf{u}$ (right-hand rule). This can be presented with the Rodrigues formula in the form

$$\mathbf{R} = \mathbf{I} + [\mathbf{u}]_\times \sin\omega + (1-\cos\omega)[\mathbf{u}]_\times^2,$$

where $[\mathbf{u}]_\times$ is a so-called cross-product matrix, i.e. $[\mathbf{u}]_\times \mathbf{v} = \mathbf{u} \times \mathbf{v}$.

The rotation matrix is an orthogonal (even orthonormal) matrix and thus it preserves angles and lengths. Furthermore, by the definition of orthogonality, the inverse of a rotation is its transpose

$$\mathbf{R}^{-1} = \mathbf{R}^T.$$

We can present *scaling* as a matrix operation using homogeneous coordinates

$$\mathbf{x}' = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix},$$

where s_x, s_y and s_z are scale factors in the directions of the coordinate axis.

Affine transformation

Translation by a vector $\mathbf{t} = (t_1, t_2, t_3)$, $\mathbf{x}' = \mathbf{x} + \mathbf{t} = (x_1 + t_1, x_2 + t_2, x_3 + t_3)$ can be presented in matrix form only using homogeneous coordinates. Translation $\mathbf{x}' = \mathbf{x} + \mathbf{t}$ is equivalent to multiplication by homogeneous transformation matrix $\mathbf{T}$,

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The inverse operation to translation is translation by the opposite vector $-\mathbf{t}$, and the inverse of the translation matrix is

$$\mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & -t_1 \\ 0 & 1 & 0 & -t_2 \\ 0 & 0 & 1 & -t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Affine transformation A combines linear mapping and translation. *A* is an affine transformation if

$$\mathbf{x}' = A(\mathbf{x}) = L(\mathbf{x} + \mathbf{t}),$$

where L is a linear mapping and $\mathbf{t}$ is a translation vector. In the matrix representation we have

$$\mathbf{A} = \mathbf{L}\,\mathbf{T},$$

where $\mathbf{A}$ is now an affine transformation matrix, $\mathbf{L}$ is a linear transformation matrix and $\mathbf{T}$ is the corresponding translation matrix.

Affine transformations can be concatenated but not commutated:

$$A1(A2(x)) = (A1(A2))(x)$$
$$A1(A2(x)) \neq A2\,(A1(x)) \text{ in general.}$$

The non-commutative law means that the order of transformations is significant, and in AR application we need to do them in the correct order. The concatenation property allows us to multiply series of transformation matrices into one matrix.

Affine transformations preserve lines and parallelism (i.e. parallel lines and planes remain parallel). They also preserve *ratios* of length, area and volume and the degree of a polynomial. Intersecting lines and planes are also transformed into intersecting lines and planes, but they do not preserve angles and shapes.

Let $\mathbf{L}$ be a 3×3 linear mapping and $\mathbf{t}$ a 3×1 translation vector, then

$$\mathbf{A} = \begin{bmatrix} & \mathbf{L}_{3x3} & & \mathbf{t}_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} l_{11} & l_{12} & l_{13} & t_1 \\ l_{21} & l_{22} & l_{23} & t_2 \\ l_{31} & l_{32} & l_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

is an affine transformation.

Perspective projection

In augmented reality, we are especially interested in perspective projections that describe how the camera projects the world into images.

Perspective projection describes how the 3-dimensional world is mapped to a 2-dimensional image in a pinhole camera. In projective geometry, it is presented using a projective mapping $P: R^3 \to R^2$. With homogeneous coordinates, a projective mapping is presented with a 4 x 4 transformation matrix, which first maps points to 4-dimensional homogeneous space. The result is then converted to image coordinates with two successive homogeneous divisions, as explained earlier (p. A2).

In perspective projection, each point projects to a point where the line connecting the point and the centre of projection intersects the image plane.

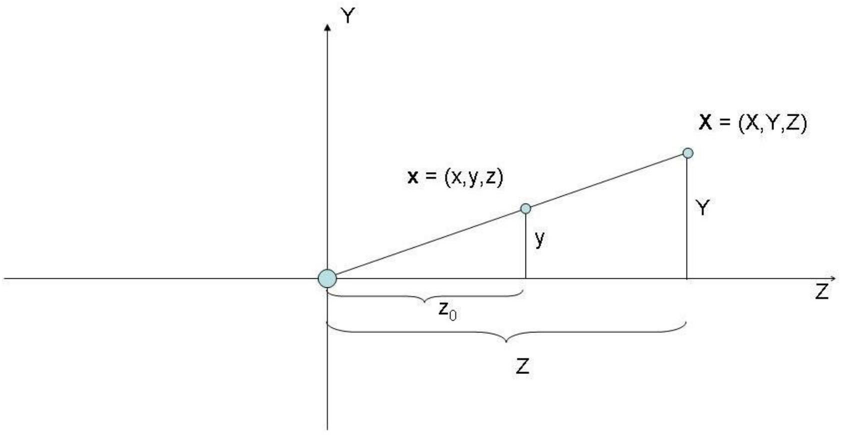


Figure 104. Principle of perspective projection

Consider the case in which a point $X = (X, Y, Z)$ is projected to the image plane $Z = z_0$ and the centre of projection is at the origin (Figure 104). The projected ray intersects the image plane at the point $\left(X\, {}^{z_0}\!/\!_Z, Y\, {}^{z_0}\!/\!_Z\right)$. We can write this in matrix form. A perspective projection $\mathbf{P}$ to the image plane $Z = z_0$, i.e. $\mathbf{x} = \mathbf{PX}$, is

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/z_0 & 0 \end{bmatrix}.$$

As in homogeneous coordinates $\lambda\mathbf{x} \equiv \mathbf{x} \Leftrightarrow (\lambda\mathbf{P})\mathbf{x} = \mathbf{P}(\lambda\mathbf{x}) \equiv \mathbf{Px}$, we may scale this with z_0 and represent the projective matrix in the form

$$\mathbf{P} = \begin{bmatrix} z_0 & 0 & 0 & 0 \\ 0 & z_0 & 0 & 0 \\ 0 & 0 & z_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The image plane of a camera is at the distance of the focal length (*f*) from the optical centre, i.e. $z_0 = f$.

A perspective projection preserves straight lines as straight lines. It does not preserve parallelism, only lines parallel to the image plane (perpendicular to the viewing direction) stay parallel in perspective projection; other parallel lines converge to the vanishing point. The effect of perspective projection is that objects become smaller the further away they are from the centre of projection. Therefore, perspective projection does not preserve ratios of lengths or areas.

The simplest perspective projection uses the origin as the centre of projection and *z* = 1 as the image plane. We can write this using homogeneous coordinates in matrix form as

$$\mathbf{x} = \mathbf{PX}$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Orthographic projection $\mathbf{O}$,

$$\mathbf{O} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

is the limit of perspective transformation as $f \to \infty$. It is a linear transformation.

Weak perspective projection

$$\mathbf{W} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \hat{Z} \end{pmatrix},$$

is the scaled orthographic projection (i.e., linear transformation). It can be used to approximate perspective projection if the object lies close to the optical axis, and object's dimensions are small compared to its average distance $\bar{Z}$ from the camera.

2D homography

A plane-to-plane projective mapping is a special case of projective geometry. It occurs, for example, when we take an image of a marker, wall, floor, book or other planar object. We consider this special case next.

A plane-to-plane 2D *projective transformation* is an invertible mapping $f(\mathbf{x}):\mathbf{P}^2 -> \mathbf{P}^2$, such that three points $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$ lie on the same line if and only if also points $f(\mathbf{x}_1)$, $f(\mathbf{x}_2)$ and $f(\mathbf{x}_3)$ lie on the same line. Here $\mathbf{P}^2$ denotes a homogeneous 3-vector. Other synonyms for projective transformation are *collineation* and *homography*. In computer vision and augmented reality research, *2D homography* and *planar homography* are the most commonly used terms for a plane-to-plane projective mapping.

2D projective transformation $f(\mathbf{x}):\mathbf{P}^2 -> \mathbf{P}^2$ can be presented in matrix form

$$\mathbf{y} = f(\mathbf{x}) = \mathbf{Hx} \Leftrightarrow$$
$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix},$$

where $\mathbf{H}$ is a non-singular matrix. The matrix H is called a *homogeneous matrix*. As we operate with homogeneous coordinates, only the ratio between matrix elements is significant and the homography is defined only up to scale.

Furthermore, the homogeneous matrix has eight independent ratios, among the nine elements. A projective transformation therefore has eight degrees of freedom. In consequence, we need at least eight independent parameters to define a homography. It has been proved that four corresponding points on both planes, with no more than any two points collinear, are enough to define a plane-to-plane homography [72]. Thus, as we know the physical dimension of a marker, we are able to solve the homography between the marker plane (world coordinates) and the ideal image coordinates, when we detect four corners of a marker from an image.

Images taken with a camera rotating around its optical centre are related to 2D homography as well as planar objects and their image. In addition, the relation between an object infinitely far away and its image can be approximated with 2D homography. Here, infinitely far away means that the distance (*d*) of the object is infinite compared with the camera's focal length (*f*) that is $d/f \approx \infty$. Besides marker-based AR, planar homographies are used for, e.g., image stitching and panoramic images, and camera calibration.

Appendix B: Camera model

In order to model how world points are mapped to the image plane in a camera, we need to understand camera geometry. In this appendix, we discuss camera models and the image formation process. Camera models and image formation have been studied in photogrammetry and computer vision, and many good presentations on the subject are available, for example [72] and [74], to mention a few.

Pinhole camera model

Camera obscura is a simple optical device (normally a box) in which light travelling through a small hole or lens forms an image. It is often called a *pinhole camera* because of the small hole through which the image forms (see Figure 105).

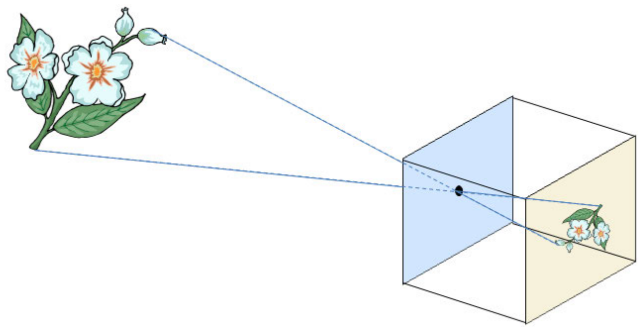


Figure 105. Image formation in a pinhole camera.

In an ideal case, the hole of the camera is infinitesimal. The *pinhole camera model* describes the perspective optics of such an ideal camera. In actual cameras, the lens has some dimension, its physical geometry is imperfect and the image forms on the image sensor, which has some physical characteristics. We need to take these imperfections into account to achieve an accurate camera model. We will discuss these deviations from the pinhole camera model later in Section 0.

In the pinhole camera model, the centre of projection, called the *optical centre,* is same as the *camera centre*. The *optical axis* is the ray going through the camera centre without refracting. The optical axis is also called the *principal axis*. The focal length of the lens is f. The *image plane* is perpendicular to the optical axis and located behind the camera centre in the *focal plane,* which is at distance f from the optical centre. The point at which the optical axis meets the image plane is the *principal point*. The plane through the camera centre parallel to the image plane is the *principal plane* (see Figure 106).

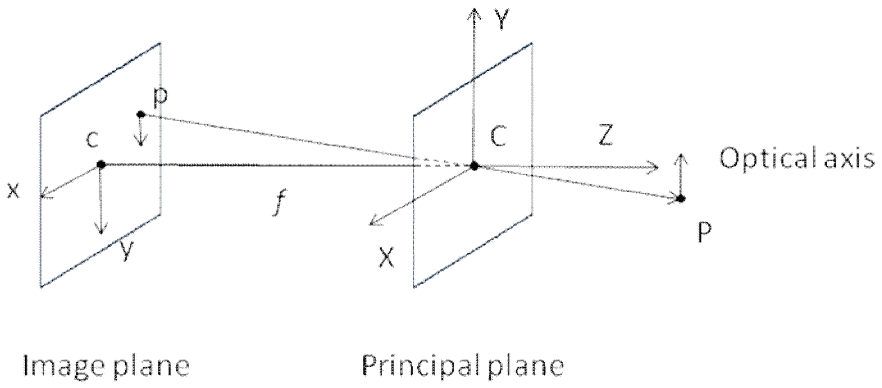


Figure 106. Principles of a pinhole camera: the optical axis goes through the camera centre (C) and the image centre (c). The image plane is perpendicular to the optical axis and at the distance of the focal length (*f*) of the camera. The principal plane is the plane parallel to the image plane and it goes through the camera centre. The world point P is mapped through the optical centre to the image plane to the image point p.

Perspective camera projection

In the pinhole camera model, the optical centre is at the coordinate origin (camera centre), and the Z-axis points forward (see Figure 106). We call this the *camera-centred view*. The image plane is located uniformly with the focal plane at the distance of the focal length (*f*) from the camera centre. Thus, the world point $\mathbf{X} = (X, Y, Z)^{\mathrm{T}}$ is mapped to the image point $\mathbf{x} = (f\, {}^{X}\!/\!_{Z}, f\, {}^{Y}\!/\!_{Z})^{\mathrm{T}}$, where the image coordinate origin is assumed to be at the principal point. Using homogeneous coordinates, we can present this in matrix form:

$$\mathbf{x} = \mathbf{PX} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & {}^{1}\!/\!_{f} & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}.$$

Camera model for a pinhole camera

In general, we may want to set the coordinate origins somewhere other than in the camera centre. We call this the *world-centred view*. The camera coordinates and world coordinates are related by rotation and translation (Figure 107 and Figure 108).

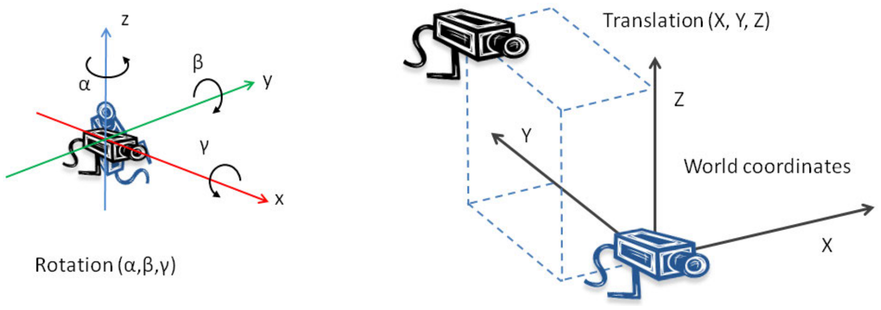


Figure 107. Camera rotation and translation into world coordinates.

Transformation $\mathbf{T}_{\mathrm{W2C}}$ between the camera coordinates and the world coordinates consists of rotation $\mathbf{R}$ and translation $\mathbf{T}$. In general, $\mathbf{T}_{\mathrm{W2C}} = \mathbf{RT}$.

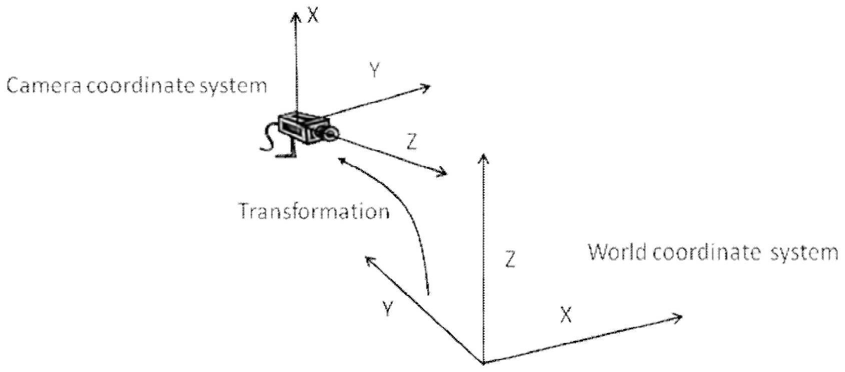


Figure 108. The camera coordinate system and world coordinate system are related through a transformation.

The pinhole camera model consists of the camera transformation and the perspective transformation $\mathbf{P}$. The camera projection matrix $\mathbf{M}_{pin}$ for the pinhole camera model is

$$\mathbf{M}_{pin} = \mathbf{PT}$$

In the case of a real camera, we also need to take into account the mapping from the image to the sensor coordinates $\mathbf{K}$. The mapping $\mathbf{K}$ from the image to the sensor coordinates consists of scaling $\mathbf{S}$ to pixel size, translation $\mathbf{T}_{img}$ and shearing $\mathbf{H}$ from the image coordinates to the actual pixel coordinates

$$\mathbf{K} = \mathbf{SHT}_{img} .$$

The camera projection matrix is

$$\mathbf{M} = \mathbf{KPT}.$$

Putting all this together, our camera model is

$$\mathbf{T}_{img}\mathbf{H}_{pix}\mathbf{S}_{pix}\mathbf{P}_{cam}\mathbf{R}_{cam}\mathbf{T}_{cam}.$$

We call this a simple camera model. In different applications, the transformations may be more convenient to calculate in a reversed way. In this case, the inverse of the matrix in question is used in the model.

Let $\mathbf{X}$ be world coordinates and $\mathbf{x}$ the corresponding image coordinates. The following illustrates the transformations into different coordinate systems

$$\underbrace{\mathbf{x}}_{\text{Image coordinates}} = \underbrace{\mathbf{T}_{img}\underbrace{\mathbf{H}_{pix}\underbrace{\mathbf{S}_{pix}\underbrace{\mathbf{P}_{cam}\underbrace{\mathbf{R}_{cam}\underbrace{\mathbf{T}_{cam}\underbrace{\mathbf{X}}_{\text{World coordinates}}}_{\text{Camera location}}}_{\text{Eye coordinates}}}_{\text{Ideal image coordinates}}}_{\text{Normalized device coordinates}}}_{\text{Pixel coordinates}}}_{\text{Image coordinates}}$$

Normally, we can multiply all these matrices together.

We divide the camera matrix into two parts. The first part consists of the camera-dependent part of the model; we call this *intrinsic camera matrix* $\mathbf{M}_{int} = \mathbf{T}_{img}\mathbf{S}_{pix}\mathbf{P}_{cam}$ and the parameters affecting this are called *intrinsic parameters*. The second part consists of the camera-independent part of the model; we call this the *extrinsic camera matrix* $\mathbf{M}_{ext} = \mathbf{R}_{cam}\mathbf{T}_{cam}$ and the parameters affecting this part are called *extrinsic camera parameters*.

$$\underbrace{\mathbf{T}_{img}\mathbf{H}_{pix}\mathbf{S}_{pix}\mathbf{P}_{cam}}_{\text{intrinsic camera matrix}}\underbrace{\mathbf{R}_{cam}\mathbf{T}_{cam}}_{\text{extrinsic camera matrix}}$$

The intrinsic parameters remain unchanged (unless the camera has a variable zoom lens). The intrinsic camera matrix is often calculated beforehand or at the start of the application in a calibration phase. The extrinsic camera matrix is updated as the camera moves. Next, we discuss the intrinsic and extrinsic parameters in more detail.

Intrinsic camera parameters

Intrinsic camera parameters describe how a particular camera forms the image. Two cameras are identical if they have same intrinsic parameters. The intrinsic parameters are the focal length, principal point offsets (x and y), skew and pixel aspect ratio.

Focal length

The perspective projection depends on the focal length of the camera f_c, and the perspective projection matrix is

$$\mathbf{P}_{cam} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f_c & 0 \end{bmatrix}$$

Sometimes other formulations for perspective projection are used.

Instead of mapping to the focal plane, we formulate the mapping to image co-ordinates $(X, Y, X)^T \rightarrow (u, v)$. We get

$$u = X\frac{f}{Z} \text{ and } v = Y\frac{f}{Z}.$$

Reformulating this

$$Zu = Xf \text{ and } Zv = Yf.$$

This may be written as

$$\begin{pmatrix} \lambda u \\ \lambda v \\ \lambda \end{pmatrix} = \begin{pmatrix} Xf \\ Yf \\ Z \end{pmatrix},$$

where $\lambda = Z$ is the homogeneous scaling factor.

We can write this in matrix form using homogeneous representation

$$\lambda\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \qquad 9.1$$

or as a 4×4 matrix

$$\lambda \begin{pmatrix} u \\ v \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{9.2}$$

We now have two representations (9.1 and 9.2) for a perspective matrix. In the latter representation, we have translated the whole image plane in the z-direction from $z = f$ to $z = 1,$ keeping the *x* and *y* coordinates as they are. On the other hand, this can also be considered as a projection to plane $z = 1$ and then scaling the *x* and *y* directions with the factor *f* (or scaling the *z* direction with the factor *1/f*).

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The *z*-coordinate is irrelevant, as is the scaling of the image at this stage, as we need to perform the scaling to the actual pixel size anyway. We see that these representations are consistent.

Principal point offset

In the ideal case, the optical axis goes through the centre of the image, but in practice due to camera assembly or structural restrictions, the centre of the image has some offset from the optical centre. For convenience, the image origin is often at the upper left or lower left corner of the image. Thus, the image coordinate origin is not located at the principal point. The difference is called the *principal point offset* (p_x, p_y) and we need to take it into account (see Figure 109).

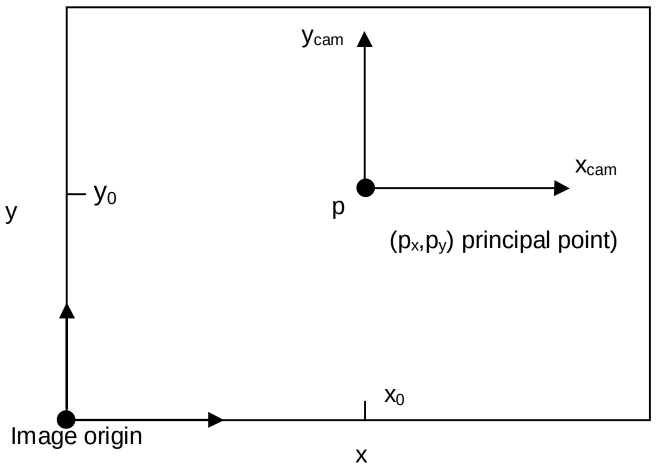


Figure 109. The principal point offset (p_x,p_y) is the difference between the camera's principal point and the image origin.

The principal point offset correction is simply a translation into a 2D image plane, and the corresponding translation matrix is

$$\mathsf{T}_{pix} = \begin{bmatrix} 1 & 0 & p_x & 0 \\ 0 & 1 & p_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

If we concatenate the perspective projection and principal point offset correction we get the following transformation matrix

$$\begin{aligned} \mathbf{T} &= \mathsf{T}_{pix}\mathbf{P}_{cam} \\ &= \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & p_x & 0 \\ 0 & 1 & p_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{aligned}$$

This gives us ideal image coordinates.

Pixel aspect ratio

We still need to project image coordinates to *pixel coordinates.* If the camera has non-square pixels, we need to take the aspect ratio into account. We mark the pixel height with $\mathrm{m_y}$ and pixel width with $\mathrm{m_x}$. The aspect ratio correction is a scaling operation, and in matrix form it is

$$\begin{bmatrix} \frac{1}{m_x} & 0 & 0 \\ 0 & \frac{1}{m_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Skew

In the ideal case, the pixel rows and columns are perpendicular and parallel to the image axes, but due to the camera assembly and structural restrictions, this is not always the case. We therefore need to take into account the rotation angle between the image axis and the pixels, called *skew.*

If the pixel rows and columns (e.g. the image sensor elements of the camera) are not perpendicular, we need to include the angle α between the pixel rows and columns, called *skew*, into the calculation.

First, we assume that we have rectangular pixels. However, the rows are shifted by the skew factor. The skew is a shear operation in a 2D image.

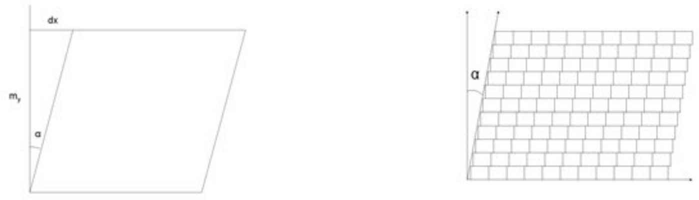


Figure 110. Skew between pixel axes.

From Figure 110 we see that $x = x' + dx;$ this is clearly a shear transform in the *xy* plane in the x-direction $x = x' + ay'$ and

$$\tan(\alpha) = \frac{dx}{y} \Leftrightarrow dx = \tan(\alpha)y.$$

Thus the shear factor is $\tan(\alpha)$, thus $x = x' + \tan(\alpha)y,$ and $y = y'.$

We can represent the skew operation using a shear transformation matrix in homogeneous format

$$\mathbf{H} = \begin{bmatrix} 1 & (\tan\alpha) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Now we have

$$\lambda \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \frac{1}{mx} & 0 & 0 \\ 0 & \frac{1}{my} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & px & 0 \\ 0 & f & py & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & (\tan\alpha) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix},$$

where $\lambda = Z$ is a homogeneous scaling factor. We multiply these matrices and get the formula

$$\lambda \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \frac{f}{mx} & (\tan\alpha)\frac{f}{mx} & px & 0 \\ 0 & \frac{f}{my} & py & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.$$

As only the pixel aspect ratio *mx/my* is important, only the ratios $\frac{f}{my}$ and $\frac{f}{mx}$ are important and we may simplify this into the form

$$\mathbf{x}_{pix} = \mathbf{K}\mathbf{x}_{img}$$

$$\begin{pmatrix} x_{pix} \\ y_{pix} \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & s & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

We call the matrix **K** the (*intrinsic) camera calibration matrix*. It is a mapping between the ideal image coordinates and the camera sensor coordinates. It is an upper triangular matrix and has *five degrees of freedom*. If $m_x = m_y$ then $f_x = f_y$ and if the pixel coordinate axels are perpendicular, that is $\alpha = 90^\circ$, the skew factor becomes zero, i.e. $s = 0$.

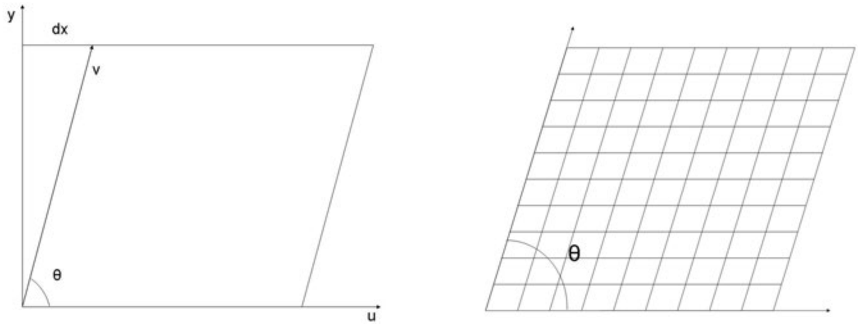


Figure 111. Skew model for skewed pixels.

Above, we assumed that the pixels have a rectangular shape. However, sometimes pixels can be skewed themselves. In this case, we need to use another skew model. Let θ be the angle between the coordinate axes (above α was the skew angle).

Now we have

$$x = \cot(\theta)\, y' + x' \text{ and}$$
$$y = \tfrac{1}{\sin(\theta)}\, y'.$$

In homogeneous matrix format this becomes

$$\begin{bmatrix} 1 & (\cot\theta) & 0 & 0 \\ 0 & \frac{1}{\sin(\theta)} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

This may be combined with other intrinsic matrices, in a similar way to the above.

Extrinsic camera parameters

We call camera parameters that are independent of the individual properties of the camera *extrinsic camera parameters* in contrast to intrinsic parameters. Extrinsic parameters are the camera location and pose relative to the world coordinates.

The extrinsic camera matrix is

$$\mathbf{M}_{ext} = \mathbf{R}_{cam}\mathbf{T}_{cam}$$

We denote the camera location with a three-dimensional vector $(\mathrm{X_c}, \mathrm{Y_c}, \mathrm{Z_c})^{\mathrm{T}}$ and in the homogeneous coordinates with the corresponding four-dimensional vector $(\mathrm{X_c}, \mathrm{Y_c}, \mathrm{Z_c}, 1)^{\mathrm{T}}$. With these notations the camera translation matrix becomes

$$\mathrm{T}_{cam} = \begin{bmatrix} 1 & 0 & 0 & X_c \\ 0 & 1 & 0 & Y_c \\ 0 & 0 & 1 & Z_c \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

As discussed in Appendix A, we can define the pose with three parameters: the rotation on the x-, y- and z-axes (α, β, γ).

$$\begin{aligned} \mathbf{R}_{cam} &= \mathbf{R}_z \mathbf{R}_y \mathbf{R}_x \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

The extrinsic camera matrix has six free parameters and therefore six degrees of freedom (DOF).

Lens distortions

The ideal pinhole camera model describes the image formation process for most cameras relatively well. However, real cameras often cause some sort of (systematic) error due to the physical properties or imperfections of the optical system. This imperfection in image formation by an optical system is called *optical aberration*.

Optical systems have several types of aberrations: piston, tilt, defocus, spherical aberration, coma, astigmatism, field curvature, image distortion and chromatic aberrations. From these aberrations, only distortions have an effect on the image geometry, and their effect on image formation must be taken into account in the camera model. The other aberrations only affect quality instead of the geometry of the obtained image and can therefore be ignored when forming the camera model.

There are two main types of distortion: *radial distortion* and *tangential distortion*. Radial distortion has a significant influence on the image geometry, especially with shorter focal lengths, whereas tangential distortion is often insignificant and can often be neglected. Radial distortion causes are mainly due to a flawed radial curvature curve of the lens elements [308].

Radial distortion

In *radial* distortion, points are moved in the radial direction from their correct position, and lines (other than in the radial direction) are bent. This type of distortion is mainly caused by a flawed radial curvature curve of the lens elements.

Depending on the way it affects the image, it is divided into *barrel distortion* and *pincushion distortion*. In barrel distortion, image magnification decreases with the distance from the optical axis, and in pincushion distortion, the image magnification increases. Barrel distortion is common for wide angle lenses and pincushion distortion for narrow angle lenses [309]. Pincushion distortion is also often seen in older or low-end cameras (e.g. camera phones). Fisheye lenses, which take hemispherical views, use barrel distortion as a way to map an infinitely wide object plane into a finite image area. On rare occasions, a mixture of these two may also occur; this is referred to as *moustache distortion*. It starts out as barrel distortion close to the image centre and gradually turns into pincushion distortion towards the image borders. Moustache distortion is observed with certain retrofocus lenses and on a large-range zoom.

In the distorted image, the straight lines are curved (see Figure 112).

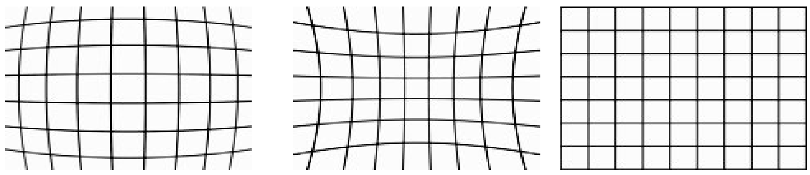


Figure 112. Barrel distortion, pincushion distortion and a non-distorted image.

Radial distortion is usually not perfectly rotationally symmetrical, but for a computation of distortion it is often assumed to be symmetrical.

The position of the point after ideal linear pinhole projection (*x,y*) is translated by radial displacement *L(r)*, which is a function of radial distance $r = \sqrt{x^2 + y^2}$ from the centre of radial distortion

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(r) \begin{pmatrix} y \\ y \end{pmatrix}.$$

The centre of the radial distortion may differ from the principal point, but as it is often close enough, we may use it as the best approximation for the centre.

The radial correction in pixel coordinates is

$$x_{ud} = x_c + L(r)(x - x_c) \text{ and } y_{ud} = y_c + L(r)(y - y_c),$$

where (x, y) are the observed coordinates, (x_c, y_c) is the *centre of radial distortion* and (x_{ud}, y_{ud}) are the corrected, undistorted coordinates and $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$. Sometimes the centre of radial distortion may be different from the principal point [310].

Due to the presumption of its rotational symmetry, radial distortion over an image can be presented as a general curve describing a dependency between a radial distance from the image centre and radial distortion. As an underlying distortion function is usually unknown and cannot be obtained by analytic means, the polynomial approximation of the radial distortion function is [311].

We may approximate the radial distortion function *L(r)* with a Taylor expansion

$$L(r) = 1 + a_1 r + a_2 r^2 + a_3 r^3 + a_4 r^4 + \dots$$

where (*a*$_1$, *a*$_2$,…, *a*$_n$) are distortion parameters. The complexity of the model is given by the number of terms of the Taylor expansion we use to approximate L(r).

Other approximations for radial distortion are also used. A commonly used approximation in a 3D reconstruction and augmented reality is Fitzgibbon's *division model*

$$\mathbf{x}_{ud} = \frac{1}{1+\lambda \|\mathbf{x}\|^2}\mathbf{x},$$

where **x** is the observed image coordinates, $\mathbf{x}_{ud}$ the undistorted coordinates and λ the estimated distortion parameter [312].

Radial distortion cannot be represented by a matrix. It can therefore not be multiplied into a calibration matrix; the system needs to perform it separately. In practice, many image processing tasks can be performed without correcting radial distortion. Radial distortion correction (i.e. warping the image) will distort the noise model by averaging and may therefore introduce an unwanted aliasing effect [72]. For this reason, feature detection should preferably be performed on the original image and then only the locations of the features will be undistorted; this is also faster. However, as geometrical characteristics change in distortion, for example, straight-line detection is better to perform on an undistorted image.

Tangential distortion

Sometimes the assembly of cameras is not precise. This may cause *tangential distortion* in addition to radial distortion. Tangential distortion bends lines starting from the centre of distortion in the tangential direction (Figure 113).

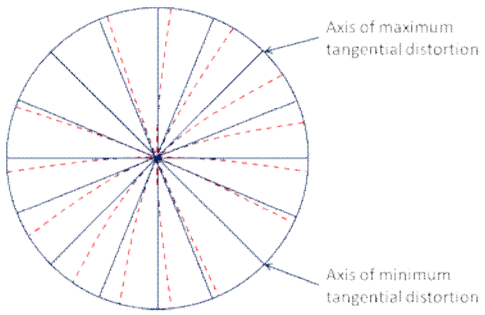

Axis of maximum tangential distortion
Axis of minimum tangential distortion

Figure 113. Solid lines show lines without tangential distortion and the corresponding red dashed lines illustrate the effect of tangential distortion.

Actual optical systems are subject to various degrees of decentring. Due to imprecise assembly, the optical axes of lens elements are sometimes not strictly collinear, and the image formation distorts. This defect introduces what is called *decentring distortion*. Decentring distortion has both radial and tangential components [308]. Sometimes the term decentring distortion is used in the meaning of tangential distortion.

Another typical source of tangential distortion is *thin prism distortion,* which arises from an imperfection in the lens design and manufacturing as well as camera assembly (for example, slight tilt of some lens elements or the image sensing array). This type of distortion can be adequately modelled by the adjunction of a thin prism to the optical system, causing additional amounts of radial and tangential distortions [308].

The ideal image plane is parallel to the lens. However, physical assembly may be imprecise and the image sensor may be askew compared with the lens, causing tangential distortion (Figure 114).

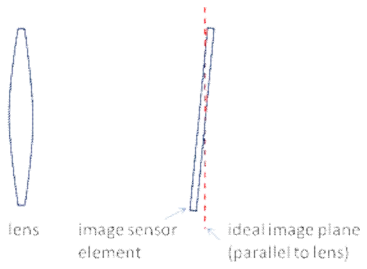


Figure 114. Tangential distortion occurs when, for example, the image sensor is askew compared with the ideal image plane that would be parallel to the lens.

The magnitude of tangential distortion is typically about 15% of the size of radial distortion [313]. Compensating for tangential distortion is therefore less critical than compensating for radial distortion. Consequently, systems may sometimes neglect it.

Different camera calibration toolboxes may use slightly different camera models and camera parameters. For instance, the Camera Calibration toolbox for Matlab uses the following model

$$\mathbf{x}_d = \begin{bmatrix} 1+k_1r^2+k_2r^4+k_5r^6 \\ 1+k_1r^2+k_2r^4+k_5r^6 \end{bmatrix} \mathbf{x}_n + \begin{bmatrix} 2k_3xy+k_4(r^2+2x^2) \\ k_3(r^2+2y^2)+2k_4xy \end{bmatrix},$$

where k_1, k_2 and k_5 are radial distortion parameters and k_3 and k_4 are tangential distortion parameters. Furthermore, $\mathbf{x}_n$ is a normalized image point after the pinhole camera projection and $\mathbf{x}_d$ is a new image point after the distortion function.

The conventional way of modelling lens distortion is to present it as a combination of radial, decentering and thin prism distortion. However, other models have also been proposed, for example, as a combination of radial distortion and a transform from the ideal image plane to a real sensor array plane [314].

Appendix C: Camera calibration and optimization methods

In this appendix, we discuss camera calibration, which is essential for computer vision applications including AR. We start the discussion with camera calibration and then review linear methods and non-linear optimization methods used for it. More profound surveys on camera calibration can be found in computer vision literature (e.g. [72, 74]), and for optimization methods, mathematical handbooks provide more information (e.g. [315]).

Camera calibration methods

Camera calibration means finding out the camera-dependent parameters for a scene model. Camera calibration includes, at least, approximating the intrinsic camera parameters and distortion functions. Sometimes it also includes finding out the extrinsic camera parameters (i.e. camera pose). Small differences in the camera assembly or physical variations in the individual lenses affect the parameters. Thus, for applications that require high accuracy, each camera needs to be calibrated individually, even if the cameras are of an identical model.

Camera calibration methods are based on the detection of known control points. Differences between measured and calculated control point coordinates are used to construct an approximation of the distortion functions and to estimate the intrinsic parameters.

Metric cameras are cameras specifically designed for photogrammetric tasks. They have a robust mechanical structure, well-aligned lenses with low distortion and a lack of autofocus and other functionalities that may, uncontrollably, change the internal geometry of the camera [316].

For metric cameras, the manufacturer usually provides exact intrinsic parameters as well as distortion coefficients, which simplifies the calibration process. However, most augmented reality applications are designed for use with ordinary (non-metric) cameras.

We can divide *non-metric cameras* into professional (high-quality) cameras and consumer (low-cost) cameras (e.g. USB cameras and camera phones). Non-metric cameras may have some of the features of the metric cameras but not all of them. In addition, autofocus, zoom lenses, image stabilizers, etc. can reduce the potential accuracy of a given camera [317]. Augmented reality applications designed for the mass market (e.g. games) are often based on low-end cameras, with various distortions and calibration results clearly affecting the visual quality of augmentation. Thus, camera calibration is an essential stage of augmented reality application development.

+Calibration patterns

Calibration methods commonly assume that the system can obtain an accurate set of correspondences between known world and image points. Calibration is often done using a known calibration pattern, *calibration rig*, e.g. a chequerboard pattern, but calibration is also possible using random features.

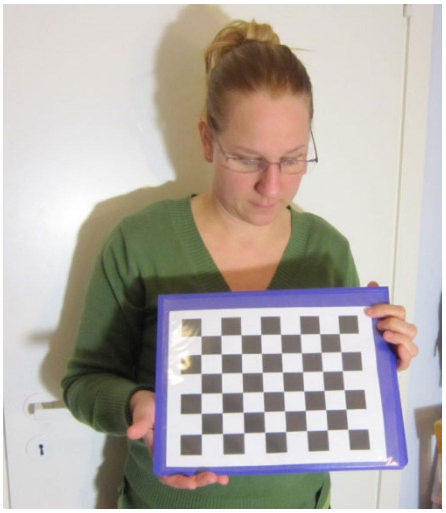


Figure 115. Example of a planar calibration rig with a chequerboard pattern.

The system can obtain these correspondences using, for example, a predefined *planar calibration pattern,* e.g. a chequerboard with a known pattern size (see Figure 115). It knows the exact position of each corner point in the pattern and can then use corner detection and/or line detection methods to find the corresponding points from the image. The advantage of a single planar calibration is that it is extremely easy to produce.

However, there are certain limitations with a single planar calibration pattern. The system can overcome these using a *multi-planar calibration object,* e.g. two (or more) perpendicular planes consisting of chequerboard patterns of known sizes. This kind of calibration pattern is relatively simple to produce.

Another solution is to use a *3D calibration pattern* of known formation, e.g. detectable points at the end of sticks pointing in different directions. A 3D calibration pattern is more complex to produce as it is not printable.

It is also possible to implement calibration without a calibration rig using feature detection. In *feature-based calibration*, features are detected and tracked, and the system calculates the camera movement and parameters based on the behaviour of the features.

Calibration using a special calibration rig is easier to implement than feature-based calibration and is thus more common. For example, in AR applications the widely used ARToolKit [64] provides a calibration tool with which the user prints a chequerboard pattern and moves it around in front of the camera. Similar calibration toolboxes are also available for Matlab and OpenCV.

Although feature-based calibration is more difficult to implement than methods using a calibration rig, feature-based methods are more user friendly as the calibration can be automated. For successful *auto-calibration*, the system needs a relatively large number of well-distributed points and at least one image must have a roll angle that is significantly different from the others [318]. However, the common user is unfamiliar with camera calibration and its requirements. The system may therefore need to guide the user to move the camera in an appropriate way.

Depending on the type of application and system assembly, it may be more convenient to solve the extrinsic parameters in the calibration process or at least the initial pose of the camera. For stationary cameras, it is convenient to solve the camera position and pose once with high accuracy rather than use valuable processing capacity to solve it at run-time. For rotating cameras the system only needs to discover the rotation for each frame in run-time, and for moving cameras it needs to calculate the pose for each frame.

For fixed lens cameras, the focal length is static and the system needs to solve it only once. For zoom cameras, the focal length changes, and even with the same zoom step (depending on the mechanical accuracy) it may vary. In the case of the zoom-lens camera, the system therefore needs to approximate the focal length every time the zoom factor is changed. The principal point may also change with zooming cameras [319].

A change in image resolution (digital zoom) does not require new calibration for intrinsic parameters, just scaling.

Calibration process

The calibration matrix can be solved using two different approaches: *linear methods (direct methods)* and *non-linear methods (iterative methods)*. In general, non-linear methods lead to a much more robust (and accurate) solution compared with linear methods. However, non-linear estimation is an iterative approach and may lead to a local minimum instead of a global minimum, if the initial guess for a calibration matrix differs too much from the real solution. It is therefore important to have a good initial guess. The common approach is to combine a linear and a non-linear method to reach the optimal solution. The linear method is used to obtain an initial estimate and the iterative non-linear method is used to optimize the solution. Direct methods for the initial estimate are computationally fast, but for the iterative process, attention needs to be paid to the computation time and an adequate stopping criterion for iteration (e.g. threshold for a reprojection error).

Calibration with a calibration rig can be carried out, for example, as follows: the system takes images from the calibration rig under different orientations by moving either the rig or the camera. It detects the known feature points (e.g. chequerboard corners) from the images. The distortion can be assumed to be small and the system can therefore ignore it when calculating the first estimates for intrinsic parameters with a liner method. These parameters give the ideal image positions of the features. Comparing them with the actual pixel positions, the system can calculate an initial guess for distortion parameters. It can then use an iterative non-linear method to calculate all the parameters by minimizing the reprojection error. This approach for calibration is proposed in, for example, [320].

The locations of the feature points in a calibration rig are known, but in feature-based calibration the locations of the features also need to be estimated.

In the following, we first review linear methods and then non-linear methods. We then go through some mathematical background related to them. At the end, we discuss implementing camera calibration with a maximum likelihood estimation problem.

Linear methods for estimating a calibration matrix

There are different approaches to calculating the intrinsic and extrinsic parameters for a specific camera setup. The basic idea is the same: we have our camera model $\mathbf{x} = \mathbf{MX}$, where $\mathbf{X} = (X, Y, Z, 1)$ is a 3D world point, $\mathbf{x} = (x, y)$ is the corresponding point in pixel coordinates and $\mathbf{M}$ is our camera model matrix, which we want to solve. We have a set of correspondences $\{(\mathbf{X}_i, \mathbf{x}_i) \mid i = 1, \ldots, N\}$ where $\mathbf{X}_i$ is a world point and $\mathbf{x}_i$ its projection to pixel coordinates. We can write

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \frac{1}{z_i}\mathbf{M}\begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix},$$

which can be written out as

$$z_i\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}\begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix}$$

Thus, we have following equations

$$\begin{aligned} z_i x_i &= m_{11}X_i + m_{12}Y_i + m_{13}Z_i + m_{14} \\ z_i y_i &= m_{21}X_i + m_{22}Y_i + m_{23}Z_i + m_{24} \\ z_i &= m_{31}X_i + m_{32}Y_i + m_{33}Z_i + m_{34} \end{aligned}$$

We substitute z_i in the first two equations with the last equation, sort the two first equations and we get

$$m_{11}X_i + m_{12}Y_i + m_{13}Z_i + m_{14} - m_{31}X_i x_i - m_{32}Y_i x_i - m_{33}Z_i x_i = m_{34}x_i$$
$$m_{21}X_i + m_{22}Y_i + m_{23}Z_i + m_{24} - m_{31}X_i y_i - m_{32}Y_i y_i - m_{33}Z_i y_i = m_{34}y_i$$

We can write these equations for N points in matrix form

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -X_1x_1 & -Y_1x_1 & -Z_1x_1 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -X_1y & -Y_1y_1 & -Z_1y_1 \\ & & & & & \vdots & & & & & \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -X_Nx_N & -Y_Nx_N & -Z_Nx_N \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -X_Ny_N & -Y_Ny_N & -Z_1y_N \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ \vdots \\ m_{33} \end{bmatrix} = \begin{bmatrix} x_1m_{34} \\ y_1m_{34} \\ x_2m_{34} \\ y_2m_{34} \\ \vdots \\ x_Nm_{34} \\ y_Nm_{34} \end{bmatrix}$$

A general 3 × 4 projective matrix has 11 degrees of freedom; it has 12 entries, but an arbitrary scale factor is involved, so one of the entries can be set to 1 without loss of generality. For example, Abdel-Aziz and Karara [321] used the constraint $m_{34} = 1$. However, if the correct value of m_{34} is close to zero, this assumption leads to undesired singularity [322]. Other constraints have therefore also been suggested. For example, constraint $m_{31}^2 + m_{31}^2 + m_{33}^2 = 1$ has been proposed in [323].

Another slightly different method for solving this is not to make any assumptions about the constraints. We have the equations

$$m_{11}X_i + m_{12}Y_i + m_{13}Z_i + m_{14} - m_{31}X_i x_i - m_{32}Y_i x_i - m_{33}Z_i x_i - m_{34}x_i = 0$$
$$m_{21}X_i + m_{22}Y_i + m_{23}Z_i + m_{24} - m_{31}X_i y_i - m_{32}Y_i y_i - m_{33}Z_i y_i - m_{34}y_i = 0.$$

We write these equations for all N points in matrix form

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -X_1x_1 & -Y_1x_1 & -Z_1x_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -X_1y_1 & -Y_1y_1 & -Z_1y_1 & -y_1 \\ & & & & & & \vdots & & & & & \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -X_Nx_N & -Y_Nx_N & -Z_Nx_N & -x_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -X_Ny_N & -Y_Ny_N & -Z_Ny_N & -y_N \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ \vdots \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

In matrix form, this is

$$\mathbf{Am} = \mathbf{0},$$

where *A* is a $2N \times 12$ matrix, **m** is a *12*-vector and **0** is a *2N* zero vector. The trivial solution $\mathbf{m} = \mathbf{0}$ has no physical significance. The non-trivial solution can be obtained by minimizing $\|\mathbf{Am}\|$

$$\min_{\mathbf{m}} \|\mathbf{Am}\|^2$$

To eliminate the trivial solution we require that

$\|\mathbf{m}\|^2 \neq 0 \text{ which implies } a\|\mathbf{m}\|^2 \neq 0, \forall a \in \mathbb{R}$,

thus we may set the constraint

$$\|\mathbf{m}\|^2 - 1 = 0$$

Thus the Lagrangian to be minimized is

$$\begin{aligned} L(\mathbf{m}, \lambda) &= \|\mathbf{Am}\|^2 - \lambda(\|\mathbf{m}\|^2 - 1) \\ &= (\mathbf{Am})^{\mathrm{T}} \mathbf{Am} - \lambda \mathbf{m}^{\mathrm{T}}\mathbf{m} + \lambda, \end{aligned} \quad 9.1$$

where $\lambda > 0$ is the Lagrange multiplier.

Differentiating this with respect to $\mathbf{m}$ and setting it equal to 0 gives us

$$\begin{aligned} \frac{\partial}{\partial \mathbf{m}} L(\mathbf{m}, \lambda) &= \mathbf{A}^{\mathrm{T}}\mathbf{Am} - \lambda\mathbf{m} = 0 \\ &\Leftrightarrow \mathbf{A}^{\mathrm{T}}\mathbf{Am} = \lambda\mathbf{m} \end{aligned}$$

Differentiating L with respect to λ and setting it to 0 gives

$$\frac{\partial}{\partial \lambda} L(\mathbf{m}, \lambda) = \mathbf{m}^T\mathbf{m} - 1 = 0$$

$$\Rightarrow \mathbf{m}^T\mathbf{m} = 1.$$

Pre-multiplying (9.1) by $\mathbf{m}^T$ gives

$$\mathbf{m}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{Am} = \lambda\mathbf{m}^{\mathrm{T}}\mathbf{m}$$

$$(\mathbf{Am})^{\mathrm{T}}(\mathbf{Am}) = \lambda$$

$$\|\mathbf{Am}\|^2 = \lambda.$$

Thus, minimizing $\|\mathbf{Am}\|^2$ equals minimizing λ. Furthermore, equation (9.1) tells us that $\mathbf{m}$ should be the eigenvector of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ with λ as the corresponding eigenvalue. On the other hand, we want to minimize λ. Putting all this together, our solution $\mathbf{m}$ is the eigenvector corresponding to the smallest eigenvalue of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.

Thus, we can solve $\mathbf{m}$ using the eigen-decomposition of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$. $\mathbf{A}$ is an $n \times m$ matrix, where $n > m = 12$. Furthermore,

$$rank\left(\mathbf{A}\right) + null\left(\mathbf{A}\right) = m,$$

where $null\left(\mathbf{A}\right)$ is the dimension of the null space of $\mathbf{A}$.

If $rank\left(\mathbf{A}\right) = 12$, then $null\left(\mathbf{A}\right) = 0$. In this case, $\mathbf{m} = \mathbf{0}$, which has no physical significance, is the only solution for the equation.

If $rank\left(\mathbf{A}\right) = 11$, then $null\left(\mathbf{A}\right) = 1$ and we have a (up to the scale factor) unique non-trivial solution.

If $rank\left(\mathbf{A}\right) < 11$, then $null\left(\mathbf{A}\right) \geq 2$. This means that there are infinite solutions for the equation.

There are two types of *degenerate configurations* that give an ambiguous solution for the camera matrix. The most important critical configurations are situations in which the camera and points all lie on a twisted cubic and the points all lie on a union of a plane and a single straight line containing the camera centre [72].

Thus, we need at least 6 world reference points in a *general position* (as opposed to degenerate configurations) to ensure a meaningful solution. (In fact, we need 11 equations, and as each point gives two equations, we thus need 5,5 points, i.e. for the 6th point it is enough to know only *x*-coordinates or *y*-coordinates.)

Non-linear methods for estimating a calibration matrix

An optimization problem in which the function $F(\mathbf{x})$ to be minimized can be expressed as a sum of squares of non-linear functions

$$F(\mathbf{x}) = \tfrac{1}{2}\sum_{i=1}^{m} f_i(\mathbf{x})^2 = \tfrac{1}{2}\left\| f(\mathbf{x})\right\|^2,$$

is called *non-linear least squares problem*. Here, $f(\mathbf{x})$ is an *m*-vector, where the *i*th component is the function $f_i(\mathbf{x})$, and the $\left\| f(\mathbf{x})\right\|$ is called the *residual* at $\mathbf{x}$. Here the constant ½ has been included to avoid the factor 2 in the derivatives. For example, the estimation of a camera calibration matrix is this type of *non-linear parameter estimation* problem.

Let $\phi(\mathbf{x},\mathbf{t})$ represent the desired model function. Here $\mathbf{t}$ is an independent variable and x_is are the parameters of the model. Let us denote the observed data points as $\{y_i\}$. The values $\{y_i\}$ are subject to experimental error. The functions f_i can be represented as $f_i(\mathbf{x}) = \phi(\mathbf{x}, \mathbf{t}_i) - \mathbf{y}_i$. If the model is to have any validity, we can expect the residual to be small. If the number of data points is not clearly greater than the number of parameters, then an arbitrary model will give a close fit to the data. Thus, we may assume that the number of data points x m is much greater than the number of estimated parameters [324].

In most cases it is advisable to take into account the specific properties of the non-linear least squares problem [324]. In particular, the gradient $\nabla F(\mathbf{x})$ and the Hessian matrix $\mathbf{H}$ of $F(\mathbf{x})$ have a special structure,

$$\nabla F(\mathbf{x}) = \mathbf{J}_{\mathbf{x}}^T f(\mathbf{x}) \text{ and}$$

$$\mathbf{H}_{\mathbf{x}} = \mathbf{J}_{\mathbf{x}}^T \mathbf{J}_{\mathbf{x}} + \sum_{i=1}^{m} f_i(\mathbf{x}) \mathbf{H}_i,$$

where $\mathbf{H}_i$ is the Hessian of $f_i(\mathbf{x})$. The least-squares methods are typically based on the assumption that the first-order terms will dominate the second-order term. This assumption is justified when the residual is small enough [324].

The calibration problem can be formulated as a non-linear least squares problem by minimizing the *reprojection error,* i.e. the distance between the image points $\mathbf{x}$ and the corresponding reprojected points $\mathbf{x}_\mathrm{i}' = \mathbf{M}_\mathrm{est} \mathbf{X}_\mathrm{i}$, where $\mathbf{M}_\mathrm{est}$ is the estimate for the camera matrix.

The reprojection error is

$$e = \sum_{i=1}^{N} \left(\left\| x'_i - x_i \right\|^2 + \left\| y_i' - y_i \right\|^2 \right)$$
$$= \sum_{i=1}^{N} \left(\left\| \frac{\mathbf{m}_1 \mathbf{X}_i + m_{14}}{\mathbf{m}_3 \mathbf{X}_i + m_{34}} - x_i \right\|^2 + \left\| \frac{\mathbf{m}_2 \mathbf{X}_i + m_{24}}{\mathbf{m}_3 \mathbf{X}_i + m_{43}} - y_i \right\|^2 \right),$$

where $\mathbf{m}_\mathrm{i}$ is the i*th* row vector of $\mathbf{M}_\mathrm{est}$ and *N* is the number of points. We see that this is a non-linear least squares optimization problem and that this can be solved using non-linear optimization methods. In this section, we introduce gradient descent, variations of the Newton method and Levenberg-Marquardt optimization methods, but first we revise Jacobian and Hessian matrices that are used in these optimization methods.

Jacobian and Hessian matrices

The *Jacobian matrix* is the matrix of all first-order partial derivatives of a vector-valued function. The important feature of a Jacobian matrix is that it represents the best linear approximation to a differentiable function near a given point.

Let f be a function $f:\mathbb{R}^n -> \mathbb{R}^m,\ \mathbf{y} = f(\mathbf{x}).$ We can present f with m component functions

$$y_i = f_i(\mathbf{x}),\ i = 1....m.$$

The Jacobian matrix of f is defined as follows,

$$J = \left(\frac{\partial y_i}{\partial x_j}\right) = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}.$$

The i*th* row of this matrix is the gradient of the function y_i, with respect to $\mathbf{x}$

$$J_i = \nabla_{\mathbf{x}} y_i,\ \text{for } i = 1,...,m.$$

If f is differentiable at a point $\mathbf{x}_0$, then the best linear approximation of f near the point $\mathbf{x}_0$ is given with the Jacobian as follows

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + J_f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0).$$

The *Hessian matrix* is a matrix of second-order partial derivatives of a multivariate function. That is, the gradient of the gradient of a function. The Hessian matrix describes the local curvature of a function of many variables. The Hessian (matrix) is sometimes also referred to by the term ‘functional determinants’.

The Hessian of the function $y = f(\mathbf{x})$ is

$$H = \left(\frac{\partial^2 f}{\partial x_i x_j}\right) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & & \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}.$$

A value of $\mathbf{x}$ for which $\nabla f(\mathbf{x}) = 0$ corresponds to a minimum, maximum or saddle point according to whether $\mathbf{H}_{\mathbf{x}}$ is a positive definite, negative definite or indefinite.

Hessian matrices are used in large-scale optimization problems within Newton-type methods because they are the coefficient of the quadratic term of a local Taylor expansion of a function

$$y = f(\mathbf{x} + \Delta\mathbf{x}) \approx f(\mathbf{x}) + \mathbf{J}(\mathbf{x})\Delta\mathbf{x} + \tfrac{1}{2}\Delta\mathbf{x}^T\mathbf{H}(\mathbf{x})\Delta\mathbf{x}.$$

A full Hessian matrix can be difficult to compute in practice: quasi-Newton algorithms are therefore often used. These methods use approximations to the Hessian, which are easier to calculate.

Optimization methods

In the following, we review commonly used optimization methods in optimization problems arising in AR. A more profound review of optimization methods can be found in literature; a good reference is, e.g., [315].

Gradient Descent Method

The gradient descent method (aka *steepest descent*) is a method of searching for the minimum of a function of many variables *f*. In each iteration step, a line search (i.e. searching for a minimum point along a line) is performed in the direction of the steepest descent of the function at the current location. In other words,

$$x_{n+1} = x_n + \lambda_n \nabla f(x_n),$$

where λ_n is a non-negative scalar that minimizes $f\left(x_n + \lambda_n \nabla f(x_n)\right)$.

Newton method

The Newton method is perhaps the best-known method for finding roots of a real-valued function. We can use the Newton method to find local minima (or maxima) by applying it to the gradient of the function.

We start the search with an initial guess x_0. To find the zero of the function f(x), we calculate a new value at each iteration step value based on the formula

$$x_{n+1} = x_n - \frac{f(x_n)}{\nabla f(x_n)},$$

To find the minimum, we calculate a new value using the equation

$$x_{n+1} = x_n - \frac{\nabla f(x_n)}{\nabla^2 f(x_n)}.$$

In a multidimensional case, this becomes

$$x_{n+1} = x_n - \mathbf{H}^{-1}\nabla f(x_n),$$

where $\mathbf{H}$ is the Hessian matrix.

The advantage of the Newton method is that its convergence is fast, the convergence of the basic method is quadratic, and there are also accelerated versions of the Newton method where the convergence is cubic.

Gauss-Newton method

The Gauss–Newton algorithm is a modification of the Newton method for solving non-linear least squares problems. The advantage of this method is that there is no need to calculate the second derivatives, which can be computationally demanding.

The Gauss-Newton algorithm is an iterative method to find the minimum of the sum of squares of m functions of n variables

$$\sum_{i=1}^{m} f_i^2(\mathbf{x}),\ \mathbf{x} \in \mathbb{R}^n,\ \text{where}\ m \geq n.$$

We have an initial guess $\mathbf{x}_0$, and at each iteration a new value is

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta,$$

where Δ is the solution of the normal equation

$$\mathbf{J}^T\mathbf{J}\Delta = -\mathbf{J}^T\mathbf{r}$$

where $\mathbf{r}$ is a vector of functions f_i. $\mathbf{J}$ is the Jacobian of $\mathbf{r}$ with respect to $\mathbf{x}$, both evaluated at the point $\mathbf{x}_n$, $\mathbf{J}_\mathbf{r}^T(\mathbf{x}_n)\mathbf{J}_\mathbf{r}(\mathbf{x}_n)\Delta = -\mathbf{J}_\mathbf{r}^T(\mathbf{x}_n)\mathbf{r}(\mathbf{x}_n)$.

As $m \geq n$ $\mathbf{J}^T\mathbf{J}$ is invertible,

$$\Delta = -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{r},$$

where $\mathbf{J}^T\mathbf{J}$ is an approximation to the Hessian ($\mathbf{H} \approx 2\mathbf{J}^T\mathbf{J}$).

In the zero residual case, where $\mathbf{r} = \mathbf{0}$ is the minimum, or when $\mathbf{r}$ varies nearly as a linear function near the minimum point, the approximation to the Hessian is quite good and the convergence rate near the minimum is just as good as for Newton's method .

Quasi-Newton Method

In practice, the evaluation of the Hessian is often impractical or costly. In quasi-Newton methods, an approximation of the inverse Hessian is used instead of the true Hessian. The approximation to the Hessian is updated iteratively, an initial matrix $\mathbf{H}_0$ is chosen (usually $\mathbf{H}_0 = \mathbf{I}$) and then it is updated each iteration. The updating formula depends on the method used.

We approximate the function with the Taylor series

$$f(\mathbf{x_k} + \Delta\mathbf{x}) = f(\mathbf{x_k}) + \nabla f(\mathbf{x_k})^T \Delta\mathbf{x} + \tfrac{1}{2}\Delta\mathbf{x}\mathbf{H}\Delta\mathbf{x},$$

where $\mathbf{H}$ is an approximation to the Hessian. The gradient to this approximation with respect to $\Delta\mathbf{x}$ is

$$\nabla f(\mathbf{x_k} + \Delta\mathbf{x}) \approx \nabla f(\mathbf{x_k}) + \mathbf{H}\Delta\mathbf{x}.$$

To find the minimum, we set this equal to zero

$$\nabla f(\mathbf{x_k}) + \mathbf{H}\Delta\mathbf{x} = \mathbf{0} \Leftrightarrow$$
$$\Delta\mathbf{x} = -\mathbf{H}^{-1}\nabla f(\mathbf{x_k}),$$

where the approximation to the Hessian $\mathbf{H}$ is chosen to satisfy the following condition

$$\nabla f(\mathbf{x_k} + \Delta\mathbf{x}) = \nabla f(\mathbf{x_k}) + \mathbf{H}\Delta\mathbf{x}.$$

This condition is not sufficient to determine the approximation to the Hessian. Additional conditions are therefore required. Various methods find a symmetric $\mathbf{H}$ that minimizes the distance to the current approximation $\mathbf{H}_{k+1} = \arg\min \left\| \mathbf{H} - \mathbf{H}_k \right\|$ for some metric.

Altogether, within each iteration step we calculate

1. $\Delta\mathbf{x}_k = -\alpha \mathbf{H}_k^{-1} \nabla f(\mathbf{x_k})$
2. $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k$
3. We calculate $\nabla f(\mathbf{x_{k+1}})$ and
   $\mathbf{y}_\mathrm{k} = \nabla f(\mathbf{x_{k+1}}) - \nabla f(\mathbf{x_k})$
4. We calculate new approximate to Hessian $\mathbf{H}_{k+1}$
   using the values calculated in item 3 or
   we may also calculate directly the inverse $\mathbf{H}_{k+1}^{1}$.

The variations in quasi-Newton methods differ in how they calculate the new approximation to the Hessian in step 4.

Levenberg-Marquardt method

The Levenberg-Marquardt method [325][326] [327] is an iterative procedure to solve a non-linear least squares minimization problem. The algorithm combines the advantages of the gradient descent (minimization along the direction of the gradient) and Gauss-Newton methods (fast convergence). On the other hand, it can be considered a trust-region method with step control [324].

The Levenberg-Marquard method is a heuristic method and though it is not optimal for any well-defined criterion of speed or final error, it has become a virtual standard for optimization of medium-sized non-linear models because it has proved to work extremely well in practice.

In a general case, there is no guarantee that it will converge to a desired solution if the initial guess is not close enough to the solution. In a camera calibration problem, for example, some linear method is therefore usually used to obtain good initial values.

Let f be a relation that maps the parameter vector $\mathbf{p}$ to the estimated values $\hat{\mathbf{x}}$. $\hat{\mathbf{x}} = f(\mathbf{p})$, where f is of the form $f = f_1^2(\mathbf{p}) + \ldots + f_n^2(\mathbf{p})$. Let $\mathbf{p}_0$ be an initial estimate for the parameter vector and $\mathbf{x}$ a vector of the measured values. We want to find a parameter vector $\mathbf{p}^+$ such that it minimizes the error $\varepsilon^{\mathrm{T}}\varepsilon$, where $\varepsilon = \mathbf{x} - \hat{\mathbf{x}}$.

Let $\mathbf{p} + \boldsymbol{\delta}_{\mathbf{p}}$ be the new parameter vector at each iteration, that is $\mathbf{p}_{n+1} = \mathbf{p}_n + \boldsymbol{\delta}_{\mathbf{p}_n}$. For small $\left\|\delta_{\mathbf{p}}\right\|$ we get a linear approximation for f using the Taylor series expansion

$$f\left(\mathbf{p} + \boldsymbol{\delta}_{\mathbf{p}}\right) \approx f\left(\mathbf{p}\right) + \mathbf{J}\boldsymbol{\delta}_{\mathbf{p}},$$

where $\mathbf{J}$ is the Jacobian matrix

$$\mathbf{J} = \frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}.$$

At each iteration step we need to find a $\mathbf{p}$ that minimizes the quantity

$$\left\|\mathbf{x} - f\left(\mathbf{p} + \boldsymbol{\delta}_{\mathbf{p}}\right)\right\| \approx \left\|\mathbf{x} - f\left(\mathbf{p}\right) + \mathbf{J}\boldsymbol{\delta}_{\mathbf{p}}\right\| = \left\|\boldsymbol{\varepsilon} + \mathbf{J}\boldsymbol{\delta}_{\mathbf{p}}\right\|.$$

This is minimized when $\mathbf{J}\boldsymbol{\delta}_{\mathbf{p}} - \boldsymbol{\varepsilon}$ is orthogonal to the column space of $\mathbf{J}$. This implies that

$$\mathbf{J}^T\left(\mathbf{J}\boldsymbol{\delta}_{\mathbf{p}} - \boldsymbol{\varepsilon}\right) = \mathbf{0} \Leftrightarrow \mathbf{J}^T\mathbf{J}\boldsymbol{\delta}_{\mathbf{p}} = \mathbf{J}^T\boldsymbol{\varepsilon}.$$

Here, the matrix $\mathbf{J}^T\mathbf{J}$ is the approximate Hessian. Equation 1.97 is a so-called *normal equation*. The Levenberg-Marquardt method uses a method called *damp-*

ing to solve $\boldsymbol{\delta}_{\mathbf{p}}$. In damping, the above equation is substituted by a so-called *augmented normal equation*

$$\left(\mathbf{J}^T\mathbf{J}+\lambda\mathbf{I}\right)\boldsymbol{\delta}_{\mathbf{p}}=\mathbf{J}^T\boldsymbol{\varepsilon}$$

Here, the elements on the left-hand side matrix diagonal are altered by a small factor $\lambda > 0$. Here, the λ is called a *damping term*.

$$\mathbf{J}^T\mathbf{J}\approx\mathbf{H}$$

is an approximation to the Hessian, which is obtained by averaging the outer products of the first order derivative (gradient). If f is linear, this approximation is exact, but in general it may be quite poor. However, this approximation can be used for regions where $\left\|\delta_{\mathbf{p}}\right\|$ is close to zero and a linear approximation to f is reasonable.

Thus, the equation (1.98) becomes

$$\left(\mathbf{H}+\lambda\mathbf{I}\right)\boldsymbol{\delta}_{\mathbf{p}}=\mathbf{J}^T\boldsymbol{\varepsilon}.$$

If the value of λ is large, the calculated Hessian matrix is not used at all; this is a disadvantage. The method is improved by scaling each component of the gradient according to the curvature, which leads to the final step of the Levenberg-Marquardt equation

$$\left(\mathbf{H}+\lambda diag(\mathbf{H})\right)\boldsymbol{\delta}_{\mathbf{p}}=\mathbf{J}^T\boldsymbol{\varepsilon}$$

and for each iteration

$$\boldsymbol{\delta}_{\mathbf{p}}=\left(\mathbf{H}+\lambda diag(\mathbf{H})\right)^{-1}\mathbf{J}^T\boldsymbol{\varepsilon}$$

Thus, the updating rule becomes

$$\mathbf{p}_{n+1}=\mathbf{p}_n+\boldsymbol{\delta}_{\mathbf{p}_n}=\mathbf{p}_n+\left(\mathbf{H}+\lambda diag(\mathbf{H})\right)^{-1}\mathbf{J}^T\boldsymbol{\varepsilon}.$$

Since the Hessian is proportional to the curvature of f, this implies a large step in the direction of low curvature and vice versa.

Stopping criteria

The iteration ends when one of the following conditions is met:

1. $\varepsilon^{\mathrm{T}}\varepsilon < \text{threshold } \varepsilon_0$
2. The relative change in magnitude of $\boldsymbol{\delta}_{\mathbf{p}}$ becomes smaller than threshold ε_2

3. The magnitude of the gradient becomes smaller than threshold $\mathbf{J}^{\mathrm{T}}\varepsilon < \text{threshold } \varepsilon_0$

4. The maximum number of iterations is reached $n > n_{\max}$.

Camera calibration as a maximum likelihood estimation

The calibration problem can be formulated as a *maximum likelihood estimation* problem [320]. We have *n* images and *m* points on a model plane. We assume that the image points are corrupted by independent and identically distributed noise. The maximum likelihood estimation can be obtained by minimizing the reprojection error

$$\sum_{1}^{n}\sum_{1}^{m}\left\|\mathbf{x}_{ij} - \hat{\mathbf{x}}_{ij}\right\|^2,$$

where $\mathbf{x}_{ij}$ is point *j* in image *i*, and $\hat{\mathbf{x}}_{ij}$ is the reprojection of the point $\mathbf{X}_j$ in image *i*. $\hat{\mathbf{x}}_{ij}$ is a function of rotation matrix $\mathbf{R}_{\mathrm{i}}$, translation $\mathbf{t}_{\mathrm{i}}$, camera intrinsic matrix $\mathbf{K}$ and point $\mathbf{X}_j$ in image *i*

$$\hat{\mathbf{x}}_{ij} = \hat{\mathbf{x}}_{ij}(\mathbf{K}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j)$$

and in a more general form also the function of radial distortion parameters k_1 and k_2

$$\hat{\mathbf{x}}_{ij} = \hat{\mathbf{x}}_{ij}(\mathbf{K}, k_1, k_2, \mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j)$$

Minimizing this is a non-linear optimization problem that can be solved using, for example, the Levenberg-Marquardt method.

Estimating a plane-to-plane homography

In a general case, world points (features) are distributed randomly in 3D space. However, there are several cases in which features are located on a plane rather than in free space, and the task is to find a plane-to-plane homography. Mapping a marker to an image plane or a calibration rig to an image plane are examples of such situations.

A plane-to-plane homography can be solved with, for example, direct linear transformation. A more accurate result can be reached using an iterative maximum likelihood estimation approach. We discuss these methods here.

Direct Linear Transformation (DLT)

We have n point-to-point correspondences $\{\mathbf{y}^i \leftrightarrow \mathbf{x}^i \mid i = 1 \ldots n\}$. From the formula $\mathbf{y}^i = \lambda \mathbf{H}\mathbf{x}^i$, we know that $\mathbf{y}^i$ and $\mathbf{H}\mathbf{x}^i$ differ by a scalar but have the same direction

$$\mathbf{y}^i \times \mathbf{H}\mathbf{x}^i = 0, \text{ for all } i = 1 \ldots n.$$

We use notation $\mathbf{x}^i = (x_1^i,\ x_2^i,\ x_3^i)^T$ and $\mathbf{y}^i = (y_1^i,\ y_2^i,\ y_3^i)^T$. Furthermore, it denotes the jth row of $\mathbf{H}$ as $\mathbf{h}^{j^{\mathrm{T}}}$. Thus, the cross-product becomes

$$\mathbf{H}\mathbf{x}^i = \begin{bmatrix} \mathbf{h}^{1^{\mathrm{T}}}\mathbf{x}^i \\ \mathbf{h}^{2^{\mathrm{T}}}\mathbf{x}^i \\ \mathbf{h}^{3^{\mathrm{T}}}\mathbf{x}^i \end{bmatrix}$$

and

$$\mathbf{y}^i \times \mathbf{H}\mathbf{x}^i = \det \begin{bmatrix} i & j & k \\ y_1^i & y_2^i & y_3^i \\ \mathbf{h}^{1^{\mathrm{T}}}\mathbf{x}^i & \mathbf{h}^{2^{\mathrm{T}}}\mathbf{x}^i & \mathbf{h}^{3^{\mathrm{T}}}\mathbf{x}^i \end{bmatrix}$$

$$= \begin{bmatrix} y_2^i \mathbf{h}^{3^{\mathrm{T}}}\mathbf{x}^i - y_3^i \mathbf{h}^{2^{\mathrm{T}}}\mathbf{x}^i \\ y_3^i \mathbf{h}^{1^{\mathrm{T}}}\mathbf{x}^i - y_1^i \mathbf{h}^{3^{\mathrm{T}}}\mathbf{x}^i \\ y_1^i \mathbf{h}^{2^{\mathrm{T}}}\mathbf{x}^i - y_2^i \mathbf{h}^{1^{\mathrm{T}}}\mathbf{x}^i \end{bmatrix} = \mathbf{0}.$$

We may write the last line as

$$\begin{bmatrix} \mathbf{0}^T & -y_3^i \mathbf{x}^{iT} & y_2^i \mathbf{x}^{iT} \\ y_3^i \mathbf{x}^{iT} & \mathbf{0}^T & -y_1^i \mathbf{x}^{iT} \\ -y_2^i \mathbf{x}^{iT} & y_1^i \mathbf{x}^{iT} & \mathbf{0}^T \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}. \qquad 9.2$$

Each line of the matrix actually represents three equations (one for each coordinate). Therefore, we can write this as $\mathbf{A}_i \mathbf{h} = \mathbf{0}$, where $\mathbf{A}_\mathrm{i}$ is a 3×9 matrix and $\mathbf{h}$ is a 9×1 vector, for each point *i*. The matrix $\mathbf{A}_i$ has rank two, as the third row is a linear combination of the first two in equation 9.2. Matrix $\mathbf{H}$ has eight degrees of freedom and each point gives us two independent equations. In consequence, we need at least four non-collinear points to solve the homography.

We combine the equations of four points into one equation

$$\mathbf{Ah} = \mathbf{0},$$

where $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4]^T$ is a 12×9 matrix of rank 8. We may use more than four points to get a statistically more reliable result. As the solution is only defined up to scale, we may choose $\|\mathbf{h}\| = 1.$ As the third row of each $\mathbf{A}_i$ is vain, we can leave them off and keep only the first two rows of each $\mathbf{A}_i$. In this case, $\mathbf{A}$ is a $2n \times 9$ matrix.

The (over-determined) system $\mathbf{Ah} = \mathbf{0}$ may not have a solution due to measurement errors, but we can find the best estimate by solving the least squares problem of minimizing $\|\mathbf{Ah}\|$ subject to $\|\mathbf{h}\| = 1.$

We can do this with, for example, *singular value decomposition* of $\mathbf{A}$,

$$\mathbf{A} = \mathbf{U\Sigma V}^T,$$

which gives us the $\mathbf{h}$ in the last column of $\mathbf{V},$ when $\mathbf{\Sigma}$ is a diagonal matrix with positive diagonal entries, arranged in descending order. Now we get the homography $\mathbf{H}$ from $\mathbf{h}$ by rearranging the terms.

The DLT, as presented above, gives us good results in an ideal case (with exact data and infinite precision). However, the real data are affected by noise and the solution will diverge from the correct result due to a bad condition number. This can be avoided with data normalization [328]. Therefore, we should include data normalization as an essential step of DLT [72].

Data normalization means simply transferring all data points so that they have a zero mean and unit variance. This means that each point x_i is replaced with a new point x_i^* such that

$$\mathbf{x}_i^* = \frac{\mathbf{x}_i - \boldsymbol{\mu}}{\boldsymbol{\sigma}},$$

where μ is the mean of the data points and σ^2 is the variance (the division here is element-wise division).

Data normalization is done independently for both images (the mean and variance are calculated separately for both data sets). Normalization transformation consists of translation and scaling, and it can be presented in matrix form

$$\mathbf{x}_i^* = \mathbf{Tx}_i.$$

After normalization, the DLT is applied as described above using the normalized data values. This gives us a normalized homography $\mathbf{H}_\mathrm{N}$.

We need to denormalize the homograph after DTL. Let $\mathbf{T}_1$ be the normalization transformation for the first image and $\mathbf{T}_2$ for the second image. The denormalized homography is

$$\mathbf{H} = \mathbf{T}_2^{-1}\mathbf{H}_\mathrm{N}\mathbf{T}_1.$$

Maximum likelihood estimate for homography

Another approach is to find the maximum likelihood estimate for a homograph based on a reprojection error.

We assume that the measurement (extraction) error has a Gaussian distribution. We also assume that errors occur in all directions with the same probability, thus they have a zero mean. These assumptions are justified with most of the key point extraction methods. We mark points in one image with $\mathbf{x}_i$ and in the other with $\mathbf{x}'_i$.

The maximum likelihood estimate for H then also maximizes the log-likelihood, which can be found by minimizing the sum of Mahalanobis distances

$$\sum_i \left\| x_i - x'_i \right\|_\Sigma^2 = \sum_i (x_i - \hat{x}_i)^T \mathbf{\Sigma}_{x_i}^{-1} (x_i - \hat{x}_i), \text{with respect to } \mathbf{H},$$

where $\hat{x}_i = \mathbf{H}x_i$.

We assume that the points are extracted independently, all with the same procedure. Consequently, we may assume that

$$\mathbf{\Sigma}_{x_i} = \sigma^2\mathbf{I}, \text{for all } i.$$

In this case, it becomes a least-squares problem of minimizing

$$\min \sum_i (x_i - \hat{x}_i)^T (x_i - \hat{x}_i), \text{ with subject to } \mathbf{H}.$$

We can solve this with, for example, the Levenberg-Marquardt method, for which the result of normalized DLT can be used as an initial estimate. In the case of finding a homography between a calibration rig and an image of it, we may assume that the locations of points in the first plane (rig) are known exactly and a feature extraction error only occurs in the image.

However, if we are looking for a homography between two images, both of which are subject to a measurement error, we need to estimate locations in both images in addition to $\mathbf{H}$. We assume that the errors for all points x_i and x'_i are independent with individual covariance matrices $\sum$ and $\sum'$. In this case, the optimal solution is

$$\underset{\mathbf{H},\hat{x}_i,\hat{x}_i'}{\arg\min} \sum_i \left\| x_i - \hat{x}_i \right\| + \sum_i \left\| x_i' - \hat{x}_i' \right\|, \text{ with subject to } \hat{x}_i' = \mathbf{H}x_i'.$$

Here $\hat{x}_i$ and $\hat{x}'$ are optimized feature positions.

A camera calibration matrix can be solved using some of the optimization methods discussed in this appendix or another variation of them or some other method. A common approach, however, is to combine a linear method (to get an initial estimate) with a non-linear method (to optimize the solution), and often to fine-tune all the parameters once more after finding the global minimum. Estimating the partial derivatives in Jacobian and Hessian matrices is often a problem when implementing optimization methods in practice. However, guidelines for solving it can be found in literature, e.g. [315] and [329]. More information can be found on matrix computations (e.g. [330]) and optimization methods (e.g. [315]) from mathematical literature and on camera calibration from computer vision literature (e.g. [72, 74]).

Series title and number
VTT Science 3

Title	**Theory and applications of marker-based augmented reality**
Author(s)	Sanni Siltanen
Abstract	Augmented Reality (AR) employs computer vision, image processing and computer graphics techniques to merge digital content into the real world. It enables real-time interaction between the user, real objects and virtual objects. In this work, we give a thorough overview of the theory and applications of AR. One of the challenges of AR is to align virtual data with the environment. A marker-based approach solves the problem using visual markers, e.g. 2D bar-codes, detectable with computer vision methods. We discuss how different marker types and marker identification and detection methods affect the performance of the AR application and how to select the most suitable approach for a given application. Alternative approaches to the alignment problem do not require furnishing the environment with markers: detecting natural features occurring in the environment and using additional sensors. We discuss these as well as hybrid tracking methods that combine the benefits of several approaches. Besides the correct alignment, perceptual issues greatly affect user experience of AR. We explain how appropriate visualization techniques enhance human perception in different situations and consider issues that create a seamless illusion of virtual and real objects coexisting and interacting. Furthermore, we show how diminished reality can improve the visual appearance of AR and the interaction with real-world objects. Finally, we discuss practical issues of AR application development, identify potential application areas for augmented reality and speculate about the future of AR. In our experience, augmented reality is a profound visualization method for on-site 3D visualizations when the user's perception needs to be enhanced.
ISBN, ISSN	ISBN 978-951-38-7449-0 (soft back ed.) ISSN 2242-119X (soft back ed.) ISBN 978-951-38-7450-6 (URL: http://www.vtt.fi/publications/index.jsp) ISSN 2242-1203 (URL: http://www.vtt.fi/publications/index.jsp)
Date	June 2012
Language	English, Finnish abstract
Pages	199 p. + app. 43 p.
Keywords	Augmented reality, AR, mixed reality, diminished reality, marker-based tracking, tracking, markers, visualization
Publisher	VTT Technical Research Centre of Finland P.O. Box 1000, FI-02044 VTT, Finland, Tel. 020 722 111

Julkaisun sarja ja numero
VTT Science 3

Nimeke	**Markkeriperustaisen lisätyn todellisuuden teoria ja sovellukset**
Tekijä(t)	Sanni Siltanen
Tiivistelmä	Lisätty todellisuus yhdistää digitaalista sisältöä reaalimaailmaan tietokonenäön, kuvankäsittelyn ja tietokonegrafiikan avulla. Se mahdollistaa reaaliaikaisen vuorovaikutuksen käyttäjän, todellisten esineiden ja virtuaalisten esineiden välillä. Lisätyn todellisuuden avulla voidaan esimerkiksi upottaa 3D-grafiikkaa videokuvaan siten, että virtuaalinen osa sulautuu ympäristöön aivan kuin olisi osa sitä. Tässä työssä esitän perusteellisen katsauksen lisätyn todellisuuden teoriasta ja sovelluksista. Eräs lisätyn todellisuuden haasteista on virtuaalisen tiedon kohdistaminen ympäristöön. Näkyviä tunnistemerkkejä eli markkereita hyödyntävä lähestymistapa ratkaisee tämän ongelman käyttämällä esimerkiksi 2D-viivakoodeja tai muita keinonäön keinoin tunnistettavia markkereita. Työssä kerrotaan, kuinka erilaiset markkerit ja tunnistusmenetelmät vaikuttavat lisätyn todellisuuden sovelluksen suorituskykyyn, ja kuinka valita kuhunkin tarkoitukseen soveltuvin lähestymistapa. Kohdistamisongelman vaihtoehtoiset lähestymistavat eivät vaadi markkereiden lisäämistä ympäristöön; ne hyödyntävät ympäristössä olevia luonnollisia piirteitä ja lisäantureita. Tämä työ tarkastelee näitä vaihtoehtoisia lähestymistapoja sekä hybridimenetelmiä, jotka yhdistävät usean menetelmän hyötyjä. Oikean kohdistamisen lisäksi ihmisen hahmottamiskykyyn liittyvät asiat vaikuttavat lisätyn todellisuuden käyttäjäkokemukseen. Työssä selitetään, kuinka tarkoituksenmukaiset visualisointimenetelmät parantavat hahmottamiskykyä erilaisissa tilanteissa, sekä pohditaan asioita, jotka auttavat luomaan saumattoman vaikutelman virtuaalisten ja todellisten esineiden vuorovaikutuksesta. Lisäksi työssä näytetään, kuinka häivytetty todellisuus, jossa virtuaalisesti poistetaan todellisia asioita, voi parantaa visuaalista ilmettä ja helpottaa vuorovaikutusta todellisten esineiden kanssa lisätyn todellisuuden sovelluksissa. Lopuksi käsitellään lisätyn todellisuuden sovelluskehitystä, yksilöidään potentiaalisia sovellusalueita ja pohditaan lisätyn todellisuuden tulevaisuutta. Kokemukseni mukaan lisätty todellisuus on vahva visualisointimenetelmä paikan päällä tapahtuvaan kolmiulotteiseen visualisointiin tilanteissa, joissa käyttäjän havainnointikykyä on tarpeen parantaa.
ISBN, ISSN	ISBN 978-951-38-7449-0 (nid.) ISSN 2242-119X (nid.) ISBN 978-951-38-7450-6 (URL: http://www.vtt.fi/publications/index.jsp) ISSN 2242-1203 (URL: http://www.vtt.fi/publications/index.jsp)
Julkaisuaika	Kesäkuu 2012
Kieli	Englanti, suomenkielinen tiivistelmä
Sivumäärä	198 s. + liitt. 43 s.
Avainsanat	Augmented reality, AR, mixed reality, diminished reality, marker-based tracking, tracking, markers, visualization
Julkaisija	VTT PL 1000, 02044 VTT, Puh. 020 722 111

Theory and applications of marker-based augmented reality

Augmented reality (AR) technology merges digital content with the real environment and enables real-time interaction between the user, the real environment and virtual objects. It is well suited for visualising instructions for manual workers in assembly, maintenance and repair, as well as for visualising interior design or building and constructions plans, for example.

This work gives a thorough overview of the theory and applications of AR for anyone interested in understanding the methodology, constraints and possibilities of augmented reality.

We explain the pipeline of AR applications, and discuss different methods and algorithms that enable us to create the illusion of an augmented coexistence of digital and real content. We discuss technological issues, application issues, and other important issues affecting the user experience. Also, we explain how appropriate visualization techniques enhance human perception in different situations and consider issues that create a seamless illusion of virtual and real objects coexisting and interacting. In addition, we discuss practical issues of AR application development, identify potential application areas for augmented reality and speculate about the future of AR.

In our experience, augmented reality is a profound visualization method for on-site 3D visualizations when the user's perception needs to be enhanced.

ISBN 978-951-38-7449-0 (soft back ed.)
ISBN 978-951-38-7450-6 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 2242-119X (soft back ed.)
ISSN 2242-1203 (URL: http://www.vtt.fi/publications/index.jsp)