Robust Augmented Reality

Oytun Akman

Robust Augmented Reality

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op maandag 3 december 2012 om
10.00 uur
door

Oytun AKMAN

Master of Science in Electrical and Electronics Engineering,
Middle East Technical University,
geboren te Ankara, Turkije.

Dit proefschrift is goedgekeurd door de promotor:

Prof. dr. ir. P.P. Jonker

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof. dr. ir. P.P. Jonker	Technische Universiteit Delft, promotor
Prof. dr. A.A. Alatan	Middle East Technical University
Prof. dr. J. Dankelman	Technische Universiteit Delft
Prof. dr. ir. B.J.A. Kröse	Universiteit Van Amsterdam
Prof. dr. H. Nijmeijer	Technische Universiteit Eindhoven
Prof. dr. R.C. Veltkamp	Universiteit Utrecht
Prof. dr. ir. A. Verbraeck	Technische Universiteit Delft
Prof. dr. F.C.T. van der Helm	Technische Universiteit Delft, reservelid

This work has been carried out as part of the FALCON project under the responsibility of the Embedded Systems Institute with Vanderlande Industries as the carrying industrial partner. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Embedded Systems Institute (BSIK03021) program.

Advanced School for Computing and Imaging

This work was carried out in the ASCI graduate school.
ASCI dissertation series number 265.

Copyright © 2012 by Oytun Akman

Cover design: Onur Kutluoğlu

ISBN: 978-94-6186-060-6

Author email: `oytunakman@gmail.com`

Contents

1 Introduction **1**

1.1 Augmented Reality and Mobile Computing . . . 2

1.2 From Art to Crime Scene Investigation . . . 5

1.3 Challenges . . . 8
  1.3.1 Challenges of mobile AR . . . 8
  1.3.2 Challenges introduced by CSI . . . 9

1.4 Requirements . . . 10

1.5 Thesis Outline . . . 12

2 System Architecture **15**

2.1 Hardware . . . 16
  2.1.1 Head Mounted Display . . . 17
  2.1.2 Cameras . . . 18
  2.1.3 Laptop . . . 22
  2.1.4 Backpack . . . 22

2.2 Software . . . 24
  2.2.1 Pose Tracking and Sparse Reconstruction . . . 24
  2.2.2 Dense Reconstruction . . . 24
  2.2.3 Human-Computer Interaction . . . 25
  2.2.4 Rendering Engine . . . 26
  2.2.5 Remote System . . . 27

3 Mathematical Framework **29**

3.1 Euclidean Transformation, Image Formation and Camera Models . 29
  3.1.1 Euclidean Transformation . . . 29
  3.1.2 Image Formation and camera model . . . 34
  3.1.3 Lens Distortion . . . 36

3.1.4 Camera Calibration . . . 37
3.2 Geometry of Two Views: Epipolar Geometry . . . 38
3.2.1 Triangulation . . . 40
3.3 Least-squares minimization methods . . . 40
3.3.1 Linear least-squares . . . 41
3.3.2 Non-linear least-squares . . . 42
3.3.3 Robust least-squares (M-Estimation) . . . 44
3.4 Random Sample Consensus (RANSAC) . . . 45
3.5 3D Reconstruction . . . 47
3.5.1 Bundle Adjustment . . . 48

4 Visual Odometry **51**
4.1 Introduction . . . 51
4.2 Related Work . . . 52
4.3 System Overview . . . 61
4.3.1 Image pre-processing . . . 63
4.3.2 2D feature detection . . . 64
4.3.3 Feature selection and new map-point initialization . . . 66
4.3.4 Temporal Inter-frame feature matching . . . 74
4.3.5 Pose Estimation . . . 82
4.3.6 Sparse Mapping . . . 87
4.4 Results . . . 89
4.4.1 Computational Time . . . 89
4.4.2 Pose Estimation Accuracy . . . 94
4.5 Conclusion . . . 103

5 Real-time Dense 3D Reconstruction **105**
5.1 Introduction . . . 105
5.2 Related Work . . . 106
5.3 System Overview . . . 107
5.3.1 Disparity Map Calculation . . . 108
5.3.2 Disparity Map Registration . . . 111
5.3.3 Disparity Map Refinement . . . 111
5.3.4 Post-processing . . . 114
5.4 Results . . . 114
5.4.1 Computational Time . . . 115
5.4.2 Map Quality . . . 115
5.4.3 Metric Measurement . . . 120
5.5 Conclusion . . . 120

6 Multi-cue Hand Detection and Tracking for HCI **121**
- 6.1 Introduction . . . 121
- 6.2 Related Work . . . 123
- 6.3 System Overview . . . 126
- 6.4 Hand Detection, Tracking and Pose Estimation . . . 126
  - 6.4.1 Color cue . . . 126
  - 6.4.2 Curvilinearity cue . . . 129
  - 6.4.3 Depth cue . . . 132
  - 6.4.4 Cue Integration and Bottom-up aggregation . . . 132
  - 6.4.5 Hand Tracking . . . 133
  - 6.4.6 Hand Pose Estimation . . . 135
- 6.5 Results . . . 137
  - 6.5.1 Tracking . . . 137
  - 6.5.2 Pose estimation . . . 140
- 6.6 Conclusion . . . 141

7 Conclusion **145**
- 7.1 System Evaluation in CSI . . . 148
- 7.2 Future Research Directions . . . 150

A Demonstration of the System **153**
- A.1 The night of the nerds . . . 153
- A.2 The opening of CSI Lab . . . 153
- A.3 Pauw and Witteman show . . . 153

Bibliography **157**

Summary **177**

Samenvatting **179**

Acknowledgements **181**

Curriculum Vitae **185**

Chapter 1

Introduction

> *In three words I can sum up everything I've learned about life:*
> *...it goes on.*
> **Robert Frost**

The field of *Computer Vision* is concerned with problems that involve interfacing computers with their surrounding environment through cameras, thus artificial vision systems can replace human perception in relatively simple, repetitive tasks. *Machine Vision* systems combine the capabilities of computer vision (such as perception) with other methods and technologies to provide services (actions) for industrial applications. Some examples are inspection of machine parts, detection of abnormal events in surveillance videos, reconstruction of the 3D world for navigation, and object recognition for pick and place robots in warehouses [2, 3]. These systems used to be static and confined to industrial sites, and often performing in real-time. Recent advances in technology, such as increase in computational power, improvement in peripherals and decreasing form-factor, allow the vision systems to be carried on moving platforms such as tablet PCs, mobile phones, but also mobile robots in which case we talk about *Robot Vision.* More general, it leads to the possibility of wearable visual computing that can assist the carrier agent in executing various perception-action tasks. For instance, (mobile) self-localization and mapping systems via vision sensors support robots to locate their position in 3D environments, perform autonomous path planning and obstacle detection while in motion. Face recognition and object recognition systems (combined with self-localization) are utilized in service robots that are employed to assist human users and perform simple tasks such as fetching objects. Moreover, web interfacing abilities of such mobile computers make the necessary sources of information (such as 3D maps, object databases or user databases) available from remote locations and enables off-site processing of captured visual data (i.e. cloud computing).

These mobile robots can be exploited in many applications. One example is a mobile robot assisting customers in a big shopping mall or visitors in a hospital by guiding them inside the buildings, showing them the way, helping them to find the necessary products and informing them about the content or procedures. However, in such applications where robotic actuation (i.e. grasping) is not necessary, the assistive services of a real, physical mobile robot can also be provided (cheaper) by a wearable visual computing device that is carried by the human user. Information and guidance can also be displayed virtually on the mobile device's display i.e. smart phone, tablet PC or head-up displays and a virtual avatar can replace the robot. These wearable visual computing devices share similar requirements for perception tasks with robots such as self-localization and mapping, Human-Computer Interaction (HCI) and face, object and action recognition. They provide similar capabilities but differ in some aspects since in contrast with a robot the actuation and the motion is done by the human user wearing the system.

Virtual and Augmented Reality technologies already have quite a history in this field and many attempts have been made to use Virtual Reality (VR) and Augmented Reality (AR) to create meaningful, immersive experiences incorporating humans and computers. It is appealing for many applications (such as in entertainment and gaming) to improve and enrich the visual perception, cognition and interaction by providing extra information and guidance that is not available in the immediate surroundings.

The work in this thesis focuses on the computer vision aspects of designing a wearable assistive AR system to allow interaction between users and their environment while providing tools, guidance and information to the on-site and off-site users to perform their tasks independently from each other.

1.1 Augmented Reality and Mobile Computing

Augmented Reality (AR) is the synthesis of real world and virtual (computer generated) graphics. In contrast to Virtual Reality (VR) in which the user is engaged in an entirely artificial world, in AR applications virtual imagery of objects is superimposed over, rather than completely replacing the real world and displayed to the user via wearable, hand-held or static displays.

Although the large potential of VR systems to alter the real world and immerse the user in an environment that can be hard to simulate in real life, such systems are not portable and are often limited to dedicated immersion rooms with screens or projection systems. Moreover, the user is not allowed to move freely in large scale environments due to the technological constraints such as displays fixed to walls or mobile resource limitations.

However, besides remote media rendering, many immersive applications such as augmented reality also require heavy interaction with the surrounding environment. The maximum interaction with a scene can be achieved when the user

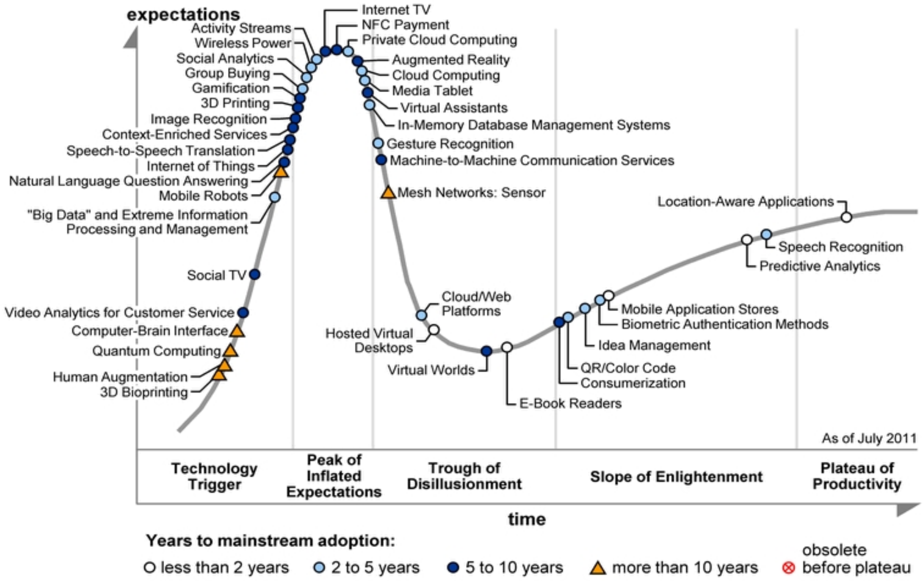
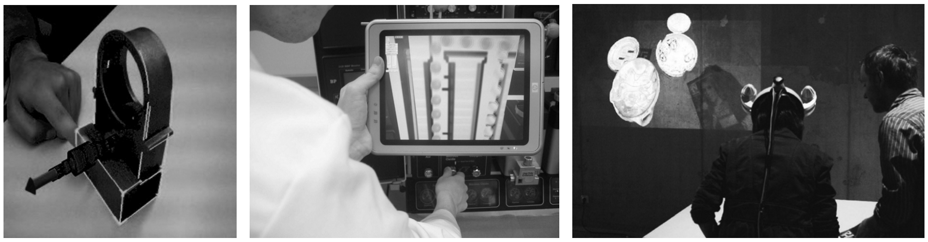


Figure 1.1: *Gartner's Hype Cycle for Emerging Technologies for 2011 [162].*

is mobile during operation. Such freedom demands a portable and easy to carry wearable setup, a personal wearable imaging/computing device with intelligence originating from the human user, while assistance is provided by the computing device [123]. Resource hungry AR applications -due to computationally expensive pose tracking and map building with natural features, HCI, high-resolution 3D rendering and video/audio analysis- are rapidly becoming available for mobile users as the necessary hardware is getting smaller and better. The introduction of small high-resolution cameras and wearable display hardware, wearable high performance computers, advanced battery and network technologies, and also the decrease in costs of off-the-shelf hardware for sensing and computing makes wearable computing and AR ubiquitous and commercially appealing.

As confirmed by Gartner's Hype Cycle for Emerging Technologies [162], expectations and complementary relation between mobile computing and AR are booming (Fig.1.1). With the developing technologies such as gesture recognition, machine-to-machine communication services and cloud computing, mainstream adoption of AR technologies is going to happen in the next 5 to 10 years and there is an enormous potential for novel immersive media technologies such as mobile AR applications. With these advances comes the possibility of offering desktop-quality (and beyond) immersive media experiences on state-of-the-art wearable computing devices that employs inside-out or ego-centric approach to sensing.

The commercial market already seems to be heading in this direction and a



(a) **(b)** **(c)**

Figure 1.2: *Recent examples of AR systems used in (a) maintenance [5], (b) medical education [160] and (c) cultural heritage [27].*

remarkable growth in the mobile devices drives the market towards the ultimate goal, which is composed of only a mobile phone (with one or more cameras, accelerometers, and GPS sensors) and a HMD (light-weight standard glasses size a.k.a. goggles or eye-wear). Some companies such as Zeiss, Sony, Epson, Apple and Google are working on such systems by creating new components or coupling their hardware (i.e. iPhone) and softwares (i.e. Street View) with the available technology.

AR systems have become commercially appealing, and applied to a number of different areas, such as design [84], medical imaging [182], medicine [160], military [60], education [90, 88], gaming [107, 199], assistance in maintenance and operation [5, 48], cultural heritage [27], civil engineering [170].

If we focus on the recent examples, Alvarez et al. [5] presented a markerless disassembly guidance system for maintenance and repair operations. Augmented virtual instructions such as virtual arrows and the next part to disassemble are generated automatically and displayed to the worker by superimposing them to the user's view. In [160] an Augmented Anesthesia Machine (AAM) which merges a Virtual Anesthesia Machine (VAM) with a real one is introduced. The system allows students to interact with a real machine while observing the results of their actions such as invisible gas flows. A so called magical lens is used for displaying virtual content on a hand-held display that tracks the real world via markers. Although their system is not see-through, they display the 3D model of the real machine combined with the virtual imagery so it appears to be a see-through experience. Caarls et al. [27] presented a truly mobile (roaming) AR system based on markers and inertia tracker data. An optical see-through Head-Mounted-Display (HMD) is used to display art and design content applied in museums.

1.2 From Art to Crime Scene Investigation

The Delft University of Technology initiated AR research in 1999 with outdoor head-mounted optical see-through AR, fusing data from a GPS, a natural feature tracking camera, and an inertia tracker, using a desktop PC in a backpack [150, 151]. Soon a switch back was made to indoor AR based on markers and inertia tracker data via Caarls' system [26] in order to improve the accuracy of the head-pose estimation by the vision system and to obtain measurements on the static and dynamic accuracy of the estimates. Meanwhile, a collaboration was set up with the AR Lab of the Royal Academy of Art in The Hague [105] to test the developed systems with art and design content, applied in museums such as Kröller-Müller, Escher, Boijmans van Beuningen and Van Gogh, as well as design manifestations such as the Salone di Mobile in Milano [27, 98]. In demonstrations, data gloves and RFID tags were utilized for interaction between the users, virtual content and the environment.

Especially the experience at the Salone di Mobile where two users, each with their own AR headset discussed the designs of the virtual furniture, lead to contacts with Dutch companies such as DAF (trucks), Fokker Services (special aircrafts) and Driessen Aerospace (crew-rests) to investigate the possibilities of using AR in collaborative product design; i.e. at the interfaces between customer and system architect, and system architect and implementation engineers. The idea was to investigate whether it is feasible to change the centrally conducted multidisciplinary design reviews of one director and a multidisciplinary team of engineers in which during a whole day a script was plowed in which all design details are discussed by all in front of a huge CAD screen, into a scenario in which all engineers wearing an AR headset roam around a 3D virtual object, hence autonomously and in parallel groups discuss the details of the design and its possible faults; whereas the role of the director is to collect the omissions and proposed improvements of the total design. A theoretical framework for such cooperated designs was set up by the University of Maastricht and the Industrial Design Faculty of the Delft University of Technology [33]. The work came to a temporarily hold as it awaited the availability of suitable AR headsets and its camera tracking mechanisms; i.e. the outcome of this thesis.

Hence, we shifted our focus onto spatial analysis using multiple AR systems, which is presented in this thesis. A close collaboration was set up with the Systems Engineering Section (SES) of the Faculty of Technology, Policy and Management (TPM) of the Delft University of Technology that was involved in a project on Crime Scene Investigation (CSI); CSI the Haag [192]. This project is under the guidance of the Netherlands Forensic Institute (NFI), an agency of the Ministry of Justice. This project acted as a simpler use case to study collaborative AR and our goal now became to design a system that supports collaboration between one or more crime scene investigators (CSIs) on the spot (the crime scene) and one or more remote experts at a distance. Note, however, that the designed platform (software and hardware) can be utilized in many different applications, such as in collaborative design of complex systems involving multiple disciplines.

In close collaboration, the image processing and the AR tools are developed by our group (Delft Bio-robotics Laboratory) and reported in this thesis, while the project domain and expertise about collaboration is brought by TPM-SES, who focused on the Computer-Supported Cooperative Work (CSCW) aspects, such as the design of the GUI, remote collaboration, communication between the collaborators, etc.. During the development process both parties were highly involved in the discussions and most of the work is a result of a close interaction.

When a severe crime is committed, a team of specialists and multiple CSIs investigate the scene to collect evidential data and clues, while preserving the scene as much as possible so that evidence is not accidentally destroyed. Meanwhile, the crime scene is digitized by either photogrammetry [174] or laser scanning [85] methods to document the structure of the scene while tagging observations manually within constrained time limits. However, this process is currently costly in time and requires expertise. There are wearable systems to support evidence recovery using RFID tags available in literature [12, 41]. Collected (digital) data is utilized for the communication between the teams, for archiving and also analysis of the incident such as line of sight determination, reconstruction of ballistic trajectories, blood pattern analysis, reconstruction of crime scene, etc.

For crime scene investigation, an AR system would be suitable since the user can perform many actions such as recording, tagging, labeling and measuring in the context of the physical environment, while receiving assistance from other co-located or remote colleagues and experts. Information about the crime scene captured in its spatial context facilitates greater efficiency and maintains the quality of the data [65]. Moreover, the evidence and reconstruction of the scene can be presented in courtrooms in a digital format to help the judge and jury to visualize the incident and improve their comprehension by establishing a common ground [25, 173].

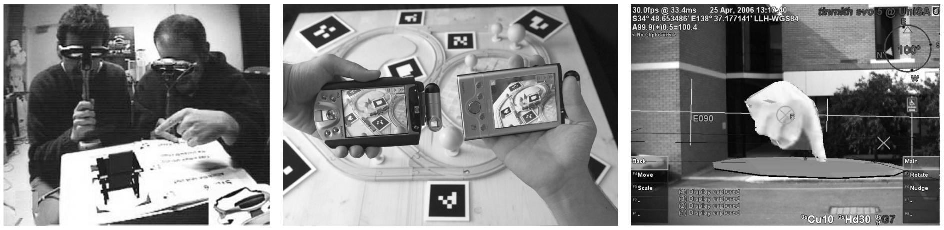
In an AR based set-up the investigators equipped with headsets and wearable computers annotate salient information in the scene by placing virtual tags. Salient information can vary from the positions of bullet shells, possible bullet paths and bullet holes, to the pose of the victims and the possible pose and paths of the suspects. The virtual annotations can also be seen and modified by other team members including remote experts. For instance, in a murder case the additional personal information of the victim can be loaded into the view of on-site CSIs by a remote expert after his/her face is seen by one of the cameras. The interaction with the scene, such as the virtual tagging, is done with hand gestures, since CSIs need to use their hands to perform their jobs. At the same time, the vision sensors capture the images of the scene and in near real time the 3D scene is reconstructed, hence indicating where the investigators have been. The whole operation can be supervised by a superior who can decouple him/her self from the on-site investigator's view and wander around in the 3D scene built up so far; possibly directing the investigators to investigate some spots closer. The 3D scene as it was found at the time of the incident can be revisited (visualized) and its search can be played back weeks, months or years after the event for proper authorities, such as attorneys, judges and juries.

In general, besides CSI, AR can provide a key to overcome contextual differences and establish a common ground and a shared understanding among users [155]. Here common ground refers to the mutual knowledge and beliefs shared by the users [36]. In AR, virtual content and the reality can be merged in the same context, leading to a common ground and therefore different knowledge types and representations can be combined [160]. This results in a better comprehension of complex concepts while performing complex tasks. For instance, increased product complexity requires teams (experts) with diverse backgrounds in the design loop in which these teams need to share expertise and knowledge, and communicate for high quality designs [33]. Or in many practical situations, as in a field worker inspecting an underground infrastructure, or mechanics at ships on high sea a small number of experts who are located in off-site locations give assistance to multiple on-site workers. In both examples, the experts have technical expertise and comprehensive understanding about the field, but are dispersed over the world and also expensive to educate. Therefore, instead of deploying them to every location, an effective system that can allow the field workers to share their environment with the remote expert while getting assistance is required [101]. Therefore, there is a growing demand for technologies to realize remote collaboration on physical tasks by creating shared visual space [63].

Sharing the same visual space (physical and augmented) introduces a strong mutual experience, while being able to alter each other's perception of reality allows enhanced communication. As presented in the seminal work of Billinghurst et al. [17], AR interfaces provide a medium for users to work in both the real and virtual world simultaneously, facilitating Computer-Supported Cooperative Work (CSCW) in a seamless manner.

There are research examples in which audio and video images are sent to the remote collaborator via a wearable active camera/laser system and the remote expert can point real object via a laser [102]. Also, in one of the earliest works on wearable collaborative systems, a field worker equipped with a HMD and a camera, transfers images to a remote expert and receives commands back in his HMD. The remote expert uses his finger to indicate regions [104]. However, while demonstrating collaboration examples these systems don't provide any augmented virtual content.

In [17], Billinghurst et al. presented a collaborative system that uses a book as the main interface object and hand held displays to provide virtual content. Several readers can read and share the story together and can move between the real and virtual world by utilizing hand held displays. Users can switch from egocentric and exocentric views and interact with the characters in the story. A more recent example with mobile AR platform is presented in [199]. In the invisible Train game of Wagner et al., virtual trains are augmented onto the real train tracks by using hand-held PDAs with cameras and the player can interact with the trains using a stylus based interface. The system allows multiple users to



(a) (b) (c)

Figure 1.3: *Collaborative AR examples from (a) [17], (b) [199] and (c) [185].*

play together. Stafford et al. [185] proposed a God-like interaction metaphor to facilitate collaboration via communication of situational and navigational information between indoor users equipped with tabletop displays, and outdoor users equipped with mobile AR systems. The outdoor user sees the indoor user's hand (or other object) appearing from the sky and pointing the location of interest to describe situational information or navigational tasks.

1.3 Challenges

In his seminal works [9, 10], Azuma defined the properties of AR systems as:

- combines real and virtual objects in a real environment
- registered in the 3D surroundings
- interactive in real time

The first property requires the AR system to track the pose of the user's head for accurate virtual image overlay. When the system is not fast or accurate enough to detect the motion of the user, then the perception of combined virtual and real content cannot be preserved due to jitter or lagging.

Another challenge introduced by the second property is capturing the structure of the 3D surroundings for accurate registration. If registration fails, then the virtual content is not attached to the real scene and displayed in unrealistic way such as inside the walls or floating in the air.

Last but not least, many immersive AR applications require heavy interaction with the surrounding environment in real-time. Natural interaction techniques for AR systems are hard to realize and are highly dependent on the scene conditions.

1.3.1 Challenges of mobile AR

As the AR user becomes mobile in order to execute certain tasks, the continuously changing context of the mobile environment introduces new challenges such as performing robust operations in unknown environments. Tracking the camera becomes more difficult when the systems enters an unknown space and solutions such as placing landmarks (fiducials) to ease navigation becomes more difficult (if not impossible) as the operation area becomes larger. In addition to that, with changing environmental conditions such as lighting or the amount of texture, as the user walks around, tracking and intuitive operation becomes more difficult.

The limited resources available on even high-end mobile devices (such as battery life, processing power) and real-time constraints currently avoid the utilization of computationally expensive but robust solutions.

Another important challenge is the selection and combination of hardware in order to provide a modular and non-intrusive design setup that includes wearing comfort. Wearable computers involve concern since they are designed to be carried on the user. They need to be minimal in the sense of weight and size, and ergonomics is another important criterion, especially when the system is carried by an on-site user for a long time. Proper wired or wireless connectivity of both displays and tracker cameras to increase the freedom of the user is another challenge that needs to be addressed.

1.3.2 Challenges introduced by CSI

The following current challenges have been identified during interviews with CSIs from the USA, UK and the Netherlands [155]:

Time needed for reconstruction: During the investigation, data capture, alignment, data clean-up, geometric modeling and analyses steps that are necessary for crime scene reconstruction are done manually. The captured data via scanners or cameras, are transferred to another group of experts and processed for further investigation. Afterwards, the final output is analyzed. Performing these steps separately with different investigators requires a lot of time and resources.

Expertise required to deploy dedicated software: The necessary software for investigation is prone to require dedicated expertise. It is not always possible for an investigator to utilize the various tools by him/her self and perform the investigation; and therefore the data needs to be transferred between various experts, which consumes time as mentioned in the previous challenge.

Complexity: Situations vary significantly between different crime scenes and this requires an adaptable system that can perform under various environmental conditions.

Time freeze: Data capture is often conducted once after a scene has been contaminated or altered. However, for a better evaluation of the case it is important to capture evidences and perform analysis before the scene is altered. Capturing all the details during the first interaction with the scene and therefore performing multiple operations together in real time is another challenge which needs to be addressed.

Physical interaction with the scene: Physical interaction with the crime scene is necessary to perform analysis and tagging, but should be as minimum as possible to reduce the scene contamination and therefore placement of markers is not preferable. Also the CSIs use their hands to perform investigation and therefore holding a device which occupies one of their hands restrains the freedom for investigation. As a result, the interaction challenge in the previous section goes one step further and the option of using auxiliary devices is eliminated [116].

1.4 Requirements

Our objective is to design an AR system that can be used for commercial applications such as art, design and serious gaming but also on the spot generation of 3D annotated worlds for crime scene investigation. Our design focuses on the selection and combination of necessary hardware and software that is required for mobile AR experience. To achieve acceptance in a consumer market and overcome the challenges mentioned in the previous section, the designed system should meet the following requirements mainly derived from the CSI applications and challenges:

Marker-less, extensible tracking for augmenting a real scene with virtual objects

Many applications and use scenarios such as art exhibits, spatial analysis for CSI, museums, serious gaming may require mobility and the users may need to move around while executing certain tasks. In order to provide assistance and context to these users while they are in motion, the AR system needs to extend its tracking region as the user moves around in an unknown environment. New parts of the scene must be added to the tracking region and new landmarks need to be added to localize the user in the scene. This enables the users to modify or create new AR experiences at locations of interest. We aim for an extensible tracking in medium sized environments such as offices and rooms of houses, and tested the system by performing tracking in these environments starting from only a part of the scene.

Also placing markers is not an option for a crime scene. The first investigator that arrives on a crime scene has to keep the crime scene as untouched as possible. Technology that involves preparing the scene is therefore unacceptable. Therefore marker-less tracking and a dynamic solution equipped with advanced adaptation algorithms are required.

The accuracy of the tracker must be within a small fraction of a degree in orientation and a few millimeters (mm) in position [8]. Also the processing time should be very low so that the user does not feel the delay between the time that the tracker takes the measurements and the time that the graphics engine renders the augmented image in the display. The system should perform in real time to maximize the experience of the user. Although the required accuracy and processing time strongly depend on the user and the application, in our requirements we set the combined latency to less than 50 milliseconds and the accuracy should be minimal such that the user doesn't feel any jitter as explained in more detail in Chapter 4.

On-line and on-site scene structure capturing

Fast, accurate reconstruction of the scene geometry is required for correct virtual image registration. Also, establishing a common ground between the remote and co-located users to perform tasks such as tagging requires a medium as explained in the previous sections. Such a 3D reconstruction can support the standard ways such as photographs and drawings, and on-line operation can decrease the time needed for reconstruction.

Reconstructed scenes are required to successfully represent the scenes and be visually satisfying. Although the maps are not created for precise measurements, a spatial resolution of a few centimeter is set as a requirement to evaluate our system. Also the maps are required to be created on the fly as the user moves and therefore the processing speed should be fast. Considering a slowly walking user looking towards a medium sized room from a couple of meters, we set the maximum computational time of creating a dense map from a stereo image pair as 1 fps.

Hand gestures for user interface operation

Exploitation of the user's hands as an interaction and pointing device instead of other auxiliary equipment introduces more freedom to the user. Also from the CSI point of view, the hands of the CSIs have to be free to physically interact with the crime scene when needed, e.g. to secure evidence, open doors, climb, etc [116]. Additional hardware such as data gloves or physically touching an interface such as a mobile device is not acceptable.

In order to provide a natural HCI, the user interface requires to operate in real time and in parallel to the tracking so that the user can interact with the system while the tracking is running in the background. As a requirement in accuracy, the system needs to operate and detect the pose of the hands as accurate as the pose of a hand sized tool that has a standard AR marker on it.

Robust and repeatable operation

Although, robust and repeatable operation is important almost for all applications including entertainment, it becomes more crucial when CSI is considered. The system should perform under challenging field conditions with minimal reconfiguration and produce similar results for consistent data gathering.

A lightweight and affordable head-mounted display (HMD) and a wearable computer

For commercial and mobile applications, the overall system should be relatively cheap, light, small, ergonomic, made of adjustable wearable computer components and mounted on a suitable location on the user to avoid interfering with the user's tasks. It became clear that the investigators whom arrive first on the crime scene currently carry a digital camera. Weight and ease of use are important design criteria. Experts would like those close to a pair of glasses. This is because hand-held or wrist worn devices do not support hands free applications, or do not allow correct positioning of virtual objects and relevant information in the 3D scene.

The following design considerations and requirements are important for an AR system for CSI and we considered/addressed them in our design. But the evaluation of these requirements are beyond the scope of this thesis and will be addressed by TPM-SES.

Remote connection to and collaboration with experts

Experts (crime scene investigators) are a scarce resource and are not often available at location on request. Setting up a remote connection to guide a novice worker (investigator) through the (crime) scene and to collaboratively analyze the (crime) scene has the potential to improve the task (investigation) quality. This leads to a requirement in which both one or more on-site CSIs as well one or more off-site experts can build up and maintain the virtual world that is built up during investigation from the unknown environment, and annotate it.

User friendliness

The attention of the user working on a relatively complex task while using the system should be focused on the task rather then the system. Therefore the expertise required to deploy dedicated software should be minimal which requires non-intrusive, intuitive and user-friendly systems. The system should be easy to start up, and intuitive to control. Important tasks such as tracking, 3D reconstruction and hand detection should be done automatically without distracting the user's attention.

1.5 Thesis Outline

This thesis describes the design of a complete marker-less mobile/wearable AR system that includes creating the necessary software modules and combining them with the necessary hardware. Moreover, the presented novel hand-based interaction method for the user to manipulate the AR content without any auxiliary device is combined with on-line scene reconstruction. According to our knowledge, this is one of the first examples of a complete 3D stereo AR system that integrates 3D marker-less AR capabilities with dense reconstruction and human-computer interaction (HCI) in a carefully engineered way, and applied to the CSI domain. The contributions of this thesis are explained in the following chapters:

In Chapter 2, the overall system, hardware and software design is presented. We discuss various options and come to a solution that satisfies the requirements and overcomes the challenges. We further present the software components in more detail and discuss how a remote user can decouple him/her self from the on-site user's view while assisting the crime scene investigation.

In Chapter 3, the mathematical background and notation of this thesis is summarized. Readers who are already familiar with the principals of computer vision can use this section to become familiar with the mathematical notation; otherwise it can be skipped.

In Chapter 4, a system is described that is able to track the 3D pose of a moving stereo-camera pair in a 3D world, while simultaneously building a sparse 3D map of that world. We also review and make choices on various tracking methods that results in our solution. We combine a smart feature selection algorithm with a two stage tracker to create a robust pose estimation system.

In Chapter 5, a system is described that is able to simultaneously build a dense 3D map of that world while the tracking is performed. We also review and make choices on various reconstruction methods that results in our solution. The estimated poses are utilized to create a 3D reconstruction of the crime scene in real-time.

Chapter 6 presents a new HCI methodology for use with a HMD-based AR system with stereo cameras. It exploits the user's hands as an interaction device instead of other auxiliary equipments. For this, we combined different cues such as curvilinearity, depth and color to detect the user's hands and their poses. This system is combined with the AR system so the user can interact with the scene and the system during investigation.

Finally, Chapter 7 concludes this thesis and gives future perspectives of the work presented.

Chapter 2

System Architecture

All animals are equal
but some animals are more equal than others.
George Orwell, Animal Farm

In this chapter, we describe the design specifications and implementation details of our AR system, which enables multiple users to explore a scene and generate and store observations on the scene in close cooperation. Specifically, this section will explain the selection of the necessary hardware, implementation details of the software for immersive AR, and communication between a remote user and on-site users.

Our goal is to design a generic, robust and affordable system that can operate in different conditions with easily replaceable sensing, computing and display devices. This leads to a modular architecture, both in software and hardware, in which each function or sensor resides in its own module. A typical AR system consists of a number of modules, including image sensing and processing hardware (cameras and computer), display hardware (HMD), tracking and pose estimation algorithms for head-pose estimation, registration algorithms for aligning the real and virtual worlds, graphics rendering hardware as well as software for virtual content rendering, communication between users and interaction methods. In the case of AR for CSI, there are two parties involved in the process: the remote experts and the on-site investigators. The remote experts provide assistance to the on-site investigators from a distance while monitoring them. Note that they are usually not mobile, but situated in front of desktop computers or displays. On the other hand, the on-site investigators are highly mobile during the operation. Therefore, the hardware and the software requirements of the on-site users and the remote experts differ from each other. For instance, a wearable setup is not necessary for remote experts. In order to exploit these differences, we separate our design into two parts, the remote system and the wearable system, as illus-

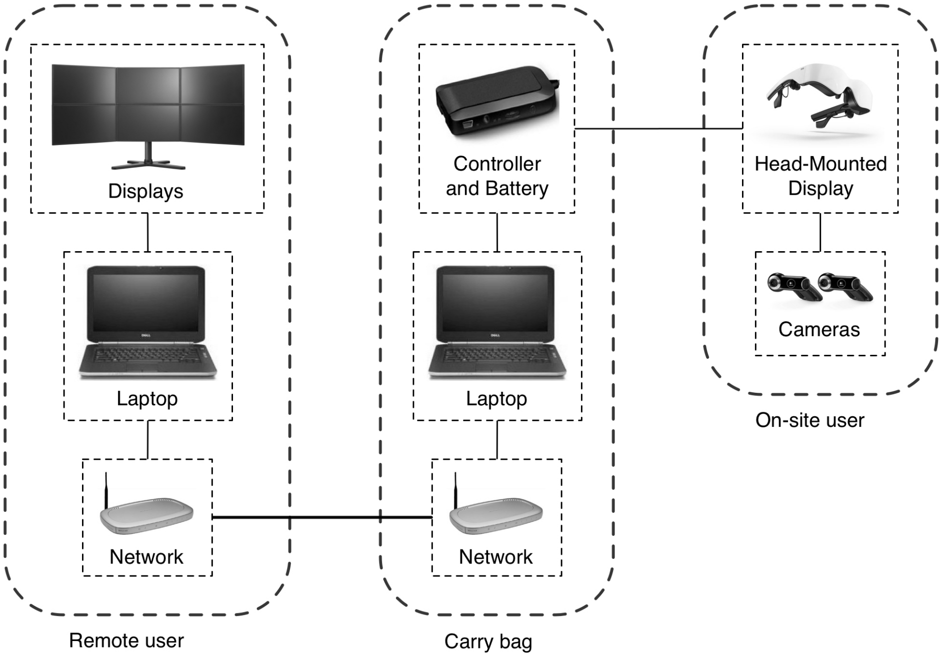


Figure 2.1: *The system diagram with two main parts: fixed setup for the remote user and the wearable part including carry-bag, HMD and cameras for the mobile user.*

trated in Fig.2.1. Also, we consider only the case of one CSI and one remote expert to simplify the design process. The mobile user wears a carry-bag, a HMD and cameras, while the remote user uses a fixed setup. Since the remote users are not mobile and only uses a fixed setup, their hardware requirements can be satisfied with a standard computer equipped with a network module. Therefore, in this thesis we only discuss the software architecture of the remote system in the software section. Both systems communicate with each other over a wireless network and transfer AR content, voice and 3D maps.

As outlined in Chapter 1, the wearable system needs to satisfy the requirements such as light weight and affordable hardware, real-time and robust operation, hands-free interaction and on-site scene reconstruction. In this section we present the hardware and software modules providing a solution satisfying these requirements.

2.1 Hardware

For our mobile wearable system we have selected the following components:

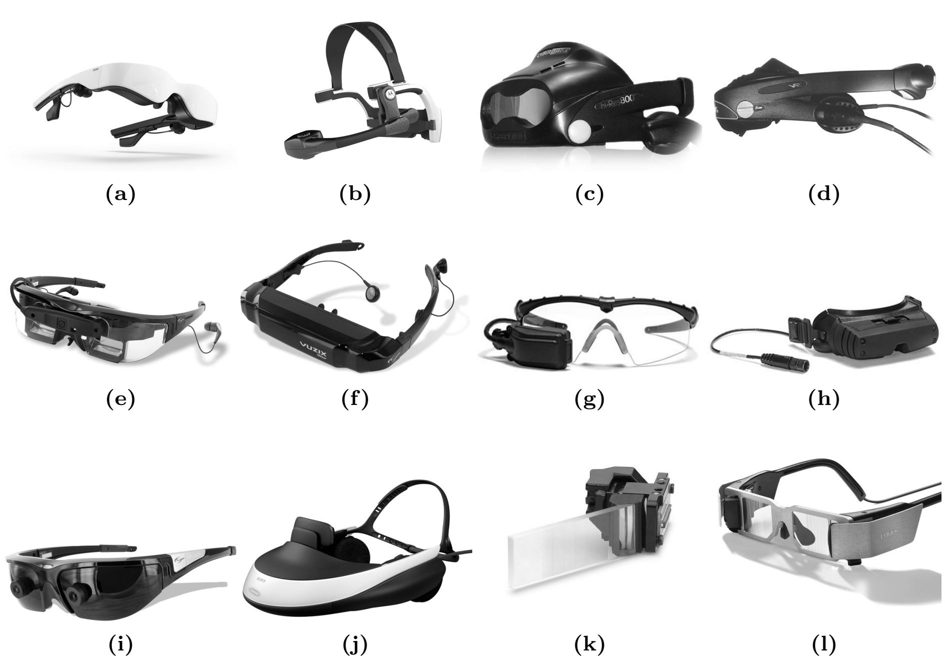

(a) (b) (c) (d)
(e) (f) (g) (h)
(i) (j) (k) (l)

Figure 2.2: *A selection of recent examples of head-mounted displays (HMDs) ranging from video see-through to optical see-through, single to stereo displays. They are used in various applications such as military, education, entertainment etc. (a) is produced by Zeiss [210] while (b) is from Kopin [99]. (c,d) and (e,f,g,h,i) are produced by Cybermind [44] and Vuzix [198] respectively. Sony [183] recently developed (j) while Lumus [117] produced (k,l).*

2.1.1 Head Mounted Display

One of the most important components of a wearable system is the Head-Mounted Display (HMDs, a.k.a. goggles or eye-wear). As discussed in the requirements section, the mobile user needs his/her hands free for executing tasks and therefore hand-**held** displays are not considered in this work. Instead, we used a head mounted display for visualization of augmented images. There are various HMDs available in the market, ranging from simple video displays to monocular or stereo see-through displays (shown in Fig.2.2). For immersive AR experiences, see-through HMDs are preferred and they can be divided into two categories: optical see-through and video see-through displays.

In optical see-through HMDs, semi-transparent mirrors or prisms placed in front of the user's eyes are used to display the virtual objects. Since the real-world is still visible through the mirrors the real and the virtual worlds are optically combined. With a video see-through HMD, the real-world is captured via cameras mounted in front of the device, and video-in-video merged images are presented on the displays.

Several challenges have accompanied the AR with see through HMDs. In video see-through displays, there is an inevitable lag since the captured images through cameras are processed before they are displayed in the headset. Although recent cameras and displays can provide higher frame/refresh rates, this effect may still cause nausea and headaches. In addition to that, the cameras should be mounted on the headset carefully to align the displays and the cameras so that the user will experience the AR like he/she is observing this through his/her own eyes. Unlike video see-through displays, the real world is perceived without any lag with optical see-through displays. However, the real world appears darker and the virtual content is displayed semi-transparent. Also the HMDs are usually bigger and more expensive relative to their video see-through counterparts, mainly due to the optics. Finally, both display systems have a narrower field of view of (∼32 degrees) compared to human eyes (∼110 degrees single eye), but this is the state-of-the-art of currently available augmented reality hardware in the consumer market.

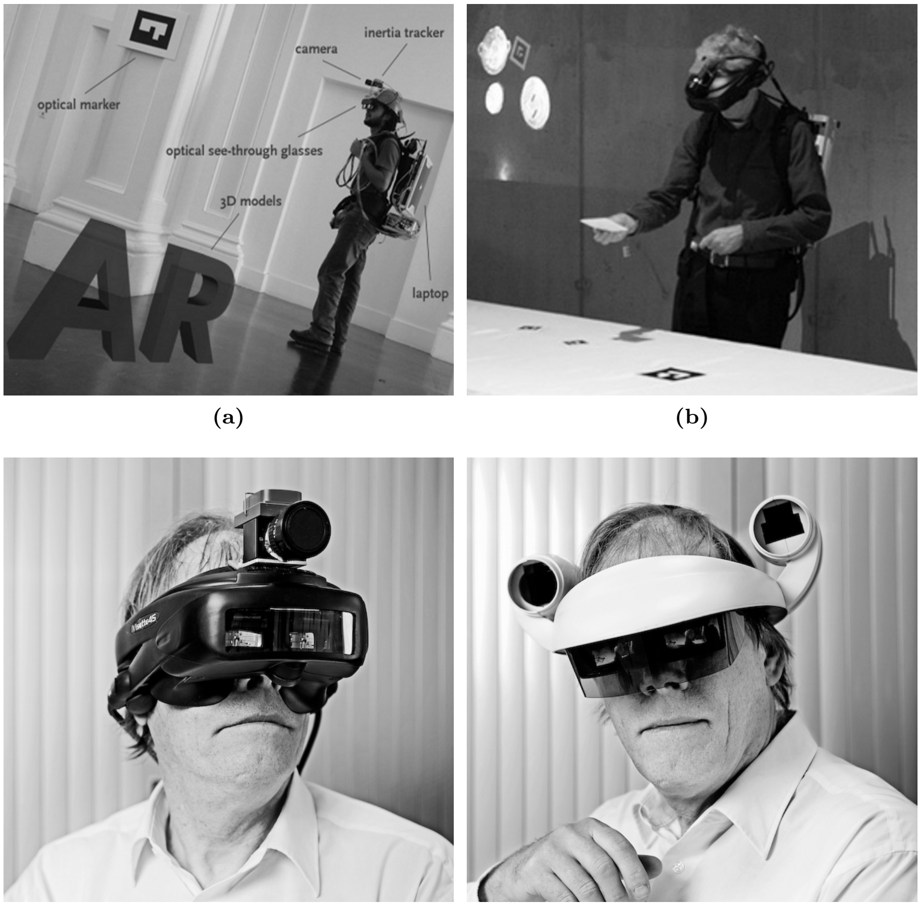
Several AR systems with optical see-through displays have been designed in the Delft University of Technology since 1999, and since 2006 in close collaboration with the Royal Academy of Art in The Hague, as shown in Fig.2.3. In our latest design we have chosen video see-through HMDs since they are more affordable and smaller then their optical see-through counterparts. We believe that for the consumer market, optical see-through HMDs are still not mature enough, and for the CSI application they are too expensive to provide them to every investigator.

In our initial design, we have used iWear VR920 (Vuzix, USA) glasses (Fig.2.2 (f)) with $640x480$ (920,000 pixels) LCD displays, 32-degree field of view and weighting approximately 91 grams. After this initial mock-up, we used Cinemizer OLED (Carl Zeiss AG, Germany) glasses (Fig.2.2 (a)). It has two high-resolution displays that can display 720p images and also has approximately a 32-degree field of view. The weight of the Cinemizer is 115 grams. Although, the Cinemizer glasses are slightly heavier than the Vuzix glasses, we preferred the Cinemizer glasses since it has better displays with higher resolution. Also, a new setup is going to be designed with HMZ-T1 (Sony, Japan) glasses (Fig.2.2 (j)).

The glasses are controlled by a control box delivered with it. The control box is connected to a computer via a VGA, DVId, HDMI, Display Port and a USB port and contains rechargeable batteries. The glasses can be powered directly through a USB connection or through the batteries in the control box.

2.1.2 Cameras

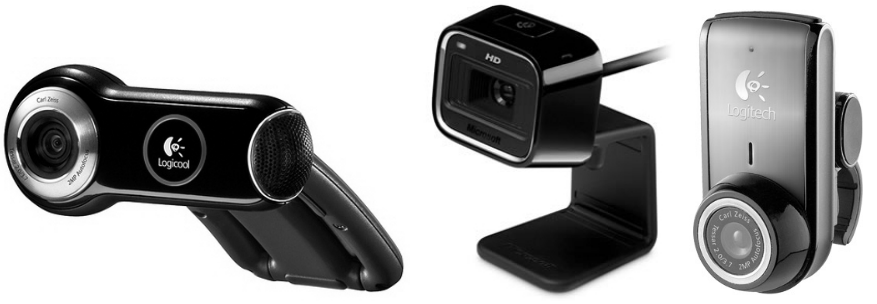
Another critical component of the AR system is its cameras. In order to capture stereo images for the two displays of the HMD and create metric maps we decided to use two cameras and create a stereo rig. The utilization of calibrated cameras also eases the initialization and the depth estimation for scene reconstruction



(a) (b)

(c) (d)

Figure 2.3: *AR systems with optical see-through HMDs of Cybermind used in the Delft University of Technology in chronological order. The system shown in (d) is designed by ARlab student Niels Mulder.*



(a) **(b)** **(c)**

Figure 2.4: *Web-cameras: (a) Logitech Pro-9000 (b) Microsoft LifeCam HD-5000 (c) Logitech C905 cameras.*

steps. The first decision was between the use of USB or Firewire (IEEE-1394) cameras. Two Firewire cameras can operate on the same bus and can supply a fixed frame rate without any change during the operation. Unfortunately, most of the commercial laptops have a single powered Firewire port (if not none) as the one chosen in our design. Also, these cameras consume too much power to run them on batteries. On the other hand, USB cameras can operate with the necessary power from the USB port. However, each camera needs a separate USB bus to transmit images, and in most of the portable computers multiple USB ports are connected to the same bus or two buses. Therefore, the selection of the laptop with sufficient USB buses becomes an important issue when more than one camera is used. Also, the USB cameras have thinner and more flexible cables, are smaller in size in comparison with the Firewire cameras, and more affordable.

Another advantage of the web-cameras is they have well developed drivers for Linux and the latest web-cameras capture high quality, high resolution images with low noise, due to their good lenses (e.g. Zeiss).

In our initial design, two Pro-9000 (Logitech, USA) webcams are mounted above the Vuzix glasses as shown in Fig.2.5 (top). In the final design two LifeCam HD-5000 720p (Microsoft, USA) webcams are stripped and mounted in front of the Cinemizer providing a full stereoscopic 720p resolution pipeline. The attached stereo rig is 65 grams and the cameras can output color images with 720p resolution at 30 frames per second (fps). They are mounted approximately 6.5 cm from each other (shown in Fig.2.5 (middle)) as with human eyes, and are calibrated off-line. Synchronization of the cameras for the acquisition of the stereo images is achieved in software. Currently, *Marty*, a new setup by combining HMZ-T1 (Sony, Japan) glasses with two C905 (Logitech, USA) webcams (shown in Fig.2.5 (bottom)) was designed by Niels Mulder on commission of the ARLab.

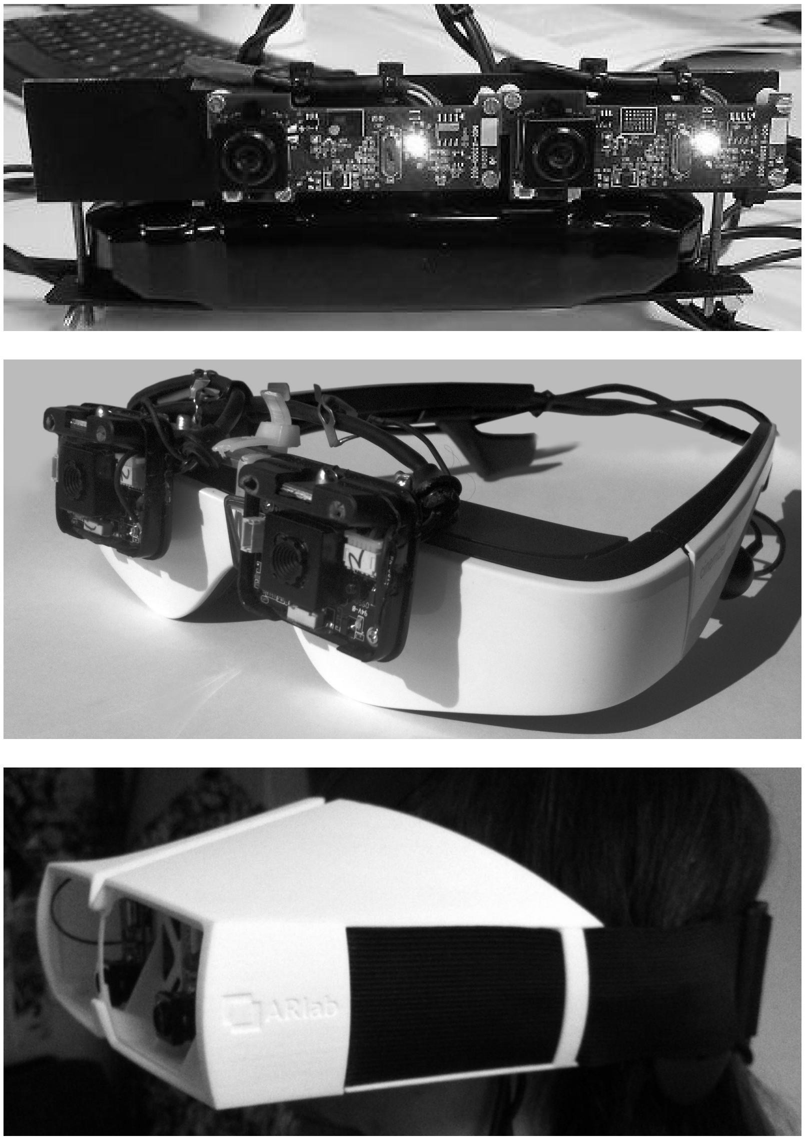


Figure 2.5: *HMDs: top to bottom: initial design with iWear VR920 glasses combined with Pro-9000 cameras, Cinemizer OLED glasses with LifeCam HD-5000 cameras and Marty, a video see-through AR headset based on a Sony HMZ-T1 combined with Logitech C905 cameras, was designed by Niels Mulder.*

2.1.3 Laptop

Head pose estimation, graphics rendering, scene reconstruction, human-computer interaction and networking are computationally highly intensive and require a high-end CPU-GPU combination. In order to achieve high frame rates and execute all tasks in parallel, we have designed multi-threaded software, utilizing multiple cores. This is summarized in the following section and explained in detail in Chapters 4 and 5. We preferred CUDA-enabled GPUs manufactured by Nvidia because of their superior graphics performance and support for Linux drivers.

We opted for an available laptop solution with high-end CPU and GPU, since the system is going to be carried by the user during operation. At the time of selection (in 2010) the Dell Latitude E6520 was one of the best choices:

- 2.7 GHz QuadCore i7 processor (Intel Corporation, USA).
- NVIDIA NVSTM 4200M (DDR3 512MB) graphics card.
- Wireless LAN and network connector (RJ-45) for communication.
- 4 GB RAM,
- VGA+DVI connectors.
- 4 USB 2.0 ports.
- 2.5kg weight.

Figure 2.6: *Dell Latitude E6520*

Although this laptop is big and the 15.4 inch display is redundant during field operation, we stick to this laptop (with the bigger display) instead of a smaller one to be able to debug and run the AR software on the same platform. All the algorithms are developed in C++ under the Linux operating system (Ubuntu 10.10).

2.1.4 Backpack

To fit all equipment we used a carry-bag with metal frame from a backpack baby carrier designed in [26]. A metal cabinet is attached at the bottom of the frame to hold the laptop and the laptop is fixed with Velcro strips.

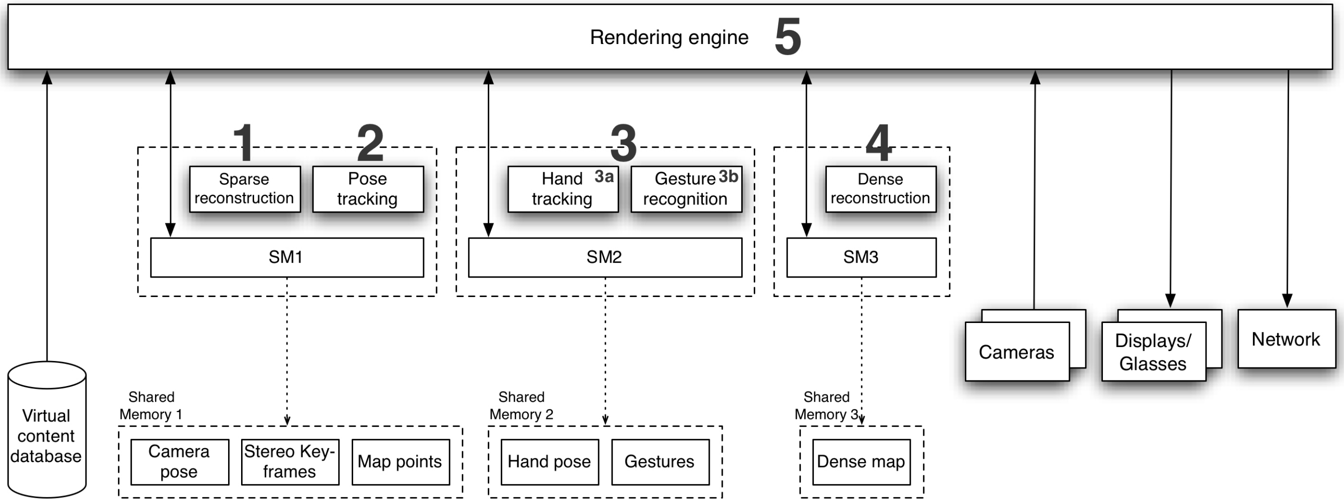


Figure 2.7: *System architecture of the mobile setup. Each module labeled with numbers has its own thread and runs in parallel with the other threads. The rendering engine (5) establishes the connection between the cameras, displays and other modules. Pose tracking (2), sparse reconstruction (1) and dense reconstruction (4) modules handle the head pose estimation and scene reconstruction. The Human-Computer Interaction interface (3) is composed of two modules (run together in a single thread), 3a and 3b, and allows users to select and use tools, and supports some basic widgets, like menus slider bars, text labels and icons.*

2.2 Software

The basic software architecture with its modules is depicted in Fig.2.7. The mobile AR system consists of 5 main components: *pose tracking*, *sparse reconstruction*, *dense reconstruction*, *HCI* (hand tracking and gesture recognition), and *rendering engine* modules.

Each module is designed to satisfy a requirement given in the previous chapter, and has its own thread and runs in parallel with the other threads. The multi-threaded design minimizes the dependency between the modules and avoids global failures due to a failure in one of the modules. For instance, the user can still use the HCI and restart the system when the tracking is lost or continue tracking when the HCI is not working due to undetected hands. However, the modules are not completely independent, i.e. the dense reconstruction will fail if the tracking module fails. Modules communicate through a shared memory structure as shown in Fig.2.7. The shared memory improves the modularity and isolates the modules from each other.

In the following sections, we give an overview of the functions of each module and elaborate on the communication between the modules. A more detailed information about each module is given in the following chapters.

2.2.1 Pose Tracking and Sparse Reconstruction

The pose tracking and the sparse reconstruction modules together handle the head pose estimation. The pose tracking module calculates the camera pose for each frame by using the map points (3D natural features) created and maintained by the sparse reconstruction module. When the user moves around, the sparse reconstruction module expands the map by adding new map points using the camera pose and stereo key-frames provided by the pose tracking module. Both modules share the camera pose, stereo key-frames and map points through a shared memory. The detailed explanation about the working principals of algorithms and the structure of the shared information are given in Chapter 4. From these two modules, the camera pose is sent to the rendering engine in order to render virtual content and reflect the changes in the user's viewpoint.

2.2.2 Dense Reconstruction

The dense reconstruction module creates the dense 3D representation of the scene. It receives the key-frames and camera poses created by the pose tracking module and creates dense point clouds. As the user observes unseen parts of the scene, the existing dense map is extended by adding new map points. If the user is operating in the previously mapped part of the scene, then the existing map is updated with the new observations. The created map is used to support the standard ways of evidence capturing such as photographs and drawings, and to establish a common ground between the remote and the on-site user. A more detailed information about this module is given in Chapter 5. The aim of the dense reconstruction module is not to generate highly accurate maps of the crime scene but to provide contextual information which can help the remote experts to navigate around the scene and the on-site users to place virtual tags.

2.2.3 Human-Computer Interaction

The Human-Computer Interaction interface is composed of two modules: hand tracking and gesture recognition. The hand tracking module exploits the depth, curvilinearity and color information to detect the user's hands and tracks them between the consecutive stereo frames. In each frame the 3D position and the surface normal of the hands are extracted. The details of the hand tracking module are given in Chapter 5. The gesture recognition module is developed by Poelman et al. [155] within the CSI project to demonstrate the capabilities of the hand tracking module and to provide simple tools for investigators. The details of this module is explained below for the sake of completeness although it is not one of the main points of this thesis.

The user interface (gesture recognition) allows users to select and use tools, and supports some basic widgets, like menus slider bars, text labels and icons. It translates the 3D information from the hand tracker into click and drag events which triggers the widgets and controls the CSI tools. The events are quite similar to mouse events from WIMP interfaces, except that they also contain 3D information. On the one hand, CSIs need to move their hands freely without triggering unintended events. On the other hand, gestures have to be intuitive and easy to make. The system distinguishes three types of gestures: Left hand thumb-up, left hand thumb-down, and right hand thumb-down. The orientation of the hand, and the 3D position are used to both draw the interface elements and the gesture recognition. This module is basically designed to demonstrate the capabilities of the hand tracker module and therefore based on simple gestures. A menu surrounding the hand appears when the left hand thumb-up is detected. The menu sticks to the hand and is locked in space until the posture changes and the thumb points downwards options can be selected. The right hand, as a pointing device, is used to select objects in the virtual scene. Effectively, recasting is used to determine with which scene point to interact. Fig.2.8 shows the gestures distinguished with the defining hand postures. A click is done by moving the left or right recognized segmented hand forward quickly, and moving it backward again. The direction of movements of the segmented hand is continuously monitored to recognize this gesture. When the pointer moves only in a forward direction, the path over which it is moving is tracked. As soon as it has moved forward, and backward more than halfway along the same path, this is registered as a click at the furthest point of the path. If anywhere in this sequence the segmented hand deviates more than a pre-defined angle from the path, the event is not recognized as a click. In this way both small and big gestures are recognized, as long as the direction of the movement is right.

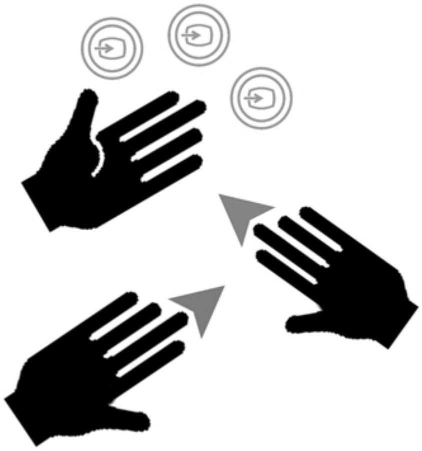


Figure 2.8: *The gestures distinguished with the defining hand postures.*

The user interface and the virtual scene are general-purpose parts of the mediated reality system. They can be used for CSI, but also for any other mediated reality application. The Tool Set, however, needs to be tailored for the application domain. The current mediated reality system supports following tasks for CSIs (identified as requirements scene mapping in the previous chapter): recording the scene, placing tags, loading 3D models, bullet trajectories and placing restricted area ribbons. Fig.6.1 shows the corresponding menu attached to a user's hand. The above tools are based on generic functions to store real-world data (video streams, measurements, photos), import data from other sources (photos, databases, the internet) and change the virtual scene (placing tags). By offering the above tools, we enable the communication between a novice on a crime scene and the guiding from the remote expert. Thereby, we further address our requirement on the remote collaboration with experts.

2.2.4 Rendering Engine

The rendering engine establishes the connection between the cameras, displays and other modules. Captured images are transferred to each module via the rendering engine and also displayed on the screens. The virtual scene, the dense map of the scene as well as the virtual artifacts that CSIs have placed, are rendered in overlay with the real crime scene using the camera pose information coming from the pose estimation module and the dense map created by the dense reconstruction module. Moreover, the Graphical User Interface (GUI) and the CSI tools are drawn on the screen using the detected gestures by the HCI module. For this work we used the OGRE [91] render engine.

Finally, the OGRE render engine organizes the connectivity between users at different locations. Thereby, it supports our requirement on a remote connection to and collaboration with experts. All image streams, camera pose estimations, dense maps and interactions with the scene are uploaded to the server by using the server-client architecture. With the same architecture the HMD wearer and remote clients receive their information from the server. The identity of the users and their access rights, known to the system, determine the user's access privileges. Further optimization of the engine, to meet the requirements on collaboration, include a 3D graphical user interface (opposed to the default 2D interface), a stereoscopic pipeline to render virtual content on top of stereo image streams, and network options that enable remote spatially oriented collaboration. Both the remote expert and HMD wearer can add and remove virtual objects [155] from the scene. Their interface, however, is different: the mediated reality system wearer has a stereoscopic first person view and the remote expert has a free 3D navigable view on the scene. Interaction between two users is synchronous and turn-based: one user must complete his/her actions before the other can perform new actions.

2.2.5 Remote System

The remote expert needs to monitor the on-site user and his/her view, and has the possibility to also place virtual objects. Therefore the software architecture used in the remote system is a simplified and stripped version of the mobile system. The pose estimation, sparse reconstruction, dense reconstruction and HCI modules are omitted. Only the simplified rendering engine is kept in the remote system. Moreover, the mode of interaction with the engine differs: the HMD wearer uses his/her hand to navigate the graphical user interface and the remote expert uses a mouse and a keyboard.

As explained in the previous section, the rendering engine establishes the communication in the remote system. The transferred information such as camera images, 3D maps and virtual content are rendered to the displays of the remote expert. In addition to displaying the information, the system also provides the remote experts with tools to augment the map. The virtual content created by the remote expert is placed on the 3D maps and shared with the on-site wearable system, so that both sides can work on the same map.

In our system, the remote expert can observe the scene relative to the on-site user from two different viewpoints: *egocentric* and *exocentric*. In the egocentric mode, the remote expert sees the scene through the cameras mounted on the HMD of an on-site user and therefore moves with that person in the scene. In teleoperation applications such as performing an analysis on the scene together with the on-site user, an egocentric viewpoint offers a better spatial representation and improves the performance [71]. However, a better overview of the scene can be comprehended if the remote expert can see the scene with a bird-eye view. In order to provide such a view, we use the 3D dense reconstruction of the scene created during the motion of the on-site user. The remote expert can decouple him/her self from the on-site user and use an exocentric (virtual) viewpoint. Moreover, he/she can travel in the virtual map and observe the scene from different viewpoints for better spatial analysis. Also, the system can benefit from merging 3D maps of multiple CSIs to make one global map which is a design requirement when multiple on-site investigators exist in the same scene. However, we didn't address this in our design and left it as a future work.

Although the remote system is a part of our system and in our design considerations, the evaluation of it is not in the scope of this thesis and will be addressed by TPM-SES in the CSCW domain.

Chapter 3

Mathematical Framework

Tell me and I'll forget;
Show me and I'll remember;
Involve me and I'll understand.
Chinese Proverb

In this chapter the mathematical background of this thesis is summarized. The main purpose of this section is to introduce the mathematical notation, and hence for readers who are familiar with the principals of computer vision it can be skipped. For readers who are less familiar with the subject, this section gives a brief introduction to the concepts and the algorithms that are used for visual odometry and 3D reconstruction in this thesis. More detailed information can be found in [118, 79, 191, 131].

3.1 Euclidean Transformation, Image Formation and Camera Models

In order to study 3D vision and reconstruction, it is important to understand the Euclidean transformation (rigid body motion) and perspective projection. The 3D motion of a moving camera (or an object) can be modeled as an Euclidean transformation while the image formation process can be described by a perspective projection.

3.1.1 Euclidean Transformation

In order to represent a camera motion (rigid body motion) in a 3D space, a map or a transformation should preserve distances between points and their orientations.

These transformations are called *special Euclidean transformations* (denoted by $SE(3)$) and can be defined as [118]:

A map $g : \mathbb{R} \to \mathbb{R}$ is a special Euclidean transformation (rigid-body motion) if the norm (and therefore the inner product) and the cross product of any two vectors are preserved.

The rigid body motion or the special Euclidean transformation between two coordinate frames O_1 and O_2 (frame O_2 relative to O_1), $g_{O_1 O_2}$, has two components: a translational part T which is the vector between the origins of the two coordinate frames and a rotational part R which is the orientation of O_2 relative to O_1. The rotational part can be represented by the $3x3$ matrix

$$R = [r_1, r_2, r_3] \in \mathbb{R}^{3x3} \tag{3.1}$$

The rotation matrix is a special orthogonal matrix in $\mathbb{R}^{3x3}$ and the space of all such matrices (a special orthogonal group or rotation group) can be denoted by [118]

$$SO(3) = \left\{ R \in \mathbb{R}^{3x3} \mid R^T R = I,\ det(R) = +1 \right\} \tag{3.2}$$

Exponential coordinates are preferred to parameterize the rotational transformations instead of other representations such as the matrix representation, quaternions or Euler angles since it is minimal, more intuitive and easy to understand. A rotation around the axis $w = [w_1, w_2, w_3]^T \in \mathbb{R}^3$ by an angle of r radians can be denoted by $R = e^{[w]_x r}$ which can also be written by absorbing r into w by setting $\|w\| = r$ so that

$$R = e^{[w]_x} = exp([w]_x) \tag{3.3}$$

where $[w]_x \in \mathbb{R}^{3x3}$ is a skew-symmetric matrix $\begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix}$ and exp is the *exponential map* that defines the mapping from the space of all skew-symmetric $3x3$ matrices $so(3)$ to $SO3$

$$exp : so(3) \to SO(3);\ [w]_x \to e^{[w]_x} \tag{3.4}$$

R can be calculated from a given w by using *Rodrigues' formula*

$$e^{[w]_x} = I + \frac{[w]_x}{\|w\|} sin(\|w\|) + \frac{[w]_x^2}{\|w\|^2} (1 - cos(\|w\|)) \tag{3.5}$$

and for a given $R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ the corresponding w is

$$\|w\| = cos^{-1} \left(\frac{trace(R) - 1}{2} \right), \frac{w}{\|w\|} = \frac{1}{2sin(\|w\|)} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$$

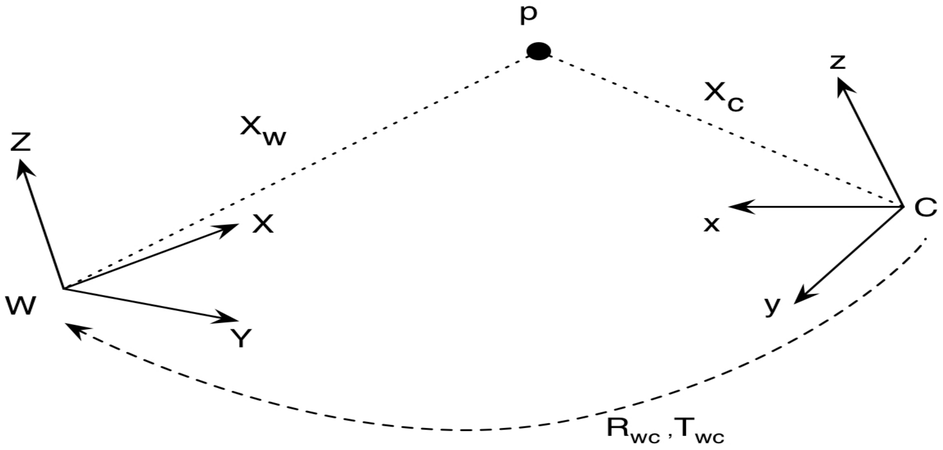


Figure 3.1: *Special Euclidean transformation, rotation R_{wc} and translation T_{wc}, between two coordinate frames W and C.*

The general special Euclidean transformation (both rotation and translation) between two coordinate frames W and C is shown in Fig.3.1. The coordinates of a 3D point p in the reference frame W, X_w, and in the reference frame C, X_C, are related with the $SE(3)$ transformation

$$X_w = R_{wc}X_c + T_{wc} \tag{3.6}$$

where $R_{wc} \in SO(3)$ and can be represented as $g_{wc} = (R_{wc}, T_{wc})$ or simply $g = (R, T)$. The subscript WC can be read as 'from frame C to frame W'. The set of all special Euclidean transformations is defined as [118]

$$SE(3) = \{g = (R, T) \mid R \in SO(3),\, T \in \mathbb{R}^3\} \tag{3.7}$$

The coordinate transformation for $SE(3)$ is not linear ($u = Av$) but affine ($u = Av + b$). However it can be converted to linear and the matrix representation for $SE(3)$ can be obtained by using homogeneous coordinates [79]. Homogeneous coordinates of a 3D point $X = [X, Y, Z]^T$ are denoted as:

$$\tilde{X} = \begin{bmatrix} X \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \in \mathbb{R}^4 \tag{3.8}$$

then the affine equation (3.6) can be rewritten in a linear form with homogeneous coordinates

$$\tilde{X}_w = \begin{bmatrix} R_{wc} & T_{wc} \\ 0 & 1 \end{bmatrix} \tilde{X}_C \tag{3.9}$$

The homogeneous representation of $g \in SE(3)$ can be written as

$$SE(3) = \left\{ \tilde{g} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \mid R \in SO(3),\ T \in \mathbb{R}^3 \right\} \tag{3.10}$$

and the motion of a continuously moving rigid body (at time t) can be shown as

$$g(t) = \begin{bmatrix} R(t) & T(t) \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4x4} \tag{3.11}$$

Similar to the rotational motion $SO(3)$, the special Euclidean transformation can also be represented in exponential coordinates such that [118]

$$g(t) = e^{\hat{\xi} t} \tag{3.12}$$

where $\hat{\xi} = \begin{bmatrix} [w]_x & v \\ 0 & 0 \end{bmatrix} \mid [w]_x \in so(3),\ v \in \mathbb{R}^3$.

A $4x4$ matrix $\hat{\xi} \in \mathbb{R}^{4x4}$ is called a twist (exponential coordinates for $SE(3)$) and the set of all twists is denoted by $se(3)$. The twist coordinates ξ of the twist $\hat{\xi}$ are defined as $\xi = \begin{bmatrix} v \\ w \end{bmatrix} \in \mathbb{R}^6$, where v is the linear velocity and w is the angular velocity. The relation between the exponential representation and the matrix representation is

$$e^{\hat{\xi}} = \begin{bmatrix} e^{[w]_x} & \frac{(I - e^{[w]_x}) e^{[w]_x} v + w w^T v}{\|w\|} \\ 0 & 1 \end{bmatrix} \quad \text{if } w \neq 0 \tag{3.13}$$

The $\hat{\xi}$ can be written by using the group generator matrices G_i and the twist coordinates $\xi = [\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6]^T (= [v_1, v_2, v_3, w_1, w_2, w_3])$ as

$$\hat{\xi} = \sum_{i=1}^{6} \xi_i G_i \tag{3.14}$$

where the generator matrices are

$$G_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},\ G_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},\ G_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$
$$G_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},\ G_5 = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},\ G_6 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.15}$$

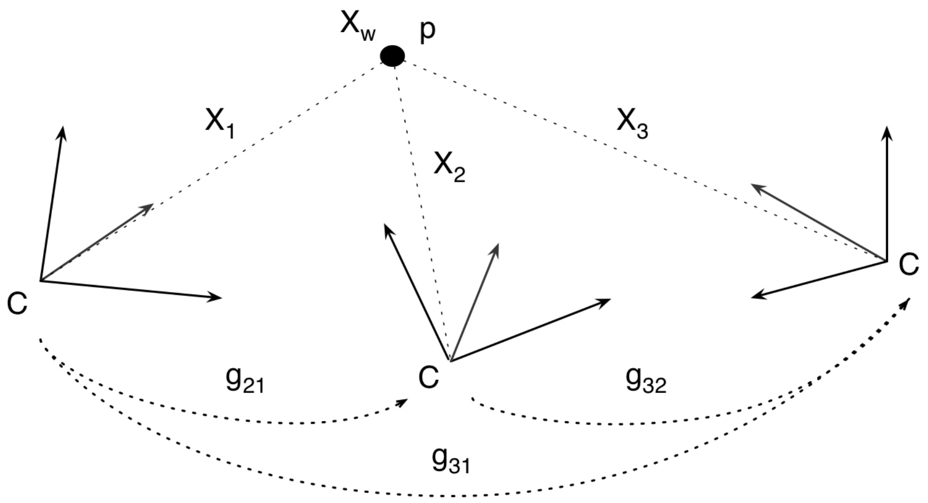


Figure 3.2: *The relation between the position of a 3D point p in a moving camera frame.*

A small (infinitesimal) motion of the camera (η) can be represented by a $M \in SE(3)$

$$M = exp(\hat{\eta}) = exp\left(\sum_{i=1}^{6} \eta_i G_i\right) \tag{3.16}$$

For small motions M can be approximated as

$$M = \begin{bmatrix} 0 & -w_3 & w_2 & v_1 \\ w_3 & 0 & -w_1 & v_2 \\ -w_2 & w_1 & 0 & v_3 \end{bmatrix} \tag{3.17}$$

and the partial derivatives of the motion matrix with respect to the motion parameters for small camera motion ($\eta = 0$) are

$$\frac{\partial M}{\partial \eta_i} = G_i \tag{3.18}$$

These simple derivations will be used in the following chapters to calculate the projection Jacobians for pose estimation and bundle adjustment.

Consecutive $SE3$s

The relation between the position of a 3D point p in a moving camera frame is shown in Fig.3.2. If the 3D position of p relative to the world frame is X_w, then its coordinate relative to the camera at time $t \in \mathbb{R}$ is

$$X(t) = R(t)X_w + T(t) = g(t)\tilde{X}_w \tag{3.19}$$

The relative motion between the time instants t_i and t_j, $g(t_i, t_j) \in SE(3)$ can be denoted as g_{ij} and it relates the two coordinates

$$X_i = g_{ij}\tilde{X}_j = R_{ij}X_j + T_{ij} \tag{3.20}$$

The position of p relative to the camera at time instant t_3 is

$$\begin{aligned} X_3 &= g_{32}X_2 = g_{32}g_{21}X_1 \\ &= g_{31}X_1 \end{aligned} \tag{3.21}$$

Therefore the composition rule $g_{ik} = g_{ij}g_{jk}$ can be used to find the position of the camera by applying the consecutive motions between the time instants. Moreover, the rule of inverse can be used to reverse the motion

$$g_{21}^{-1} = g_{12} \tag{3.22}$$

3.1.2 Image Formation and camera model

The simplest approximation of the thin lens camera is a *pinhole camera model.* In the (frontal) pinhole camera model a 3D point $\mathbf{X} = [X, Y, Z]^T$ and its image $\mathbf{x} = [u, v]^T$ on the image plane are related by *perspective projection* [131]. From the similar triangles shown in Fig.3.3

$$u = f\frac{X}{Z},\ v = f\frac{Y}{Z} \tag{3.23}$$

where f, the *focal length*, is the distance between the *camera center* C and and the *principal point* p. The Z axis is called the *principal axis* and its intersection with the image plane is the principal point. The image plane can also be defined as the $z = f$ plane.

In an ideal perspective camera the projection of a point $\mathbf{X} = [X, Y, Z]^T$ relative to the camera coordinate frame onto the image plane can be written as

$$\mathbf{x} = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{f}{Z}\begin{bmatrix} X \\ Y \end{bmatrix} \tag{3.24}$$

which can be denoted in homogeneous coordinates as

$$Z\tilde{\mathbf{x}} = Z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\tilde{X} \tag{3.25}$$

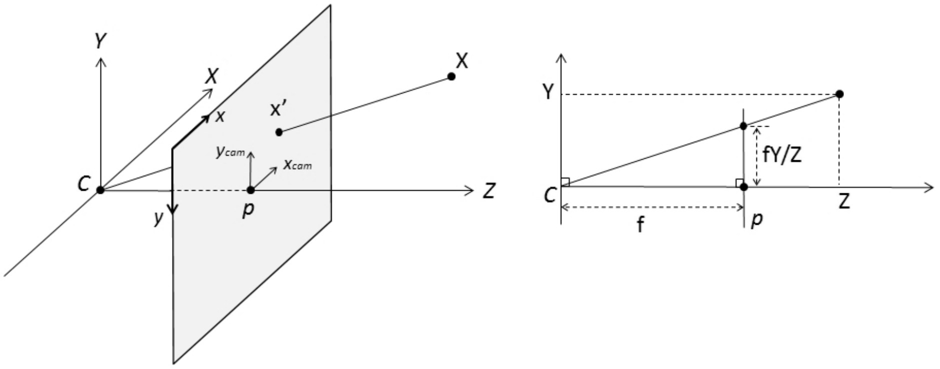


Figure 3.3: *Image formation.*

The equation above can be decomposed into two matrices

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} [I|0] \tag{3.26}$$

The coordinates of the perspective projection $\mathbf{x}$ of the 3D point $\mathbf{X}$ are given with respect to the principal point p. So the coordinates in the image coordinate frame (in which the origin is the top left corner of the image) can be written as

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} fX/Z + p_x \\ fY/Z + p_y \end{bmatrix} \tag{3.27}$$

where (p_x, p_y) are the coordinates of the principal point p in the image coordinate frame. Therefore the Eq.(3.26) becomes

$$Z\tilde{x} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} [I|0]\tilde{X} \tag{3.28}$$

Until now the image coordinates are measured in the units of the camera centered reference frame. The conversion between metric coordinates and pixel coordinates is obtained by multiplying the x and y coordinates with m_x and m_y which are the number of pixels per unit distance in image coordinates in the x and y directions. So

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} m_x u' \\ m_y v' \end{bmatrix} = \begin{bmatrix} m_x(fX/Z + p_x) \\ m_y(fY/Z + p_y) \end{bmatrix} \tag{3.29}$$

$$\lambda\tilde{x} = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} [I|0]\tilde{X} \tag{3.30}$$

where $f_x = fm_x$, $f_y = fm_y$, $x_0 = m_x p_x$ and $y_0 = m_y p_y$, the scalar s is the skew parameter (which is zero for most standard cameras) and λ is the (projective) depth of the point $\tilde{X}$. This matrix is called the *intrinsic calibration matrix* and is represented as an upper triangular matrix K

$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.31}$$

In general, points in space are expressed in the world coordinate frame. Therefore, the coordinates of the point need to be converted into the camera coordinate frame. If the special Euclidean transformation from world to the camera coordinate frame is $g_{cw} = g(R, T)$ then the coordinates of the point X_w relative to the camera coordinate frame is $\tilde{X}_c = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \tilde{X_w}$, where R and T are called the *extrinsic parameters.*

The final projection from the world coordinate frame to the image plane is given as

$$\lambda\tilde{x} = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} [I|0] \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \tilde{X} = K\pi g\tilde{X} \tag{3.32}$$

The $3x4$ matrix $P = K\pi g$ is called the *projection matrix* of the camera and defines the behavior of the camera. So the effect of the camera is characterized by two stages: the 3D world coordinates to the $z = 1$ plane (the normalized coordinates) and the normalized coordinates to the image coordinates. Also, it is possible to calculate intrinsic and extrinsic parameters back from P by decomposing it via QR-decomposition [79].

3.1.3 Lens Distortion

The perspective projection only models the linear part of the image formation process, since the 3D point, image point and the camera center are collinear. However the real cameras suffer from non-linear distortions. The most important of these distortions are called *radial* and *tangential* distortions. Radial distortion is an alteration in magnification from a center to any point in the image, measured in a radial direction from the center of distortion. This distortion effect takes place in the lens and therefore the distortion should be placed between the intrinsic and extrinsic parameters.

The ideal (distortion-free) normalized image coordinates ($z = 1$ plane coordinates) (x', y') of $\mathbf{x} = R[X, Y, Z, 1]^T + T = [x, y, z]^T$ are calculated as

$$x' = x/z, \; y' = y/z \tag{3.33}$$

Then the distorted (measured) normalized image coordinates [24, 203, 213] are

$$\mathbf{x_d} = \begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{bmatrix} \tag{3.34}$$

where $r^2 = x'^2 + y'^2$ (assuming that the center of radial distortion is the same as the principal point), and k_1, k_2, k_3 are the radial distortion coefficients (the first three lower order terms of the model).

Moreover there exists a tangential distortion [206, 80] which is resulting from the lens not being exactly parallel to the imaging plane. The tangential distortion is minimally characterized by two additional parameters, $p1$ and $p2$, such that the final correction including the radial distortion is

$$\mathbf{x_d} = \begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 y' + p_2(r^2 + 2x'^2) \\ y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' \end{bmatrix} \tag{3.35}$$

Then the final projection onto the image plane is

$$\lambda \tilde{x} = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tilde{\mathbf{x_d}} \tag{3.36}$$

The inverse of the polynomial distortion function in Equation.3.35 is difficult to calculate, but it can be obtained by an iterative way as explained in [213].

3.1.4 Camera Calibration

The process of determining the intrinsic and the extrinsic parameters of a camera is known as *camera calibration.* Extrinsic calibration is also known as *pose estimation.* Calibration objects are used to establish easy and accurate correspondences between 3D scene points and their 2D image projections.

The existing camera calibration methods can be classified in three main categories, direct nonlinear minimization (DNM), closed-form solution (CFS) and hybrid (two step) methods [206]. In direct nonlinear minimization, the camera parameters are searched by using an iterative algorithm with the objective of minimizing a cost function (residual errors). A closed-form solution directly computes the camera parameters with a non-iterative algorithm based on a closed form solution. Both methods have their own drawbacks; DNM needs good initial estimates of parameters to converge to a global minimum, while CFS is sensitive to noise and cannot incorporate non-linear distortions. The hybrid methods [213, 194, 80] initially estimate the intrinsic parameters assuming that the non-linear distortion parameters are zero by using CFS and afterwards utilizes these estimated parameters as initial estimates for DNM.

The camera calibration method proposed in [213] only requires the camera to capture at least 3 images (if both extrinsic and intrinsic parameters are to be estimated) of a planar pattern from different orientations instead of a complex

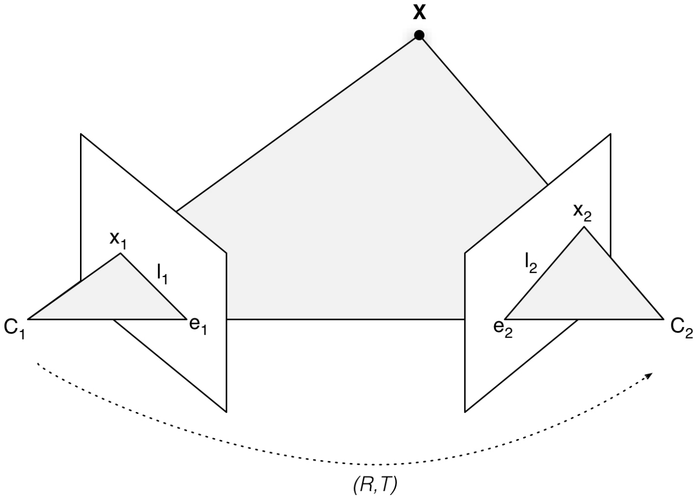


Figure 3.4: *Epipolar geometry.*

3D calibration object with known dimensions. Camera motion or pattern motion need not to be known. Because of this flexibility, and also the superior real-data performance compared to some other methods [190], we have utilized this calibration method in this thesis. Initially the five intrinsic and all extrinsic parameters are estimated by using the closed-form solution created by using the homographic relations between the calibration pattern and its projection on the image plane. Afterwards, the nonlinear distortion parameters are estimated by solving the linear least-squares. Finally, all estimated parameters are refined by using non-linear minimization (as explained in Section 3.3.2).

In his seminal work [194], Tsai proposed that only radial distortion and only one term needs to be considered for industrial applications. According to Sun and Cooperstock [190], including the two de-centering distortion components generally guarantees a high calibration accuracy for a camera with unknown lens distortions. Therefore, we only considered k_1 and k_2 in our calibration.

3.2 Geometry of Two Views: Epipolar Geometry

A 3D point $\mathbf{X}$ and two camera optical centers C_1 and C_2 form a triangle as shown in Fig.3.4. This geometry is usually exploited to simplify the search for corresponding points in stereo matching with given rotation and translation matrices, R and T. In the case that R and T are not known, it can be used to solve for the camera poses given enough (correspondence) points.

Given two images of the 3D point **X** from two different vantage points with the image coordinates x_1 and x_2: if the calibration matrix K is known, then the normalized coordinates of two image points are

$$\dot{x}_1 = K^{-1}x_1 \text{ and } \dot{x}_2 = K^{-1}x_2. \tag{3.37}$$

The normalized coordinates can be related with 3D coordinates as

$$\lambda_1\dot{x}_1 = X_1 \text{ and } \lambda_2\dot{x}_2 = X_2 \tag{3.38}$$

where X_1 and X_2 are the 3D coordinates of the point **X** relative to the two camera frames. Two camera frames are related with the Euclidean rigid body motion $X_2 = RX_1 + T$. This relation can be written as

$$\lambda_2\dot{x}_2 = R\lambda_1\dot{x}_1 + T \tag{3.39}$$

By multiplying both sides with $[T]_x$ we obtain

$$\lambda_2[T]_x\dot{x}_2 = [T]_xR\lambda_1\dot{x}_1 \tag{3.40}$$

Premultiplying this equation with $\dot{x}_2^T$ leads to $\dot{x}_2^T\lambda_2[T]_x\dot{x}_2 = \dot{x}_2^T[T]_xR\lambda_1\dot{x}_1$. Since $\dot{x}_2$ and $[T]_x\dot{x}_2$ are perpendicular

$$\dot{x}_2^T[T]_xR\dot{x}_1 = 0 \tag{3.41}$$

The matrix

$$E = [T]_xR \tag{3.42}$$

is called the *essential matrix* and encapsulates the relative pose between the cameras, and the constraint

$$\dot{x}_2^TE\dot{x}_1 = 0 \tag{3.43}$$

is called the *epipolar constraint.*

The plane determined by the camera centers and the point **X** is called the *epipolar plane.* The projection of camera centers to the other camera's image plane are called the *epipoles.* The intersection of the epipolar plane with the image planes are called *epipolar lines.* In each image, both the image points and the epipoles lie on the epipolar lines such that

$$\ell_i^Te_i = 0,\ \ell_i^Tx_i = 0,\ i = 1,2 \tag{3.44}$$

The projection of all 3D points on the ray passing from C_1 and **X** onto the second image frame lie on the epipolar line l_2. Therefore, in the case that only x_1 is known, the possible image location of the corresponding point x_2 is constrained to the epipolar line l_2. The benefit is the search for the point corresponding to x_1 in calibrated stereo setup, *epipolar search*, need not cover the entire image, but can be restricted to the epipolar line l_2.

3.2.1 Triangulation

Triangulation is the problem of finding the position of a point in space given its position in two images taken with cameras with known calibration and pose [78]. We utilize calibrated cameras in our system and therefore we can use the normal coordinates of the image points, $\mathbf{x}' = K^{-1}\mathbf{x}$. Assuming that the first camera frame as the reference then

$$\lambda_1 x'_1 = [I|0]X \text{ and } \lambda_2 x'_2 = [R|T]X \tag{3.45}$$

If we cross product the right side of the equations with the left sides to eliminate the λ then

$$[x'_1]_x[I|0]X = 0 \text{ and } [x'_2]_x[R|T]X = 0 \tag{3.46}$$

Four linear equations are obtained from two views in the coordinates of x, which may be written as

$$AX = \begin{bmatrix} -1 & 0 & x_1 & 0 \\ 0 & -1 & y_1 & 0 \\ x_2r_{31} - r_{11} & x_2r_{32} - r_{12} & x_2r_{33} - r_{13} & x_2t_3 - t_1 \\ y_2r_{31} - r_{21} & y_2r_{32} - r_{22} & y_2r_{33} - r_{23} & y_2t_3 - t_2 \end{bmatrix} X = 0 \tag{3.47}$$

The solution x can be found by using Singular Value Decomposition (SVD) (subject to the condition $\|x\| = 1$) and this method is called *linear triangulation.*

Moreover, since we use rectified images, the rotation between two cameras is $R = I$ and the translation is $T = [b, 0, 0]^T$ where b is the baseline, and the y coordinates of the same 3D point in two images should be the same $y = y_1 = y_2$. Therefore, Eq 3.47 becomes

$$\begin{bmatrix} -1 & 0 & x_1 & 0 \\ 0 & -1 & y & 0 \\ -1 & 0 & x_2 & -b \end{bmatrix} X = 0 \tag{3.48}$$

which makes the calculation of X trivial.

3.3 Least-squares minimization methods

The least-squares minimization methods are used in many parts of computer vision. Thus we give a short description of the concepts. For instance, the linear least squares is used in homography estimation, fundamental matrix calculation and image alignment. The non-linear least squares is used for pose estimation, bundle adjustment, etc., while the combination of the two methods is used in different applications such as camera calibration.

A more general optimization problem is simply to minimize the cost (objective) function $g(\mathbf{p})$ over all the values of unknown parameter $\mathbf{p}$ and therefore find $\mathbf{p}^*$, a global minimizer for $g(\mathbf{p})$, such that

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} g(\mathbf{p}) \tag{3.49}$$

In the least-squares minimization, the cost function is the squared distance objective function

$$g(\mathbf{p}) = \frac{1}{2}\sum_i (\epsilon_i(\mathbf{p}))^2 \tag{3.50}$$

which can be denoted in the matrix form as

$$g(\mathbf{p}) = \frac{1}{2}\|\epsilon(\mathbf{p})\|^2 = \frac{1}{2}\epsilon(\mathbf{p})^T\epsilon(\mathbf{p}) \tag{3.51}$$

where $\epsilon(\mathbf{p}) = [\epsilon_1, \epsilon_2, ..., \epsilon_n]^T$.

In computer vision, it is very common that $\epsilon_i(\mathbf{p})$ is defined as the difference between the measurement b_i and the prediction $f(a_i, \mathbf{p})$ and referred to as residual

$$\epsilon_i(\mathbf{p}) = f(a_i, \mathbf{p}) - b_i \tag{3.52}$$

Then the cost function becomes

$$g(\mathbf{p}) = \frac{1}{2}\sum_i (f(a_i, \mathbf{p}) - b_i)^2 \tag{3.53}$$

If $f(a_i, \mathbf{p})$ is linear in the unknown parameter $\mathbf{p}$ then it is a linear least-squares minimization problem. Otherwise it is a non-linear least-squares problem.

3.3.1 Linear least-squares

The cost function (3.53) can be written in the linear least-squares form (if $f(a_i, \mathbf{p})$ is linear) as

$$g_{LLS}(\mathbf{p}) = \sum_i |a_i\mathbf{p} - b_i|^2 = \|\mathbf{A}\mathbf{p} - \mathbf{b}\|^2 \tag{3.54}$$

where $\mathbf{A} = [a_1, a_2, ..., a_n]^T$ and $\mathbf{b} = [b_1, b_2, ..., b_n]^T$.

This minimization can be used if a solution for linear equations $\mathbf{A}\mathbf{p} = \mathbf{b}$ where $\mathbf{A}$ is a mxn matrix, does not exist, since it is still important to find a vector $\mathbf{p}$ that is closest to a solution to the system. In order to do that $g_{LLS}(\mathbf{p})$ is minimized and such $\mathbf{p}^*$ is known as the least squares solution to the overdetermined system.

The $\mathbf{p}^*$ corresponding to the minimum value for $g_{LLS}(\mathbf{p})$ can be found by solving the associated normal equations

$$(\mathbf{A}^T\mathbf{A})\mathbf{p}^* = \mathbf{A}^T\mathbf{b} \tag{3.55}$$

which can be obtained by setting the derivative of $g_{LLS}(\mathbf{p})$ equal to zero

$$\frac{\partial g_{LLS}(\mathbf{p})}{\partial \mathbf{p}} = 0 \tag{3.56}$$

The final $\mathbf{p}^*$ is

$$\mathbf{p}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} = \mathbf{A}^+\mathbf{b} \tag{3.57}$$

if $\mathbf{A}^T\mathbf{A}$ is invertible. $\mathbf{A}^+$ is called the *pseudo-inverse* of $\mathbf{A}$. Also $\mathbf{p}^*$ can be computed numerically by using the Singular Value Decomposition (SVD) [79].

In the case of homogeneous equations such as $\mathbf{Ap} = 0$, the homogeneous squared error is

$$g_{HLS}(\mathbf{p}) = \sum_i |a_i\mathbf{p}|^2 = \|\mathbf{Ap}\|^2 \tag{3.58}$$

If $\mathbf{p}$ is a solution, so is $k\mathbf{p}$ for any scalar k. A reasonable constraint would be $\|\mathbf{p}\| = 1$. Then the solution is the last column of V (corresponding to the smallest singular value) where $\mathbf{A} = \mathbf{UDV}^T$ is the SVD of $\mathbf{A}$ (which is the same as the eigenvector of $\mathbf{A}^T\mathbf{A}$ corresponding to the smallest eigenvector).

3.3.2 Non-linear least-squares

When the cost function given in Eq.(3.53) is not linear in the unknown parameter $\mathbf{p}$, it is minimized iteratively by relinearizing it around the current estimate of $\mathbf{p}$ using the gradient derivative (Jacobian) $J^p = \frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}$ evaluated at the current estimate and computing an incremental improvement $\triangle\mathbf{p}$. First start with an initial estimate $\mathbf{p}_0$ which is close to minimum and then refine the estimate under the assumption that the function $f(p)$ is locally linear around $\mathbf{p}_0$

$$f(\mathbf{p}_0 + \triangle\mathbf{p}) = f(\mathbf{p}_0) + J^0\triangle\mathbf{p} \tag{3.59}$$

where the Jacobian J is evaluated at the current estimate.

We seek a point $\mathbf{p}_1 = \mathbf{p}_0 + \triangle\mathbf{p}$, which minimizes $\frac{1}{2}\sum_i(f(a_i, \mathbf{p}_1) - b_i)^2$ that can be approximated by

$$\frac{1}{2}\sum_i(f(a_i, \mathbf{p}_0) + J^0\triangle\mathbf{p} - b_i)^2 \tag{3.60}$$

This cost function can be written as

$$\frac{1}{2}\sum_i(J^0\triangle\mathbf{p} + \epsilon_i(\mathbf{p}_0))^2 \tag{3.61}$$

and requires to minimize $\|J^0\triangle\mathbf{p}+\epsilon_i(\mathbf{p}_0)\|$ over $\triangle\mathbf{p}$, which is a linear least-squares problem and the vector $\triangle\mathbf{p}$ can be obtained by solving the normal equations (3.55). The iterations continue until convergence is obtained.

The objective function can also be approximated around $\mathbf{p}_0$ by using Taylor series

$$g(\mathbf{p}_0 + \triangle) = g(\mathbf{p}_0) + g_p(\mathbf{p}_0)\triangle + \frac{1}{2}\triangle^T g_{pp}(\mathbf{p}_0)\triangle + \ldots \tag{3.62}$$

where *gradient* $g_{\mathbf{p}} = \frac{\partial g(P)}{\partial P}$ and *Hessian* $g_{pp} = \frac{\partial g_p(P)}{\partial P}$ are evaluated at $\mathbf{p}_0$. To minimize $g(\mathbf{p}_0 + \triangle)$ with respect to $\boldsymbol{\Delta}$, we need to differentiate with respect to $\triangle$ and set to 0;

$$g_p + g_{pp}\triangle = 0 \implies g_{pp}\triangle = -g_p \tag{3.63}$$

If $g(\mathbf{p})$ is a least-squares cost function as given in Eq.(3.51), then various update equations can be used such as

1. Newton update equation
$$g_{pp}\triangle = -g_p$$
where $g_p = \epsilon_p^T \epsilon$ and $g_{pp} = \epsilon_p^T \epsilon_p + \epsilon_{pp}^T \epsilon$ evaluated at $\mathbf{p}_0$
2. Gauss-Newton update equation
$$\epsilon_p^T \epsilon_p \triangle = -\epsilon_p^T \epsilon$$
under the assumption that $g(\mathbf{p})$ (ϵ_p) is linear around the solution $\mathbf{p}_0$, and the Hessian is approximated as $g_{pp} = \epsilon_p^T \epsilon_p = J^T J$ since $\epsilon_{pp}^T \epsilon = 0$, and $\epsilon_p = J$.
3. Gradient descent
$$\lambda\triangle = -g_p$$
The negative gradient vector defined the direction of most rapid decrease of the cost function. So this is moving in that direction iteratively and λ defines the length of the step. It is similar to Newton but instead the Hessian is approximated by the scalar matrix λI.

Then the update $\triangle$ can be found by using one of the update equations and the updates are repeated until the objective function is minimized.

Levenberg-Marquardt (LM) iteration

The LM algorithm varies parameter updates between Gauss-Newton updates and gradient descent updates by adjusting λ

$$(J^T J + \lambda I)\triangle = -J^T \epsilon \tag{3.64}$$

a small λ results in a Gauss-Newton update, which will cause rapid convergence in the neighborhood of the solution, while a large λ is a gradient descent step, which will guarantee a decrease in the cost function when the convergence is difficult.

In order to avoid 'error valley' problem, Marquardt proposes the update (augmented normal equations)

$$(J^T J + \lambda diag(J^T J))\triangle = -J^T \epsilon \tag{3.65}$$

Since the Hessian H is proportional to the curvature of g, the above equation implies a large step in the direction with low curvature and a small step in the direction with high curvature.

3.3.3 Robust least-squares (M-Estimation)

Regular least-squares methods are minimizing the residuals of all the measurements and they are optimal and reliable when the noise in the measurements is Gaussian. However, the Gaussian assumption is violated when there are outliers in the measurements and the least-squares result is skewed in order to approximate a Gaussian with the data. Therefore a robust version of it is necessary in the presence of outliers among the measurements. An M-Estimator in which a robust penalty function $\rho(r)$ is applied to the residuals, can be used for such a case. The cost function is defined as

$$g_{RLS}(\mathbf{p}) = \sum_i \rho\left(\epsilon_i(\mathbf{p})/\sigma_i\right) \tag{3.66}$$

where $\rho(r)$ is a continuous, symmetric and monotonically increasing function with a minimum value at $r = 0$, and σ_i is the scale of the residual error ϵ_i.

The rationale for using a robust penalty function (such as Tukey, Cauchy etc.) is to decrease the effect of the measurements with gross errors in the estimation process [76]. In the least-squares method, the residuals for any measurement can be arbitrarily large. However, in robust loss functions, small error values correspond to Gaussian noise and are included in the minimization process while the influence of large errors are either bounded or totally eliminated.

The minimization of the cost function can be done by taking the derivative and equating it to zero

$$\sum_i \psi\left(\frac{\epsilon_i(\mathbf{p})}{\sigma_i}\right) \frac{\partial \epsilon_i(\mathbf{p})}{\partial \mathbf{p}} \frac{1}{\sigma_i} = 0 \tag{3.67}$$

where $\psi(r)$ is the derivative of $\rho(r)$ and is called the *influence function*. If the weight function $w(r) = \psi(r)/r$ is introduced to solve the equation above, then

$$\sum_i w\left(\frac{\epsilon_i(\mathbf{p})}{\sigma_i}\right) \epsilon_i(\mathbf{p}) \frac{\partial \epsilon_i(\mathbf{p})}{\partial \mathbf{p}} \frac{1}{\sigma_i^2} = 0 \tag{3.68}$$

This minimization can be done by an *iteratively reweighted least squares* algorithm which is used to minimize cost functions in the form of $\sum_i w(r)r^2$. At each iteration the weights $w_i = w(\epsilon_i(\mathbf{p})/\sigma_i)$ are calculated by using the previous iterations estimate of $\mathbf{p}$, $\mathbf{p}_t$, and the estimate $\boldsymbol{\Delta}\mathbf{p}_t$ is calculated with the weights fixed by solving the normal equations (explained in Sec.2.3.1)

$$J^T C J \triangle \mathbf{p}_t = -J^T C b \tag{3.69}$$

$$\triangle \mathbf{p}_t = -(J^T C J)^{-1} J^T C b \tag{3.70}$$

$$\mathbf{p}_{t+1} = \mathbf{p}_t + \triangle \mathbf{p}_t \tag{3.71}$$

where C is a diagonal matrix, $diag(w_1, w_2, ..., w_n)$, corresponding to the independent weights and J is the Jacobian, both evaluated at the local point $\mathbf{p}$. The iterations continue until $\mathbf{p}$ converges to a final value.

Various robust penalty functions and corresponding weight functions are given in Table 3.1 and illustrated in Fig.3.5 together with the least-squares cost function. For instance, Tukey's robust cost function rejects the outliers that satisfy

$$|r| = \left| \frac{\epsilon_i(\mathbf{p})}{\sigma_i} \right| > a \tag{3.72}$$

where 95% asymptotic efficiency can be obtained by tuning $a = 4.6851$ as shown in Fig.3.5. Therefore, residuals $|\epsilon_i(\mathbf{p})| > 4.6851\sigma_i$ are suppressed as outliers.

Incorporating σ_i solves the scaling problem that is introduced when the constant a is used as a threshold for outlier detection. A proper selection of scale will improve outlier detection and therefore the estimate of the scale should be robust and not affected by outliers. Since the σ_i are not available beforehand, a robust standard deviation estimate of the errors can be used as the scale [212]

$$\tilde{\sigma} = 1.4826 \left(1 + \frac{5}{n-d} \right) \underset{i}{\text{median}} \, |\epsilon_i(\mathbf{p})| \tag{3.73}$$

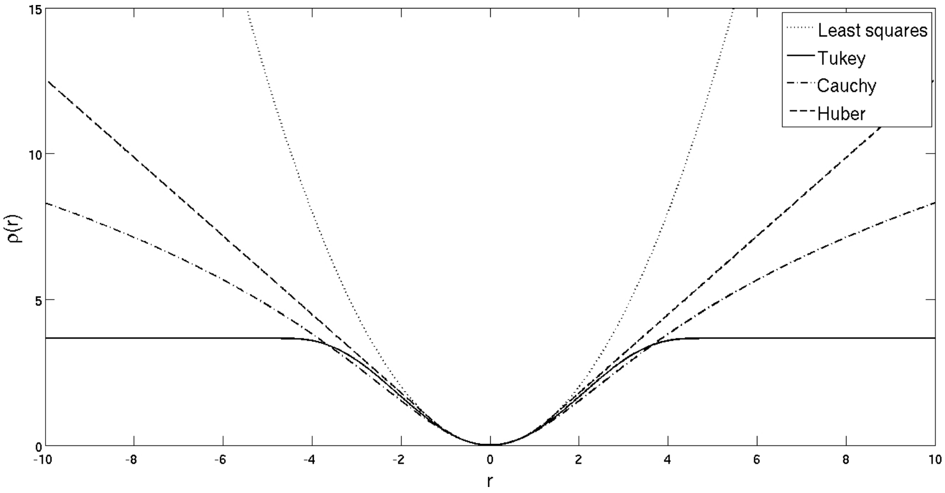
where n is the size of the data set and d is the dimension of the parameter vector $\mathbf{p}$. The constant 1.4826 is equal to the median of the absolute values of random numbers sampled from the Gaussian normal distribution $N(0, 1)$.

3.4 Random Sample Consensus (RANSAC)

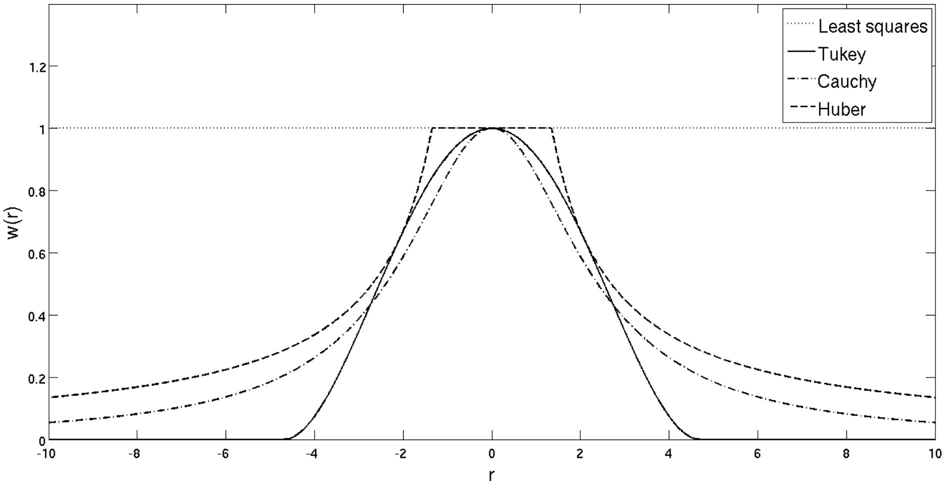
The RANSAC methods is used for fitting a model to experimental data which can tolerate more than 50% outliers in the data [59]. As explained in the previous sections, classical techniques for parameter fitting such as the least-squares method are minimizing the errors for all measurements (including the outliers) and therefore the estimation fails to represent the correct model in the presence of outliers. In the robust least-squares method, outliers are detected by using

Table 3.1: *Robust loss functions and corresponding weight functions.*

	$\rho(r)$	$w(r)$
Least-squares	$r^2/2$	1
Tukey	$\begin{cases} \frac{a^2}{6}[1-(1-(\frac{r}{a})^2)^3], & \lvert r\rvert \le a \\ \frac{a^2}{6}, & \lvert r\rvert > a \end{cases}$	$\begin{cases} [1-(\frac{r}{a})^2]^2, & \lvert r\rvert \le a \\ 0, & \lvert r\rvert > a \end{cases}$
Cauchy	$\frac{b^2}{2} log(1+(\frac{r}{b})^2)$	$\frac{1}{1+(\frac{r}{b})^2}$
Huber	$\begin{cases} \frac{1}{2}r^2, & \lvert r\rvert \le c \\ \frac{1}{2}c(2\lvert r\rvert - c), & \lvert r\rvert > c \end{cases}$	$\begin{cases} 1, & \lvert r\rvert \le c \\ \frac{c}{\lvert r\rvert}, & \lvert r\rvert > c \end{cases}$



(a) *Robust loss functions*



(b) *Robust weight functions*

Figure 3.5: *Robust loss functions and corresponding weight functions, parameters are tuned to a=4.6851, b=2.3849, c=1.345 to obtain 95% asymptotic efficiency.*

Table 3.2: *RANSAC: finding pose using random sample consensus.*

```
Select
    $n$: smallest number of points required to fit the model
    $k$: the number of iterations required
    $d$: the number of supporting samples required to assert a model fits well
    $t$: the threshold used to identify the samples that fit well to the model
Until $k$ iterations have occurred
    Select $n$ points from the data uniformly and randomly
    Calculate the model parameters (fit the model) by using that $n$ points
    For each data point outside of the $n$ points set
        Evaluate the reprojection error between the point measurement
        and its backprojection with the estimated model. If the error is
        less than $t$, then the point fits well to the model
    end
    If there are $d$ or more points that fit well with the estimated model then
    there is a good fit. Recalculate the model parameters using all these
    points
end
Use the best fit from this collection, using the total reprojection error of the
final fitting as a criterion
```

the median of all the samples' residuals and these outliers are weighted less as their residuals greater then a value determined by the median. The RANSAC is another approach to detect outliers and select the inliers that are used for model fitting. Although it is computationally more expensive due to its iterative select-fit-evaluate nature, it is very robust against the outliers.

In Table.3.2 we explain the method by describing the application of RANSAC on pose estimation which can be generalized to any model fitting problem by changing the desired model type [79]. For instance the pose model can be replaced by the plane model and instead of the reprojection errors to evaluate the quality, the distance between the 3D points and the plane can be used as an error criterion to evaluate the samples and the model. Then the same algorithm can be used for fitting planes to 3D points. The pose can be calculated by using and n-point method [79] from 3D point and 2D feature matches which is explained in more detail in Section 4.3.5.

3.5 3D Reconstruction

3D reconstruction is the process of recovering the geometric structure of a scene from one or more of its images. More formally, recovering the 3D world coordinates $[X, Y, Z]^T$ of a point $\mathbf{P}$ given the pixel coordinates $\mathbf{p}_i$ of its projections onto the images. 3D reconstruction from a single image is an under-determined problem. However, if two images of the point $\mathbf{P}$ captured by cameras with known internal and external parameters are given, then the position of the point $\mathbf{P}$ can be recovered by triangulation as explained in Section 3.2.1.

If the non-linear distortions are corrected then the perspective projection equations can be written

$$\lambda_1 p_1 = K_1 R_1^T (P - C_1) \ , \ \lambda_2 p_2 = K_2 R_2^T (P - C_2) \tag{3.74}$$

When all intrinsic and extrinsic parameters K_1, K_2, R_1, R_2, C_1 and C_2 are available (via stereo camera calibration), the 3D world coordinate of the point P can be estimated which results in full reconstruction.

The relation between the two image coordinates of the 3D point in a single stereo image pair captured by a stereo rig is:

$$\lambda_1 p_1 = K_1 P' \text{ and } \lambda_2 p_2 = K_2 R_2^T R_1 P' + K_2 R_2^T (C_1 - C_2) \tag{3.75}$$

where $P' = R_1^T (P - C_1)$.

The pose of the first camera in the world coordinate frame (R_1, C_1) cannot be recovered from a single stereo image pair but the intrinsic parameters and the relative pose of the second camera with respect to the first camera are known. Therefore, the scene can be reconstructed by the image pair only up to a Euclidean transformation $P' = R_1^T (P - C_1)$ with respect to the world and is called Euclidean reconstruction. When the (metric) distance between the cameras is also not known (only K_1, K_2, $R_2^T R_1$ and the direction of the translation between two cameras, $R_2^T (C_1 - C_2)$ up to a scale are known) then the scene is reconstructed up to an Euclidean and 3D similarity transformation. This reconstruction is referred as the metric (similarity) reconstruction. If the internal camera parameters are also not known (only the relative pose information without the precise inter-camera distance is known) then this is called affine reconstruction. Finally, if we have no knowledge about the camera and the scene, we can only find the fundamental matrix with image correspondences and 3D reconstruction can be done up to a projective transformation. This is called the projective reconstruction.

The projective transformations preserve only collinearity and coplanarity. The affine transformation preserves parallelism additional to the collinearity while the metric transformations preserve the angles but not metric size. In our case the pose information supplied by the tracking module can be used to recover the absolute information about the cameras' extrinsic parameters in the real world and P can be recovered from P'.

3.5.1 Bundle Adjustment

Errors on the estimated parameters such as the combined 3D feature coordinates, camera poses, and calibrations propagate in time and accumulate in each new

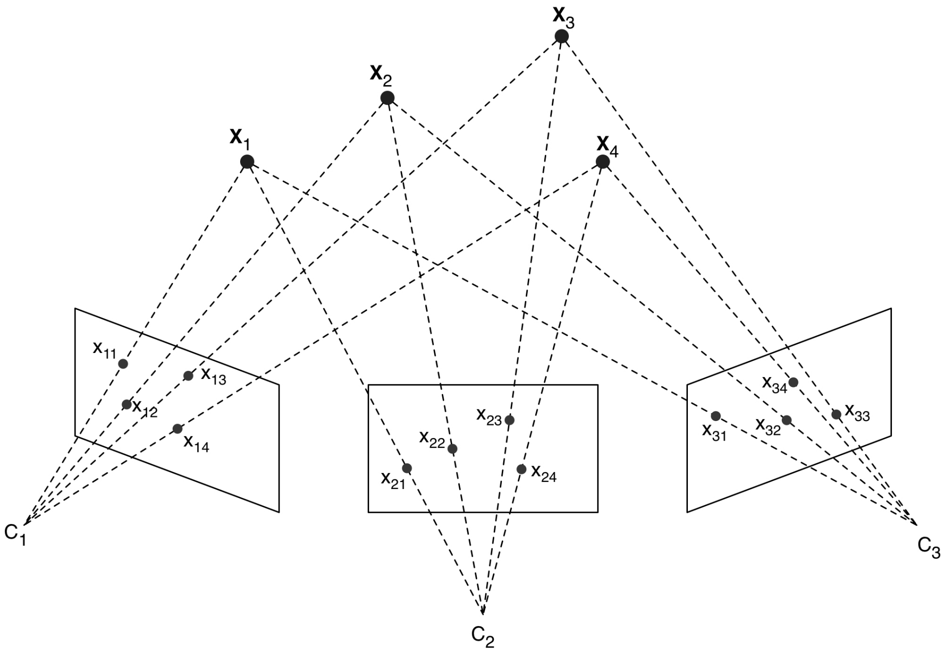


Figure 3.6: *The goal of bundle adjustment is to minimize the distance between the reprojected 3D points and the extracted 2D features that they are reconstructed from. The larger points represent the 3D points in space and the smaller points are their reprojections onto the each camera frame.*

camera pose. In order to solve this problem, the global reprojection error of all 3D points in the images in which they are visible is minimized. This optimization problem is solved by using *Bundle adjustment* [193] which is the refinement of a 3D reconstruction to produce jointly optimal 3D structure and viewing parameter estimates. Therefore it enables accurate map creation and extension while correcting errors that can occur due to drift.

The goal of bundle adjustment is to minimize the distance between the reprojected 3D points and the extracted 2D features that they are reconstructed from, as illustrated in Fig.3.6. The global reprojection error, ϵ_G, is defined as

$$\epsilon_G = \sum_i \sum_j D(x_{ji}, P_j \mathbf{X}_i)^2 \tag{3.76}$$

where $\mathbf{X}_i$ is the ith 3D point, $P_j\mathbf{X}_i$ is its projection onto the camera C_j with the projection matrix P_j and x_{ji} is the measurement of the point X_i in the camera frame C_j.

This problem is formulated as a non-linear least-squares problem as explained in Section 3.3.2 and can be optimized by using Levenberg-Marquardt algorithm

(Section 3.3.2). In every iteration, new estimates are found for camera poses, calibration parameters and 3D feature coordinates that minimizes the error function. Also each 3D point is not visible in all the images and therefore the Jacobian matrices are sparse. This sparsity decreases the computational load of the algorithm. The implementation and utilization details of the bundle adjustment algorithm are discussed in more detail in Section 4.3.6.

Chapter 4

Visual Odometry

Everything must be made as simple as possible, but no simpler.
Albert Einstein

4.1 Introduction

The key goal of AR is combining real-world objects and representations of virtual objects such that the combination is practically indistinguishable to the user. If such a blending can be achieved, then a dynamic real-scene can be augmented with additional information that can aid in many tasks. This requires a proper alignment of the real and virtual objects with respect to each other or the perception that the two coexist in the same 3D space cannot be preserved.

Tuceryan et al. [195] defines a number of factors for the success of a realistic AR experience, which includes camera (viewpoint) tracking, the modeling of the real-world geometry and graphics rendering. Tracking the camera (head-pose of the user) is necessary to reflect the changes in the user's viewpoint in the rendered graphics. Extracting the real-world geometry is crucial for registering virtual entities with the real scene. Graphics rendering hardware/software and display hardware (such as HMDs and game engines) are important to create virtual content for augmenting the real world.

In order to provide a realistic AR experience and preserve the spatial consistency between the real and the virtual world, the relative pose between the camera/display and the objects needs to be estimated with high precision. This is usually accomplished by tracking the (mostly static) landmarks (or real objects) available in the scene and using this information to update the pose of the camera. However, this is very difficult to accomplish due to the precision required. Even a small misalignment can be detected by the human visual system due to the resolution of the fovea and the sensitivity of the visual system to differences [9].

Moreover, the resolution of the displays are usually lower than the human eye and therefore errors of just a few pixels are noticeable.

Also, the failure in the tracking[1] and registration processes, and the lag may result in higher sickness symptomatology such as nausea, oculomotor disturbances and disorientation [186] which results in some aftereffects such as decrease in eye-hand coordination, and postural stability.

Almost two decades ago, Azuma [8] defined 3 main requirements of a tracker for AR as:

1. The accuracy of the tracker must be within a small fraction of a degree in orientation and a few millimeters (mm) in position
2. The combined latency (delay) between the time that the tracker takes the measurements and the time that the graphics engine renders the augmented image in the display must be very low
3. The tracker must work at long ranges (in contrast to its VR counterpart) such that the extended-range trackers must support walking users

Following these requirements, we aimed for a combined latency less than 50 milliseconds (> 20 fps) and defined our working range as room-size environments. In order to satisfy these requirements, we focus on vision-based trackers since they are non-invasive, accurate and relatively low-cost. However, they suffer from various challenges such as lighting conditions, occlusions, lack of texture, cluttered backgrounds and repetitive patterns. For instance, partial or full occlusion may prevent the feature to be detected, or due to lighting conditions, specular reflections or fast motion of the camera, a motion blur may result in failure of the feature detection. Also observing the same scene from a different point of view might lead to a confusion of the tracker since the features look very different.

In this chapter, a system is described that is able to track the 3D pose of a moving stereo-camera pair in a 3D world, while simultaneously building a sparse 3D map of that world. The sparse map is utilized as landmarks while performing the tracking. The system presented satisfies the requirements and overcomes the challenges mentioned.

4.2 Related Work

Pose tracking techniques can be separated into 3 main parts, as illustrated in Fig. 4.1: sensor-based, vision-based and hybrid tracking techniques [215].

Sensor-based methods utilize sensors such as magnetic, acoustic, mechanical, inertial, optical, etc. or their combinations, to track the user's pose and each

---

[1] In AR and VR contexts, vision based tracking refers to pose estimation or visual odometry. However, in computer vision, tracking is mostly used for data association such as matching between features, models or images. In this chapter tracking refers to the first context until otherwise stated.

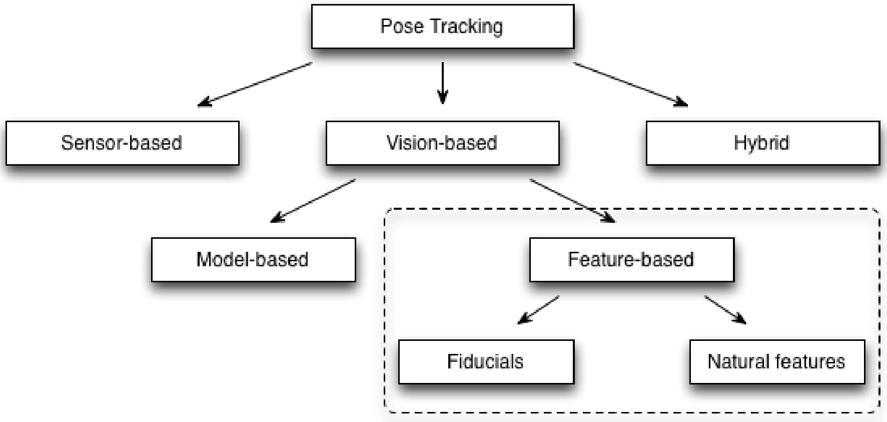


Figure 4.1: *Overview of pose tracking techniques.*

method has its own advantages and limitations [163]. For instance accelerometers don't need any reference while the error in position measurements increases due to integration. Magnetic sensors are sensitive against the presence of metals in the environment and inertial trackers accumulate the error due to drift with time. In [136], a scalable, indoor system was presented that used a combination of ultrasonic sensors (Time-of-Flight sensing) and fixed receivers, with an inertial tracker on the HMD. In [137], Newman et al. extended their previous concept by introducing a *Ubiquitous Tracking* method, in which the measurements from widespread and diverse sensors are automatically fused. Azuma and Bishop [11] proposed a system in which four optical sensors aimed at an array of infrared LEDs mounted in ceiling panels combined with inertial sensors. A similar active LED system was used in guiding construction workers and as a 'touring guide' in [57]. A 6 degrees-of-freedom magnetic tracker system is used to track the camera pose in [195]. However, the sensor based systems have no feedback on the accuracy of the pose estimates or the matching between the real and virtual world. Moreover they lack any error correction mechanism, and therefore analogous to an open loop system whose output is perceived to have an error. However, vision based AR systems are closed loop systems since they can provide feedback to the system by using features in the real environment and their projections in 2D images and correct mis-registration errors by altering the pose estimate dynamically [13].

Vision-based tracking techniques employ optical cameras and computer vision algorithms, and therefore they are non-invasive, accurate and relatively low-cost. They can be divided into two main classes: model-based and feature-based algorithms [157].

Model based tracking methods use a (3D) model of the tracked object and usually provides a more robust solution. In the basic concept of the model-based tracking, the pose information is updated in each video frame first by using a dynamic model via a prediction filter and then by measurements in the video frame. After projecting the model edges of the object to the image plane by using the latest camera pose, edge search is performed near the projected edges and in the direction perpendicular to the rendered edges. Then, the error between the projected edges and the actual edge locations is minimized by using (non-linear) least-squares methods (explained in Section 3.3). The estimated pose corresponding to the minimum error is used in the next frame. Many systems share these basic principles.

For instance, in [52, 207, 124] tracking is done by seeking the alignment of object's 3D Model (CAD model) edges and the area of high image gradient via a robust estimator and an iterative re-weighted least-squares (Section 3.3.3). Comport et al. [39] used CAD models and parametric representations of geometric shapes for tracking. However, relying only on edge information might fail in highly textured environments or objects, and lead to erroneous pose estimation. Reitmayr et al. [161] proposed a hybrid outdoor system in which an edge-based tracker was combined with a textured 3D model. In [158, 157], edge detection and texture analysis are combined to increase the robustness of the tracker. Vacchetti et al [196] introduce another robust estimator to handle multiple hypotheses (image corners and model edges) in the pose estimation which minimizes the errors in different measurement types together. For every frame, corner features (Harris) are extracted and matched with interest points in the latest offline-reference frames in which the 3D locations of the interest points are known beforehand. Then the camera pose is estimated by using the matching between the 2D image features and the 3D world points. They use the initial pose estimate, model edges and interest points from the previous frame and the current frame to refine the pose estimate and also the absolute 3D position of the tracked features in the world coordinate frame by using their robust estimator. Bleser et al.[20, 18] presented a hybrid tracker that exploits the object CAD model for initial pose estimation and tracks the model as long as it is in the field of view. Also 3D locations of new features (corners) are added as landmarks and tracked by Extended Kalman Filtering (EKF) as explained in the SLAM based trackers later. Model-based tracking methods need off-line data such as 3D models of the tracked objects or the environments, and are therefore difficult to maintain in an unknown environment and it restricts the camera motion.

Feature-based tracking algorithms use interest points, i.e. corners, present in the scene that are different than their neighbors and can be detected easily, and therefore don't require any model. Feature-based tracking algorithms fall into two parts: fiducial-based feature tracking and natural feature tracking. The former approach utilizes known-markers while the latter one focuses on tracking 2D features such as corners, edges, or texture.

Fiducials (i.e. landmarks or markers) are exploited to introduce a set of easy to extract correspondences between 3D features and image features. In general

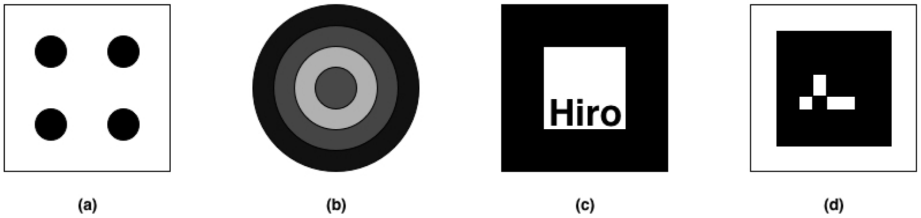


Figure 4.2: *Different markers are utilized in the development of AR applications. Some samples of circular markers (a) and (b) are used in [37] and [35] while rectangular ones (c) and (d) are exploited in [89] and [27]*

fiducials are easy to detect and the image locations of the markers can be detected with a higher precision relative to natural features. Moreover, 3D positions of the corresponding 2D marker features can be measured with very high accuracy (for instance by hand).

The topic of 3D pose estimation from fiducials has been extensively researched in the past and various marker types are used as shown in Fig.4.2. Circular markers and their centers are used in various research related to AR [35, 133, 197, 144]. For instance in [35], Cho et al. utilized colored circular fiducials in different sizes for a scalable system. The pose is found by using the RANSAC method [59] as explained in Section 3.2. The pose is estimated using three randomly selected correspondences between 3D features and image features, and evaluated by using the reprojection errors. This procedure is repeated until a good estimate with less then the minimum reprojection error is found or maximum number of iterations is reached.

Planar rectangular markers are also exploited in many research efforts [89, 187, 211, 27]. With these markers and a single camera, the pose of that camera can be determined up to a scale factor. The marker detection is based on the search of contours of (possibly skewed) rectangular objects in the camera, followed by the detection of a valid pattern (ID) in its contour's interior. The four corner points of a marker are used to calculate the rotations of the marker with respect to the camera. In order to obtain a higher accuracy of the positions of the corners than the size of a single pixel, i.e. *sub-pixel accuracy*, the corners of the skewed rectangle are omitted, whereupon high accuracy line fits –using a Gaussian line model on the gray-value pixels that form the lines– are done on the four line-pieces making up the rectangle. Geometric intersections are used to determine the corner positions at high accuracy, making it possible to calculate the pose of the marker with respect to the camera up to a scale factor. If the size of the marker is linked to its ID, the 3D camera pose can be accurately tracked in real-time.

As an another approach, [37] presented a fiducial detector based on machine learning which has significant performance improvements. They train a classifier with sample fiducial images under varying conditions and label the input image pixels as either fiducial or not.

Negative points are that the markers must be put precisely in place in the scene beforehand, and that for markers far away the corners (or centers) are too close together to make an accurate pose calculation possible. In this case, setting up an exact grid of markers may overcome this, but setting up such an exact grid is very cumbersome and, moreover, not possible in many unknown environments. Also, it is vulnerable to occlusions of the markers [98].

Due to new algorithms and faster computers, real-time natural feature tracking becomes feasible for pose estimation. In contrast to the model-based and marker-based methods, landmarks are acquired during the operation of the system and therefore the system can perform extendible tracking. This eliminates the dependency on prior information about the environment and extends the range of operation. These methods can also be classified as Structure-from-Motion (SfM) approaches, which start from scratch and build the scene structure and landmarks while estimating the camera pose.

Initial methods utilize natural features to support marker based tracking or require a marker based tracking for calibration, prior to the natural feature based tracking. One of the first examples of natural feature tracking for robust AR is presented in [146]. Park et al. used fiducials to initialize their system and find the pose of the camera. Afterwards, they detect natural features, register them relative to the fiducials and utilize them to find the pose when the fiducials are not visible. However, they were not able to achieve real time operation.

In [68], Genc et al. proposed a two stage tracking method for AR that combines markers, natural features and *Bundle adjustment* [193] (see also Section 3.5.1). During the first stage, the training stage, the pose of the camera is estimated by using fiducials while the 2D natural features are tracked and their 3D positions are estimated by using triangulation (Section 3.2.1). During this stage, *bundle adjustment* is used to refine the 3D positions of the points. When the system is trained and the number of 3D points are enough, the system switches to the second tracking stage in which only the learned features are used for camera tracking. The features are tracked by using an optical flow tracker (see Section 4.3.4) and the pose is computed using the algorithm by Tsai [194]. The map is only modified during the training stage and not extended in the tracking stage. In [189], Subbarao et al. improved the system in [68] by introducing the Heteroscedastic Errors-In-Variables Regression (HEIV) estimator [126] instead of Tsai's method to calculate the camera pose and the structure which claimed to be robust against noise and nonlinearity of the system. However, the computational load is very high and the system cannot perform in real-time.

Simon et al. [179] presented a marker-less camera tracking system for planar environments. A plane is specified manually when the system starts up. Afterwards, the interest points on the specified plane are matched between the current and the next frame, and from the set of correspondences a planar homography [79] is calculated. The planar homography is a projective mapping that maps coplanar points x_i on the plane π_1 to coplanar points $x_i^{'}$ onto the plane π_2. Therefore it is applicable to rigid and planar surfaces. The homographic mapping can be represented by $3x3$ matrix H, which can be derived by using the mapping between the points x_i and $x_i^{'}$. The matrix H contains 9 entries, but it can be defined only up to a scale. Thus, the total number of degrees of freedom in a 2D projective transformation is 8. Therefore, minimum 4 points are necessary to compute the H matrix, since a 2D point has two degrees of freedom corresponding to its x and y components. By decomposing the homography matrix into the position and the rotation of the camera, the pose is calculated. The selected plane imposes a natural coordinate frame for the alignment of the virtual and real content. However, the planarity assumption is difficult to satisfy in many environments.

Skrypnyk and Lowe [180] proposed a two stage system for markerless AR. In the first offline-stage SIFT features [113] are extracted from the reference images and multi-view correspondences are calculated. Then these correspondences are used to create a metric model of the world and perform self-calibration. In the second stage, features are extracted from the captured images and matched with a world model by using a Best-Bin-First search algorithm [113]. The Best Bin First search finds the nearest neighbor (or a very close neighbor) of the query point in the world model. Afterwards, the pose of the camera is calculated by using correspondences. However, due to the high computational load of SIFT features, the system can only perform at 4 fps. Park et al. [147] split the object detection and tracking stages by using key-frames. Key-frames are selected from the captured images when there is a significant motion in the camera pose (see Section 4.3 for more detail). Object recognition is performed on the key-frames with pre-learned features and the detected object pose is estimated by feature matching and robust pose estimation (see Section 3.3.3 and 4.3.5) frame-by-frame.

In [139] the motion of a stereo-rig is estimated by using a traditional feature tracker such that normalized correlation over an $11x11$ window is used to evaluate the potential matches. The relative pose of the new frame is estimated by using a 5-Point algorithm and preemptive RANSAC (see Section 3.2) followed by an iterative refinement. No scene priors such as planarity or motion priors such as the velocity of the camera are used. Also, they did not use any global optimization method such as batch techniques (see Section 3.5.1) to refine/correct the pose estimates.

In [67], Geiger et al. presented a work similar to our system, which can track the camera pose while creating a dense 3D reconstruction, but mainly designed for vehicles. They assumed that the 3D landmarks are visible for a couple of frames and therefore they match 2D key-points between current left-right and previous left-right stereo pairs. If the feature is successfully matched between 4 images then it is used for pose estimation by minimizing the backprojection error in the stereo view. Then they combine stereo-disparity maps by using the pose estimates. However, they don't utilize any batch processing (bundle adjustment) to refine the estimates in order to be real-time.

Standard Structure-from-Motion algorithms suffer from drift and therefore are usually followed by a global optimization scheme (bundle adjustment) explained in Section 3.5.1 or reformulated using a Kalman filter [205] as explained below.

Simultaneous localization and mapping (SLAM) and its adaptations such as EKF-SLAM [47] and FastSLAM [55] are the most popular incremental mapping methods (mostly used in robotics and autonomous vehicles). The current camera pose and the position of every landmark (clearly visible natural feature) are updated with every incoming image. The uncertainties in the measurement and the predictions are stored in a Kalman filter. Davison et al. [46] presented a traditional single camera SLAM framework for a wearable robot (a camera on an extended pan-tilt-zoom unit) with active vision. The robot (a moving lens-camera system with three rotational degrees of freedom) mounted on the user's shoulder is located continuously so that the user can get assistance from a remote expert who can manipulate the camera-robot. In [30, 31] a system in which (planar) object recognition and SLAM are combined for wearable camera systems is presented. They perform object recognition at regular intervals by observing the scene and matching the detected SIFT features with a previously created object database with known metric dimensions. When an object is detected its pose is estimated by decomposing the homography transformation between the image features and the stored model. Then the detected object's 3D positions are embedded into a mono-SLAM map as 3D point measurements. However, these methods suffer from a high computational load due to the data association, and no more than a few dozen points can be tracked and mapped in real-time. Monocular setups suffer from the lack of metric distance and their maps are created up to scale (if a calibration marker is not used). Se and Lowe [175] used a stereo setup and SIFT [113] features as natural features to track the pose of a mobile robot. They assume a 2D planar motion of their platform assuming it operates on planar surfaces such as floors. However, their assumption is not valid for AR setups because the human user frequently violates the 2D planar motion assumption with his/her head by moving it up and down while investigating the scene, and their algorithm is computationally expensive. Efforts to handle large-scale maps and improve the robustness of SLAM have been made [159, 34, 109]. Pupilli et al.[159] employ a particle filter instead of EKF, and [34] replaces correlation-based search with multi-resolution descriptors in a Kalman filter framework for robustness against rapid camera motion and outliers. Lee and Song [109] detect salient regions in the environment and exploit them as landmarks in SLAM. They combine different cues for saliency and register the detected regions into a map for scene exploration.

However, they are still either confined to small-resolution images for real-time operation or not fast enough for head-pose estimation.

Bundle adjustment based methods and their variants become popular as the speed of the processing hardware increases. Compared to EKF, bundle adjustment has a complexity that is linear in the number of landmarks and cubic in the number of poses [193] while EKF has quadratic complexity.

In [166], Royer et al. compute the 3D reconstruction of the environment off-line from a learning sequence. They select key-frames from a reference sequence and calculate the pose of each key-frame and create a 3D environment from matched 2D features between the key-frames. They run bundle adjustment once for each new 3 key-frames since the training step is done off-line. When the map is ready, they start real-time localization by matching the image features to the closest key-frame features and estimating the pose by using 2D features and corresponding 3D points in the map.

Klein and Murray [93] proposed a parallel tracking and mapping (PTAM) system in which they split the pose estimation and mapping processes by using a key-frame structure. They also utilized a local and a global bundle adjuster to correct the positions of landmarks. Their system shows impressive performance in small spaces such as on office desks and forms the basis for the visual tracking system described in this thesis. Guan et al. [73] proposed a similar system to PTAM in which they utilize *ferns* for keyframe recognition and relocalization when the system is lost. Ferns are non-hierarchical structures used to classify image patches [143]. Each fern consists of a small set of binary tests and returns the probability that a patch belongs to a class created during a training stage. All possible appearances of the image patches surrounding a keypoint in a keyframe are assigned to the same class during a training stage. The random ferns are used to assign a new keypoint to the most likely class and therefore perform keyframe recognition. In [32] the object recognition framework presented in [31] is combined with PTAM. Object recognition is performed on key-frames by using SIFT features.

In [132] similar incremental reconstruction concepts with the shape-from-motion paradigm such as calculating the pose, reconstructing new landmarks with the new pose and repeating the pose estimation and the reconstruction steps, are exploited to track the pose. They match the Harris features between two consecutive frames by using a normalized correlation score evaluated over the patches around the features and estimate the pose from the matches. However, instead of running the bundle adjuster in parallel to the tracking, they only run a local bundle adjuster on the last N key-frames when a new key-frame is added to the system. They aim to reduce the computational load of batch processing by using a local bundle adjuster which refines the last added frames instead of all the frames.

Recently there are new research efforts published for drift-free tracking with constant-time batch processing (bundle adjustment). Mei et al. [127, 128] presented a constant time SLAM system (RSLAM) using a stereo setup for a large-scale mapping. They represent the 3D map and the robot position by using a graph structure in which the camera poses are the nodes of the graph and these nodes are connected to the other spatially close ones (camera poses) by edges. Therefore each camera pose is represented by relative transformations (edges) to its neighbors (connected nodes) and this representation is called Continuous Relative Representations (CRR). The current pose of the camera is used to define the active nodes of the graph. Active regions which contain the current camera pose and spatially close N other poses (nodes), define the landmarks used for pose estimation and behaves like a local environment around the robot. They calculate optimal structure and pose locally by using a bundle adjuster only in the active region (with fixed amount of nodes and edges) in constant time which is called relative bundle adjustment (RBA). Therefore a global minimization step that doesn't scale with the size of the map is avoided. In order to detect previously seen scenes by matching the current view with the previously seen ones, they exploited the work in [43] and use this information to correct the current pose and the map of new discovered areas. When a previously seen scene is detected, the current pose of the camera and its connected nodes are updated with the measurements from the old scene node and RBA is performed on the new active region including the old node. This reduces the errors originated from drift or pose estimation. Although the presented work performs better relative to PTAM in outdoor datasets, PTAM shows similar or better results for indoor and desktop environments in which the map size is small and a global bundle adjustment can be performed in near real-time.

As different features than corners, edge features are exploited for pose estimation. [181, 54] used straight lines and edges for EKF SLAM. In [94], Klein et al. used edgelets and combine them with corner-based tracking to improve the resilience against rapid camera motion. Their approach is similar to the model-based methods, but in this case they create edgelets during the operation and extend their edge-map in a similar way to the corner-based methods. The main working principle (and drawback) is very similar to model-based trackers in which the edge features are searched on high gradient regions.

Current research efforts are shifting towards mobile phone applications because of their market penetration. [200] presented the first natural feature based pose tracking at real-time on mobile phones. There are also new systems that employ simplified versions of the previously explained methods [95] and include the various sensors available on mobile phones [103]. A good survey of their applications can be found in [141]. However, they still suffer from the shortcomings of mobile processors and are not suitable for wide-area extensible tracking.

In practice, even the best vision-based tracking methods fail due to fast motion blur, occlusions, lack of texture, etc. In order to cope with that, various sensor data (inertial sensors, gyroscopes, etc.) was combined with image data. These sensors predict the camera position and then this is refined using vision techniques similar to the ones previously described. In [19, 86, 92], various combinations of sensors with vision (mostly in an EKF framework) are given. Recently, depth cameras (RGB-D cameras) such as XBox 360 Microsoft Kinect are becoming popular and are exploited for pose estimation [111, 134, 148]. However, the depth cameras are highly undesirable for our system due to their size and we believe a pure vision-based approach without additional sensors is more generic and hence desirable.

All these approaches have different objectives than we have for our system. This effects the choice of algorithms. Vision based tracking is preferred over sensor-based methods since it is a closed loop system which can provide feedback on the estimate and an error correction mechanism with relatively low-cost hardware while performing in real-time. Model-based visual tracking is not suitable for our application since off-line data such as 3D models of the tracked objects or the environment itself is necessary when the system starts up. Although they can be extensible in time, dependence on models during operation makes it difficult to maintain in unknown environments and it restricts the camera motion. On the other hand, marker based visual tracking methods require setting up an exact grid of markers in the environment, but setting up such an exact grid is very cumbersome and, moreover, not possible in many unknown environments. Also, it is vulnerable to occlusions of the markers. Finally, most of the natural feature based systems are not scalable; they are either confined to small indoor workspaces, difficult to extend and sensitive to lighting conditions, or designed for large outdoor scenes which don't need high precision and real-time operation. Monocular setups such as PTAM suffer from a difficult initialization step, a pure rotational motion which provides no triangulation baseline, unobservability and lack of correctly scaled (metric) camera motion estimation. Also the points needs to be tracked over a couple frames before it contributes to the pose estimation process, which delays its effect. Furthermore, they mostly reconstruct a sparse map which is not suitable for meshing in dense reconstruction and occlusion handling.

4.3 System Overview

Our real-time visual odometry pipeline consists of 5 stages as illustrated in Fig.4.3: image pre-processing, 2D feature detection and selection, temporal inter-frame feature matching, pose estimation and sparse map making including bundle adjustment. The benefit of such a modular system is that, in order to cope with new tasks or hardware setups, one can modify and update every part of the system largely independently of the other parts.

When visual tracking is considered, small inter-frame motion is desirable for narrowing down the search space. However, spatially and temporally close frames carry redundant 3D information and increase the computational load of the (sparse and dense) reconstruction process for creating map points. Especially, the cost of global bundle adjustment grows $O(n^3)$ with the number of frames. Therefore, we use separate threads for tracking, sparse reconstruction and dense stereo reconstruction and perform them in parallel. We separate the threads by using a key-frame based approach, as used by Klein and Murray [93], in which the tracker performs feature detection, temporal matching and robust pose estimation, and creates the key-frames from the input images. A new key-frame is created when the camera moves far from the previous positions into an unobserved part of the

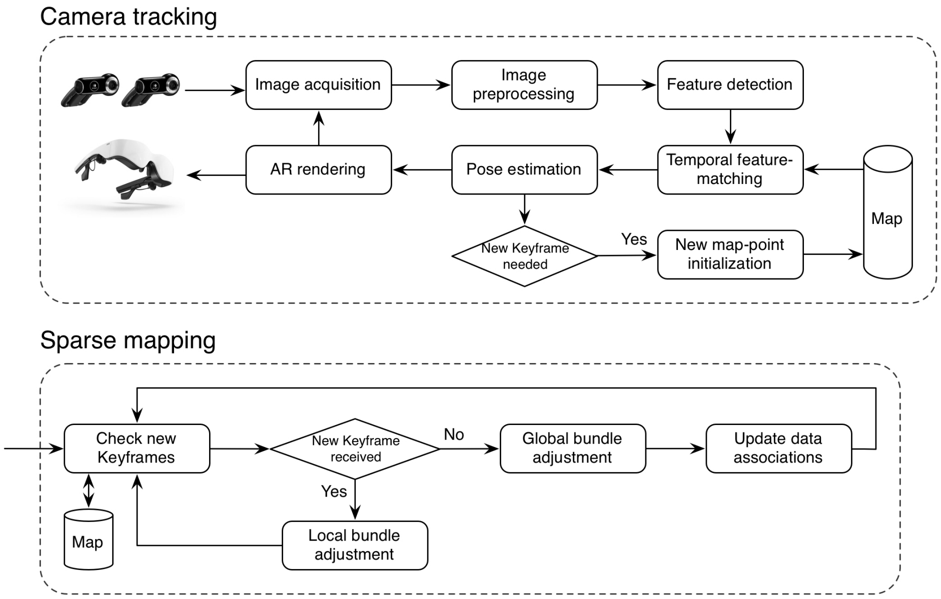


Figure 4.3: *Process flow with independent threads: camera tracking and sparse mapping.*

scene and therefore the created new key-frame is spatially far from the existing key-frames. Afterwards, the new key-frame is sent to the sparse and dense reconstruction threads by inserting it into their key-frame buffers. Then, the sparse reconstruction thread refines the pose estimates and adds new map points from the key-frame if necessary, and the dense reconstruction module generates a dense 3D point cloud and registers them with the existing map (see Chapter 5). Also, the sparse reconstruction module shares the map points with the tracker and those points are matched between the frames for robust pose estimation. The key-frame approach also avoids map pollution with the same frames when the camera is stationary and allows new key-frames when the camera is in motion.

The multi-threaded nature of the overall system is illustrated in Fig.4.4 in which the camera tracking module includes image pre-processing, 2D feature detection, feature matching, pose estimation stages. Four separate threads (including the dense 3D reconstruction process in Chapter 5 and the hand tracking process in Chapter 6) are running in parallel and share information via key-frames and map points. Dark and light shaded background regions represent the time steps and equal to 1/30 seconds since the cameras run at 30 fps.

Stereo images are captured via a stereo rig which is pre-calibrated as explained in Section 3.1.4 and the rectification parameters that puts the cameras into coplanar and row aligned position with the same focal length and principal points are calculated. Synchronization of the cameras for the acquisition of the stereo im-

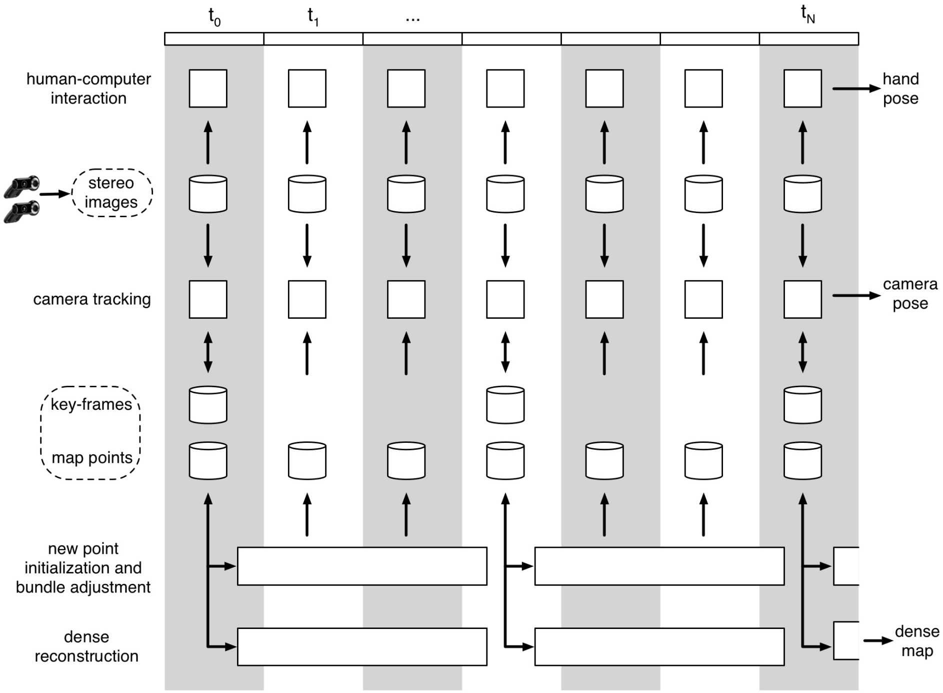


Figure 4.4: *Multi-threaded system architecture. Image pre-processing, 2D feature extraction and feature matching stages are included in the camera tracking thread. Human-computer interaction, sparse mapping and dense reconstruction modules run in parallel to camera tracking in separate threads.*

ages is achieved in software by applying a multi-threaded design, in which each camera has its own thread for image acquisition.

4.3.1 Image pre-processing

The image pre-processing step is depicted in Fig.4.5. Initially the RGB input image pair is converted into 8bpp gray-scale images. Then the intensities of the left and the right images are corrected to obtain the same mean and variance values by adjusting the right image with respect to the left image (reference image). This intensity correction step improves the stereo matching performance [171]. We update the intensity value of each pixel in the right image as

$$I_{ij}^r = \frac{\sigma^l}{\sigma^r}(I_{ij}^r - \mu^r) + \mu^l$$

where μ^l and μ^r are the image intensity means and σ^l and σ^r are the standard deviations of the left and right images respectively.

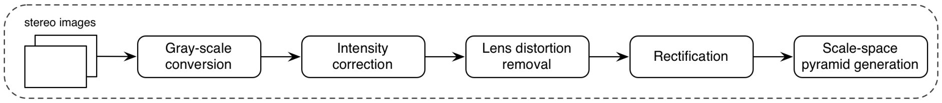


Figure 4.5: *Image preprocessing flow.*

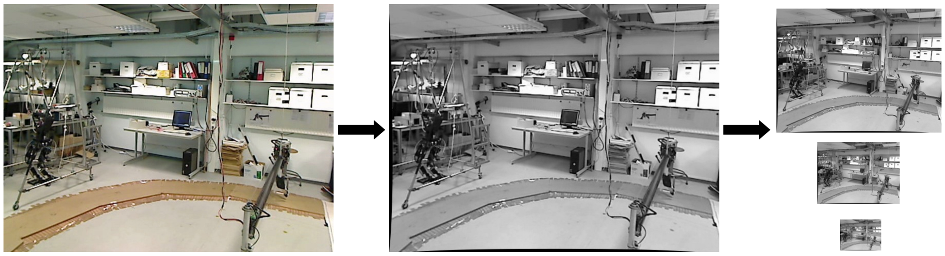


Figure 4.6: *Image preprocessing stage: input color images are converted into gray-scale images and the lens distortion is removed. Afterwards they are rectified and a scale-space pyramid of four images including the original size image is created.*

Afterwards, the radial and the tangential lens distortions are removed from the corrected input images by using the distortion model explained in Section 3.1.3. Then, the left and the right images are rectified so that they are row-aligned and the virtual rectified cameras are coplanar and have the same focal length and principal point. Therefore all tracking and mapping algorithms are performed on the undistorted and rectified images. This step makes the epipolar lines collinear and parallel to the x-axis of the images which relaxes the stereo correspondence problem by reducing it to a 1D (line) search problem.

The need for a multi-scale representation arises when dealing with images of real world scenes [112]. Since the scales of the measurements (features) are not known a priori and the features need to be extracted from images automatically, the images are analyzed at multiple scales. Moreover, the detected features at coarse scales in the multi-scale representation are strong features since they are preserved after sub-sampling and smoothing steps [112] and therefore resilient against image blur. Hence, a scale-space pyramid of four images is created by sub-sampling and smoothing the original image as shown in Fig.4.6. Combination of features detected in different scales provide resilience against fast camera motion and image blur.

4.3.2 2D feature detection

Feature detection is used as the initial step of tracking since the matching is performed between the initial models of the map points and the 2D features. Local features such as corners have been proved to be suitable for matching and recognition since they are robust to illumination changes, occlusions and background clutter. A large number of local feature detectors, such as SIFT [113] and SURF [16] exist in literature and were tested in order evaluate their performance. However, their computational load increases the time complexity of the tracker and decreases the robustness against fast camera motions. Moreover, in contrast to matching algorithms for wide-baseline images (such as reconstruction from unordered images), the camera motion between two consecutive frames is smooth and relatively small, and therefore features with high repeatability and low computational load are preferred instead of the ones with strong descriptors. Therefore, for performance reasons, ultimately the AGAST (Adaptive and Generic Accelerated Segment Test) corner detector [120] has been used to detect salient points in the scene. AGAST detector utilizes the same corner criterion as FAST (Features from Accelerated Segment Test) [164], but provides a performance increase for arbitrary environments.

In the FAST detector, the intensity values of the pixels on a discretized circle of 16 pixels surrounding the center pixel are compared to the nucleus (segment test) as shown in Fig.4.7. If there are at least 9 connected pixels (for FAST-9) that are all darker or all brighter than the center pixel by a certain threshold, then the center pixel is regarded as a corner. To perform these comparisons very fast, the FAST detector finds the best decision tree using a machine learning method on the training images captured in the application environments (and implemented as a long if-else statements). Also, a more recent version FAST-ER [165] is presented in literature. The main difference in the new version is that the thickness of the circle is increased to 3 instead of 1. The repeatability is slightly increased with respect to FAST at the cost of a higher computational time. However, both method are highly optimized for a specific training environment and has to be trained again for different scenes to provide the best performance [120]. In order to decrease the dependency on training images, the AGAST detector dynamically adapts to the environment during the operation. Two trees, one optimized for structured and another one for homogeneous regions, are built and the detector switches between two decision trees dynamically according to the detection performance.

Its speed outperforms many feature detectors ($\sim$20 times faster than Harris detector and $\sim$13% speed-up regarding FAST-9) while providing the same repeatability as FAST. Moreover it has high levels of repeatability under various transformations and different environments. Although it is sensitive to high noise compared to features such as SIFT (DoG), its high levels of repeatability and speed makes it a good selection for our purpose.

The feature detection step is depicted in Fig.4.8. In order to obtain resilience against fast camera motion and image blur, the AGAST corners are detected for every image in the pyramid. The feature detection process is performed twice both on the left and the right images of the stereo image pair. Typical features extracted from the various scenes are shown in Figure 4.9. Features extracted from different scales are shown with different colors, the level 0 features in the image pyramid (largest image) are shown in red while the level 1, 2 and 3 are shown in yellow, green and blue respectively. Non-maximum suppression is not

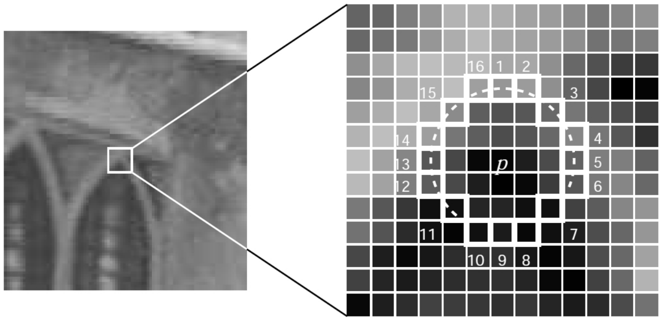


Figure 4.7: *12 point segment test corner detection in an image patch as used by Rosten and Drummond [164]. The pixels on a discretized circle of 16 pixels surrounding the center pixel are compared to the nucleus p.*

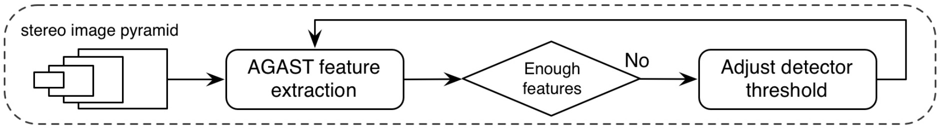


Figure 4.8: *Feature detection flow.*

performed on the detected features in order to keep the search space large so that a more precise matching can be done.

A minimum amount of features for every scale of the new image is obtained by using a variable detection threshold. Depending on the number of features detected, this threshold is decreased or increased until a fixed number of features are found. Variable thresholding guarantees a minimum number of features and increases the robustness against illumination changes and low contrast as shown in Fig.4.10.

4.3.3 Feature selection and new map-point initialization

When the system is first started or a relatively new part of the scene is explored, new map points are added to either create a new map or extend an already existing map. Feature selection and new map-point initialization step is depicted in Fig.4.11. Map points represent the 3D points corresponding to the 2D AGAST features. In order to perform robust and accurate pose estimation we expect to have an approximately similar number of map points uniformly distributed in an image.

Map points are created by using extracted AGAST features in the key-frames that are selected by the pose estimation stage when there is a significant camera motion. Before the depth estimation, a non-maximum suppression is performed on the left-image as explained in [164] in order to select the strongest feature amongst

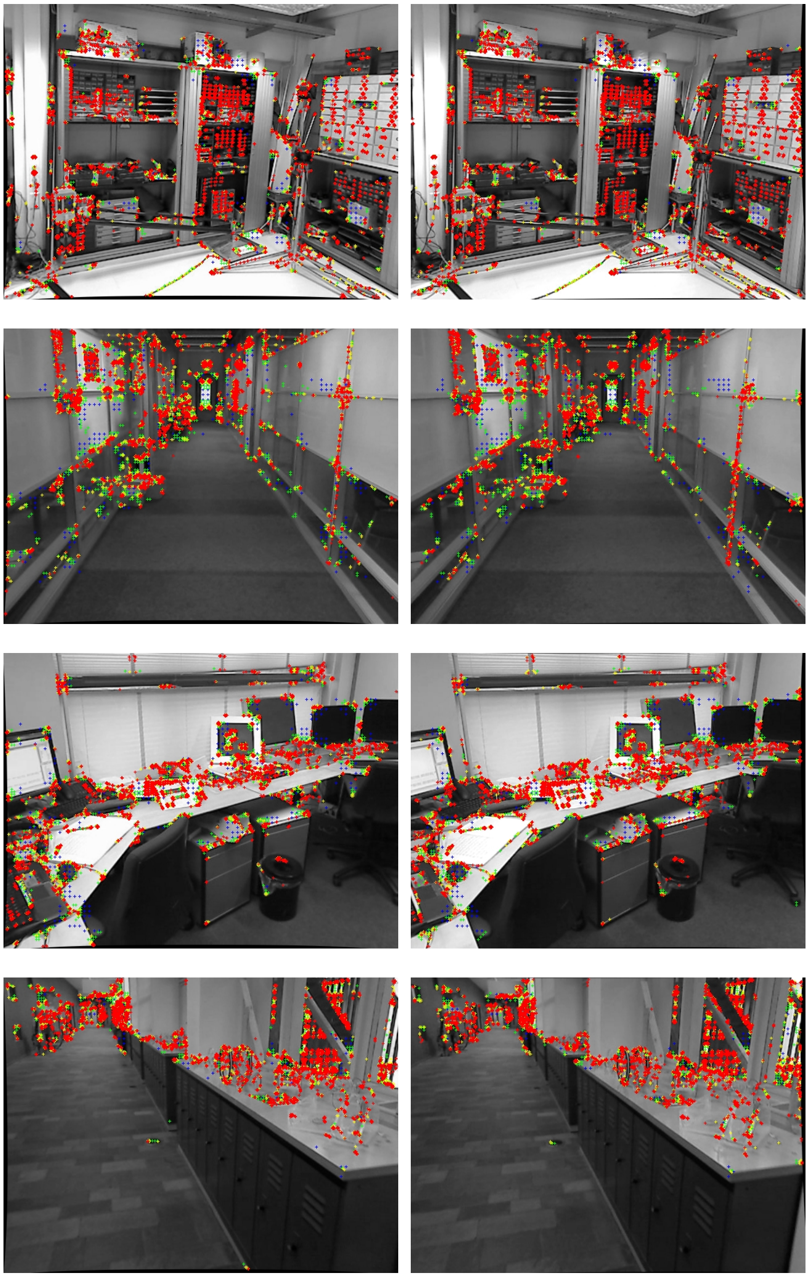


Figure 4.9: *Extracted AGAST features in multiple scales from the left and the right images, level 0 features are shown in red while the level 1, 2 and 3 are shown in yellow, green and blue respectively.*

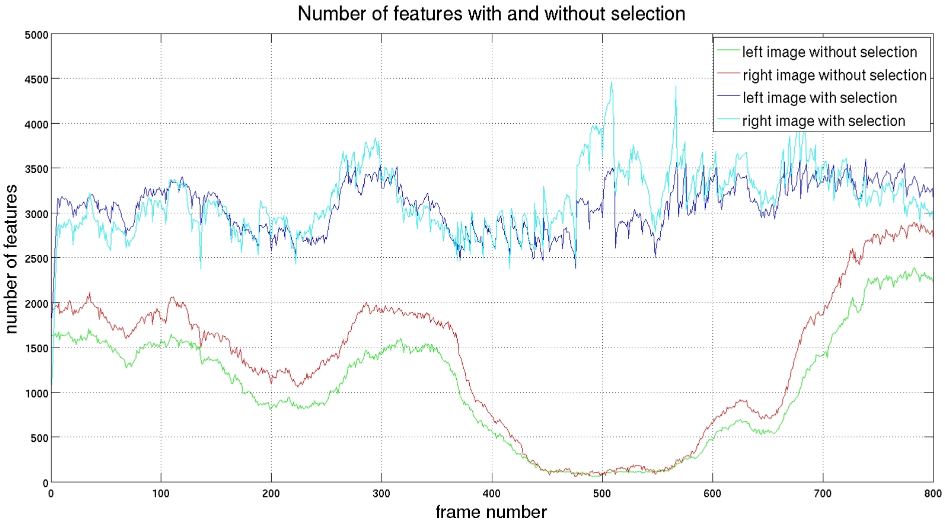


Figure 4.10: *The number of detected features for left and right images in a 800 frames sequence. The features count is stable and close to a certain number when adaptive thresholding is used.*

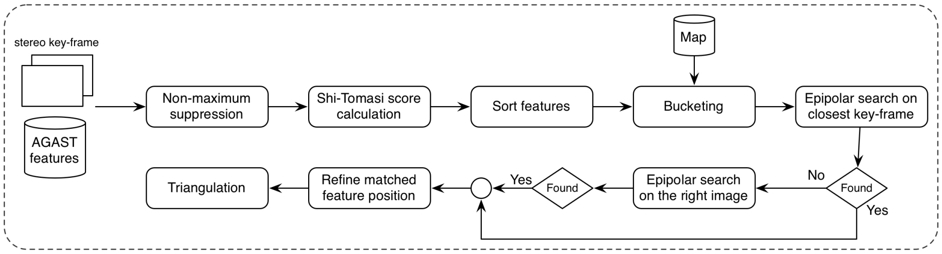

stereo key-frame
AGAST features
Non-maximum suppression
Shi-Tomasi score calculation
Sort features
Map
Bucketing
Epipolar search on closest key-frame
No
Found
Yes
Epipolar search on the right image
Found
Yes
Refine matched feature position
Triangulation

Figure 4.11: *Feature selection and new map-point initialization.*

the multiple features that are detected adjacent to one another. For each detected corner, a score function based on the sum of the absolute differences between the center pixel and the pixels on the discretized circle of 16 pixels around the center pixel is calculated. Then the corners which have an adjacent corner with higher score function are removed resulting in strong candidates for map points. This simple suppression method can be performed very fast and is therefore suitable for eliminating weak features before using a computationally more expensive but robust scoring method such as the Shi-Tomasi corner score [178]. The results of non-maximum suppression is illustrated in Fig.4.15 (a) and (b).

Since the segment test doesn't compute a corner response function, we utilize the Shi-Tomasi corner score on the selected features (features that passed the non-maximum suppression) to obtain a more global and robust score than the score function explained above. The Shi-Tomasi corner score can be calculated by using the Harris matrix [77]:

$$H = \begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_y I_x & \sum_W I_y^2 \end{bmatrix} \tag{4.1}$$

where I_x and I_y are the partial derivatives of the image I and W is the window around the corner feature. Then the Shi-Tomasi score is defined as

$$C = min(\lambda_1, \lambda_2) \tag{4.2}$$

where λ_1 and λ_2 are the eigenvalues of H.

The Shi-Tomasi corner score for each candidate is calculated and the candidates with scores smaller than some threshold are rejected. Then the candidates are sorted according to their corner score from the highest to the lowest.

A uniform distribution of features is achieved by using a bucketing technique similar to [214] as illustrated in Fig.4.12. The minimum and maximum coordinates of the left-image features are calculated and the region between these points are evenly divided into NxN buckets. Then starting from the strongest feature, each new feature is assigned to a bucket according to its 2D coordinates. When the number of assigned features exceeds the allowed number of features (T) for a bucket, the rest of the features falling into this bucket are neglected. Therefore, only the strongest first T features are used to create new map points from each bucket. Until a global threshold for a maximum number of features for all the new map points is reached, this procedure is repeated.

If there is already a map, first, existing 3D map points are projected back into the left image and assigned to the buckets. Then the new detected features are distributed into the bins as explained before, but this time with already existing initial features in the buckets. Therefore, if there is a feature-rich region with already existing map points, less (or no) new map points will be added from this region while other regions with less strong-features will be represented with more map points. To obtain a spatially uniform distribution inside the buckets, features that are close to the existing map point projections are discarded. This avoids

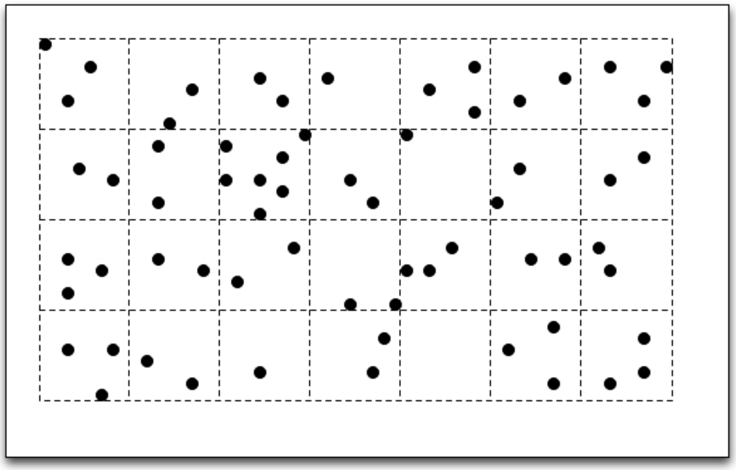
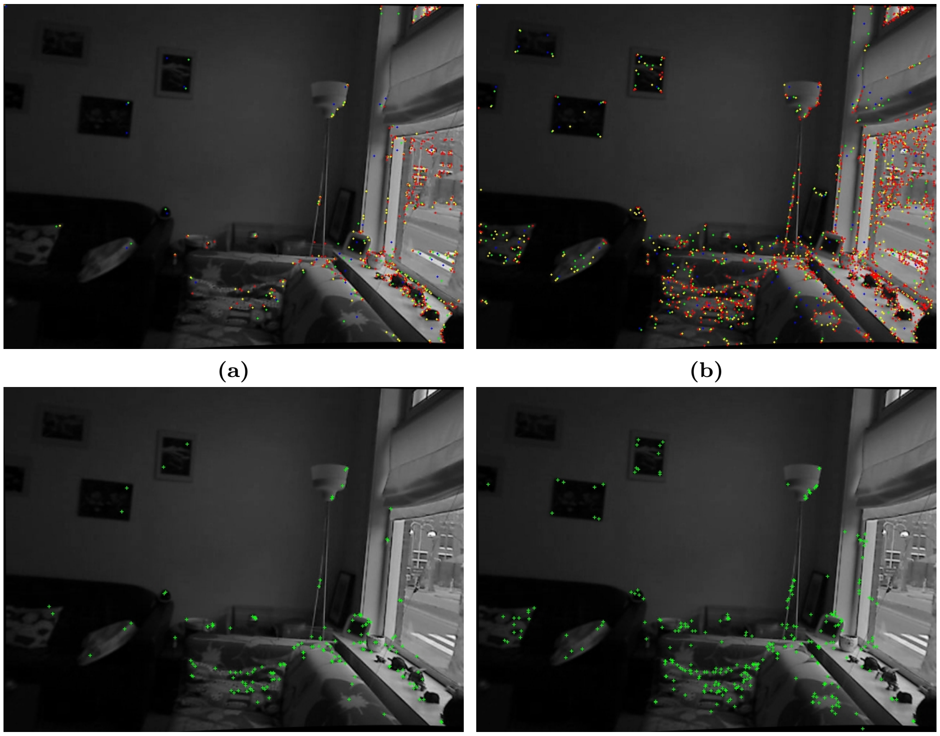


Figure 4.12: *Illustration of bucketing. The region defined by the border points is divided into buckets.*

adding new map points corresponding to the same (or very close) corners. A typical map is given in Fig.4.13. For instance, if the camera aperture is adjusted for outside light and the inside scene is comparatively dark, then low-contrast images are captured and the map points are created around the better illuminated parts of the image. However, bucketing and adaptive feature thresholding detects more features and distributes them more evenly in the image.

When the left-image features are filtered by using their corner scores and bucketing, the remaining corners are searched along the corresponding epipolar-lines first on the spatially closest key-frame to the current key-frame. Since the baseline between the two key-frames is larger (by default) than the baseline between the stereo-cameras, the depth estimation is more accurate if the matches are established correctly. Therefore, better map points can be obtained in the intersecting ROI between the two key-frames. If the matching fails then a second search is performed on the right-image.

We perform epipolar search using an error tolerance of 5-pixels for the key-frame and 1-pixel in the y-coordinates for the right-image. In order simplify epipolar search, we use the mean and standard deviation of depth of the map points visible in the image. The 3D coordinates of the two points, start and end points, are calculated as $(\mu - \sigma)\hat{n}$ and $(\mu + \sigma)\hat{n}$ where μ and σ are the mean and standard deviation of depth of the visible map points, and $\hat{n}$ is the unit vector with the direction from camera center O_l to the candidate's, C_i, normalized image coordinates (illustrated in Fig.4.14). Then their projections to the closest key-frame or the right-image are used as the search limits in the matching step. The μ and σ values are updated in each frame using the visible map-points and therefore



(a) **(b)**

(c) **(d)**

Figure 4.13: *Detected features and map points on a low-contrast image. (a) AGAST features detected with a fixed threshold and (b) adaptive threshold, (c) added map points without any selection and (d) with a selection (adaptive thresholding and bucketing). With the selection, map points are more uniformly distributed in the image.*

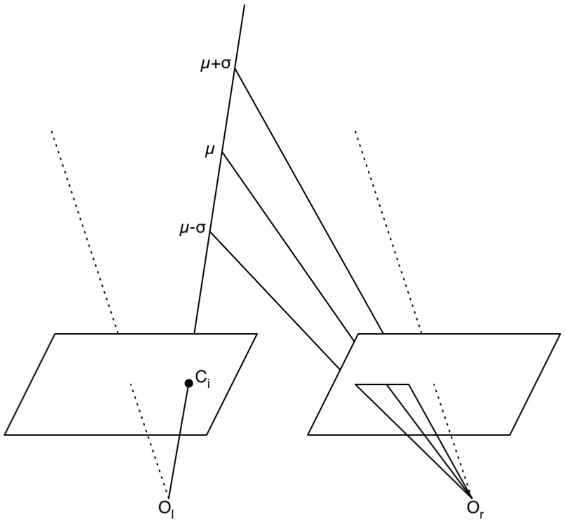


Figure 4.14: *Bounded epipolar search. The search range in the closest key-frame or the right image is decided by using the mean and standard deviation of depth of the map points visible in the image.*

the new features corresponding to the map points further from the average depth are usually discarded.

Initially, the search is done on AGAST features in the closest key-frame or the right-image. Features that are far from 5-pixel or 1-pixel (depending on the source of the second image) to the epipolar line are discarded and the search is performed on the rest. An $8x8$ patch around the searched feature is made and compared with the other patches calculated around the selected AGAST features in the second-image by using ZMSSD (Zero Mean Sum of Squared Distances) between the patches. If the smallest ZMSSD is greater than some threshold, then matching fails. If the second matching also fails for the right-image, a third dense search is performed on all the pixels along the epipolar line in the right-image. This step enables the matching of features that are not found by the AGAST feature detector on the right-image.

When the match is found, its position is refined by using the *inverse compositional approach* method explained in [14]. The refinement gives sub-pixel precision and also serves as a double check since the patches that move more than a pixel are discarded. The sub-pixel accuracy improves the depth estimation especially for the high pyramid scales in which the integer pixels are not sufficient to represent the feature position.

Finally, the 3D coordinate of the map point is calculated from matched pair by using *linear triangulation* as explained in Section 3.2.1. The 3D map points and corresponding visible 2D projections are shown in Fig.4.16. Also, when a map point is created its source images and the image coordinates are saved in order to search for the map point in the following frames.

A new map point is also searched in the previous key-frames to associate it with the previous images. This step is not crucial for the performance of the system, however obtaining more measurements of the new map point improves the quality of the map since bundle adjustment can be performed on all the measurements of the new point. Since the 3D coordinates of the map points and the pose of the key-frames are known, this search can be done by projecting the 3D map points into the key-frames and searching around the 2D projections. A similar search method is also used and explained in Section 4.3.4.

The overall method can be summarized as:

1. Perform non-maximum suppression on the detected AGAST features of the left image
2. Calculate the Shi-Tomasi score on the remaining features
3. Sort the features according to their scores and remove features with scores smaller than some threshold
4. Assign the features starting from the strongest one into buckets until either all buckets are full, maximum number of features are added or all features are assigned
5. Calculate the mean and standard deviation of depth of the visible map points, μ and σ
6. Perform epipolar search for the features in the buckets in the closest key-frame
    (a) Detect the spatially closest key-frame and calculate the epipolar line
    (b) Select all features that are 5-pixel close to the epipolar line bounded by $(\mu - \sigma)\hat{n}$ and $(\mu + \sigma)\hat{n}$
    (c) Calculate the ZMSSD for each pair
    (d) Select the best match with the smallest ZMSSD
7. If the match is not found or good enough, repeat the previous step by using the right key-frame of the stereo pair
    (a) Select all features that are 1-pixel close to the y-coordinate of the original feature and bounded by $(\mu - \sigma)\hat{n}$ and $(\mu + \sigma)\hat{n}$
    (b) Calculate the ZMSSD for each pair
    (c) Select the best match with the smallest ZMSSD

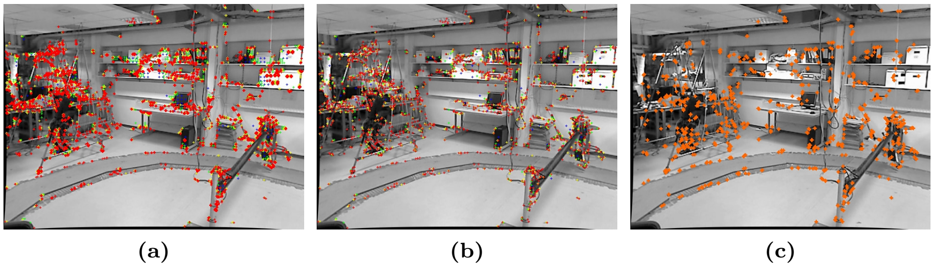

(a) (b) (c)

Figure 4.15: *(a) AGAST features and (b) features after non-maximum suppression, (c) added map points with a selection (adaptive thresholding and bucketing).*

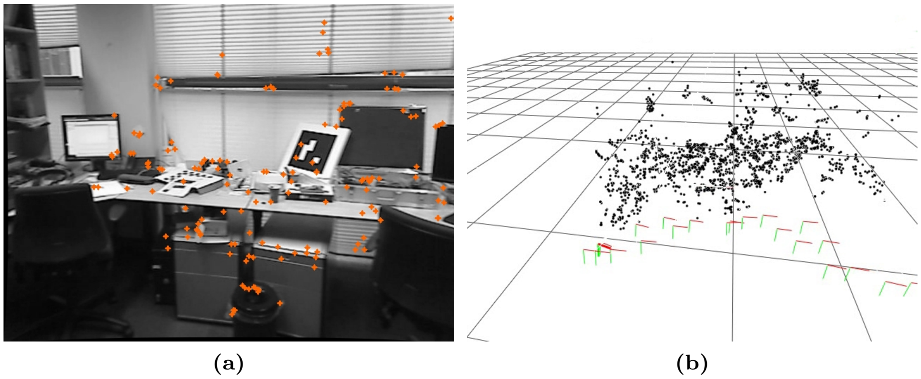

(a) (b)

Figure 4.16: *(a) Visible map points, (c) 3D map points in the map.*

   (d) If the match is not good enough repeat the previous 2 steps for all the pixels on the bounded epipolar line

8. If the match is found, refine its position to sub-pixel accuracy

9. Triangulate the point and calculate its 3D position

10. Add the new point to the map

4.3.4 Temporal Inter-frame feature matching

Temporal feature matching is performed by associating 2D image patches around the projections of a 3D map point between consecutive frames. Although this process can suffer from drift, using consecutive frames and the patches created in the previous and the current frame is highly robust against lighting changes and easier to perform due to small camera motions between the two frames. A solution to this problem is to introduce key-frames and perform matching between

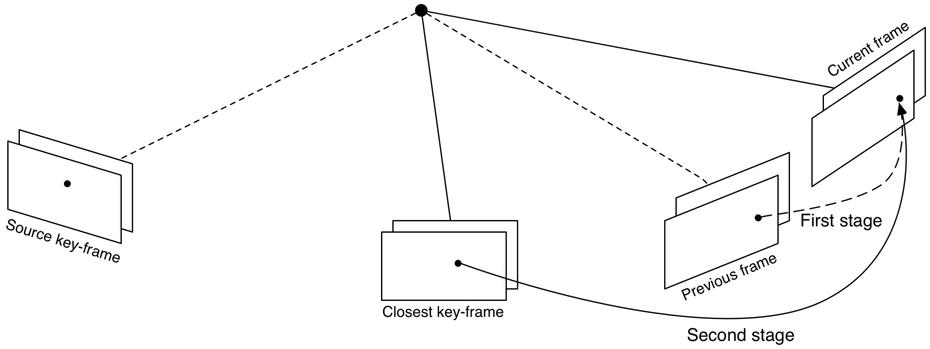


Figure 4.17: *Two stage tracking. First stage utilizes the previous frame to be robust against fast motion while the second stage uses the closest key-frame to eliminate drift.*

the patches created in one of the key-frames and the current frame. Since a key-frame and corresponding features in it are registered beforehand, it serves as an absolute reference (although there could be errors in the key-frame pose). Moreover the pose of the key-frames are refined by using bundle adjustment as explained in Section 4.3.6. A straight forward method to select a reference key-frame for a map-point is to use the source key-frame in which the point is detected the first time and created. Then the surrounding patch in the source key-frame can be used for matching in other frames for a drift-free estimate. However, this is more difficult than the matching between consecutive frames due to a (larger) viewpoint difference and a possible variation in the lighting conditions. Therefore, the patch should be modified for better matching, otherwise the initial patch fails to represent the point and the matching fails.

Selecting the closest key-frame to the last estimated location, instead of the source key-frame of the map point, can improve the performance since the viewpoint and the lighting differences are smaller. A better viewpoint invariance can be gained by warping the image patch in the key-frame to the current frame by using the last estimated location of the camera. This process creates a viewpoint corrected patch that can be used for matching via conventional methods. However, it is still less accurate due to the wider baseline compared to the consecutive frames. Also as explained in [62], by matching against the key-frames no temporal (and therefore spatial) consistency is enforced on the pose estimates which might lead to jitter.

Creating new map points in similar positions to the existing ones which are lost due to the viewpoint changes is another option to continue tracking. However, adding several map points corresponding to the same 3D point in the scene adds redundant information to the map and increases the computational load of later map refining steps (bundle adjustment).

For robust and drift-free tracking, we propose a two-stage approach that handles the large motions while not suffering much from drift as depicted in Fig.4.17. Initial frame-to-frame matching results are refined by using a second key-frame to current frame matching. This two stage tracking step is depicted in Fig.4.18.

First, the detected map points in the previous frame are projected into the current frame and used as an initial estimate for matching. Then a sparse pyramidal optical flow algorithm [21] is used to refine the initial estimates by using an $11x11$ window around the previous image and the current image projections.

The goal of the optical flow algorithm is to find the 2D displacement (or the image velocity) of the feature point p between the two images, which is the vector $\mathbf{d} = [d_x, d_y]^T$ also known as the optical flow at p. The gray-scale values of a point at $[x, y]^T$ in the previous image and the current image are given as $I_p(x, y)$ and $I_c(x, y)$ respectively. Then the optical flow $\mathbf{d}$ at point $\mathbf{p} = [u_x, u_y]^T$ in the previous image can be calculated by finding the point in the current image, $\mathbf{v} = \mathbf{u} + \mathbf{d} = [u_x + d_x, u_y + d_y]$, such that $I_p(\mathbf{u})$ and $I_c(\mathbf{v})$ are similar. This can be done by minimizing the residual function

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} (I_c(x, y) - I_p(x + d_x, y + d_y))^2 \tag{4.3}$$

where w defines a $w_x x w_y$ neighborhood of the point p. In our experiments we set the $w_x = 5$ and $w_y = 5$.

A standard iterative Lucas-Kanade method [115] can be utilized for this minimization. Applying Taylor series expansion around $\mathbf{u}+\mathbf{d}$ to $I_p(\mathbf{u}+\mathbf{d}) = I_p(u_x + d_x, u_y + d_y)$ and neglecting the higher order terms we obtain

$$\epsilon(\mathbf{d}) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} ((I_c(x, y) - I_p(x+d_x, y+d_y)) + \nabla I_p(x+d_x, y+d_y)^T \mathbf{\Delta d})^2 \tag{4.4}$$

where $\nabla I_p(x + d_x, y + d_y) = \left[\frac{\partial I_p}{\partial x}, \frac{\partial I_p}{\partial y}\right]^T$ is the image gradient at $(x + d_x, y + d_y)$.

If we write the above equation as

$$\epsilon(\mathbf{d}) = \sum_W (I_i^t + \nabla I_p(\mathbf{x_i} + \mathbf{d})^T \mathbf{\Delta d})^2 \tag{4.5}$$

where $I_i^t = I_c(\mathbf{x_i}) - I_p(\mathbf{x_i} + \mathbf{d})$, then this least-squares problem can be minimized by solving the associated normal equations

$$\mathbf{A \Delta d} = \mathbf{b} \tag{4.6}$$

where $\mathbf{A} = \sum_W \nabla I_p(\mathbf{x_i} + \mathbf{d}) \nabla I_p(\mathbf{x_i} + \mathbf{d})^T$ and $\mathbf{b} = \sum_W I_i^t \nabla I_p(\mathbf{x_i} + \mathbf{d})$ and updating $\mathbf{d} = \mathbf{d} + \mathbf{\Delta d}$ iteratively.

The pyramidal optical flow algorithm [21] performs the standard optical flow algorithm in multiple scales and therefore is able to handle large pixel motions

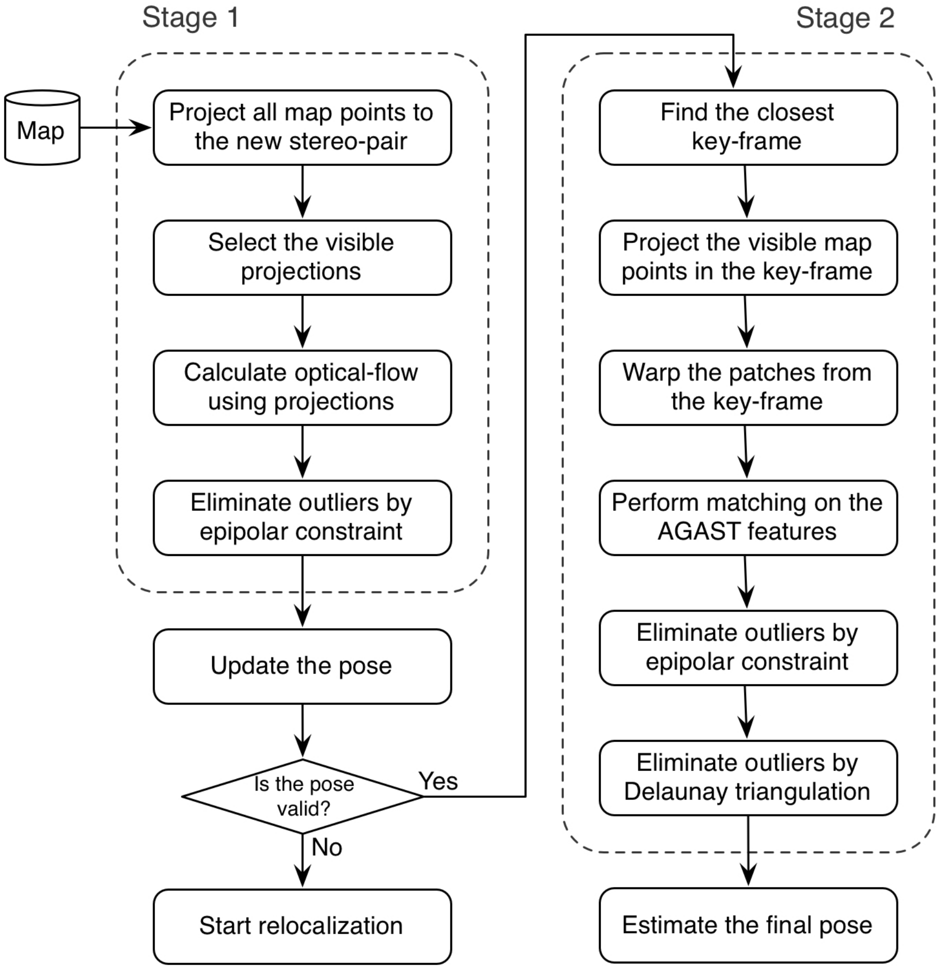

Stage 1
Stage 2
Map
Project all map points to the new stereo-pair
Select the visible projections
Calculate optical-flow using projections
Eliminate outliers by epipolar constraint
Update the pose
Is the pose valid?
Yes
No
Start relocalization
Find the closest key-frame
Project the visible map points in the key-frame
Warp the patches from the key-frame
Perform matching on the AGAST features
Eliminate outliers by epipolar constraint
Eliminate outliers by Delaunay triangulation
Estimate the final pose

Figure 4.18: *Two stage tracking flow.*

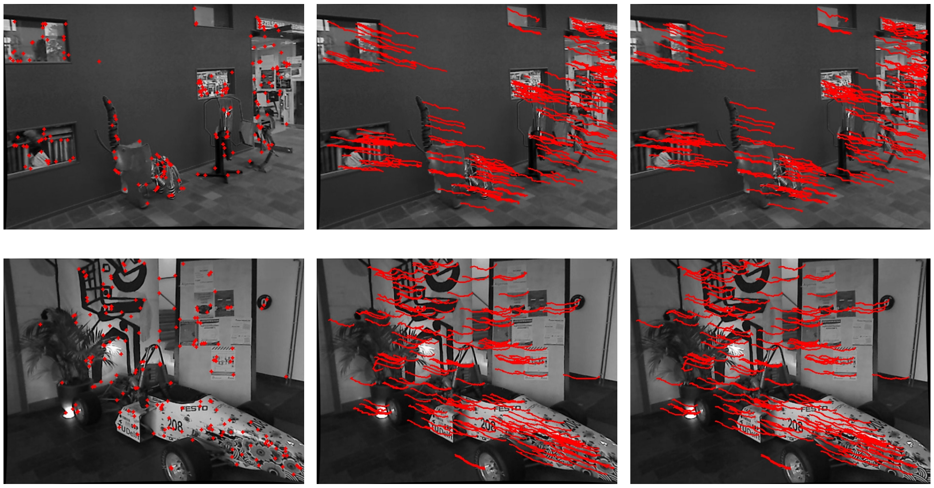


Figure 4.19: *The first stage of the temporal inter-frame feature matching. The selected map points (shown in the left images) for two datasets are tracked in the left and the right images (the middle and the right images) using the previous images and the projections of the map points. Also the outliers that are violating the epipolar constraint are removed. The trajectories of the tracked map points in both left and right images are shown in red in the middle and the right images respectively.*

(larger than the integration window sizes w_x and w_y). Initially the optical flow is calculated in the highest pyramid level and the estimated flow is propagated to the lower scale as an initial guess for the pixel displacement. This is repeated until the lowest level (the original image) is reached. Also for the highest level we use the back-projections of the map points as the initial guess for the flow.

The clear advantage of using the pyramidal implementation is that the large pixel displacement can be calculated while keeping the size of the integration window relatively small. Moreover, the image pyramid calculated for feature extraction is directly used for tracking. This is repeated for both the left and the right images. If the found positions of the same map point in both images satisfy the epipolar constraint,

$$abs((\mathbf{x_l} + \mathbf{u_l})[0,1]^T - (\mathbf{x_r} + \mathbf{u_r})[0,1]^T) < e_{th} \tag{4.7}$$

then the matching is successful and the point is labeled as an inlier. The tracking examples with the selected inliers are illustrated in Fig.4.19. Finally, the pose of the camera is estimated using the inliers via robust estimation as explained in Section 4.3.5.

As the second stage, the pose obtained from the first stage and the closest key-frame to it are used to eliminate the drift. The map points that are visible in the closest key-frame are projected into the current image by using the new estimate of the pose. Then the 2D AGAST features in the close proximity of the projections are evaluated to find the best match (in the sense of photo-consistency) with the corresponding point's model. We only consider the 2D AGAST features that are closer than 10 pixels to the projections as the candidates for matching. This time in contrast to the optical flow tracker, we perform matching over the corner features.

As a descriptor we use affine warped patches around the projections. In the affine model, each pixel in the patch $W(x)$ undergoes the same transformation such that

$$h(x) = \mathbf{A}x + \mathbf{d}$$

where $\mathbf{A} \in \mathbb{R}^{2x2}$, $\mathbf{d} \in \mathbb{R}^2$ and $\forall x \in W(x')$.

The parameters of the affine model are calculated by using the projections p and p' from the closest key-frame and the current image [93]. One-pixel right (u_s) and one-pixel down (v_s) at the source level l (original level that the feature is detected) of the projection is first projected to level 0, and denoted by

$$p_{OR} = p + 2^l[1,0]^T \text{ and } p_{OD} = p + 2^l[0,1]^T$$

respectively as illustrated in Fig.4.20. Then these two points are back-projected onto the plane passing through the map point P with the surface normal $\frac{P-C_{kf}}{\|P-C_{kf}\|}$ rotated relative to the key-frame, and denoted by P_{OR} and P_{OD}.

Finally, these two points are projected onto the current image plane and the difference between the one-pixel right and one-pixel down points are calculated as $u_c = p'_{OR} - p'$ and $v_c = p'_{OD} - p'$. Then the affine matrix (from key-frame to the current image) $\mathbf{A}$ and $\mathbf{d}$ are [93]

$$\mathbf{A} = \begin{bmatrix} \frac{\partial u_c}{\partial u_s} & \frac{\partial u_c}{\partial v_s} \\ \frac{\partial v_c}{\partial u_s} & \frac{\partial v_c}{\partial v_s} \end{bmatrix} \quad \text{and} \quad \mathbf{d} = p' - p$$

where the matrix $\mathbf{A}$ is the same for both stereo images.

In order to preserve scale invariance, projections are done from the detected pyramid level S_n. Although searching a map point in the same pyramid level that was detected in the closest key-frame, seems a reasonable approach, large camera motions will change the scale of the feature and the affine matrix, which results in sampling errors. To cope with that, we select the search pyramid level l as explained in [20, 93] such that

$$\frac{det(\mathbf{A})}{2^{2l}} \tag{4.8}$$

is closest to unity. $det(\mathbf{A})$ corresponds to the area of the source pixel (in square pixels) in the level 0 image. Therefore, selecting the level that makes the Eq.4.8

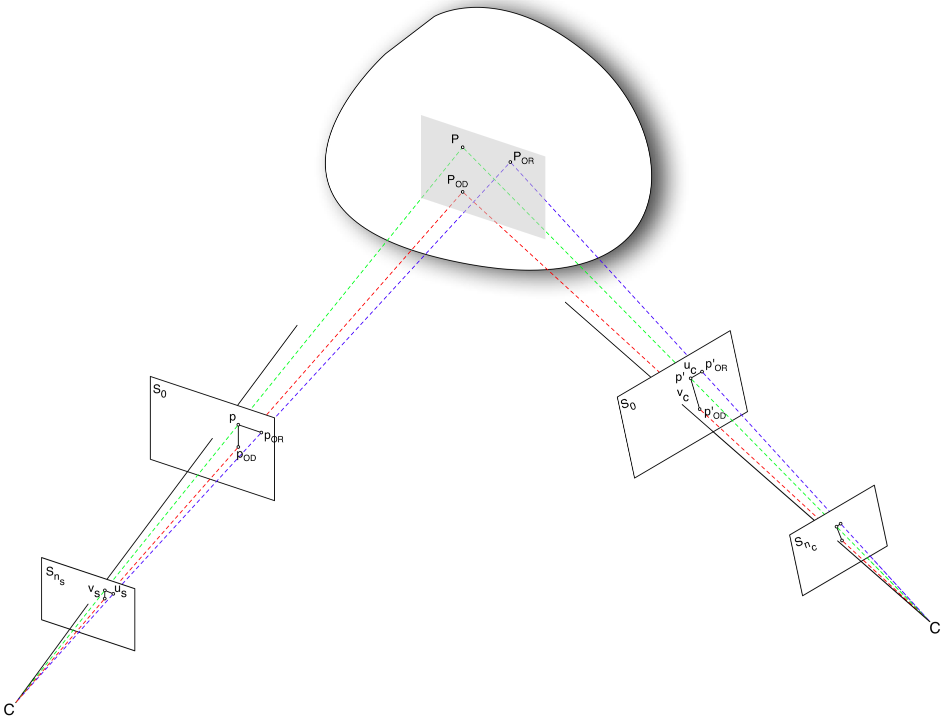


Figure 4.20: *Affine transformation between two frames corresponding to a map point is calculated by reprojecting one right and one down pixels in the source coordinate frame to the current frame.*

closest to unity results in the patch that most closely matches the scale of the original patch.

Then the image patch in the key-frame is warped by using $\mathbf{A}/2^l$ and compared with the patches around the image features in the current image and the feature with the minimum ZMSSD is selected as the match. The earlier explained second stage is also repeated for the right image and false matches are eliminated using the epipolar constraint. The results of the second stage tracking are illustrated in Fig.4.21. The locations of the map points tracked by the first stage are shown in red while their corrected locations with the second stage tracker are shown in blue. The displacements between the results of the two stages are shown in green.

As a final clean up, a Delaunay triangulation [177] is utilized to detect sporadic outliers. The triangulation is performed on the found image coordinates of the map points after the second stage of the tracking. When the triangulation is performed, each feature is expected to have minimum 2 neighbors with similar displacement. The displacement is calculated as the distance between the pro-

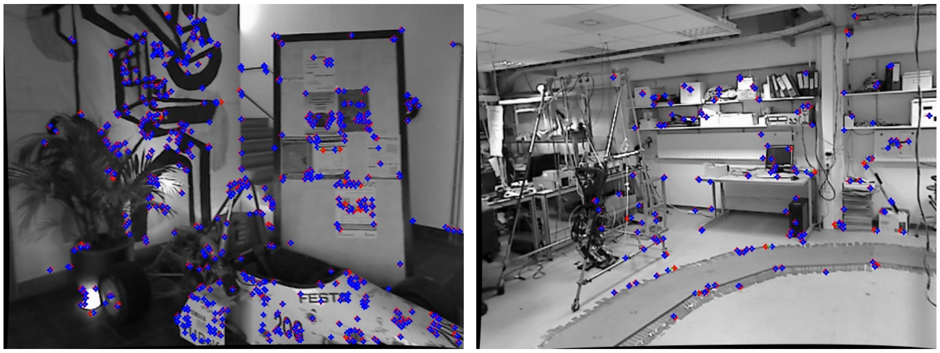


Figure 4.21: *The second stage of the temporal inter-frame feature matching. The locations of the map points tracked by the first stage are shown in red while their corrected locations with the second stage tracker are shown in blue. The displacements between the results of the two stages are shown in green.*

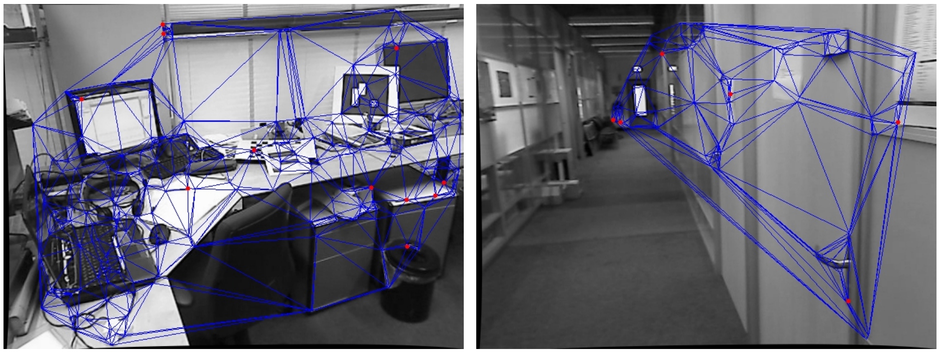


Figure 4.22: *The Delaunay filtering examples in two images. The outliers with less than 2 neighbors with similar flow are shown in red.*

jected image coordinate and the converged tracking patch. If this condition is not satisfied then the point is discarded as an outlier. The Delaunay filtering is illustrated in Fig.4.22. In our experiments two neighbors have similar flow if the difference between their displacements is less then 5 pixels. The outliers with less than 2 neighbors with similar flow are shown in red. When all matches are ready, a final pose estimation is done.

In order to be resilient against fast camera motion and image blur, when performing in real time, we track a maximum number of $N = 1000$ features in each frame. The features are selected starting from the highest pyramid level down to the lowest pyramid level.

4.3.5 Pose Estimation

The pose of the camera needs to be estimated in each incoming video frame and the pose estimation is performed after the first and the second tracking stages separately. The parameterization of the pose has an impact on the behavior of the estimation process and its stability. As explained in [193], the most suitable parameterizations for the pose are more uniform and well-behaved (locally close to linear) near the current state estimate and the pose should be parameterized using a local incremental pose update Δg to the existing pose g, where Δg can be any well-behaved special Euclidean transformation. Therefore, in each frame the new pose of the camera is updated by multiplying the current pose of the camera with a small motion matrix.

Given the camera pose $g_t = g(R,T) \in SE(3)$, the new pose is calculated by using the camera pose update, a small motion matrix, $U_t \in SE(3)$ as

$$g_{t+1} = U_t g_t$$

In order to avoid numerical problems such as singularities (e.g gimbal lock) we used an exponential twist to parameterize U. The motion U is also a member of the Lie Group SE(3) and can be minimally parameterized with the twist coordinates $\xi = [v,w]^T \in \mathbb{R}^6$ by using the exponential map under exponential coordinates such that

$$U = exp(\hat{\xi}) \tag{4.9}$$

where v is the linear velocity and w is the angular velocity. With this representation, the partial derivatives of the motion matrix with respect to the motion parameters can easily be calculated for a small (infinitesimal) camera motion (see Section 2.1.1 for more detail).

This motion matrix between the consecutive frames can be estimated by using the projections of the 3D map points and their corresponding 2D image coordinates found by the 3D-2D feature matching explained in the previous section. This method is also known as PnP (Perspective-n-Point), recovering the pose from n 3D-2D correspondences. It is based on the minimization of the reprojection error in the images.

In our system we minimize the reprojection error in both images of the stereo pair. The pose of the left and the right cameras can be denoted as

$$g^l_{cw} = g(R,T) \quad \text{and} \quad g^r_{cw} = g(R, T - [b,0,0]^T)$$

which are special Euclidean transformations from the world to the left and right camera coordinate frames respectively and b is the baseline between the stereo cameras.

Then, the projection of a map point $\tilde{X}_w = [X,Y,Z,1]^T$ in the world coordinate frame to the camera image planes can be given

$$x_l = \pi(g^l_{cw}\tilde{X}_w) \quad \text{and} \quad x_r = \pi(g^r_{cw}\tilde{X}_w)$$

where $\pi()$ represents the projection from camera frame to the image coordinates as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \pi \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & c_u \\ 0 & f & c_v \end{bmatrix} \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix} \tag{4.10}$$

assuming zero skew and square pixels (explained in more detail in Section 2.1.2). The effect of lens distortion is not included in the projection formulation since we are using rectified images with corrected lens distortion, and projections for both cameras are identical due to the same reason.

Finally, the reprojection error in both views can be defined as

$$\begin{aligned} \epsilon(\xi) &= \sum_i \|\epsilon_l^i(\xi)\|^2 + \|\epsilon_r^i(\xi)\|^2 && (4.11) \\ &= \|\hat{x}_l^i - x_l^i\|^2 + \|\hat{x}_r^i - x_r^i\|^2 && (4.12) \\ &= \sum_i \|\hat{x}_l^i - \pi(exp(\hat{\xi}) g_{cw}^l \tilde{X_w^i})\|^2 + \\ &\quad \|\hat{x}_r^i - \pi(exp(\hat{\xi}) g_{cw}^r \tilde{X_w^i})\|^2 \end{aligned}$$

where $\hat{x}_l^i$ and $\hat{x}_r^i$ are the detected feature locations of the 3D map point X_w^i in the current left and right images respectively.

In the simplest case this cost function can be minimized iteratively by using the Gauss-Newton method and finding the pose update $\Delta\xi$ by solving the normal equations

$$\mathbf{J}^T \mathbf{J} \Delta\xi = \mathbf{J}^T \epsilon$$

where the Jacobian J_i for the errors $\epsilon_l^i(\xi)$ and $\epsilon_r^i(\xi)$ corresponding to the map point X_w^i is calculated as

$$J_i = f \begin{bmatrix} 1/Z_l & 0 & -X_l/Z_l^2 & -X_l Y_l/Z_l^2 & 1 + X_l^2/Z_l^2 & -Y_l/Z_l \\ 0 & 1/Z_l & -Y_l/Z_l^2 & -1 - Y_l^2/Z_l^2 & X_l Y_l/Z_l^2 & X_l/Z_l \\ 1/Z_r & 0 & -X_r/Z_r^2 & -X_r Y_r/Z_r^2 & 1 + X_r X_l/Z_r^2 & -Y_r/Z_r \\ 0 & 1/Z_r & -Y_r/Z_r^2 & -1 - Y_r^2/Z_r^2 & X_l Y_r/Z_r^2 & X_l/Z_r \end{bmatrix} \tag{4.13}$$

where $X_l^i = [X_l, Y_l, Z_l]^T = g_{cw}^l \tilde{X}_w$ and $X_r^i = [X_r, Y_r, Z_r]^T = g_{cw}^r \tilde{X}_w$ are evaluated at the last calculated pose and therefore $X_r^i = X_l^i + [b, 0, 0]^T$, $Y_r^i = Y_l^i$, $Z_r^i = Z_l^i$ and the parameter updates are given as

$$\Delta\xi = [v_1, v_2, v_3, w_1, w_2, w_3]^T$$

However, as explained in the Section 3.3.3, outliers due to incorrect matches violate the Gaussian noise assumption and cause incorrect pose estimates. In order to reduce the sensitivity to the outliers we have utilized a robust estimator

(M-estimator) [83] in our least-squares minimization scheme. The M-estimate of ξ is

$$\tilde{\xi} = \arg\min_{\xi} \sum_i \rho(\epsilon_l^i(\xi)/s_l) + \rho(\epsilon_r^i(\xi)/s_r)$$

where $\rho(r)$ is a robust loss function. We have selected Tukey's bi-weight function since it is one of the most aggressive robust loss functions against the outliers. Tukey's function completely rejects outliers by giving them zero weight and therefore detected outliers have no effect on the camera motion estimation. We selected the minimum of both median of the errors for more aggressive rejection in the calculation of a robust standard deviation estimate of the errors

$$\sigma = min\left(\operatorname*{median}_i \epsilon_l^i(\xi), \operatorname*{median}_i \epsilon_r^i(\xi)\right)$$

As explained in Section 3.3.3, minimizing this function is equivalent to minimizing the iteratively re-weighted least squares problem

$$\sum_i w_l^i \|\epsilon_l^i\|^2 + w_r^i \|\epsilon_r^i\|^2$$

where $w_{l,r}^i$ are calculated at the current estimate of ξ. In our system we do 10 (Gauss-Newton) iterations during minimization.

Quantization errors in the pixel positions at higher scales dominate the reprojection errors and therefore the estimates. In order to cope with that, each measurement is normalized with its scale such that the new error is

$$\hat{\epsilon}^i = \frac{\epsilon^i}{2^{2s}}$$

where s is the level in the image pyramid.

One of the problems with the M-estimators is its dependence on the accurate initial estimates and sensitivity to a large number of outliers (maximum approximately % 20) [100]. First a *decaying velocity model* [93] is used to predict the possible new pose of the camera. The decaying velocity of the camera is found by using the previous velocity ξ_p and the current motion of the camera U. The current motion of the camera between the previous and current frames U is given in equation (4.9). Then the twist coordinates corresponding to this motion can be calculated as

$$\xi_{cm} = ln(U) = ln(g_{t+1} g_t^{-1})$$

Then the velocity, ξ_c of the camera for the next frame is

$$\xi_c = \alpha(\xi_p/\beta + \xi_{cm}/\beta)$$

We use $\alpha = 0.9$ and $\beta = 2$ and since we multiply average velocity by the constant α, the motion of the camera decays to zero quickly. This model imposes a certain smoothness on the motion and thus is fragile against rapid head motions.

Table 4.1: *Pose estimation algorithm.*

(i)	Estimate the camera velocity by using a *decaying velocity model* and the previous motion of the camera.
(ii)	Compute the initial pose by using a velocity estimate and the previous pose.
(iii)	Refine the initial pose estimate by using the combination of RANSAC and EPnP algorithms, and select the inliers.
(iv)	Estimate the final pose by minimizing the reprojection errors of inliers in both views by using a robust estimator.

On top of that, a RANSAC scheme (see Section 3.2) is used to obtain an accurate initial estimate before performing a robust estimation. During the RANSAC iterations we use a non-iterative PnP algorithm, the EPnP algorithm [110], whose computational complexity grows linearly with the number of points used for estimation. Its main idea is based on representing the coordinates of the n 3D points as a weighted sum of four virtual control points and estimating the coordinates of these control points in the camera referential which can be done in $O(n)$ time. This method is an effective choice because it is very fast for small numbers of points and therefore can easily be executed in the RANSAC iterations. The overall method can be summarized as:

1. Randomly select $N = 7$ samples from the 3D-2D matches.
2. Estimate the pose by using the EPnP algorithm.
3. Determine the number of inliers that are within a distance threshold T of the pose and calculate the reprojection error using the new estimate.
4. Repeat the step 1, 2 and 3 until the number of inliers is greater than a certain threshold or maximum number of attempts has been made.
5. Select the best estimate that gives the largest consensus-set (inliers), and reestimate the pose with the inliers.

When the inliers are selected, the robust estimation can be done safely. The overall method is summarized in Table.4.1 and estimated poses are illustrated in Fig.4.23.

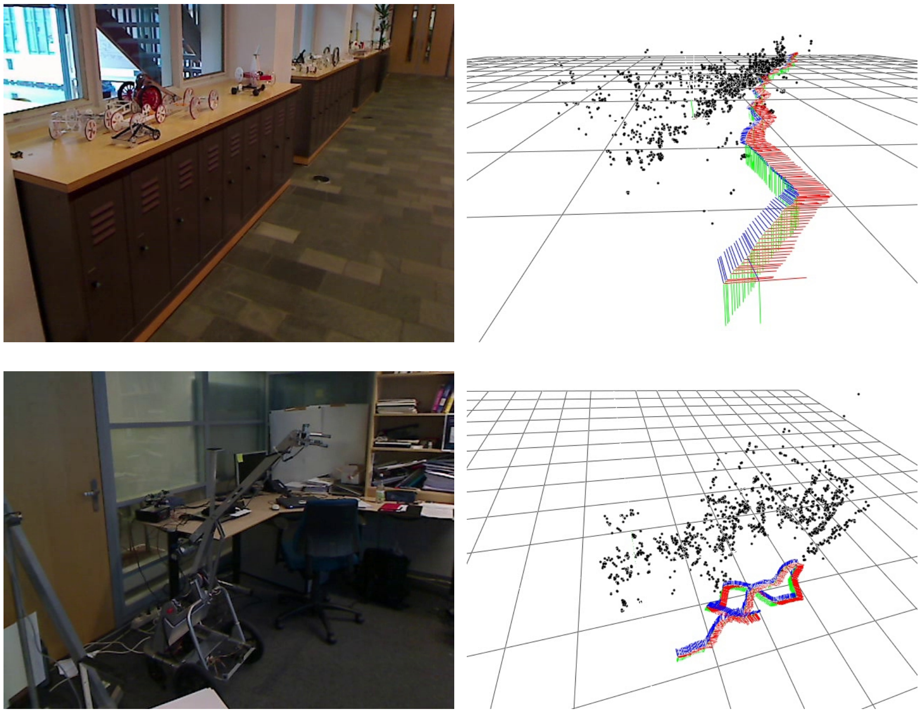


Figure 4.23: *The pose estimation examples. On the left side the scene is shown. On the right side the map points and the estimated camera poses for each input frame are illustrated. X, Y and Z axis of the camera are represented in red, green and blue respectively.*

Relocalization

When the tracking fails, a relocalization is done by extracting the features in the current image and matching them against the points in the map. A non-maximum suppression is performed on the current frame features in order to limit the number of features to a given maximal number by selecting the strongest ones. Then the image patches around the features (without any prior affine-warp) are compared and the key-frame with the maximum number of matches is selected as the closest key-frame. This matching stage is very similar to the second stage of the tracking except no warping is done in relocalization. Then the initial pose is estimated by using RANSAC and refined by using a non-linear robust estimation as explained in this section. When the pose is found, the temporal inter-frame feature matching continues to track the map points.

4.3.6 Sparse Mapping

The sparse map encapsulates the key-frames and the measured 3D map points relative to a global coordinate frame. Overall sparse mapping is depicted in Fig.4.24. We have designed a two stage method to define the world coordinate frame. When the first map points are created as explained in Section 4.3.3, a predefined marker is searched in the key-frames. If the marker is detected, then the world coordinate frame is aligned with the marker such that the origin will be on the marker and the z-axis will be the surface normal of it. Since we use the marker only for the coordinate frame alignment, we didn't consider the presence of multiple markers and if there are more than one markers exist in the scene, our systems picks the one that is detected first. If no marker is detected, then the major plane in the scene is detected as the world coordinate frame. First the inliers for the major plane are detected using RANSAC: many plane hypotheses calculated via randomly selected 3 points are evaluated by the remaining points, and the points voted for the best plane are selected as inliers. Then given a set of m points in 3D belonging to the plane, a direct solution for the surface normal n can be found by solving

$$\sum_{i=0}^{k}(\mathbf{p}_i - \bar{\mathbf{p}})^T(\mathbf{p}_i - \bar{\mathbf{p}}) \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \lambda \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$$

$$A\mathbf{n} = \lambda \mathbf{n}$$

where $\bar{\mathbf{p}}$ is the mean of all inliers which is also used as the origin of the world coordinate frame. The minimum solution corresponding to the surface normal is given by the eigenvector of A corresponding to its minimum eigenvalue. Finally, all the map points are rotated and translated from initial key-frame coordinates to the world coordinate frame.

In the absence of 3D map points with pre-known coordinates, as in extensible tracking, error-buildup (drift) in the map is inevitable, which results in tracking

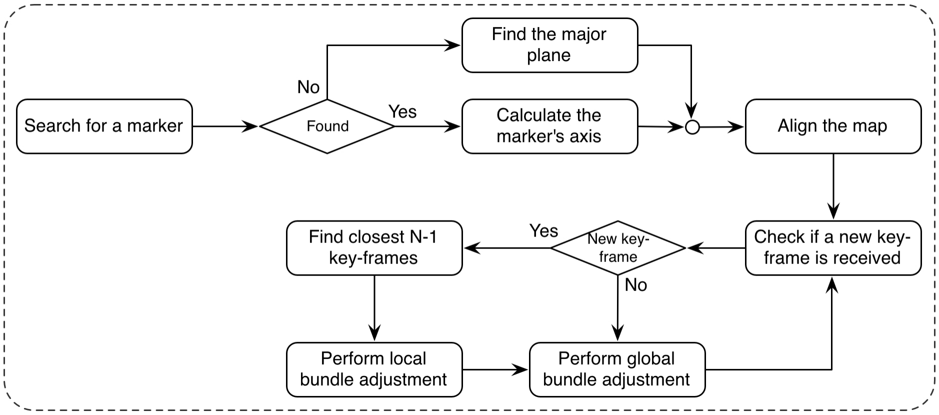

Find the major plane
No
Search for a marker
Found
Yes
Calculate the marker's axis
Align the map
Find closest N-1 key-frames
Yes
New key-frame
Check if a new key-frame is received
No
Perform local bundle adjustment
Perform global bundle adjustment

Figure 4.24: *Sparse mapping flow.*

failure. This accumulation of errors can be corrected by using bundle adjustment. Bundle adjustment improves the accuracy of the map points and key-frame poses, and performing it sequentially for each new key-frame can prevent tracking failures since the future estimates will be improved [56].

Bundle adjustment is based on the minimization of the error function

$$\{\tilde{\xi}^2, \tilde{\xi}^3, ..., \tilde{\xi}^i, \tilde{M}^1, \tilde{M}^2, ..., \tilde{M}^j\} = \arg\min_{\xi,\mathbf{M}} \sum_i \sum_j \rho(\epsilon^{ij})$$

with respect to all the 3D map points M^j, and all the key-frame poses $\xi^i \in SE(3)$ except the reference (initial) frame. Since we use a calibrated setup, intrinsic camera parameters don't need to be adjusted. $\rho(r)$ is the same robust loss function as explained in Section 4.3.5 and ϵ^{ij} is the reprojection error of map point M^j in the key-frame ξ^i defined as

$$\epsilon^{ij} = \|x^{ij} - \pi(exp(\xi^i) g^i_{cw} \widetilde{M^j_w})\|^2$$

where x^{ij} is the found position of the jth map point in the ith key-frame. This error function can be minimized using the Levenberg-Marquardt algorithm (Section 3.3.2). We used the bundle adjustment implementation in [93] which follows [79].

This bundle adjuster exploits the sparsity which arises because each map point is usually not visible in every key-frame. Therefore the reprojection errors of these points do not exist in the key-frames where they are not visible and thus the time performance of the adjuster is improved. As we have mentioned before, the bundle adjustment has a complexity that is linear in the number of landmarks and cubic in the number of poses $O(N_c^3)$. When the map scale is small and the number of key-frames is relatively smaller than the number of map points, the complexity $O(N_c^3)$ is dominated by $O(N_pN_c^2)$ until N_c approaches N_p.

Bundle adjustment of as many key-frames as possible in backwards direction, starting from the latest key-frame is the most desirable, but also computationally very expensive. In order to decrease the computational load, we use a local bundle adjustment when a new key-frame is added and a global one when the camera is relatively still.

For local adjustment, we perform n iterations of bundle adjustment over the N key-frames when a new key-frame I_t is added. N key-frames are selected as the last key-frame added and the spatially closest $N-1$ frames to it as illustrated in Fig.4.25. The rest of the key-frames are locked down and not adjusted. Thus, only the key-frame poses $\xi = \{\xi_t^1, .., \xi_t^N, \}$ and the map point set M^t which contains all the map-points that are visible in the selected key-frames are refined. The cost function is the sum of reprojection errors of points M^i in the N key-frames ξ_j

$$\sum_{\xi_i \in \{\xi_t^1, .., \xi_t^N\}} \sum_{M^j \in M^t} \rho(\epsilon^{ij})$$

The complexity of the local bundle adjustment is $O(N_pN_c)$. In order to limit the computational time we perform bundle adjustment only on the key-frames coming from the left camera and update the measurements on the right camera by using the refined pose of the left camera and the known baseline between the cameras.

4.4 Results

In our experiments we utilized $640x480$ images and our system was tested on the hardware explained in Chapter 2: a laptop with a 2.7 GHz Core i7 processor (Intel Corporation, USA) and a Linux operating system.

Various test scenarios in indoor scenes which demonstrate typical investigation procedures such as exploring the room with changing illumination conditions, coming closer to salient regions etc., were recorded and tested by using the algorithms described in the previous sections. The main objective of our experiments is to measure the computational time of the designed modules and compare the performance of our overall visual odometry algorithm with a marker based pose estimation algorithm.

4.4.1 Computational Time

In order to measure the computational times of the designed modules and different stages of the tracker, we have tested our algorithms on two different data sets.

In the first dataset, the user explores a relatively large room by walking around the room. The illumination conditions vary inside the room due to the brighter

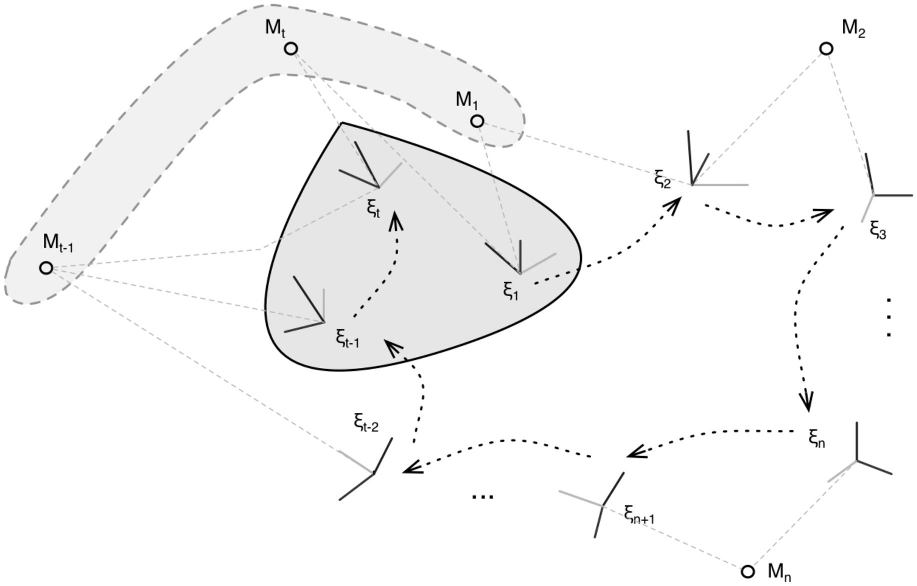


Figure 4.25: *The keyframes are represented by* ξ_i *and the map points are shown with* M_i*. As the camera moves, new map points are created. When a new key-frame* ξ_t *is added, the* $N-1$ *closest key-frames (*$N = 3$ *in this figure) and the visible map-points from these* N *keyframes (shown with the gray area) are adjusted using local bundle adjustment.*

outside and large windows. 39 key-frames are created during operation and used to create new map points. In the second dataset, the user moves along a long corridor and comes back while exploring the scene by coming closer and further to the furniture around the corridor and the walls surrounding it. 102 key-frames are created during operation and were used to create new map points, since the corridor is spatially larger relative to the first dataset.

In Fig.4.26 and Fig.4.27, the computational times of all modules are given for the test datasets. Fig.4.26 (a) and Fig.4.27 (a) give the computational times of the image preprocessing, feature extraction and Delaunay elimination modules. Fig.4.26 (b) and Fig.4.27 (b) show the number of map points created during the operation. In Fig.4.26 (c) and Fig.4.27 (c), the stage 1 and stage 2 tracking and pose estimation times are given. Finally, Fig.4.26 (d) and Fig.4.27 (d) give the computational time of the complete visual odometry module and tracking modules.

Image preprocessing and feature extraction take approximately 10 and 5 milliseconds respectively in the first dataset, and 7 and 4 ms in the second data set. The fluctuation in the computational times (especially in the first dataset) result from the illumination variance and texture difference between the views of the two cameras. These two steps only depend on the image size and the texture/lighting in the scene since we repeat the feature detection if enough features cannot be found. However, we repeat the feature extraction maximum two times to be able to continue real-time tracking. The Delaunay elimination time is negligible compared to the other modules. The Delaunay triangulation has a running time $O(nlogn)$ regardless of the distribution of the points. Since we limit the number of tracked features to 1000 and use binning, the time complexity of this step is independent of input data.

The total number of the map points is similar to each other in both datasets and approximately 3000. One of the reasons that the number of map points in the first dataset is larger than the second dataset, although it is shorter, is that the amount of texture in the first dataset is higher than in the second one.

Between the tracking and pose estimation modules, the first tracking stage takes the most time since it is handling the initial stage of tracking and attempts to track all the visible features. The other tracking stage works on the already found features by the first stage tracking and refines their 2D image positions. The stage one tracking takes approximately 5-10 milliseconds depending on the amount of visible features. The pose estimation stages take approximately 2-3 ms and are computationally very cheap compared to the stage one tracking. The pose estimation using EPnP algorithm can be done in O(n) and we fix the Gauss-Newton iterations in non-linear robust pose estimation stage to 10. The total visual odometry takes approximately 20 ms and its time complexity is not effected by the number of key-frames or map points, since the maximum number of tracked map points, the maximum/minimum number of features and iterations in optimization are set to fixed values. Moreover, we use binning and do not add new map points around the already occupied locations, which helps to keep the

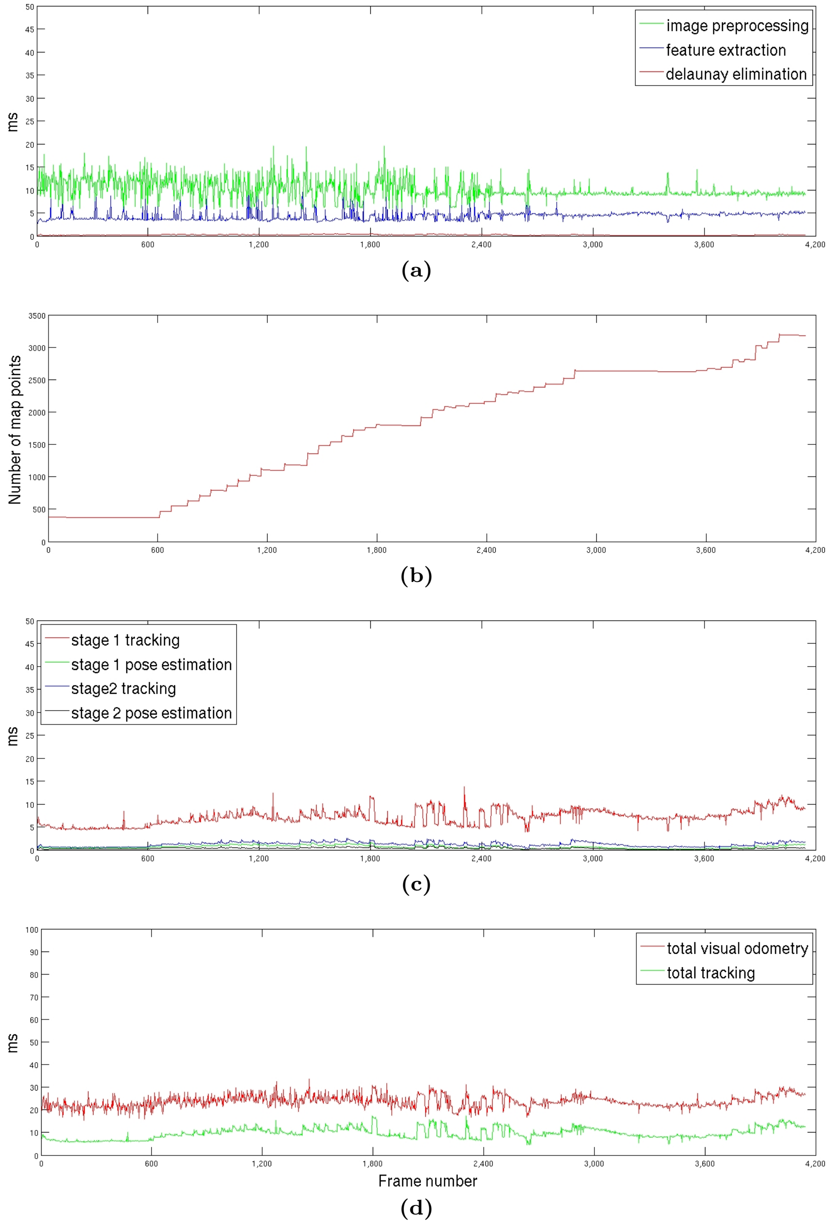


Figure 4.26: *Computational times of the designed modules and different stages of the tracker tested on the first dataset. (a) the computational times of image preprocessing, feature extraction and Delaunay elimination modules, (b) the number of map points created during the operation, (c) stage 1 and stage 2 tracking and pose estimation times and finally (d) the computational time of the complete visual odometry module and tracking modules.*

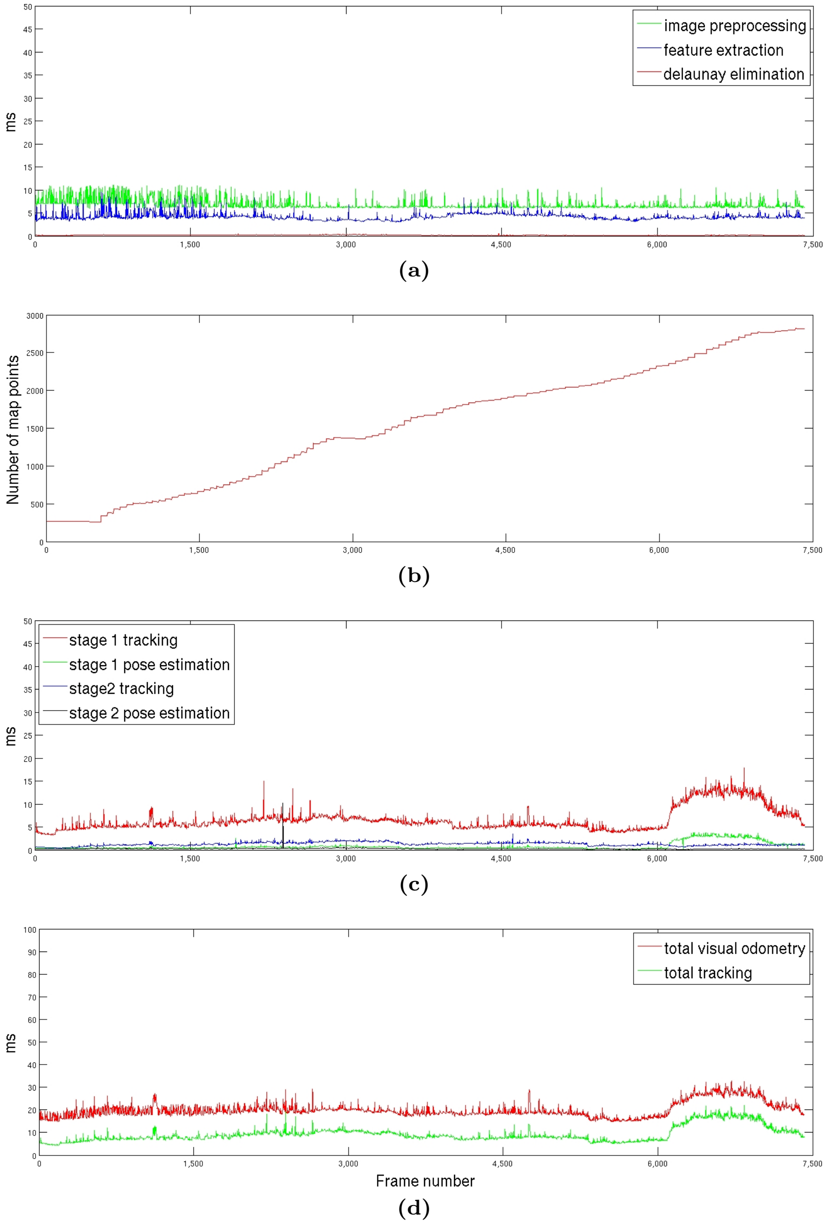


Figure 4.27: *Computational times of the designed modules and different stages of the tracker tested on the second dataset. (a) the computational times of image preprocessing, feature extraction and Delaunay elimination modules, (b) the number of map points created during the operation, (c) stage 1 and stage 2 tracking and pose estimation times and finally (d) the computational time of the complete visual odometry module and tracking modules.*

map size in manageable levels.

The computational performance of the sparse mapping module is given in Fig.4.28 for two datasets. In both figures the computation times of the local and global bundle adjustment are given against the number of map points. The sparse bundle adjustment can be done in $O(n_p n_c)$ where n_c is the number of key-frames and n_p is the number of visible map points in these key-frames. It is not affected by the number of key-frames since it is performed on the closest $N = 3$ key-frames. Also, since we are limiting the number of map points via binning the effect of the number of map points is very limited on computational time.

The time complexity of the global bundle adjustment is $O(n_p n_c^2)$ therefore its computational time increases as the number of map points and key-frames increases since it is performed on all the map points visible in all the key-frames. Very short global bundle adjustment in the figures happen due to the fact that when a new key-frame is added to the sparse mapping module, all the bundle adjustment steps currently running on the map are canceled and new map points are initiated.

When we change the threshold for the maximum number of new map points for each bin (the default value is 10) for new map point selection, the total number of map points increase. Although this increases the computation time of the bundle adjustment modules, its effect on the tracking is negligible since the system performs already more than 30 fps.

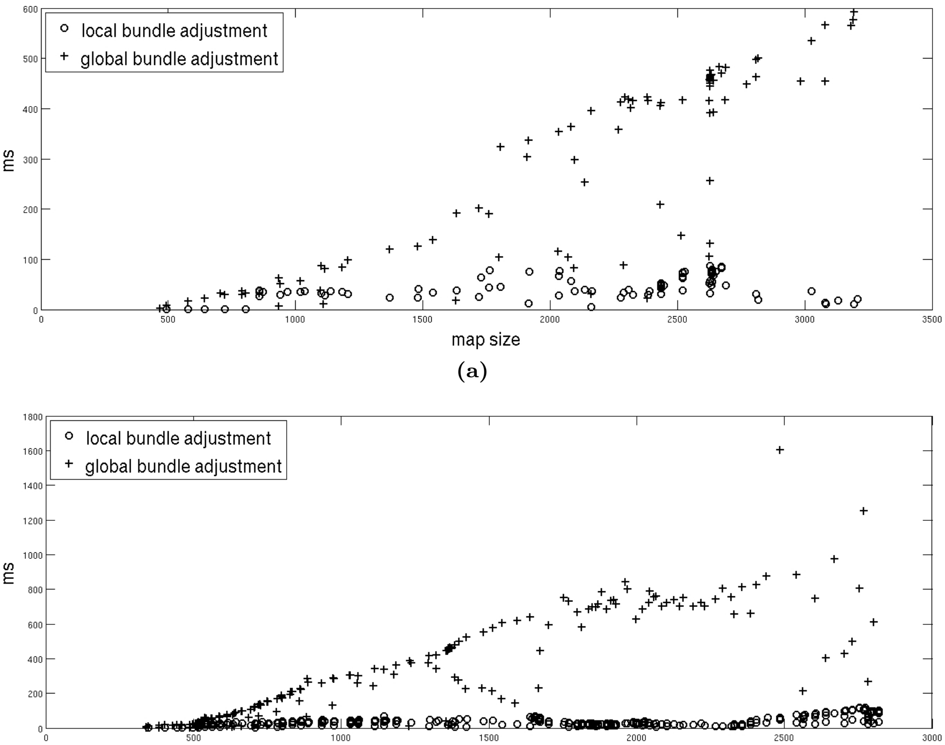
Finally, in Fig.4.29 we demonstrated the relocalization module's performance. When the tracker is lost, the relocalization algorithm is executed to find the pose of the camera (shown in red).

4.4.2 Pose Estimation Accuracy

In order to compare our pose estimation's accuracy with a marker-based method, we have placed a marker into the scene as shown in Fig.4.30 and used AR Toolkit [89] to detect the pose of the camera with respect to the marker. We used a square marker of $30x30$ cm which gives us an maximum effective tracking range between $1.6-1.8m$ according to [89]. In the marker-based system, the world coordinate frame is placed onto the marker. Therefore, when we first initialize our system, we detect the marker and align our world coordinate frame with the marker-based world coordinate system.

In order to compare the pose estimation results, we detect the position, C^m, and the orientation, R^m, of the camera with the marker-based method.. Afterwards we calculate the position, C^n, and the orientation, R^n, of the camera with our natural feature based method. In order to compare the results of the two methods, we use the difference between the positions and the angle between the two principal axis (Z axis) of the pose estimates as an error measure.

$$\epsilon_R = arccos(R_z^m \cdot R_z^n) \text{ and } \epsilon_C = |C^m - C^n| \tag{4.14}$$



(a)

(b)

Figure 4.28: *The computation times of local and global bundle adjustment in the sparse mapping module for two datasets.*

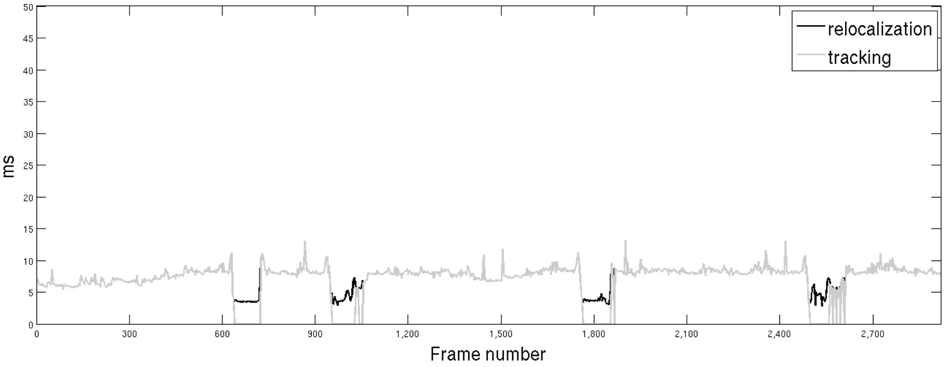


Figure 4.29: *The relocalization module. When the tracker is lost, the relocalization module tries to find the pose. When the pose is found, the tracker keeps working from the re-found position.*

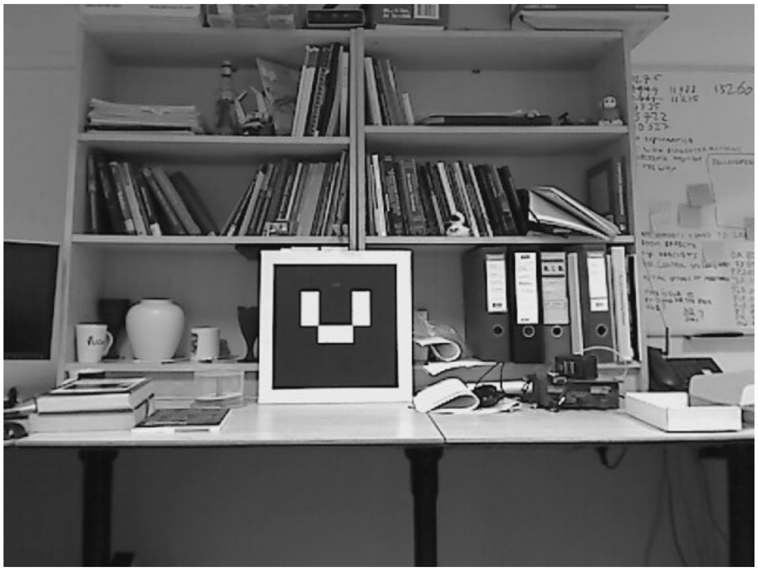


Figure 4.30: *A marker is placed into the scene for the comparison with a marker-based method.*

We have tested our algorithm on 5 different datasets. The characteristics of the test sequences are given in Table.4.2. In the first dataset, the user is asked to move parallel to the marker's surface from approximately $1.5m$ while looking in the direction of the marker's surface normal (Z axis). This gives us a translational motion in the $X - Y$ plane of the marker. In the second dataset, the user moves in the direction of the marker's Z axis from $40cm$ to $2.5m$ while looking in same direction. This results in a translational motion along the Z axis of the marker. In the third dataset, the user rotates his head around the X, Y and Z axis while looking at the marker from $2m$ which results in mostly rotational motion. In the fourth dataset, the marker is placed on the floor and the user follows a circular trajectory around the marker while looking towards the marker. In the last dataset, the user makes a random motion and walks around the room. In all datasets, the user always keeps the marker in the view.

In Fig.4.31 - Fig.4.35 the trajectories and the errors, ϵ_R and ϵ_C, for each frame are given. We also present the average errors and the sizes of the created maps for each dataset in Table.4.3.

Our algorithm performs better than the marker-based algorithm when the camera is relatively further away ($-1.5m$) then the marker. As the distance between the camera and the marker increases, the pose estimates of the marker-based algorithm become inaccurate and the virtual object jitters as shown in Fig.4.32. Moreover, a similar performance decrease is observed as the angle between the camera's principal axis and the marker's surface normal increases. As

Table 4.2: *Properties of the test video sequences.*

seq. id	type of motion	# of frames
1	translation parallel to the marker	1802
2	translation in the marker's normal direction	1705
3	rotation around the head	1471
4	circular around the marker (rot. + trans.)	1151
5	random (rot. + trans.)	1976

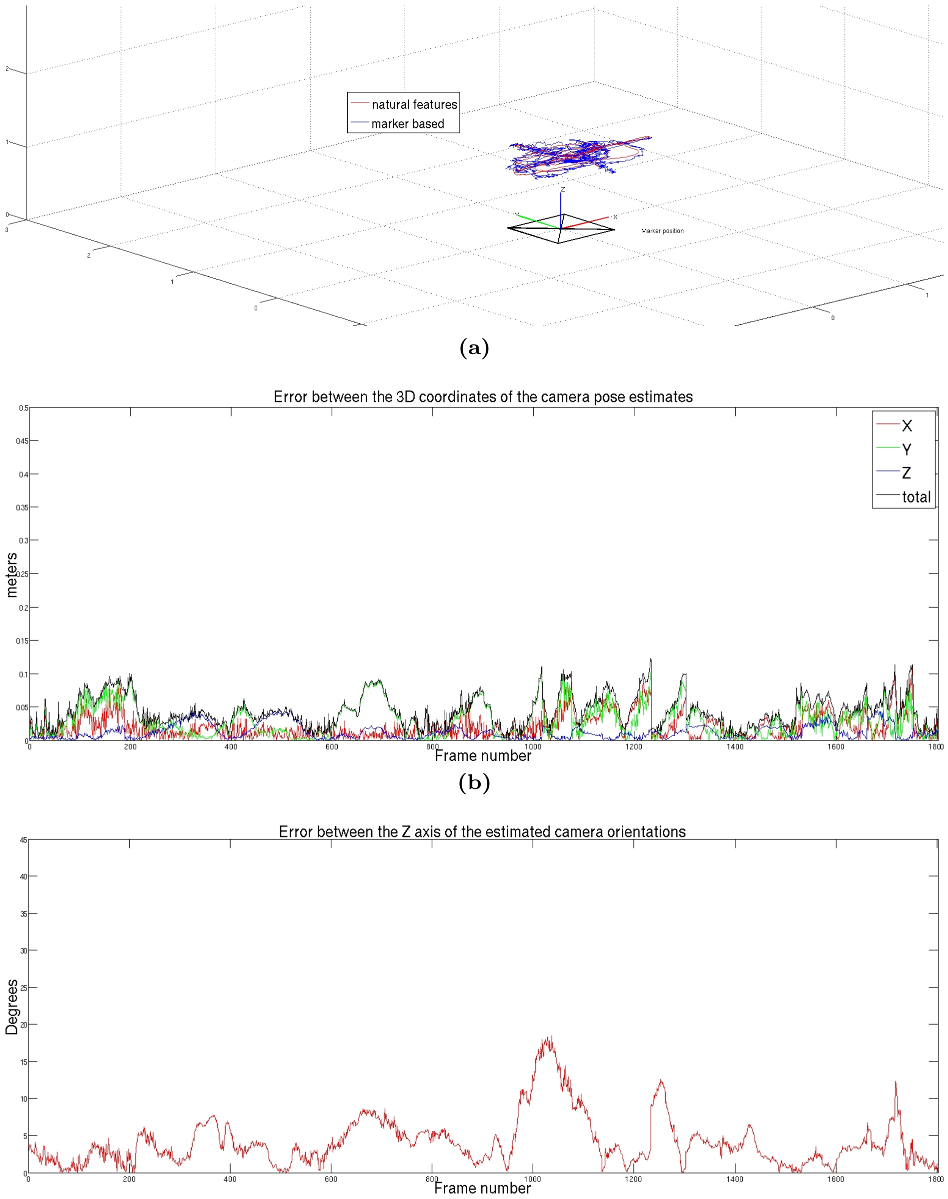
Table 4.3: *Results of the test video sequences.*

seq. id	average rot. error (°)	average trans. error (m)	# of map points	# of keyframes
1	4.1536	0.0434	684	6
2	3.5554	0.0511	630	6
3	8.1931	0.0541	249	2
4	5.6345	0.0831	2291	35
5	14.715	0.0747	1473	21

shown in Fig.4.32, the error increases as the angle increases and the error gets closer to zero as the angle decreases. These drawbacks result in the large average errors as shown in Table.4.3.

However, our algorithm is not affected by the increasing or decreasing distance between the camera and the marker, since new features are added as the camera moves and the bundle adjuster imposes a smooth trajectory by using the previous pose estimates. Therefore, an accurate virtual object placement and almost no-jitter are observed in the test datasets. When the camera is close to the marker, the performances of the marker-based method is similar to our method. However, the orientation estimates of the camera are more consistent and smooth with our method.

The presence of the marker during the experiments to test our method introduces the corners of the marker as natural features. However, the marker also hides other natural features behind it and therefore the marker corners don't make any favor to our system. Moreover, as the camera comes closer to the marker, the camera's view is mostly occupied by the marker and less natural features are available since the marker is composed of texture-less black and white regions. Although this introduces a clear drawback, our system still performs better than the marker-based algorithm.

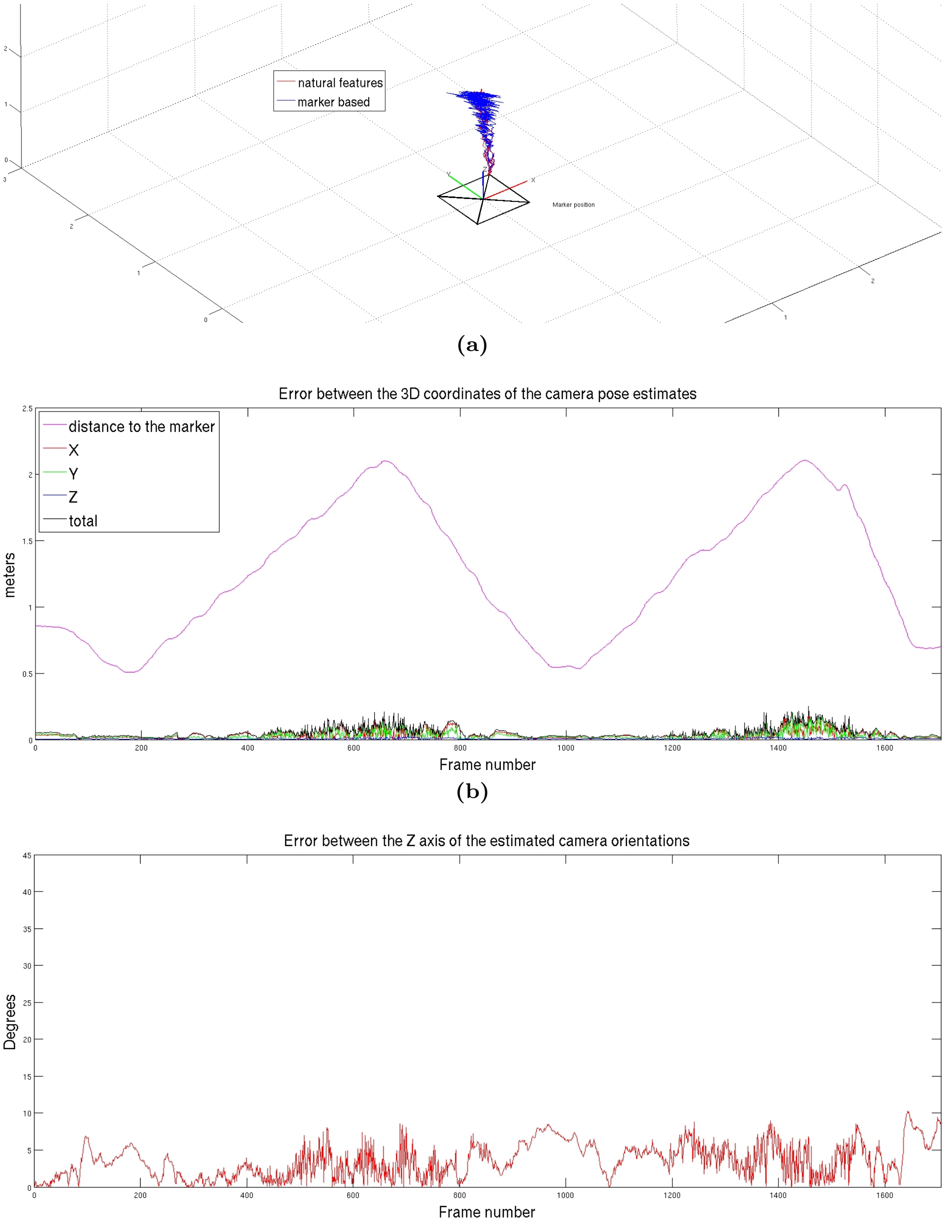


(a)

(b)

(c)

Figure 4.31: *Results of the first dataset. (a) the 3D trajectory of the camera in the world coordinate frame placed on the marker. The red line represents the natural-feature based pose estimation while the blue one represents the marker-based methods pose estimates, (b) the error between the position of the camera pose estimates, (c) the error between the Z axis of the estimated camera orientations.*

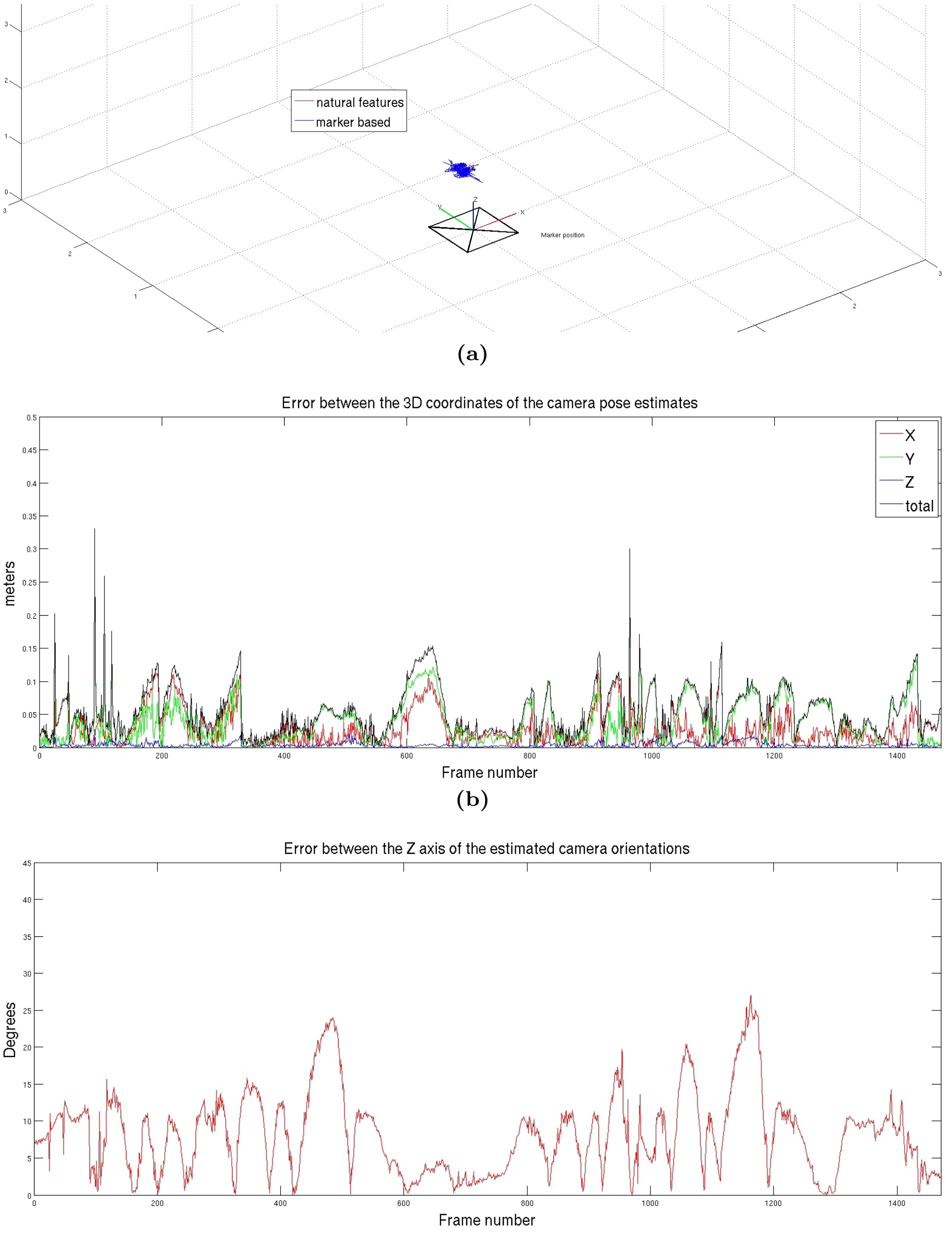


(a)

(b)

(c)

Figure 4.32: *Results of the second dataset. (a) the 3D trajectory of the camera in the world coordinate frame placed on the marker. The red line represents the natural-feature based pose estimation while the blue one represents the marker-based methods pose estimates, (b) the error between the position of the camera pose estimates. Purple line shows the distance between the camera and the marker, (c) the error between the Z axis of the estimated camera orientations.*

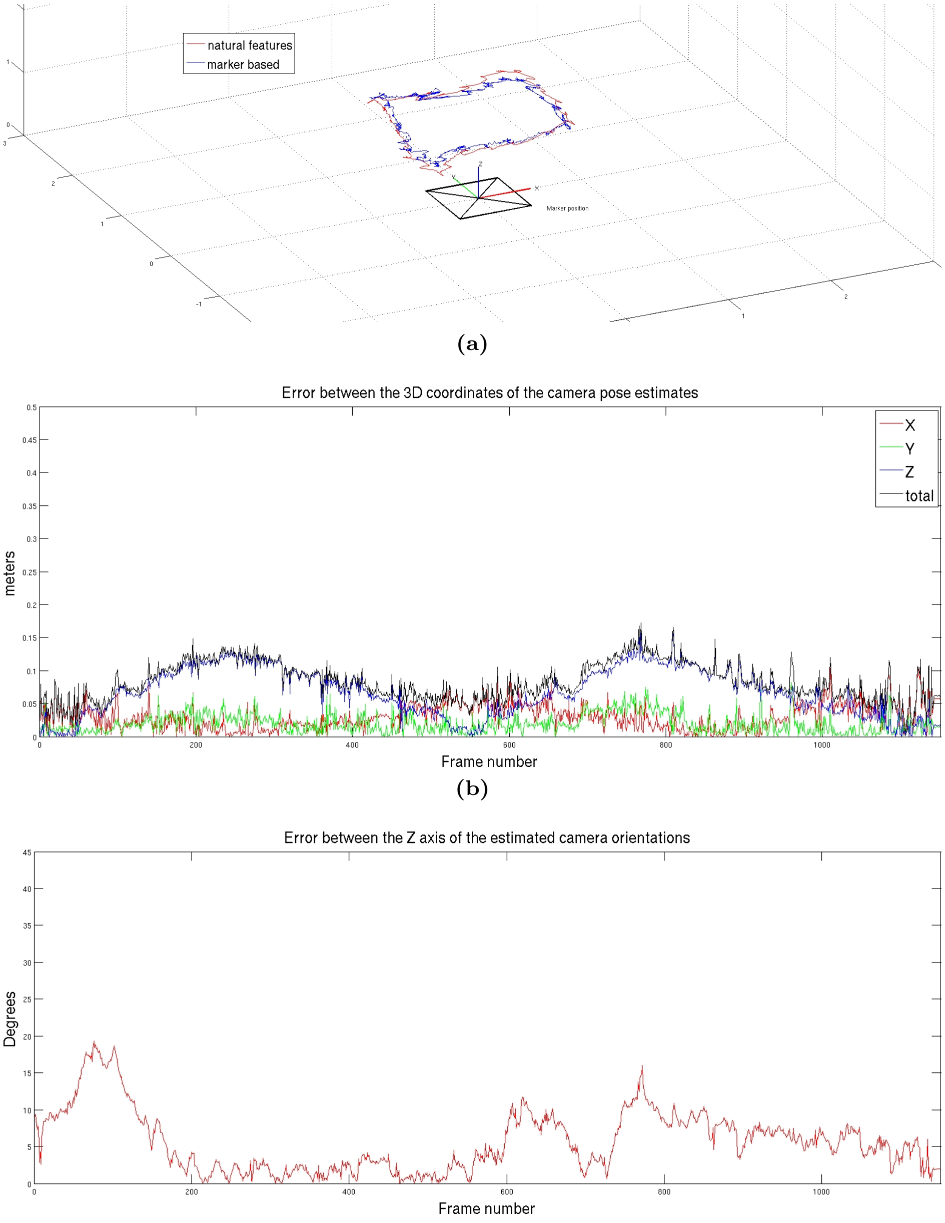


(a)

(b)

(c)

Figure 4.33: *Results of the third dataset. (a) the 3D trajectory of the camera in the world coordinate frame placed on the marker. The red line represents the natural-feature based pose estimation while the blue one represents the marker-based methods pose estimates, (b) the error between the position of the camera pose estimates, (c) the error between the Z axis of the estimated camera orientations.*

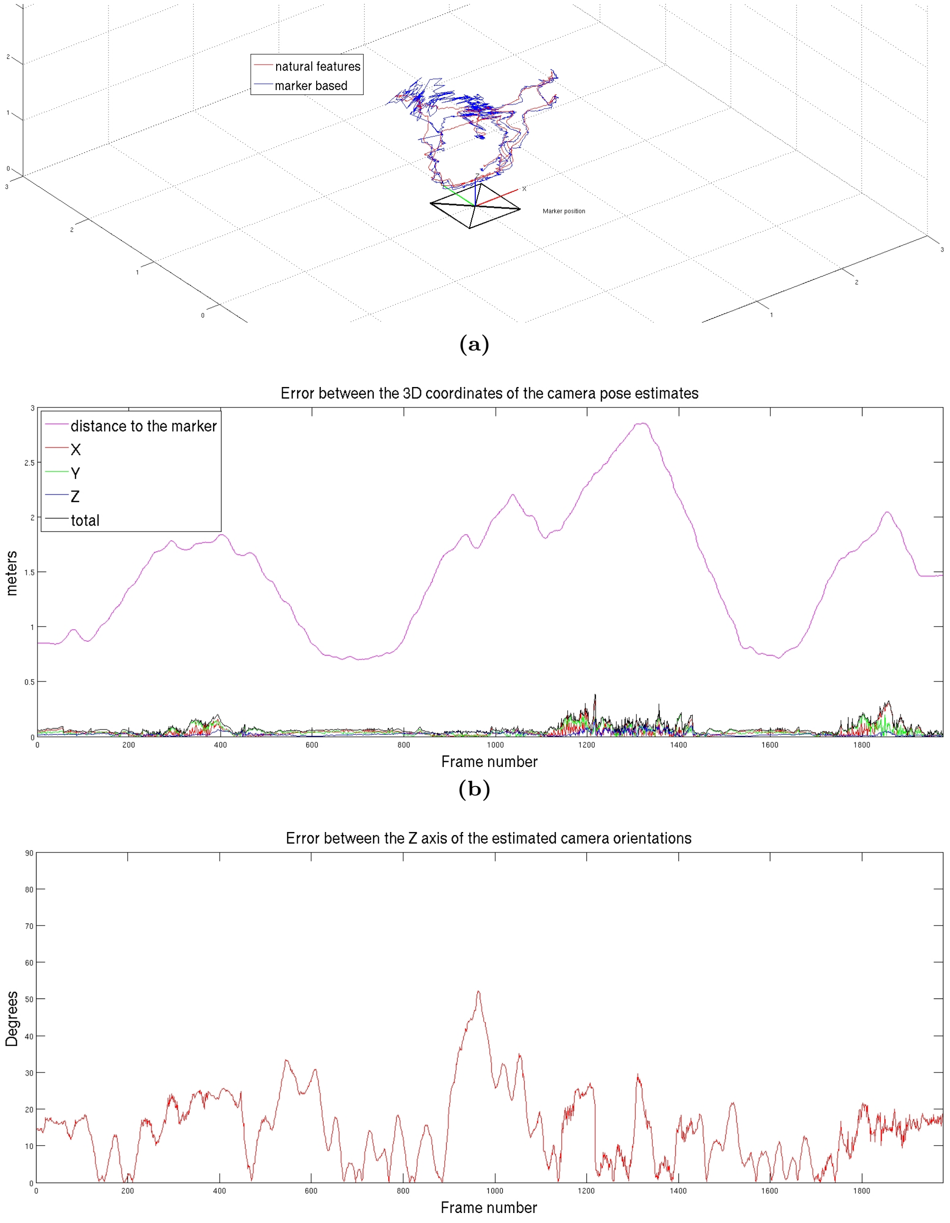


(a)

(b)

(c)

Figure 4.34: *Results of the fourth dataset. (a) the 3D trajectory of the camera in the world coordinate frame placed on the marker. The red line represents the natural-feature based pose estimation while the blue one represents the marker-based methods pose estimates, (b) the error between the position of the camera pose estimates, (c) the error between the Z axis of the estimated camera orientations.*



(a)

(b)

(c)

Figure 4.35: *Results of the fifth dataset. (a) the 3D trajectory of the camera in the world coordinate frame placed on the marker. The red line represents the natural-feature based pose estimation while the blue one represents the marker-based methods pose estimates, (b) the error between the position of the camera pose estimates. The purple line shows the distance between the camera and the marker, (c) the error between the Z axis of the estimated camera orientations.*

4.5 Conclusion

In this chapter we proposed a marker-less visual odometry algorithm for fast and accurate camera pose estimation by using natural features available in the scene. We created a two stage real-time tracking method to minimize the drift while being robust against fast camera motions. The visual odometry algorithm is able to track the pose of the camera at approximately 45-50 fps. The sparse 3D map of the scene is also created and extended during the operation of the user, so that the motion of the user is not confined to small spaces. Compared to the marker-based methods our algorithm is not sensitive to the distance between the camera and the marker (or initial map points), robust against different viewing angles, and leads to more accurate and consistent pose estimation results. Experimental results showed that the pose of the camera is determined accurately in indoor environments such as rooms. Accurate and the real-time operation of our system enables marker-less, extensible tracking for augmenting a real scene with virtual objects. This robustness makes it suitable for head-mounted display based augmented reality applications.

Chapter 5

Real-time Dense 3D Reconstruction

"Men have forgotten this truth," said the fox.
"But you must not forget it.
You become responsible, forever, for what you have tamed."
Antoine de Saint Exupry, The Little Prince

5.1 Introduction

Modeling the real-world geometry is another important part of Augmented Reality (AR). If the geometry of the scene can be captured, virtual entities can be registered correctly while performing tracking. This provides a spatial consistency between the real and the virtual world which leads to a realistic AR experience for the user.

From the collaborative AR point of view, a 3D reconstruction of the scene can help to introduce a common ground between the collaborators. Sharing the reconstructed space and being able to alter it by adding virtual content allows enhanced communication (as explained in Chapter 1). In collaborative CSI applications, a remote investigator that is collaborating with the on-site user can be supported with the reconstructed real-world geometry by enabling them to move freely in the created world independently from the user on-site (as explained in Chapter 2). Moreover, the real 3D scene helps to introduce a common ground between the investigators so that the discussions about the crime scene can be done over the created map instead of or in addition to the standard ways such as photographs and drawings. In addition to aiding the discussions between the teams, if the maps are created with calibrated cameras and therefore the metric information is available, initial measurements such as the distance between two locations or the dimensions of the objects can be extracted from the map.

Although there are off-line solutions for dense reconstruction via laser scanners or multiple-view stereo algorithms, real-time and robust operation is still not achieved completely.

In this chapter, a system is described that is able to build a dense 3D map of that world. The system presented satisfies the requirements and overcomes the challenges mentioned.

5.2 Related Work

Our pose estimation method explained in Chapter 4 calculates the 3D location and orientation of a camera at each frame and it builds up a sparse 3D map of landmarks that is used for tracking. Sparsity is necessary in order to cope with the computational constraints. Dense 3D map reconstruction is a process that can be performed in parallel to pose estimation. Consequently, another software module is necessary that is able to create dense 3D maps and merge them with each other.

Geometry extraction from a single image via learning has have been investigated in various researches [74, 169]. An image is segmented into superpixels, and a 3D configuration of each pixel is estimated by using local appearance (texture) and global constraints (planarity). Although the results are impressive, they assume that the scene is made up from a number of planes and hence reconstructs the -mostly outdoor- scenes with a couple of planes, which usually represent the ground, walls of the buildings, and background. Therefore the maps are not accurate enough for 3D dense scene reconstruction, especially for indoor environments.

3D reconstruction from uncalibrated images is also demonstrated in various works [156, 176, 75] by using structure-from-motion (SfM) algorithms combined with multiple-view stereo (MVS). Their extensions for urban scene reconstruction [1, 64, 129] are also available in literature. The reconstruction of a city is also demonstrated with similar methods in [61]. These algorithms are not motivated by real-time operations, require an immense computational power and take easily a couple of hours to obtain an accurate semi-dense reconstruction. Moreover, they require the visibility of map points in many different frames which is not always available in our system. Comparatively faster extensions utilize other sensors such as GPS or laser scanners, or use various assumptions, such as the planarity of the scene or the visibility of points depending on the context. However, these assumptions are not always valid in our application where auxiliary sensors such as laser scanners are not desired due to time constraints and GPS is not applicable since we use the system mostly in indoor environments. Moreover, crime scenes can have non-planar regions such as bodies, which violates the planarity assumptions.

Recently, real-time solutions for multi-view reconstruction are also presented. Newcombe and Davison [135] showed that it is possible to reconstruct very detailed and dense representations of a highly textured desktop-scale environments by combining PTAM with multi-view approaches. In [188], a very similar system is presented with near real-time dense map creation. However, their method is confined to small workspace areas. Also their primary goal is to create very detailed 3D maps which is not always necessary for our application.

Also recently, reconstruction via RGB-D cameras are investigated by many researchers [111, 134, 148]. Dense 3D maps obtained from the depth cameras are combined by using the pose information estimated via the aforementioned methods. In [154, 81] the Microsoft Kinect RGB-D sensor is used for modeling indoor environments. However, these systems are targeted for large-scale building mapping, work on recorded videos and do not perform any real-time tracking to provide real-time performance needed for augmented reality. Vision based hand-held active scanners for creating dense 3D scans of objects are also studied. In [204], a high quality scan is created with a fixed Time-of-Flight (ToF) camera and by moving the object in front of it, in [42] the hand-held ToF camera is moved around the object. However, these systems are motivated by small scale reconstruction with high quality scans. In general, although RGB-D camera based systems perform well on texture-less surfaces and independent from indoor lighting conditions, they are constrained by the limitations of the sensors such as sensitivity to day-light and maximum depth limitations, and introduce extra weight and power consumption. Therefore they are not considered in our design.

All these approaches have different objectives and therefore they are not suitable for our application. Utilization of single camera image or scene priors to create a map do not provide correct metric maps that are required for our application. Computationally intensive off-line solutions for dense reconstruction are also not applicable for our on-site fast scene reconstruction application. Other approaches that are using online reconstruction algorithms with cameras or other sensors are either confined to small desktop environments or needs extra hardware which is not suitable in our application.

5.3 System Overview

Various reconstruction types are briefly discussed in Section 3.5. We focus on full reconstruction in which the collinearity, parallelism, angles and metric transformations are preserved. As explained in Chapter 2 the aim of the dense reconstruction module is not to generate highly accurate maps of the crime scene but to provide contextual information which can help the remote experts to navigate around the scene and the on-site users to place virtual tags while performing initial measurements. Our overall dense reconstruction method is depicted in Fig.5.1.

When a new key-frame pair is added to the sparse map, the sparse map maker pushes the pair after refining its pose into a key-frame buffer. Presence of key-frames in this buffer triggers the dense reconstruction thread. For each stereo pair, a disparity map is calculated by utilizing stereo matching and a continuous stream of disparity maps is generated when the camera moves around the scene. The colored point clouds obtained from the disparity maps are aligned using their

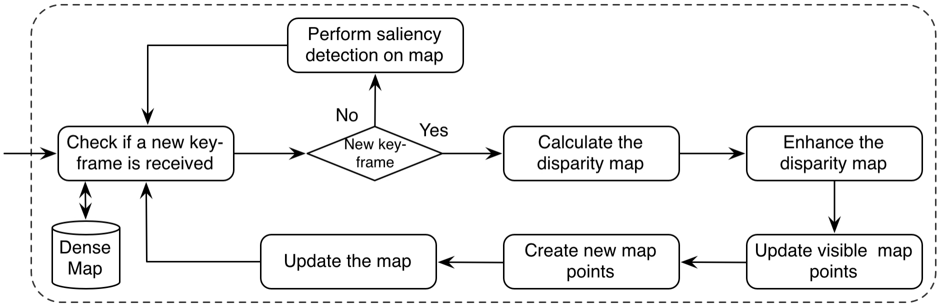

Perform saliency detection on map
No
Yes
Check if a new key-frame is received
New key-frame
Calculate the disparity map
Enhance the disparity map
Dense Map
Update the map
Create new map points
Update visible map points

Figure 5.1: *Dense reconstruction flow.*

estimated relative poses to merge them into a final dense reconstruction of the environment. The coordinate frame derived from the previous section is selected as the main coordinate frame and all points coming from further stereo image pairs are projected to that main coordinate frame.

5.3.1 Disparity Map Calculation

The term *disparity* is used to describe the difference between the image coordinates of the projection of a 3D point in the left and the right images of a stereo pair. Since our cameras are placed horizontally and calibrated (therefore the images are rectified), calculated disparity values are always in the horizontal axis (x-axis in image coordinates). The disparity value d_i for pixel $\mathbf{p}_i(x, y)$ is given with respect to a reference image (usually the left image) and the relation between the projections of p_i into the left and the right images, (x_l, y_l) and (x_r, y_r)

$$x_l = x_r - d, \quad \text{and} \quad y_l = y_r$$

Then, the 3D coordinates of a point can be calculated by using the disparity values and triangulation as explained in Chapter 3. Therefore, the goal of a dense stereo matching algorithm is to calculate the disparity values for each pixel in the key-frame (stereo) pair such that the produced disparity values (image) best describes the 3D geometry in the scene. Two broad classes of stereo matching algorithms are local window-based algorithms and global algorithms [171].

In local window-based algorithms, the disparity of a point is calculated by using only the intensity values of its neighbors within a finite window and assuming that the disparity values within the support window are smooth. For instance, in traditional sum-of-squared-differences (SSD) based block-matching (BM) algorithm [171] the squared difference (SD) of intensity values at a given disparity value is used as the matching cost. The matching costs are aggregated by summing the SDs within a support window assuming that all the pixels in the window have a constant disparity. Then, the aggregated costs for different disparities are calculated and the disparity associated with the minimal aggregated value (minimum cost) is selected as the final disparity at each pixel. Since the costs are computed using only the pixels in a support window, local algorithms strongly dependent on the window size. Although they are computationally cheap, small window sizes leads to a low matching ratio on poorly-textured surfaces, while bigger window sizes can eliminate the edges by smoothing them out.

Algorithms based on global correspondences [58, 22] attack the aforementioned problems with local correspondences by introducing smoothness constraints on the disparities and solving an optimization problem by minimizing an energy (global cost) function that combines a sum of a data fitting term and a smoothness term. As explained in [171], the goal is to find the disparity value that minimize a global energy function

$$\epsilon(d) = \epsilon_{data}(d) + \lambda\epsilon_{smooth}(d).$$

where the data term $\epsilon_{data}(d)$ measures how well the assigned disparities values (the disparity function) minimize an global aggregated matching cost, similar to the minimal aggregated value in local algorithms but for all pixels. The smoothness term $\epsilon_{smooth}(d)$ enforces a disparity smoothness assumption between the neighbor pixels. A more detailed information about energy functions types and optimization methods for disparity calculation can be found in [171]. In general, global methods suffer from a high computational complexity since a global optimization scheme is necessary and cannot be performed in real-time.

For stereo matching we utilized 3 different algorithms: a traditional block-matching (BM) algorithm [171], modified Semi Global Matching (SGBM) algorithm [82] and Efficient LArge-scale Stereo (ELAS) algorithm [66]. For the BM and SGBM algorithms, we used the implementations in OpenCV [142] and the implementation provided in [66] is used for ELAS algorithm. We present the overall reconstruction results in the next section.

The BM algorithm is an example of local window-based algorithms and uses only the intensity values within a finite window without any global constraints.

In the original SGM [82], the pixel matching cost is calculated hierarchically by mutual information and aggregated in multiple directions. An approximation of a global energy function (2D smoothness term) by combining many 1D constraints is minimized by path-wise optimizations from all directions (along 16 orientations) through the image. The winner which corresponds to the minimum cost is selected as the match. Since the method uses an approximation of a global energy function, it is relatively faster than algorithms based on global correspondences and more accurate than the local ones. In the modified version of SGM, SGBM, the pixel matching costs are calculated over small blocks instead of individual pixels. Also the number of passes for path-wise optimizations from all directions is limited to five rather than a traditional eight passes. Moreover, in the OpenCV implementation there are filtering and post processing steps for improving and refining the final disparity maps.

Lately, the ELAS algorithm [66] is introduced, which shows impressive results

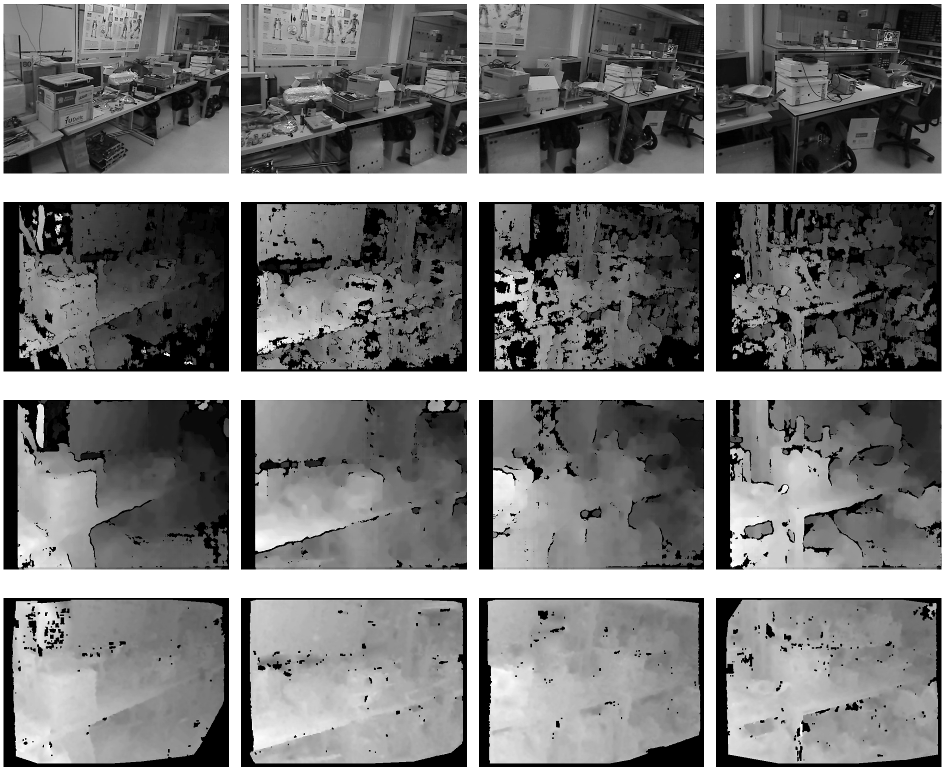


Figure 5.2: *Examples of the left key-frames and calculated disparity maps from the corresponding key-frame pairs. From top to bottom, left key-frames, disparity maps created with block-matching, SGBM and ELAS algorithms respectively.*

in real-time. It uses small aggregation windows for dense matching by reducing ambiguities on the correspondences. First, *support points* which are a set of robustly matched correspondences are calculated. Then, a prior over the disparity space is built by forming a triangulation on the support points. Because the built prior is piecewise linear, the algorithm performs well in the presence of poorly-textured and slanted surfaces. This advantage of ELAS makes it a good candidate for our application.

In Fig.5.2, the examples of the created disparity maps are given. In the first row, four left key-frames selected from the same dataset are given. In the following rows, the results of the block-matching, SGBM and ELAS algorithms are given respectively. We selected a search window size of $11x11$ and fix the disparity search range between 0 and 48 pixels.

5.3.2 Disparity Map Registration

When the disparity map is calculated and the 3D points are created by linear triangulation, the next step is to register new map points with the existing ones. A straight-forward way of merging point clouds is to register new points directly into the existing map. However, this would increase the size of the map to unmanageable amounts mostly with redundant points, while not improving the quality of the map. Moreover, utilization of the multi-view approaches are computationally too expensive as discussed in the related work section.

Instead, we reproject the existing map points into the current key-frame and compare it with the calculated disparity map. The projection of an existing map point $\tilde{M_i} = [X, Y, Z, 1]^T$ in the world coordinate frame to the current left key-frame K_j can be given

$$x_l = \pi(g_{cw}^{K_j} \tilde{M}_i)$$

where $\pi()$ represents the projection from camera frame to the image coordinates and $g_{cw}^{K_j}$ is the pose of the camera when the key-frame K_j is captured. If the projection is inside the image (the map point is visible) and there is a valid disparity value at the projected location x_l of the existing map point M_i, then the 3D coordinate of the map point is updated by using a weighted average. The weights are obtained by counting the number of updates for each point. For instance, if a map point is observed N times before, then the new average position when a new measurement comes is

$$M_i^t = \frac{N M_i^{t-1} + M_i^t}{N + 1}$$

where M_i^t is the 3D coordinates of the map point at time t. The only exception to this updating is when the new point is not spatially close to an existing point. In this case, a new map point is initiated. This approach adds new map points if there are no existing points in the scene, or updates the existing ones as illustrated in Fig.5.3. Moreover, averaging reduces the measurement noise.

Finally, when a map point is updated, we calculate the standard deviation of the 3D coordinates of the map point calculated in different key-frames in order to evaluate the consistency of its estimated 3D position. If the standard deviation of the 3D coordinates is larger than $5cm$ we label the point as a 'bad point' and remove it from the map.

An example of a dense map is shown in Fig.5.4. Map points created from the same key-frame pair are given the same color and the overall map is created from 4 key-frames (pairs) shown in Fig.5.2

5.3.3 Disparity Map Refinement

After disparity map generation we use an algorithm of Sarkis et al. [167] called the tritree meshing algorithm [167] in order to enhance the map and also create a

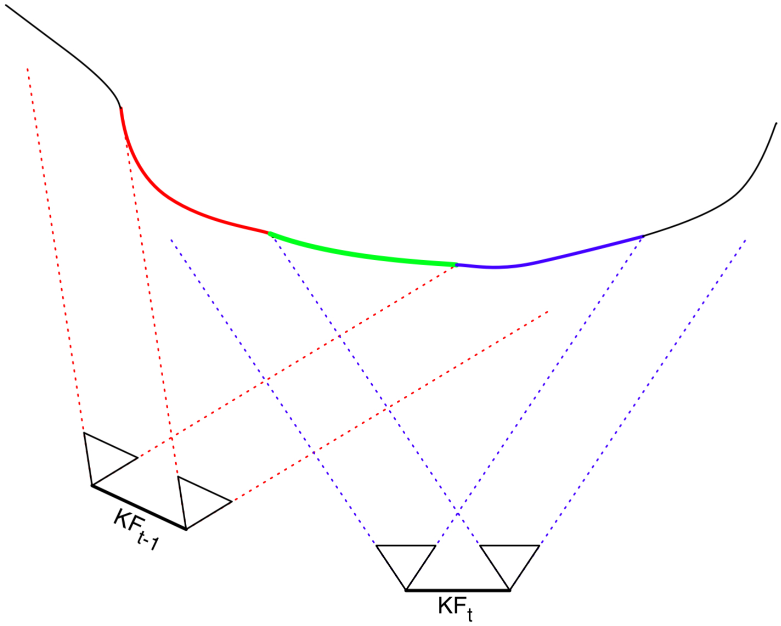


Figure 5.3: *Dense 3D reconstruction. Existing reconstructed parts of the scene from key-frame at $t-1$ are illustrated in red. The green color represents the updated points with the new key-frame while the blue color shows the new map points reconstructed from the key-frame at t*

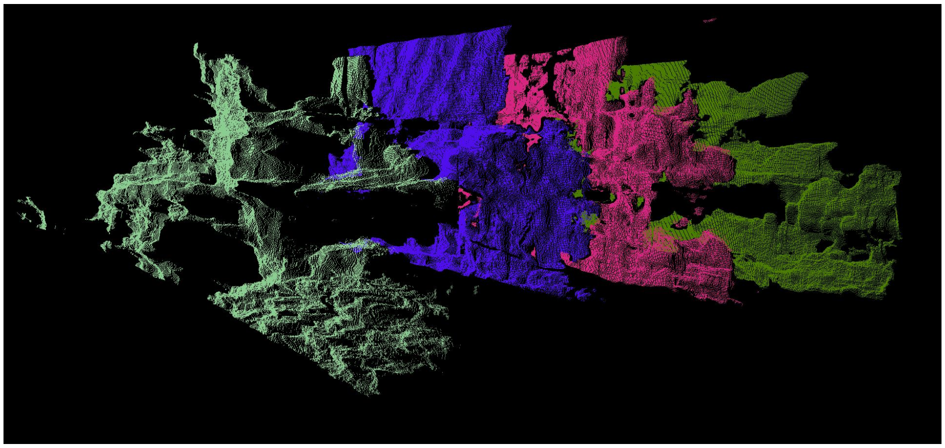


Figure 5.4: *Dense 3D reconstruction from 4 image pairs. Different colored map points are originally created from different key-frame pairs and updated by the measurements available in the other key-frames.*

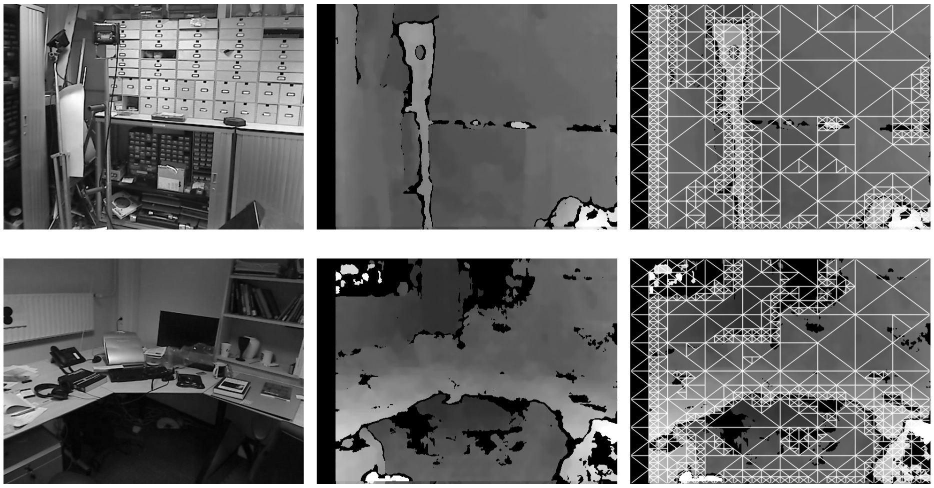


Figure 5.5: *Examples of tritree meshing. Input images are given on the left side and the corresponding disparity maps are presented in the middle. Calculated tritree meshes are shown on the left side images. As can be seen in the pictures, the edges are preserved by representing them with more triangles while relatively smooth regions have bigger triangles.*

sparse mesh out of the map points. Later on this mesh is used for fast visualization of the map. This algorithm was originally developed for image (disparity map) compression.

Initially the disparity image is divided into two triangles. The vertex j of the triangle i is defined as $v_j^i = (x, y, d)$ where (x, y) are the image coordinates of the vertex and d is the disparity value at point (x, y) in the corresponding disparity map. For each triangle, the plane $P^i = (\mathbf{c}, \mathbf{n})$ passing through its three vertices is calculated where $\mathbf{c}$ is the mean of vertices and $\mathbf{n}$ is the normal of the plane

$$\mathbf{c} = \frac{1}{3}\sum_{j=0}^{2} \mathbf{v}_j^i \quad \text{and} \quad \mathbf{n} = \mathbf{v}_0^i \times \mathbf{v}_1^i$$

Then the pixels inside the triangle are projected onto the plane and the distance between the original points and their projections are calculated. The disparity value, d_p, of the projected point p_i onto the plane can be calculated solving the plane equation $(p_i - c) \cdot n = 0$ and it is given as

$$d_p = \frac{-(xn_x + yn_y - c \cdot n)}{n_z}$$

where (x, y) is the image coordinates of the point p_i. If the original disparity value of the point p_i is given as d_o, then the distance between the original point and its projection, ε_i, is

$$\varepsilon_i = |d_p - d_o|$$

A point is labeled as an outlier if the distance $\varepsilon_i > t_d$ and the ratio of the outliers to the total number of points inside the triangle is used as a cost function. If the cost is greater than some threshold, this implies that the variation inside the triangle is high and then the triangle is separated into two. In our experiments we set $t_d = 1$ and split the triangle if the outlier ratio is greater than 0.1. Uncertain disparity values are excluded during the computation of the cost. Also, in order to avoid very small and very large triangles we use minimum and maximum area thresholds for triangles. A triangle is not separated into two if its area is smaller than 100 pixels or definitely separated into two if its area is larger than 8000 pixels. This division continues until there is no triangle with a high disparity variation is found. The final structure is called a *tritree* [167] and triangles inside the tree represent homogeneous regions. The recursive subdivision is called *meshing*.

The proposed method segments the disparity maps into homogeneous regions while preserving the edges since the edge regions are represented by smaller triangles. We also obtain a planar approximation of the scene which can be used to extract shapes. Finally, the disparity values are updated by their projections onto the triangles to eliminate noise and uncertain disparities. Instead of replacing the disparity values with the projected ones, we use the average of the two values to create a more visually acceptable map. Also, the missing parts of the disparity map is filled with the values obtained from the triangles if there are valid triangles on those regions.

Some examples of the tritree meshing are given in Fig.5.5. From left to right, the input images, the corresponding disparity maps and calculated tritree meshes are shown. As can be seen in the pictures, the edges are preserved by representing them with more triangles while relatively smooth regions have bigger triangles.

5.3.4 Post-processing

When new key-frames are not available, some post-processing can be done on the dense map. For instance we have demonstrated that a saliency detection method explained in [4] and a color based saliency detection method [125] can be used to label the map points as salient or not. Moreover, the existing dense map can be improved by using filtering sporadic outliers by calculating the local statistics of each map point and removing it if it is an outliers. Since this step is not crucial for the system's performance we do not explain it in detail and leave it as a future work.

5.4 Results

In our experiments we utilized 640x480 images provided by the tracker module explained in the previous chapter and our system was tested on the hardware explained in Chapter 2: a laptop with a 2.7 GHz Core i7 processor (Intel Corporation, USA) and a Linux operating system.

5.4.1 Computational Time

In order to measure the computational times of the different stages of the dense reconstruction, we have tested our algorithms on a dataset that is composed of 16 key-frames.

In Fig.5.6, the computational times of all modules for different stereo matching algorithms are given for the test dataset. In (a), the total number of map points created during the operation from the stereo key-frames are given. The computational time of the complete dense reconstruction is given in (b). In (c), (d) and (e) the disparity map calculation, disparity map registration and tritree meshing times are given respectively.

When the BM algorithm is used, overall reconstruction of a single frame takes approximately 170 milliseconds ($\approx$ 6 fps), while with SGBM it takes approximately 307 milliseconds ($\approx$ 3 fps) and with ELAS it takes approximately 483 milliseconds ($\approx$ 2 fps). When the registration of map points is considered, the BM performs the fastest since the number of created map points is less than the other two methods. The tritree meshing takes more time when the ELAS is used due to the highly irregular (non-homogeneous) disparity maps created by the algorithm.

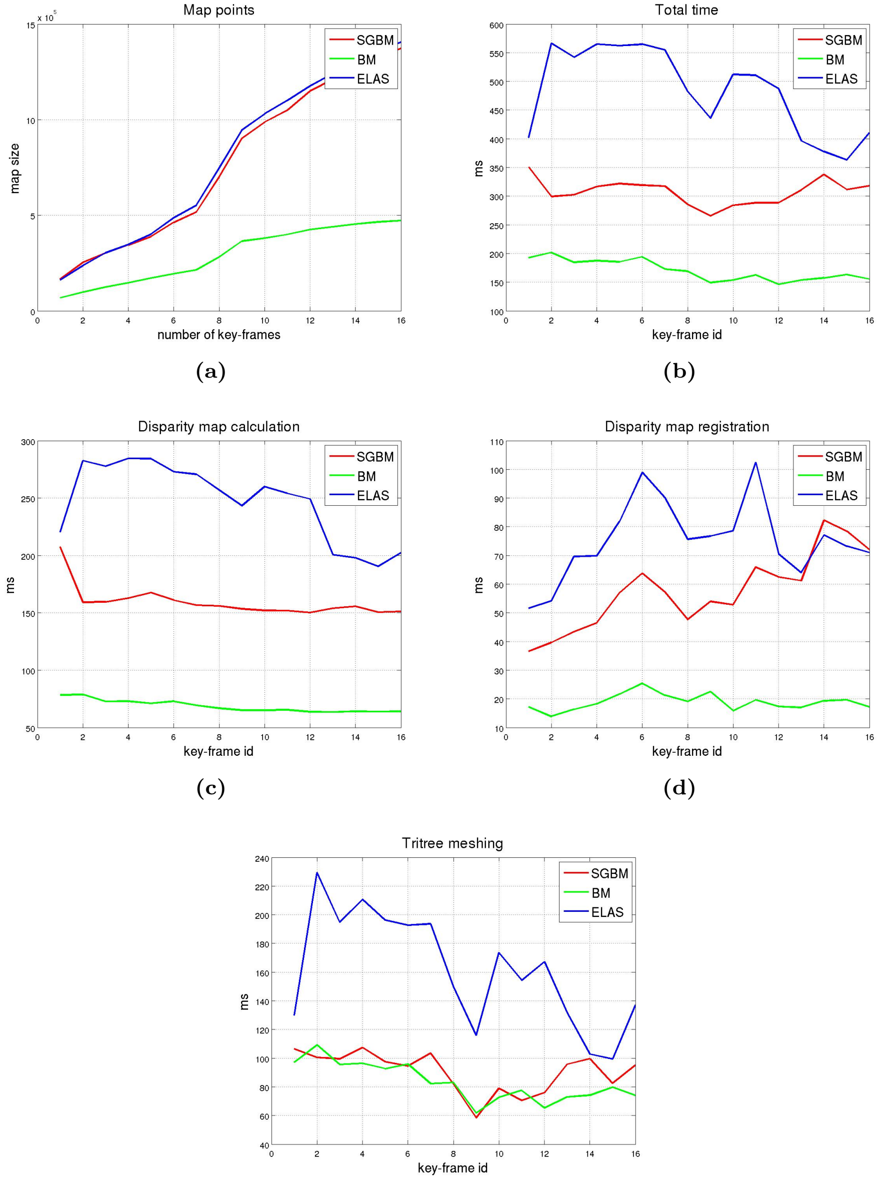
When the SGBM algorithm is used for reconstruction, approximately 3 key-frames can be processed per second. The 3D reconstruction is performed when a new key-frame is created by the tracker module. Since the new key-frames are created as the user moves, the computational performance of 3 fps provides a successful real-time reconstruction when the user is walking at a regular speed and investigating the scene.

5.4.2 Map Quality

The 3D dense maps created with the 3 different stereo algorithms are given in Fig.5.7 and Fig.5.8.

The block matching algorithm results in sparser and less accurate maps although it is computationally faster. The ELAS and SGBM algorithms show similar performance on the tested data sets, although the disparity maps and dense 3D maps created with ELAS are more irregular than the SGBM results, as shown in Fig.5.9.

Initially, we used the BM algorithm to calculate the disparity images. However, this method is very sensitive to noise and since only the pixels that are in the close-neighborhood are used as the supporting region for matching, it fails in relatively low-textured areas. Therefore in the final system we picked the SGBM algorithm [82] over Efficient LArge-scale Stereo (ELAS) algorithm [66] since it gives better maps and performs faster.



(a) (b) (c) (d) (e)

Figure 5.6: *Computational times of the designed dense reconstruction module for three different stereo matching algorithms. (a) the total number of map points created during the operation, (b) the computational time of the complete dense reconstruction, (c) the disparity map calculation times, (d) the computational times of the disparity map registration step and finally (e) tritree meshing times.*

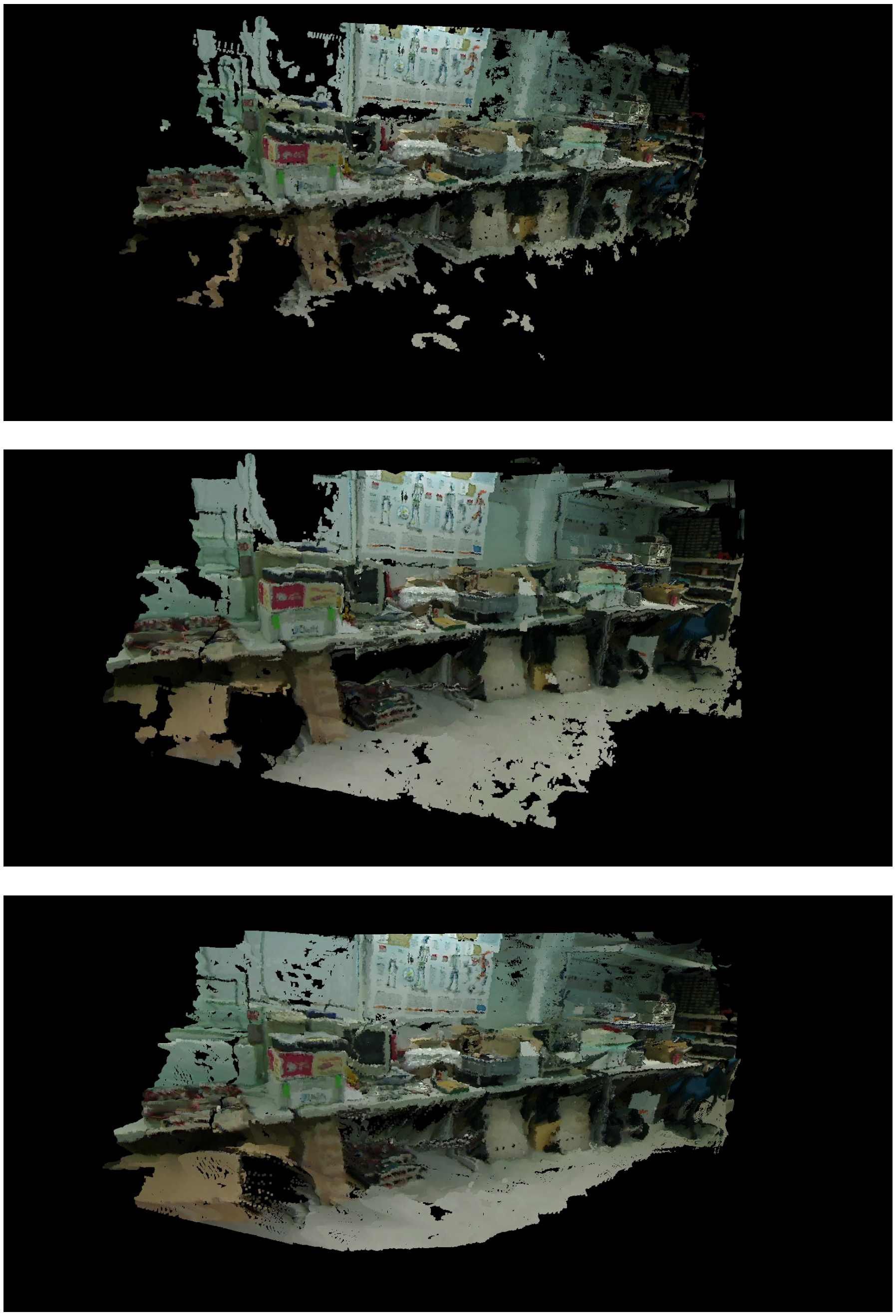


Figure 5.7: *Results of the dense mapping with 3 different stereo matching algorithms. From top to bottom, (a) BM, (b) SGBM, (c) ELAS.*

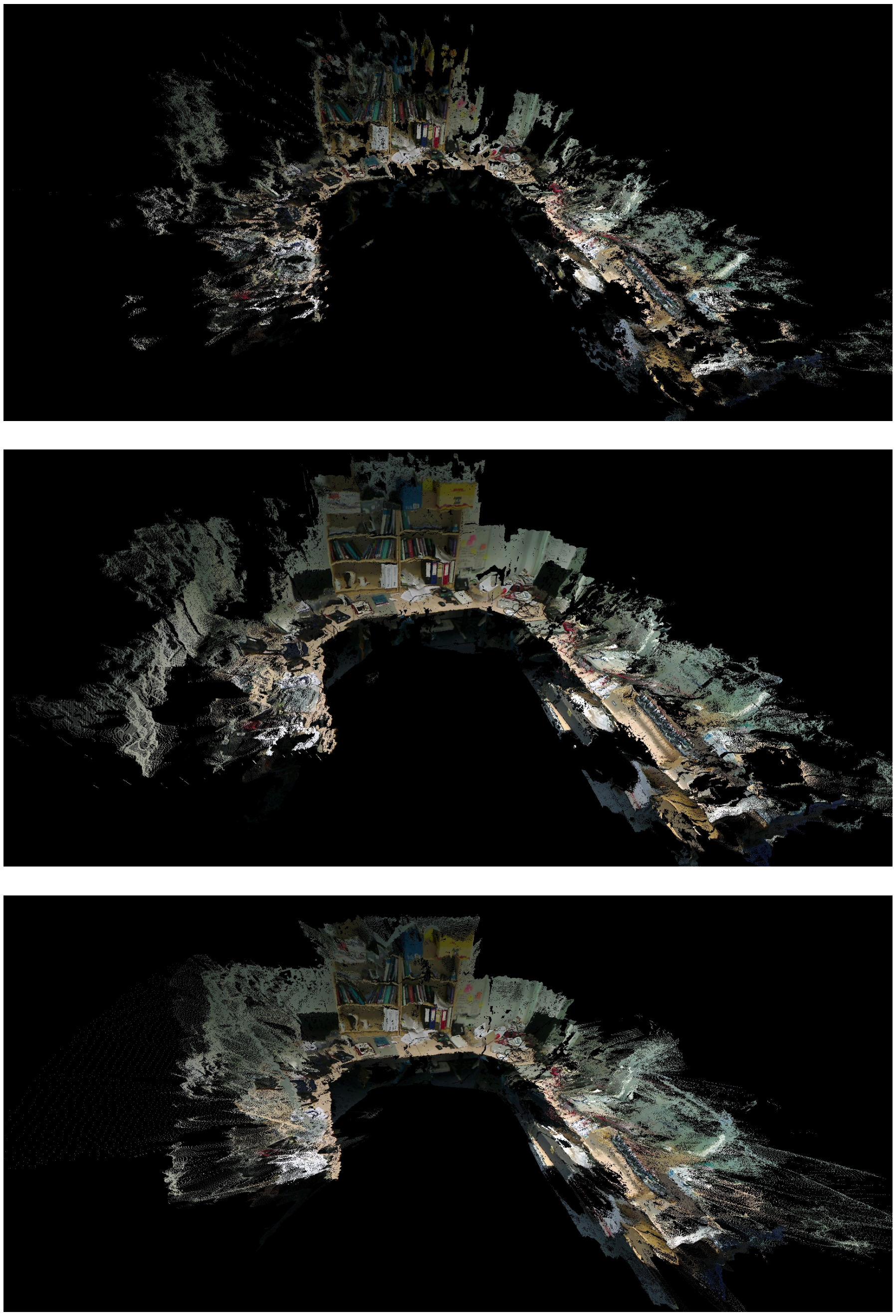
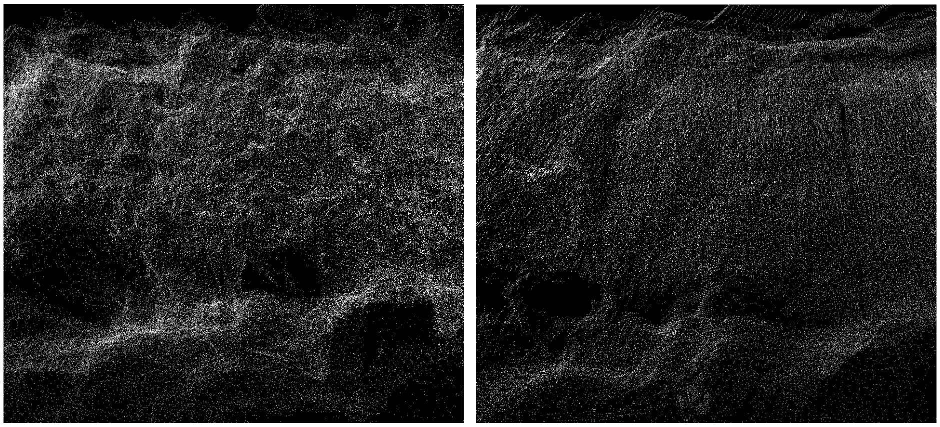


Figure 5.8: *Results of the dense mapping with 3 different stereo matching algorithms. From top to bottom, (a) BM, (b) SGBM, (c) ELAS.*



(a) **(b)**

Figure 5.9: *Map points corresponding to the poster on the wall. Points created by ELAS (a) are highly irregular and 'bumpy' compared to SGBM (b).*

Table 5.1: *Measured distances in centimeters between predetermined points on objects in three different the maps of the same scene.*

Obj.	Orig.	Map1	Map2	Map3	Total
1	179	181.35 ± 1.67	182.56 ± 1.59	178.89 ± 2.18	180.93 ± 2.26
2	41.5	41.32 ± 1.01	41.31 ± 3.08	40.36 ± 1.66	40.99 ± 1.88
3	58	59.94 ± 1.74	61.09 ± 0.53	55.76 ± 0.65	58.93 ± 2.61
4	72.5	73.36 ± 1.75	70.80 ± 1.31	68.89 ± 2.12	71.02 ± 2.47
5	32.5	31.64 ± 1.25	30.53 ± 1.16	30.14 ± 0.91	30.77 ± 1.18

5.4.3 Metric Measurement

In order to evaluate the precision of the metric measurements obtained from the reconstructed maps with the SGBM algorithm, we create 3 different maps of an environment in which five objects with known dimensions are placed. Then for each map, we manually tag the pre-measured points on the objects in 5 different zoom levels and orientations, and measure the distances between them. The average measurements and standard deviations for each map are given in Table.5.1 as well as the overall values.

The measurements show that the created maps are consistent in spatial domain and a few centimeters spatial resolution (≈ 2 cm) can be obtained. These maps are accurate enough to perform some initial measurements especially to support visual perception. However, due to the errors in the disparity maps and missing data in less-textured regions, very accurate measurements are not possible in the maps. But this is not surprising, since our goal is to create visually satisfying maps in real time instead of very accurate maps. Also we believe that higher resolution images and combination of stereo estimation methods can improve the performance.

5.5 Conclusion

In this chapter we introduced a dense 3D reconstruction method by using the estimated camera poses and stereo key-frames. The disparity maps are calculated using stereo key-frames and 3D map points are created. Then created map points are registered with the existing map points after they are refined with the tritree meshing algorithm. In order to evaluate the quality and the computational times of the maps, we tested three stereo matching algorithms and created various maps. The created maps are consistent in spatial domain and accurate enough to perform some initial measurements. Also they are visually satisfying and successfully represent the scenes. The overall reconstruction can be performed in approximately 3 fps and a few centimeters spatial resolution is achieved. For more accurate measurements, the quality of the maps needs to be improved which can be done by using (slower) multi-view stereo algorithms and higher resolution images.

Chapter 6

Multi-cue Hand Detection and Tracking for HCI

Bran thought about it. "Can a man still be brave if he's afraid?"
"That is the only time a man can be brave," his father told him.
George R.R. Martin, A Game of Thrones

6.1 Introduction

The majority of previous Augmented Reality (AR) research focused on pose tracking and virtual object registration for precise information overlay. With recent developments in wearable AR, the role of natural human computer interaction is becoming more and more important. However, considerably less research has been done on *interacting* with virtual objects in AR settings [106], especially for wearable systems. When wearable systems are considered, the world around the user can serve as a 3D interaction space in which Human-Computer-Interaction (HCI) can be performed. Necessary information such as tools, menus etc. for HCI can be visualized in this space. Moreover, auxiliary devices for interaction and control such as keyboard, mouse, joystick are not desired since they introduce extra complexity, weight and cost to wearable AR systems. The human hand is a natural input device to communicate and interact with the immediate surroundings. Therefore the utilization of the hands would be a natural choice for an interaction device.

We designed a new HCI methodology to go with a Head-Mounted-Display (HMD) based AR system with stereo cameras, which exploits the user's hands as an interaction device instead of other equipment. The pose of the hand is utilized to render graphical components onto. The 3D trajectory of the user's

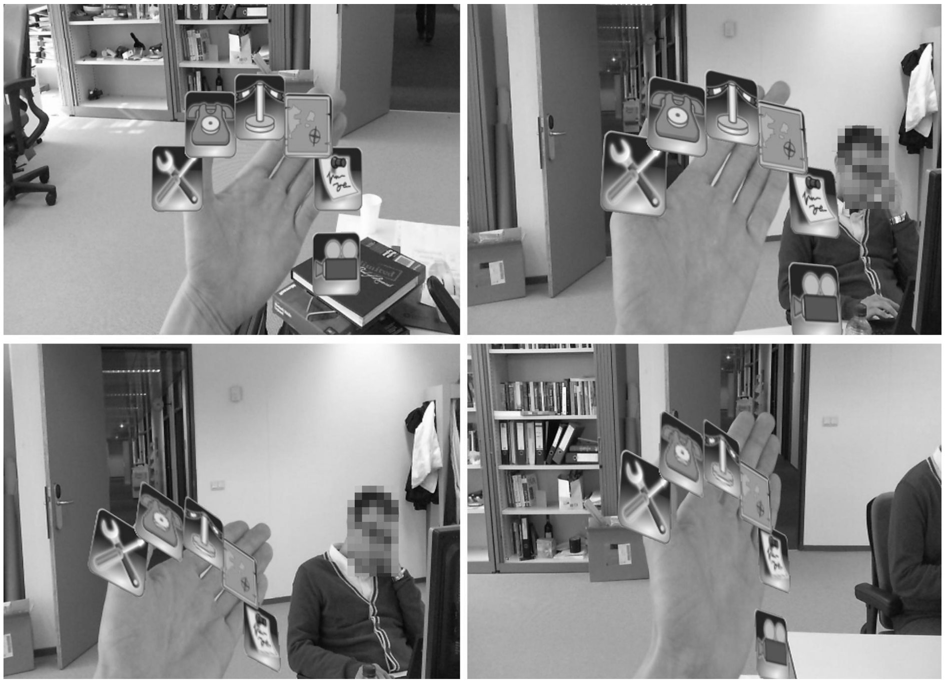


Figure 6.1: *Graphical user interface examples*

hands is converted to click and drag events which trigger widgets and control tools similar to traditional interfaces with mouse interaction from Windows; a.k.a Icons, Menus, Pointers (WIMP) interfaces [155]. This WIMP interface allows users to select and use tools, and supports many familiar widgets, such as menus, slider bars, text labels and icons (Fig.6.1). Our work broadens the freedom of the user by eliminating extra hardware or markers for interaction. With our system, the user can manipulate and control the AR system and interact with the environment.

For precise menu overlay and use, the user's hands need to be localized and tracked with high precision. In order to achieve robust, accurate and real-time operation we propose a hand detection and tracking method that probabilistically combines visual cues such as color, depth, curvilinearity and intensity. This enables robust and precise hand detection and tracking under challenging conditions. The 3D trajectory and the 3D pose are extracted from the tracked hand.

In the following section, related work is presented. Then, the framework is described, followed by a detailed description of the detection and tracking algorithms. Afterwards, we present experimental results. In the final section we conclude and discuss the future research.

6.2 Related Work

A wide variety of auxiliary equipment is employed for vision based HCI including color glove [202], LEDs [145], infrared transponders [130] and markers [153, 149]. However, utilization of additional devices decreases the mobility of the system while increasing the hardware complexity. Some of these systems limit the performance of the user since holding a device which occupies one of his hands restrains the freedom for interaction. It is highly impractical or even impossible for some users (such as surgeons) to utilize these devices during operations.

A more natural way of interaction, without any external hardware, is by using ones bare hands, which has already been investigated in many research related to AR and HCI. There have been many attempts to detect hands and gestures in a 2D image plane from a single color camera and use them as a mouse replacement. Most of these approaches use only the skin color information to segment the hands [108, 7, 106]. Although skin color detection is efficient and easy to implement, the robustness of such methods suffers from variances in lighting conditions and backgrounds in which the hand is not the only skin colored object. Another common approach to detect hands is background removal [172, 15], in which the motion in an image is detected by subtracting the background and hand detection is performed on foreground regions. Also the combination of skin color and motion detection (via background subtraction) is used [40, 138, 216]. However, background subtraction is not an option for a moving wearable setup since neither the foreground nor background are static. Detecting faces prior to the hand segmentation [119] is another popular method to improve the skin color model from detected faces and reduce the search space to the spatially close regions to the face. These methods are mostly designed for teleconferencing applications in which the user is positioned in front of the camera and hence they are not applicable to wearable systems where the cameras are not able to see the user's face.

Model based techniques are also used for hand segmentation and gesture recognition with monocular setups [49, 97]. In [49] the 3D pose of the hand is found from a single image by using texture and shading. However, the proposed method assumes a static camera and a light source that is fixed relative to the camera. In [97], flock tracking of hands is introduced. They combine KLT features [97] with color for tracking hands. However, the proposed method has the same drawbacks as other single camera methods when the background is of a similar color as the hands. Although 2D methods are useful for segmenting hands in a single view and can be extended to a stereo version, alone they are not able to provide robust 3D position of the hands or fingers and therefore are not suitable for our application. 3D model based hand shape and pose detection is also employed in [50, 51, 114] and they mostly rely on fitting articulated 3D hand models on 2D image features. However, they are either confined to desktop environments/static backgrounds and require a precise hand segmentation or are computationally too expensive for real-time applications.

Different sensors besides standard cameras that are used for hand segmentation and gesture recognition are Time-of-Flight (TOF) cameras [184, 69], thermal cameras [6, 140] and structured light systems [121]. But additional sensors to a standard stereo camera (which is also used for head-pose estimation in our framework) are highly undesirable since they introduce extra weight, power consumption and cost to the wearable setup.

Recognizing hand gestures by using stereo vision is studied in multiple research efforts. The disparity cue provided by the stereo camera can help to solve the problems of monocular methods such as background subtraction and occlusions. However, most researchers utilize the depth information after 2D segmentation to locate the 3D position of the hand [7], which still suffers from the drawbacks of 2D setups such as cluttered backgrounds. Assumptions such as static background/camera (for background subtraction) and a fixed distance between the camera and the user [209, 72], and the visibility of the user's face in the image [201, 45, 122] are also employed in different research efforts. However, these assumptions are not valid in our case where the camera is moving and the face of the user is not visible.

In [28, 152], other multi-cue hand detection and tracking methods are presented. However, both systems are monocular and don't utilize the depth information. In [152], Petersen and Stricker used a color tracker to estimate the main orientation of the hand which makes their system sensitive to lighting changes and similar colored backgrounds. In [28], the presented algorithm does not involve tracking and uses only parallel lines and curvature to detect hands.

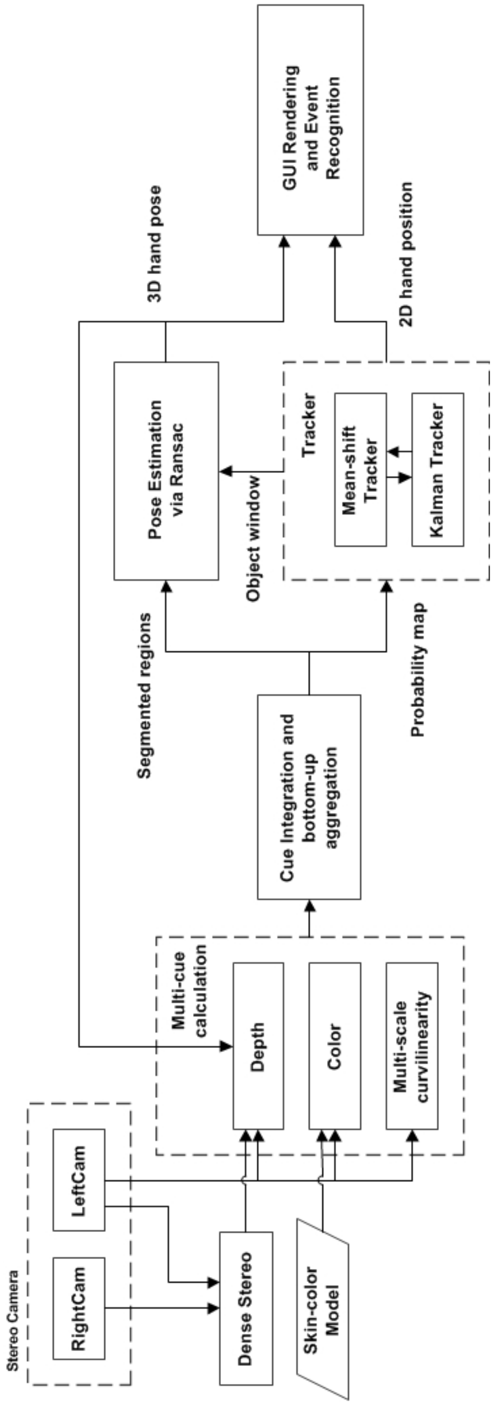


Figure 6.2: *System architecture.*

6.3 System Overview

In order to realize the GUI and the tools explained in Section 2.2.3, the user's hands must be detected and tracked. In this section we focus on the hand detection, tracking and pose estimation. The performance constraints and the requirements are derived from the GUI and we developed a system as depicted in Fig. 6.2.

The proposed method combines color, gradient, intensity and depth cues in order to detect hand/finger regions while achieving robust, accurate and real-time operation. The 6 DOF pose of the hand in the camera coordinate frame is approximated by a plane and its surface normal, and the center of the segmented hand is used to render a planar menu around it. Afterwards, the extracted regions are tracked between the frames in order to compute their trajectories. Finally, the trajectories and the pose are used to overlay necessary tools on the display for interaction or recognize the "click" gesture or perform drag. The proposed method enables the HCI system to be used in dynamic environments with a cluttered background and a moving camera, as the combination of multiple cues eliminates the drawbacks of such environments. Moreover, the pose estimation method is robust against outliers originating from erroneous disparity estimates. The best plane is approximated by using the pixels (inliers) classified as hand and have valid disparity values.

6.4 Hand Detection, Tracking and Pose Estimation

The local gradient magnitude and orientation are used to detect curvilinear structures on multiscale. These structures are strong candidates for the fingers of the user's hand. However, there are other structures that are similar to fingers which exist in the background. These false detected regions are eliminated by integrating color and intensity cues. Moreover, the depth cue is involved in the detection process to reduce the search space and decrease false detections. Afterwards, a score representing the similarity to a hand is given to each pixel by combining all the cues. A gradient directed parameter free probabilistic bottom-up aggregation method by using these scores is performed to group hand-like pixels and very small regions or outliers are eliminated.

6.4.1 Color cue

Prior to hand segmentation, a color model for skin-color is created. Many visual cues such as color, motion, texture and shape can be used to represent and segment human body parts. Among these features, color has an important role in detecting human presence in an environment due to the distinctiveness of skin-color and the fact that the major similarity between different skins lies in the chrominance rather than intensity [70].

Skin Color Model

Color is employed in detecting skin in an image, due to its robustness against scale variation, affine transformations and occlusion. Color space selection is an important issue for successful skin-color modeling [208] and various color spaces have been used to detect skin pixels [216, 168]. We preferred the HSV color space, and use hue (H) and saturation (S) values, as H and S are independent of the brightness (V) which makes it more robust compared to the RGB color space.

Most methods that have been proposed to create a skin-color model can be separated into two categories: parametric and non-parametric methods [87]. Jones and Rehg [87] compared histogram color models (non-parametric) with Gaussian mixture models (parametric) and found histogram models to be superior in accuracy and computational cost. Therefore, we decided on the non-parametric color model in our research and used a Parzen-window density estimation method [53] to eliminate the drawbacks of histogram based methods such as bin location selection and discontinuity.

A non-parametric color model can be created by Parzen-window method using training samples (pixels). Given training skin pixels $\mathbf{c}_i = (h_i, s_i)^T$ with color values (h_i, s_i), the value of the estimated probability density function at the point $\mathbf{c} = (h, s)^T$ (the probability of measuring a color value $\mathbf{c}$ from a skin pixel class w_s), $p(\mathbf{c}|w_s)$ is given as

$$p(\mathbf{c}|w_s) = \frac{1}{n_s} \sum_{i=1}^{n_s} \frac{1}{h_s} \varphi \left(\frac{\mathbf{c} - \mathbf{c}_i}{h_s} \right) \tag{6.1}$$

where n_s is the number of training skin pixels, $\varphi(\mathbf{u})$ is a window function and h_s is the smoothing parameter called the bandwidth. A bivariate Gaussian kernel would be a proper window function choice to represent the contribution of a training pixel to the density estimation since the contributions are equal to each other and the contribution of each training pixel decreases as the distance to the pixel increases. Therefore, $\varphi(\mathbf{u})$ is selected as

$$\varphi(\mathbf{u}) = \frac{1}{2\pi} e^{-\frac{\mathbf{u}^T \mathbf{u}}{2}} \tag{6.2}$$

Then the overall contribution of the training pixels $\mathbf{c}_i$ to the skin-color model and the estimate of the probability of measuring a particular color value $\mathbf{c}$ from a skin pixel is given as

$$p(\mathbf{c}|w_s) = \frac{1}{n_s} \sum_{i=1}^{n_s} \frac{1}{2\pi\sigma_s} e^{-\frac{(\mathbf{c}-\mathbf{c}_i)^T (\mathbf{c}-\mathbf{c}_i)}{2{\sigma_s}^2}} \tag{6.3}$$

where σ_s is the kernel bandwidth. The suitable kernel bandwidth σ_s must be carefully chosen since too small a bandwidth might result in too little a resolution while too large a bandwidth might lead to an over-smoothed density estimate.

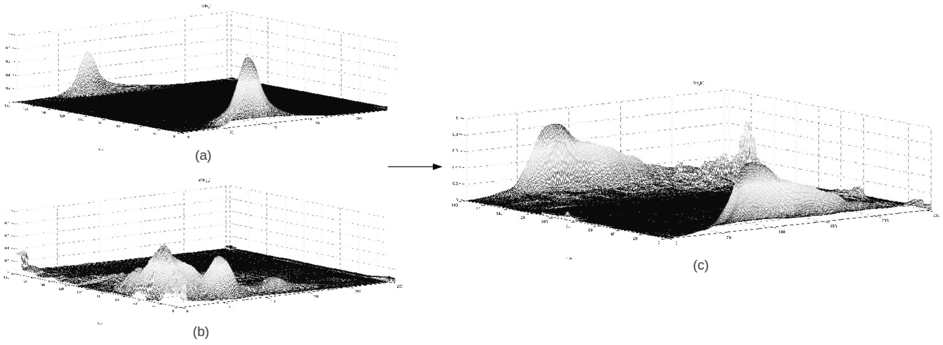


Figure 6.3: *(a) skin color model, (b) non skin-color model and (c) the probability of being a skin-color pixel based on (a) and (b).*

We use the standard deviation of the color values, σ_{hue} and σ_{sat} of the training skin pixels and select σ_s as

$$\sigma_s = min(\sigma_{hue}, \sigma_{sat}) \tag{6.4}$$

The same procedure is repeated with non-skin training pixels to create a non-skin model $p(\mathbf{c}|w_{ns})$.

$$p(\mathbf{c}|w_{ns}) = \frac{1}{n_{ns}} \sum_{i=1}^{n_{ns}} \frac{1}{2\pi\sigma_{ns}} e^{-\frac{(\mathbf{c}-\mathbf{c}_i)^T(\mathbf{c}-\mathbf{c}_i)}{2\sigma_{ns}{}^2}} \tag{6.5}$$

Training images captured under different illumination conditions are manually labeled as skin and non-skin pixels, and used in the training step. The final models are given in Fig.6.3.

Skin Color Score

The probability of being a skin-color pixel (probability mass function) is used as a score, S_c, for the final decision process. It is constructed by using the given skin and non-skin densities as a Bayesian posterior

$$S_c = P(w_s|\mathbf{c}) = \frac{p(\mathbf{c}|w_s)P(w_s)}{p(\mathbf{c}|w_s)P(w_s) + p(\mathbf{c}|w_{ns})P(w_{ns})} \tag{6.6}$$

where $P(w_s)$ and $P(w_{ns})$ can be defined as the ratio of the total number of skin and non-skin pixels in the training images

$$P(w_s) = \frac{n_s}{n_s + n_{ns}} \text{ and } P(w_{ns}) = \frac{n_{ns}}{n_s + n_{ns}} \tag{6.7}$$

Color Model Update

The created hand models are robust against the illumination changes up to some level. However, as the user moves and the lighting conditions change drastically, the created models fail to represent the hands. We increase the robustness of the system by employing a second group of models that are calculated by using the pixels that are classified as skin in the most recent N frames. Then the final score for the pixel $\mathbf{c}$ is defined as

$$\alpha P(w_s|\mathbf{c}) + (1-\alpha)P_N(w_s|\mathbf{c}) \tag{6.8}$$

This rule encapsulates the current dynamics of the environment as well as the initial conditions. The parameter α can be adjusted according to the level of expected variation. However, setting the value $\alpha = 0.8$ and $N = 5$ gives relatively good adaptation to illumination variation in our case.

6.4.2 Curvilinearity cue

A finger can be approximated as a curvilinear structure (a line with finite thickness) in an image since it is surrounded by almost parallel edges and it has a uniform color/intensity inside the finger. Moreover a line structure (a finger) can be represented at different scales depending on the distance between the camera and the hand. We use a bar-shaped profile to represent a 1D segment of a finger region as shown in Fig.6.4. The center of the line represents the center of the finger. An ideal line centered at x=0 with width w and height h can be represented as

$$f_b(x) = \begin{cases} h & |x| \leq w/2 \\ 0 & |x| > w/2 \end{cases} \tag{6.9}$$

The lines with the profile given above can be detected by convolving the image with the second derivative of the Gaussian smoothing kernel

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{6.10}$$

and locating the points where the convolution has a minimum [29]. In order to have a clear minimum at x=0, the center of the line, the sigma should be $\sigma \geq \frac{w}{2\sqrt{3}}$ [96]. Also, to keep the maximum effect of the filter inside the line (between -w/2 and w/2), σ should be less than w/2. Moreover, the convolution of the line profile with the first derivative of the Gaussian kernel will be 0 at $x = 0$ for all $\sigma > 0$.

If we write the second order derivative as a discrete derivative of the first derivative, then

$$g''(x) \approx k\frac{g'(x+w/2) - g'(x-w/2)}{2} \tag{6.11}$$

assuming that $\sigma = w/2$.

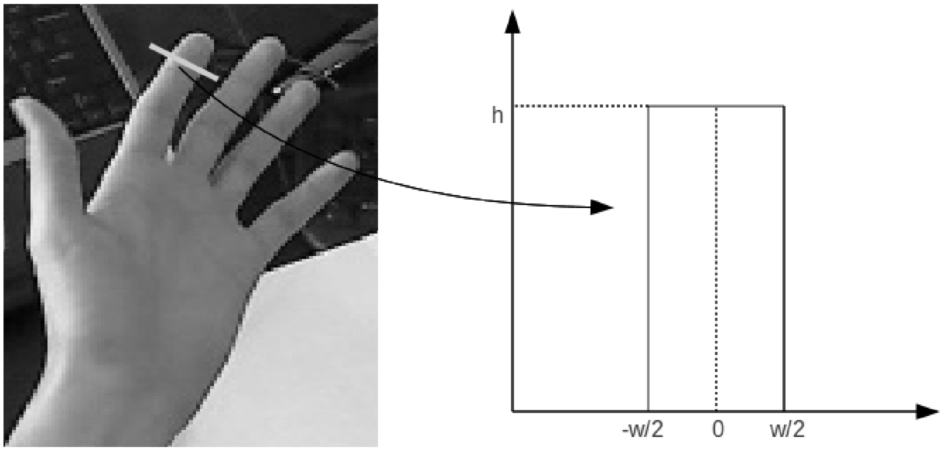
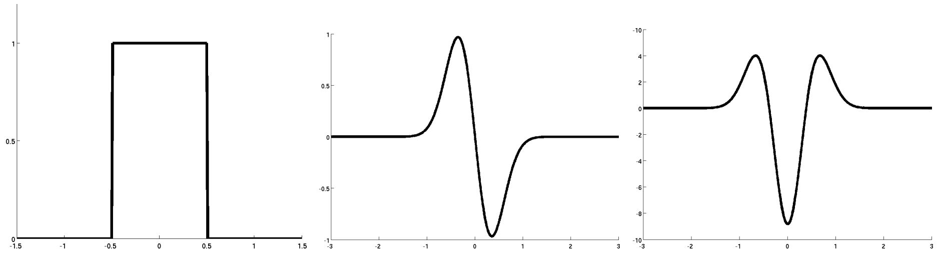


Figure 6.4: *Bar-shaped profile to represent a 1D segment of a finger.*



(a) **(b)** **(c)**

Figure 6.5: *Left to right: bar-shaped profile with w=1 and h=1, convolution with Gaussian derivative (*$\sigma = 0.5$*), convolution of (b) with the difference of two Dirac-delta functions.*

If we omit the constant $k/2$ and write the above equation as the convolution with the Dirac-delta function δ, then it becomes

$$g''(x) \approx g'(x) * (\delta(x + w/2) - \delta(x - w/2)) \tag{6.12}$$

The convolution with the first derivative of Gaussian detects the right and left edges (at $x = w/2$ and $x = -w/2$) of the bar-profile and the convolution with $\delta(x + w/2) - \delta(x - w/2)$ gives the minimum response at x=0 as shown in Fig.6.5.

A 2D line encapsulates the characteristics of a 1D bar-shaped profile in the direction perpendicular to the line. Therefore, we can extend the 1D approach to 2D images by using the gradient direction.

In order to detect the center of the 2D line section, we calculate the local orientation of the line and then apply the line-filter for a range of scales in this direction. The location with the minimum filter response is selected as the center of the line. The local orientation can be found by calculating the direction of the gradient vector at the edge pixels of the line. The direction of the gradient at pixel $\mathbf{x} = (x, y)^T$ is

$$\theta^{\mathbf{x}} = atan2\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right) \tag{6.13}$$

where $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial x}$ are the derivatives of the image I at pixel $\mathbf{x}$.

In order to decrease the computational load and make the algorithm faster for real-time operation, we approximate the first order derivative of Gaussian with the Sobel operator and utilize it to calculate the gradient of the image. We used the magnitude of the gradient vector instead of the first derivative of Gaussian filter output. The magnitude of the gradient detects the left and right edges of a line, but it lacks the direction information since it is always positive. Therefore, we search for the pixels where the convolution with $\delta(x+w/2)+\delta(x-w/2)$ gives the maximum response instead of a minimum.

We enforce the orientation information by using the direction of the gradient. We assume that the background is homogeneous and the finger is either a dark line on a bright background or a bright line on a dark background. Then the difference between the orientations of the edges (gradient vectors) around the line pixel should be π in an ideal case. To enforce the orientation constraint onto the final score we use an exponential function that has higher values as the difference between two edge orientations $\Theta_{\mathbf{l}}$ and $\Theta_{\mathbf{r}}$ at the edge pixels $\mathbf{l}$ and $\mathbf{r}$ is close to π. The orientation score $S_o(\mathbf{l}, \mathbf{r})$ is defined as

$$S_o(\mathbf{l}, \mathbf{r}) = e^{-\frac{(|\Theta_{\mathbf{l}}-\Theta_{\mathbf{r}}|-\pi)^2}{\sigma^2}} \tag{6.14}$$

where $\sigma = \pi/18$ in order to tolerate small variations in the orientation.

Moreover, the utilization of the gradient information (direction) and the explained line filter (with Dirac-delta functions) enables the system to focus on only the pixels with strong gradient magnitudes and decreases the computational load of the scale-space search. Instead of filtering the whole image, we only consider the pixels with strong gradient magnitude and the filter explained above is applied around those pixels. If the magnitude of the gradient in the direction of the normal of the gradient at pixel $\mathbf{x}$ is greater than a certain threshold, $\psi(\mathbf{x}) > \theta_\psi$, the line-filter response at pixel $\mathbf{y} = (\mathbf{x} + r\mathbf{s})$ is calculated as

$$\omega(\mathbf{x}, r) = \psi * (\delta(\mathbf{x}) + \delta(\mathbf{x} + 2r\mathbf{s})), -r_m < r < r_m \tag{6.15}$$

where ψ is the gradient magnitude image and $\mathbf{s}$ is the gradient vector at $\mathbf{x}$

$$\mathbf{s} = \nabla I(\mathbf{x}) \tag{6.16}$$

Therefore, the line-filter response for all the pixels that are $||r_m||$ far in the direction of the gradient from the pixels with high gradient values is calculated.

Considering the minimum and maximum distance of the hands, we set $r_m = 10$ in our experiments. Then the r value corresponding to the maximum line-filter response for pixel $\mathbf{x}$, r_s, is defined as

$$r_s = \arg\max_r (\omega(\mathbf{x}, r) S_o(\mathbf{x}, \mathbf{x} + 2r\mathbf{s})) \tag{6.17}$$

The center of the line is found at $(\mathbf{x} + r_s\mathbf{s})$. Then the curvilinearity score at the center of the line is defined as

$$w_{r_s} = \omega(\mathbf{x}, r_s) S_o(\mathbf{x}, \mathbf{x} + 2r_s\mathbf{s}) \tag{6.18}$$

To eliminate the sensitivity to noise and enforce the smoothness we consider the weight if the gradient of the center is smaller than the minimum gradient magnitude of the edges and the final curvilinearity score is

$$S_l = \begin{cases} w_{r_s} & \text{if } \psi(\mathbf{x} + r_s\mathbf{s}) < min(\psi(\mathbf{x}), \psi(\mathbf{x} + 2r_s\mathbf{s})) \\ 0 & otherwise \end{cases} \tag{6.19}$$

6.4.3 Depth cue

In order to utilize the depth cue for hand segmentation, we employ a simple and fast block matching stereo algorithm on the gray image pairs to extract disparity maps. However, the disparity maps are not dense and contain many uncertain disparities due to lack of texture in the scene and on the hand. Although accurate and dense disparities can be obtained by computationally expensive procedures such as belief propagation that require spatial smoothness priors [82], they are not suitable for our real-time application. In order to eliminate uncertain disparities, we perform a simple interpolation by assigning the closest valid disparity to the uncertain disparity.

We score the pixel $\mathbf{x}$ with disparity d at time $t + 1$ as

$$S_d = e^{\frac{(d-\mu_d^t)^2}{\left(\beta * \sigma_d^t\right)^2}} \tag{6.20}$$

where μ_d^{t-1} and σ_d^{t-1} are the mean and the standard deviation of the disparities of the hand pixels segmented in the previous image at time t. β is used to adjust the sensitivity of the score and set to 3 in our experiments.

When the hands are lost or previous depth values are not available, we use the a priori values of 0.7 meters and 0.3 meters for the mean and standard deviation, respectively. These values are originated from the simple assumption that the user's hands cannot be far from that distance anatomically.

6.4.4 Cue Integration and Bottom-up aggregation

The final score of each pixel $\mathbf{x}$, S_f is calculated by multiplying individual scores of $\mathbf{x}$,

$$S_f = S_c S_l S_d \tag{6.21}$$

Various scenarios in which the individual cues fail to represent the fingers while the combined score detects correctly are given in Fig.6.6.

Finally, a gradient oriented bottom-up aggregation is performed to segment pixels with similar scores into hand/finger regions and eliminate the outliers. Non-maximum suppression is done to detect local maxima and then region growing is performed in the orthogonal direction of the gradient, and the pixels that have similar score with the seed pixels are aggregated into finger segments. By using region growing, pixels that have relatively low probability of being hand or are uncertain because of stereo matching failure are considered again. If they are spatially close to the seed points and satisfy looser constraints, they are labeled as hand. This is done by using hysteresis thresholding similar to [29]. All segments are used for the following pose estimation step.

6.4.5 Hand Tracking

The trajectory of the hand is extracted by associating the pixel scores between consecutive frames. Normalized final scores of the pixels can be interpreted as the probability of being a hand/finger pixel and therefore the input image data can be converted into a probability distribution f_h.

$$f_h(\mathbf{x}) = \frac{S_f(\mathbf{x})}{\sum_{\mathbf{y}} S_f(\mathbf{y})} \tag{6.22}$$

Since this probability map comes for free with the proposed method, using the mean-shift algorithm [38] to find the mode of the distribution would be a natural choice. The mean-shift algorithm operates on the probability distribution and detects the closest mode (local maximum) by climbing in the direction of the distribution gradients. The mean-shift vector at pixel $\mathbf{x}$, $m(\mathbf{x})$ which points in the same direction as the distribution gradient $\nabla f_h(\mathbf{x})$ can be defined as

$$m(\mathbf{x}) = \frac{\sum_i \mathbf{x}_i f_h(\mathbf{x}_i)}{\sum_i f_h(\mathbf{x}_i)} - \mathbf{x} \tag{6.23}$$

where $\mathbf{x}_i$ are the pixels within the search window [38].

Starting from a guess for a local maximum, $\mathbf{y}_k$, the position of the local maximum can be found by climbing in the direction of $m(\mathbf{y})$ iteratively. The local maximum in the next iteration can be found as

$$\mathbf{y}_{k+1} = \mathbf{y}_k + m(\mathbf{y}_k) \tag{6.24}$$

The current position of the tracked hand can be used as the initial location of the search window in the next image. Although the size of the search window can be modified during operation, we have fixed the size to 40×40 since the operation range of the hands can be estimated. The detected mode corresponds to the new position of the hand. The overall method can be summarized as:

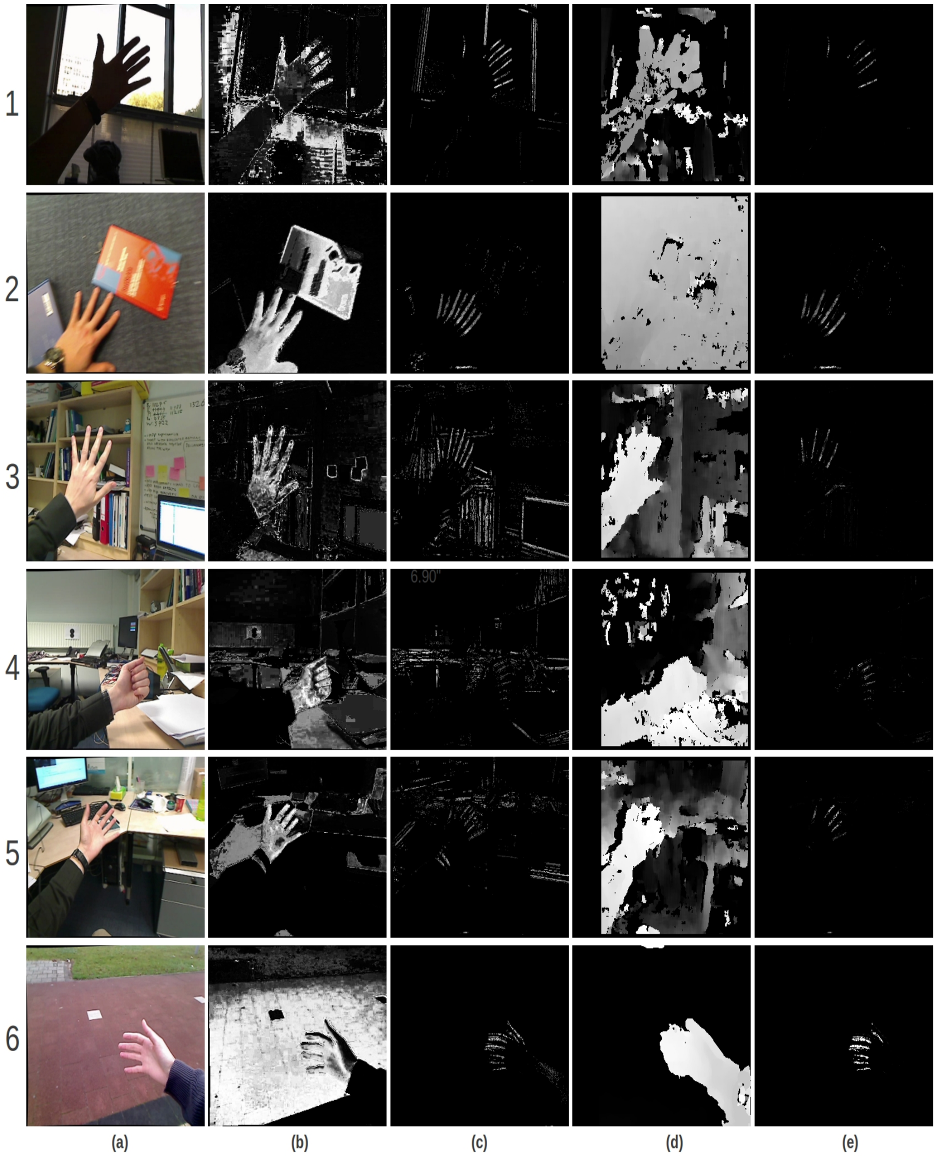


Figure 6.6: *(a) Input images (b) color cue (c) curvilinearity cue (d) depth cue (e) combined scores. In the first data set where the color is almost not available, weak color and depth scores are dominated by the curvilinearity cue. In the second set where the depth and color information is not distinctive, the curvilinearity cue again detects the fingers. In the third one, false lines in the background are eliminated by using the depth and color information. In the fourth data set, a closed hand is tested. In the last two, a weak color model is compensated by the depth and the curvilinearity information.*

1. Calculate the probability distribution from the input image by using the normalized scores of the pixels
2. Choose the previous position of the hand as the initial position of the search window
3. Calculate the mean-shift vector in the search window
4. Shift the search window to the new mean location
5. Repeat step 3 and 4 until the mean-shift vector is smaller than a certain threshold
6. Set the detected mode as the new location of the hand

Moreover, a Kalman filter [205] is employed together with the tracker to cope with situations where the hands are not visible, lost due to the image noise, motion blur or occlusions. A constant velocity model is used and the filter is updated with the measurements (detected 2D position of the hand) coming from the main tracker. Given the position vector $\mathbf{x} = (x, y)^T$ and velocity vector $\mathbf{v} = (v_x, v_y)^T$, the state equation is given by

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix}^{(t)} = \begin{pmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix}^{(t-1)} + \epsilon \tag{6.25}$$

and the output equation is

$$\mathbf{x}^{(t)} = (\mathbf{I} \quad \mathbf{0}) \begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix}^{(t)} + \nu \tag{6.26}$$

since $\mathbf{x}$ is directly observable. Here, $\mathbf{I}$ denotes a $2x2$ identity matrix, ϵ is the process noise and ν represents the measurement noise, respectively.

When the main tracker fails, Kalman predictions are used as the 2D positions of the hand for n frames. If the main tracker cannot track the hand for n frames, then the hand is labeled as "lost" and the maximum score in the image is searched to reinitialize the tracker. If the maximum score is smaller than a certain threshold or isolated from other high-scored pixels then the search is repeated in the next frame. We perform the search in the top-left quarter of the image in order to decrease false initializations. Therefore, a user can initiate the tracker by moving his hand to that corner. Some frames illustrating the hand tracker are given in Fig.6.7.

6.4.6 Hand Pose Estimation

The pose of the hands is important for interaction and realistic Graphical User Interface (GUI) rendering. We believe that approximating the segmented hands with planes is enough for many applications in which the hands are necessary as a

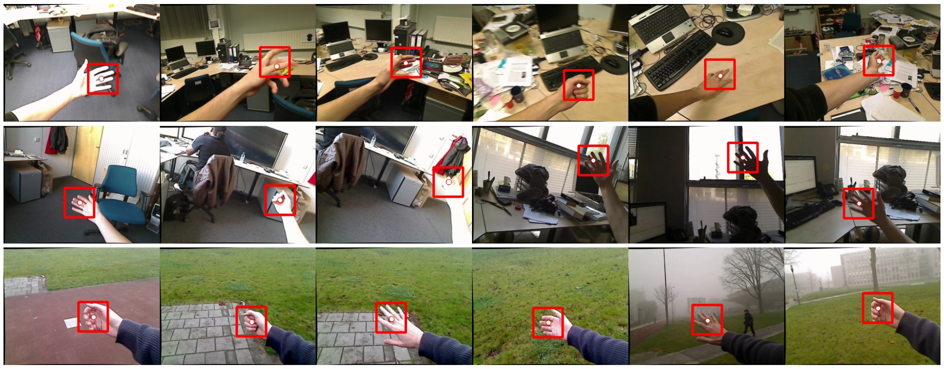


Figure 6.7: *Tracking results with non-stationary (head-mounted) camera under various indoor/outdoor conditions such as changing illumination and hand pose, cluttered and similar colored backgrounds.*

Figure 6.8: *Snapshots from each test sequence.*

3D pointing device to select or place virtual content and switch between different modes of operation.

We obtain the plane hypotheses for every hand region by using the RANSAC method [59]. Grown regions in the track window (as explained in the previous section) are used as the candidate regions to obtain the plane hypotheses and the sampling-scoring steps are performed on these regions.

- **Sampling.** A plane model can be obtained from three points in the disparity map sampled at random. Instead of selecting random three points from all the points inside the track window, we only consider the local maximums for better accuracy since they are more consistent than the other pixels.
- **Scoring.** Each model generated by the selected 3 points is evaluated against all the points inside the window. The quality of the plane is scored by the inlier count (number of points within a threshold distance Q_d to the plane). After repeating these steps M times, the plane with the highest score is selected and its normal is used as the z-axis of the coordinate frame located on the hand.

The average of 3D coordinates of all the inliers is defined as the 3D position of the hand and as the center of the coordinate frame located on the hand.

Table 6.1: *Properties of the test video sequences.*

seq. id	fast hand motion	outdoor scene	complex backgr.	changing handpose	# of frames
1	√	-	√	√	800
2	√	-	√	√	2500
3	√	√	-	√	600
4	-	-	-	-	1600
5	√	√	√	√	1300
6	-	√	√	-	1045
7	√	√	√	√	980

The image projections of the inliers are used to calculate the ellipse that fits these 2D points best in a least-squares sense. The major axis of the ellipse fit on the hand segments is used as the x-axis: the 3D coordinates of the pixels on the 2D major axis of the ellipse are used to find the 3D line. Afterwards, the y-axis is defined as the cross product of the x and z axes.

6.5 Results

We utilized 320x240 images and a standard block-matching algorithm to calculate the disparity images. Our system was tested on a laptop with a 2.7 GHz Core i7 processor (Intel Corporation, USA) and a Linux operating system. The system can operate at 25-30 fps depending on the background clutter.

6.5.1 Tracking

Various test scenarios and indoor/outdoor scenes, which demonstrate static and dynamic backgrounds and users, changing illumination conditions, presence of skin colored objects in the surrounding, etc., were recorded and tested by using the algorithm described in the previous section. The characteristics of the test sequences are given in Table.6.1 and the snapshots of the test sequences are given in Fig.6.8.

The main objective of our experiments is to compare the performance of our combined cue method with the individual cues as they are used in other algorithms. We compared the tracking with only color, only curvilinearity, color and curvilinearity, and combination of color, curvilinearity and depth. We didn't consider the depth only option since it is not possible to segment the hands from arms without further processing. We also compared our method with a standard CAMShift algorithm [23] in which the backprojection image is used as a probability distribution instead of our cues, and the tracker window size is adapted online. In all experiments, the same color model and camera calibration are used.

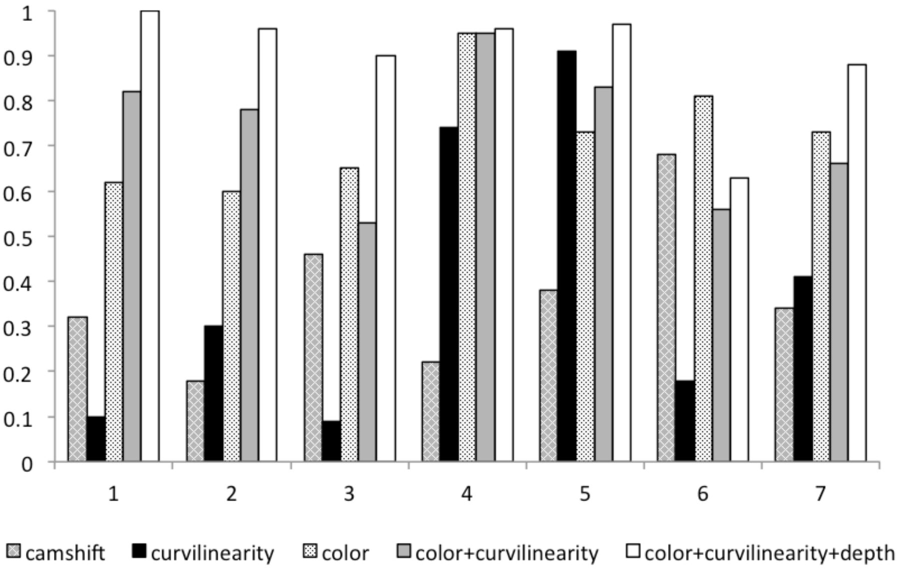


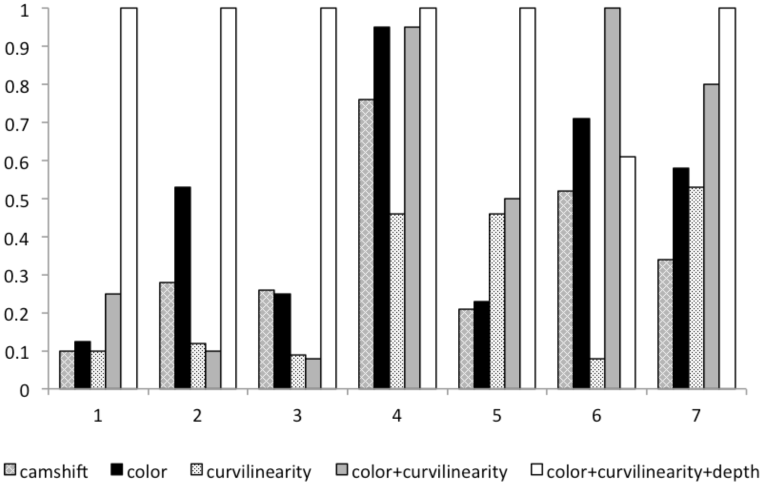
Figure 6.9: *Total number of successfully tracked frames with non-stationary (head-mounted) camera under various indoor/outdoor conditions such as changing illumination and hand pose, cluttered and similar colored backgrounds. Results are normalized w.r.t the best performing tracker.*

We also exclude the Kalman filter in order to compare the tracker performances independently from the filter predictions.

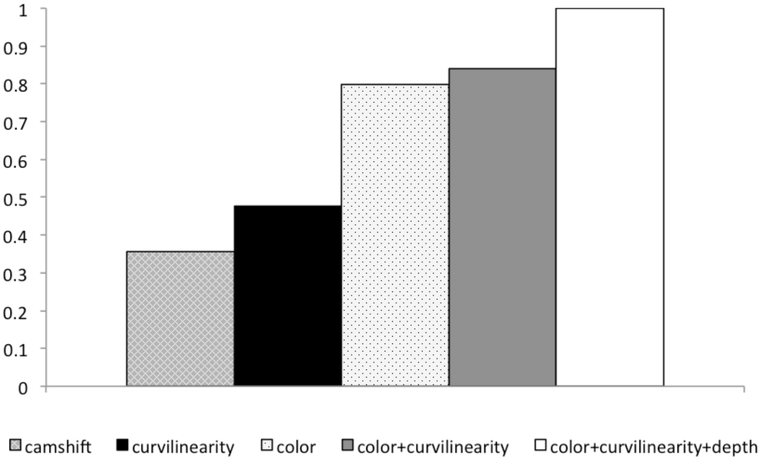
We define the tracking unsuccessful when the detected mode of the hand probability distribution (which represents the detected location of the hand) is not on the hand. Also the tracker is lost, if the maximum score in the search window is smaller than a certain size. When the tracker is lost, we re-initiate it by searching for the maximum scored region in the image. In order to decide on the success of the tracker, we visually inspected the videos and manually annotated the results.

In Fig.6.9, the total number of successfully tracked frames (normalized with respect to the best performing method) for each test sequence is given. The normalization is done by dividing the total number of successfully tracked frames of each tracker by the best performing tracker's total number of successfully tracked frames. Also the number of successfully tracked frames until the tracking is lost first time are given in Fig.6.10 (a). The overall performance of the trackers, the total number of successful frames in all the sequences, are given in Fig.6.10 (b). Again, the results are normalized with respect to the best performing method.

Our algorithm performs better than the standard CAMShift algorithm. When the background is simple and does not include any skin colored objects (as in se-



(a)



(b)

Figure 6.10: *The total number of successfully tracked frames (a) until the first lost frame, (b) in all the test sequences. Results are normalized w.r.t the best performing tracker.*

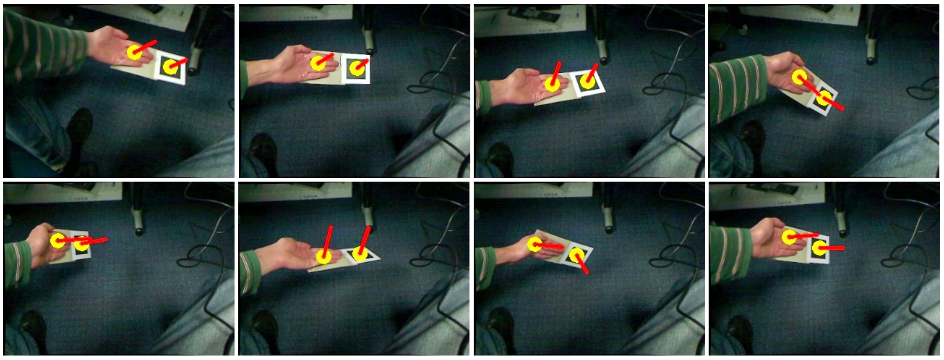


Figure 6.11: *Snapshots of the detected hand and the marker surface normals. The detected hand locations are shown in yellow while the normals are illustrated in red.*

quence 4) then the performances of all the trackers are good (except the CAMShift tracker since the arm of the user is naked and the tracker grows over it). However, when the background is complex and the lighting conditions are changing drastically, our cue combining algorithm significantly outperforms the others. Moreover, the utilization of the depth information as a cue reduces the search space when the tracker is lost. Therefore, reinitialization is more accurate and the overall performance of the tracker is improved, especially in cluttered scenes. Exploitation of the curvilinearity improved the performance of the tracker when the background is uniform and skin-colored. Also it makes the tracker focus on the fingers and copes with the situations in which the user's bare arms are visible to the camera. Only in test sequence 6, other trackers perform better than our tracker. This is due to the existence of highly curvilinear structures (bricks) with similar distance to the possible hand regions. Also the poor illumination conditions (overexposed images) in the sequence decreases the effect of the color cue.

As shown in Fig.6.10 (a), the combined method tracks the hands for a longer period without losing them and in general outperforms the other methods. Therefore, it is better suited for Human-Computer interaction since it minimizes the reinitializations.

6.5.2 Pose estimation

In order to evaluate the pose estimation results, we have mounted a marker onto the hand as shown in Fig.6.11 and used AR Toolkit [89] to detect the position, C_m, and the surface normal, θ_m, of the marker. Afterwards we calculate the position, C_h, and the surface normal, θ_h, of the hand with the proposed method. In order to compare the results of the two methods, we use the angle between the two normals and the difference between the positions as an error measure.

$$\epsilon_\theta = arccos(\theta_m \cdot \theta_h) \text{ and } \epsilon_c = |C_m - C_h| \tag{6.27}$$

We have tested our algorithm on 5 different data sets (10000 frames in total) and compared it to the marker based method. In Fig.6.12 the average errors, ϵ_θ, for different angles between the marker and the camera are given. The average error is close to 5 degrees when the angle between the normal of the marker and the camera principal axis is less than 63 degrees. Above 72 degrees the surface normal of the marker cannot be found in the test sequence and therefore there are no measurements for that range.

The angle between the marker normal and the camera, and between the hand normal and the camera are given for two test sequences in Fig.6.13. Also the distance of the hand and the marker to the camera in the direction of the principal axis are given in Fig.6.14. When the marker or the hand is lost in test sequences, the corresponding estimates are set to zero in the graphs. As shown in the figures, the estimates are very close to each other when the marker and the hand are detected. Also, the hand is detected successfully while the marker is lost in some frames.

6.6 Conclusion

In this chapter we proposed a multi-cue stereo based hand detection and tracking system which exploits depth, color and gradient information to track and estimate the 3D pose of the hands. Experimental results showed that the locations and the surface normal of the hands are determined accurately (approximately as good as a marker or better). Compared to the other tested methods our algorithm is less sensitive to the background color, robust against camera and background motion and leads to more accurate segmentation results. This robustness makes it better suited to operate in different environments than current algorithms, which make it suitable for head-mounted display based augmented reality applications.

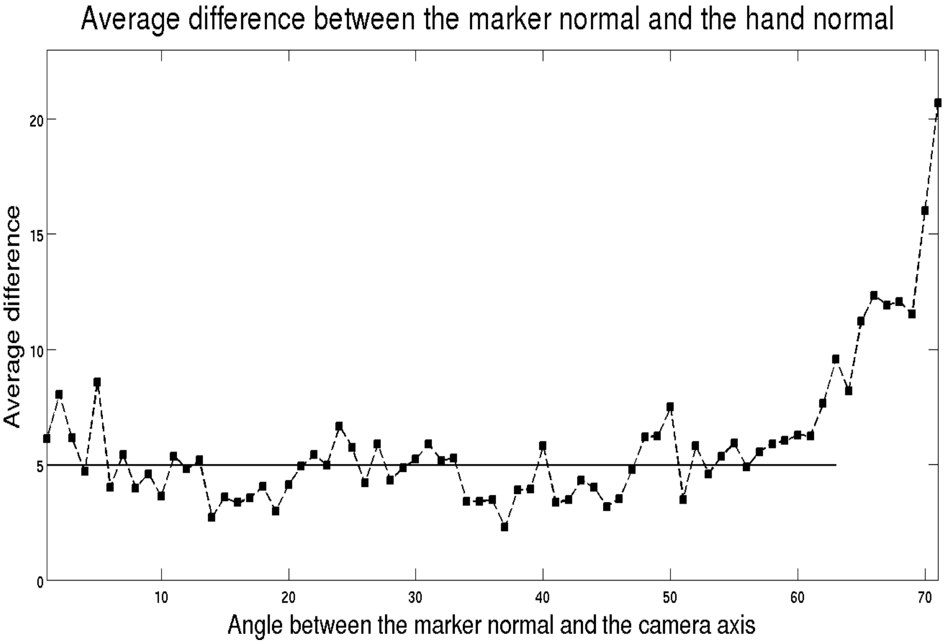


Figure 6.12: *Average error for different angles between the marker and the camera. In the experiments the marker cannot be detected above 72 degrees and therefore there are no measurements.*

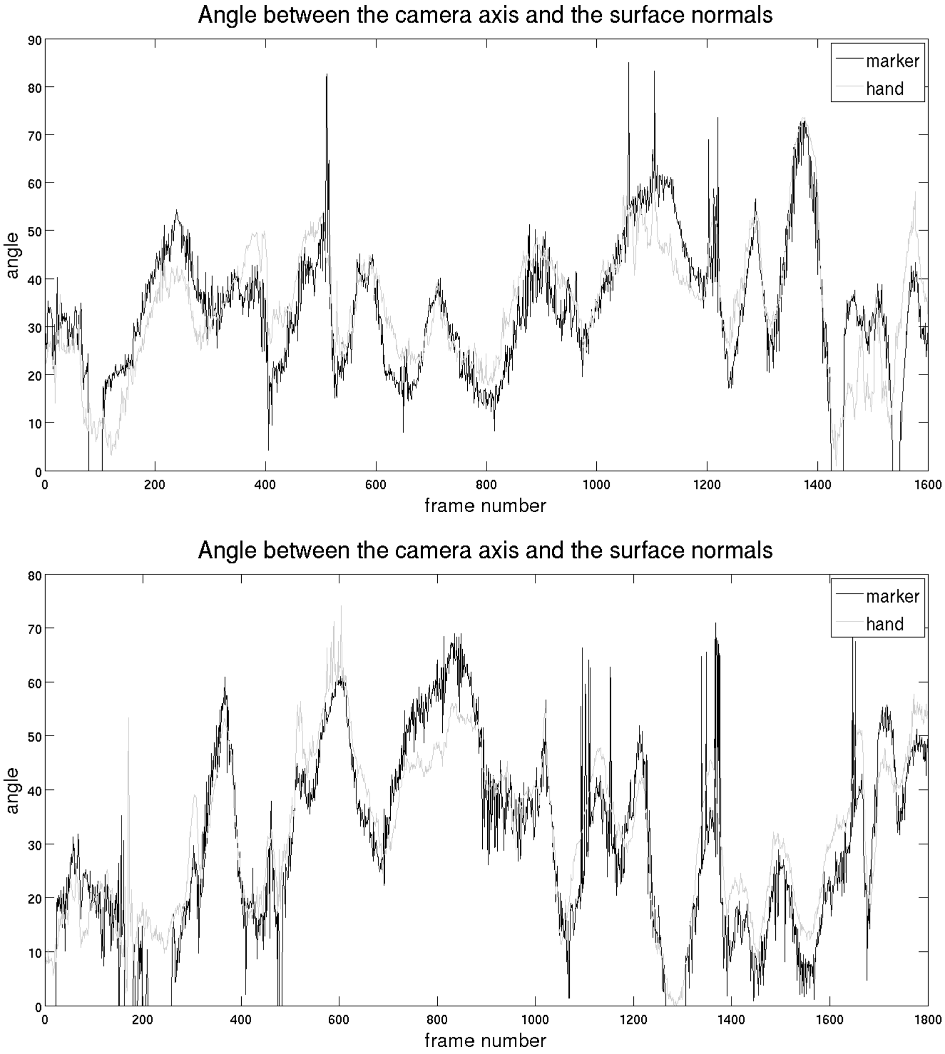


Figure 6.13: *The angle between the marker normal and the camera, and between the hand normal and the camera for sequences 1 and 2.*

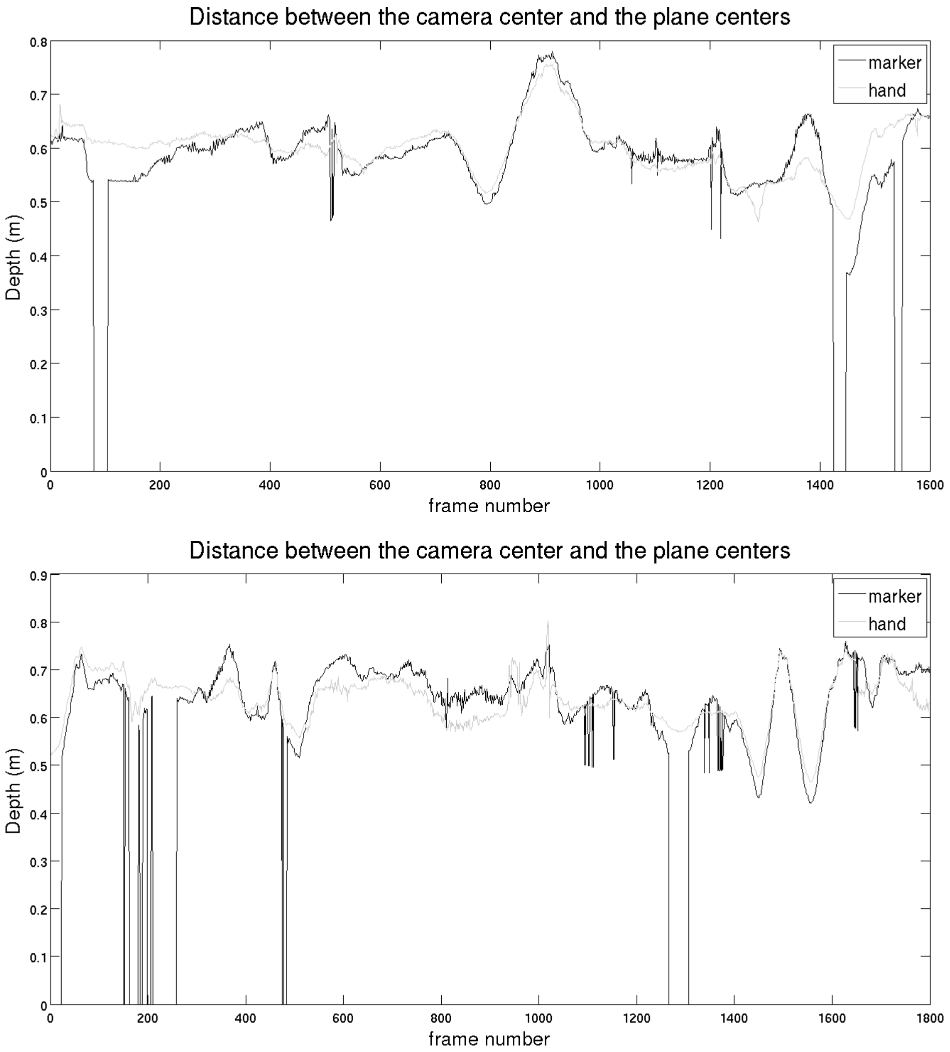


Figure 6.14: *The distance of the hand and the marker to the camera in the direction of the principal axis for sequences 1 and 2.*

Chapter 7

Conclusion

If you only read the books that everyone else is reading,
you can only think what everyone else is thinking.
Haruki Murakami, Norwegian Wood

This thesis has presented and demonstrated the application of computer vision algorithms to create a marker-less mobile/wearable Augmented Reality (AR) system that can provide assistive services to its users. The overall design of the system includes creating the necessary software modules and combining them with its hardware. The final design allows interaction between two parties and their environment while providing tools, guidance and information to the on-site and off-site users to perform their tasks independently from each other.

AR technologies already have quite a history in this field and many attempts have been made to use AR to create meaningful, immersive experiences incorporating humans and computers. The Delft University of Technology initiated AR research in 1999 with outdoor head-mounted optical see-through AR, fusing data from a GPS, a natural feature tracking camera, and an inertia tracker, using a desktop PC in a backpack [150, 151]. Soon a switch back was made to indoor AR based on markers and inertia tracker data via Caarls' system [26] in order to improve the accuracy of the head-pose estimation by the vision system and to obtain measurements on the static and dynamic accuracy of the estimates. In this work, we shift our focus from art and design [27] onto spatial analysis using multiple AR systems, more specifically we selected the Crime Scene Investigation (CSI) as our application domain.

We introduced the challenges originated from mobile AR and CSI and derived the requirements of a system that is necessary to overcome these challenges, such as marker-less, extensible tracking for augmenting a real scene with virtual objects, on-line and on-site scene structure capturing, hand gestures for user interface operation, remote connection to and collaboration with experts, a lightweight and affordable head-mounted display (HMD) and a wearable computer and real-time and robust operation.

In order to design a modular, lightweight and affordable system for the on-site user, in contrast with Caarls [27] who uses optical see-through display, we selected a video see-through head-mounted display (HMD), a wearable computer and two web-cameras as input sensors. The off-site users use standard computers and displays since they are confined to desktop environments and don't perform scene exploration.

We used Cinemizer OLED (Carl Zeiss AG, Germany) glasses which has 720p high-resolution displays with approximately a 32-degree field of view. We attached a stereo rig that is composed of two cameras that can output color images with 720p resolution at 30 fps. Together with the glasses the system weighs 170 grams. According to our discussions with the users, the resolution of the displays creates a satisfying user experience and the total weight of the system does not disturb the user during operation. So a comfortable, lightweight and affordable wearable hardware requirement is satisfied with our system. The cables of the glasses and cameras make it difficult to wear the system and the overall design can benefit a lot from a wireless headset, however there is still no affordable wireless headset available in the market.

Our software works in multiple threads to perform visual odometry, reconstruction and HCI tasks in parallel. Each module is separated from the other modules with the multi-threaded design which increases the performance and the robustness of the system. This system enables virtual content augmentation of a real scene by both the on-site and the remote user simultaneously. Two sides can communicate over the network and manipulate their perception by altering the scene. The system provides a remote connection for collaboration with experts. Although the evaluation of the remote connection is out of the scope of this thesis, we address this requirement in our design and a successful communication and virtual content augmentation is observed during tests. According to our discussions with the remote users, adding virtual content to on-site user's view helps the communication between the users.

In this thesis, we explain the mathematical background and briefly discuss most of the computer vision algorithms used for readers that are not familiar with the basic and advanced concepts.

One of the requirements is, a marker-less extensible camera tracking system, necessary to preserve the spatial consistency between the real and the virtual world, and to augment the scene with virtual objects. We aimed for a latency less than 50 milliseconds and the accuracy should be minimal such that the user doesn't feel any jitter during a tracking in medium-sized environments. Therefore, we present a visual odometry method that tracks the head-pose of the user in real time by using natural features available in the scene. We created a two stage real-time tracking method to minimize the drift while being robust against fast camera motions.

The visual odometry algorithm is able to track the pose of the camera at approximately 45-50 fps which results in approximately 20 milliseconds latency between the time that the tracker takes the measurements and the time that the graphics engine renders the augmented image. Therefore, the processing time requirement is satisfied. Also a sparse 3D map of the scene is created and extended during the operation to not to limit the motion of the user to small spaces. Utilization of bundle adjustment continuously refines the maps and reduces the errors accumulated during operation. Since the global error is minimized continuously, bigger and more accurate maps can be tracked for longer times. Also running the bundle adjuster in a separate thread enables refinement without degrading the performance of the tracker. Moreover, the adaptive feature detection algorithm ensures a fixed amount of features in varying environment conditions and helps the tracking for continuous and robust operation. In our experiments we compared our system with a marker-based method and showed that our algorithm is not sensitive to the distance between the camera and the marker (or initial map points), robust against different viewing angles, and leads to more accurate and consistent pose estimation results. Experimental results showed that the system is able to track the camera pose in medium sized environments such as rooms and corridors with varying environmental conditions. High accuracy and the fast operation of our system enables marker-less, extensible tracking for augmenting a real scene with virtual objects.

A dense 3D reconstruction of a crime scene is necessary to capture the initial structure of the scene and give a medium to the on-site and off-site users to perform measurements. As described in the requirements section in Chapter 1, reconstructed scenes are required to successfully represent the scenes, be visually satisfying and have a spatial resolution of a few centimeter. Also the maps should be created in near real-time as the user moves around and we set the maximum computational time of creating a dense map from a stereo image pair as 1 fps.

The proposed system utilizes the stereo images and the camera pose estimated during the motion of the on-site user, and creates a metric, dense 3D map of the scene in real time. This map is enhanced when the user sees the same part of the scene and extended when the user explores new parts of the scene. In our experiments, we tested three stereo matching algorithms and created various maps. In general, the created maps are consistent in spatial domain, visually satisfying and successfully represent the scenes. However, the quality of the maps decreases as the texture in the scene decreases. Also a couple of stereo pairs with inaccurate pose information can degrade the quality of the maps significantly. Therefore, the maps can be improved using a global (slower) multi-view stereo algorithms and higher resolution images. The overall reconstruction can be performed in approximately 3 fps and a few centimeters spatial resolution is achieved which satisfies our requirements. Moreover, since the created map is transmitted to remote experts, this on-line and on-site scene structure capturing establishes a common ground between the remote and co-located users to perform various tasks.

Human-Computer Interaction (HCI) is an important aspect of a mobile AR setup. Exploitation of the user's hands as an interaction device instead of other auxiliary equipment frees the user's hands for other tasks. Also, from the CSI point of view, the hands of the user should be free for interaction with the scene. e.g. securing evidences, when needed. Also, the user interface is required to operate in real time and in parallel to the tracking so that the user can interact with the system while the tracking is running in the background. Therefore, we introduced a HCI method that uses multiple cues such as depth, curvilinearity and color to detect and track the hands of the user.

The pose of the hand is calculated from the detected hand regions and used for displaying menus and utilizing various tools such as tagging and system control tools. The proposed system detects and tracks the hands successfully in real-time, is less sensitive to the background color, robust against camera and background motion and leads to more accurate segmentation results then the other tested methods. In our experiments, we also compared our system with a marker-based pose estimation method to evaluate the accuracy. The results showed that the locations and the surface normal of the hands are determined accurately (approximately as good as a marker or better). This robustness makes the HCI better suited to operate in different environments than current algorithms, which makes it suitable for head-mounted display based AR applications.

According to our knowledge, this is one of the first examples of a complete 3D stereo AR system that integrates 3D marker-less AR capabilities with dense reconstruction, remote collaboration and HCI in a carefully engineered way that can be applied to the CSI domain. The developed system was demonstrated in various events as illustrated in the Appendix.

7.1 System Evaluation in CSI

Although the main focus of this thesis is the computer vision aspects of an assistive AR system, we have used the domain of Crime Scene Investigation (CSI) as the use case to study AR and remote collaboration. We summarize the evaluation results of the system from the CSI point of view which is also extensively explained in [155]. Some interesting points mentioned are:

Within the experiment, the privacy of the HMD wearer turned out to be a major concern. Literally, every move is recorded, if someone missed something it can be retraced. One investigator remarked: "I consider this somewhat privacy intruding, but based on our car GPS they already know where we are all the time, which took some getting-used-to too". Another one remarked: "Some specialist teams already have mounted cameras specifically used for their protection". Compared to the above, our system not only records the investigation but also allows someone else to look at what you do from a remote location. A lot of the crime scene investigators that are the first to arrive on a crime scene handle the case from experience.

When they are asked to write a report about their on-going investigation they are generally incomplete in their reporting (according to the observers). Quote when they are asked to write about researching a certain trace they will tell you but never write that down. They prefer not to be bothered with desk work and the team has to trust their experience. Recording their activity can be either positive or negative. When they do not have to write everything down, every move is monitored. If the recording is perceived as a tool to monitor their work, it will encounter resistance. According to the observers, only with new recruits it will be possible to change the way of working. Furthermore, because our research is aimed at the pre-assessment on location, we do not specifically focus on investigators such as judges and court options. According to the observers a judge would appreciate the option to review the research of an investigator, the record of an investigation. Again privacy is at stake here but according to the observers it may improve the quality of the research in the end.

During our experiment it became clear that the presence of the expert is not experienced. In particular, when the expert creates or moves objects in the scene without verbal indications, disconnection between both participants occurs. There is a need to raise collaboration awareness to indicate the status of the other or where he is looking at, etc. (for instance by displaying an eyes avatar representing the gaze of the remote user) Our mediated reality system supports a whole new range of interaction and indicates a need for new mechanisms to achieve virtual presence.

Within our experiment the experts also remark that some investigators still would like to see the location with their own eyes. In our system the virtualized interaction results in only one or two predominant human senses. Further experiments are needed to explore how to support experts in their presence on the location and to understand what they miss when they use our mediated reality system with respect to be physically there?

One of the observers is also crime scene investigation trainer and liked to know if it was possible to setup a scenario, not for real investigation but for pure training purposes. He, e.g., asked: can you inflict virtual injury on a dummy or actor? Looking at the capabilities of the movie industry it seems possible, however, that it is currently still a manual procedure that uses pre-scanned objects, high dynamic range imagery and considerable render time. With the current system virtual objects stand out from real objects.

7.2 Future Research Directions

Although the work in this thesis presents a complete solution for a robust AR system to assist (collaborating) users, there are many unsolved problems that needs to be investigated in the future research. Some examples of these directions are:

Utilization of different sensors: After the introduction of depth cameras (RGB-D cameras) such as XBox 360 Microsoft Kinect to the market for affordable prices, depth sensors are becoming more and more popular. With the growing demand of consumers, smaller and more accurate depth sensors are becoming available with less power consumption and cheaper prices. In the near future, these sensors can be mounted on the headset to support the visual tracking, 3D reconstruction and human-computer interaction. Moreover, inertial sensors can also be reintroduced [27] to take the burden from the visual tracking especially during fast camera motions.

Combining real-time visual odometry and reconstruction with other computer vision algorithms to extend the application field: The overall system can be enhanced and its assistive capabilities can be increased by incorporating other algorithms such as face recognition, automating tagging etc.. The system can also be combined with object recognition modules to locate and recognize objects and guide the user towards them or inform the user about them. Such a combination can be used for educational purposes (for example in the museums to inform people about paintings, sculptures, etc.) as well as for guiding purposes (guiding people in unknown buildings). Such image processing combinations can extend the usability of the system and improve the performance.

New cameras and glasses with better hardware: The immersive aspects of the system can be improved by using better glasses with higher field-of-view, higher resolution, smaller size and lighter weight. Also new camera generations supply faster frame rates with higher image quality. Finally, the computers can be replaced by dedicated hardware that will provide a setup that is easier to carry. One possibility to support walking more freely would be to port our algorithms to a mobile platforms that can be worn on a belt.

New awareness forms: New methods of tele-presence becoming available with the technologies like our system. Therefore, either traditional patterns for computer-mediated interaction support awareness in mediated reality or rather new awareness forms need to be designed. The communication support of our current system needs further thought, especially the virtual presence of the expert in relation to the layman, privacy and security.

Incorporation of different senses: Besides the visual system, sound and smell can also be used for investigation and annotation of the scene. Smell can be used to detect various situations such as presence of gas or decomposing organic substances in the scene. Also the sound around the investigated scene can be localized and recorded. This extra information can be tagged in the maps, and further investigation can be done by utilizing them.

More complex HCI methods: Recognizing more complex hand gestures would enable more complicated interaction with the scene, upscaling the resolution with faster algorithms are therefore high on the list.

Support for multiple on-site and off-site users (collaborators): Establishing a common ground between multiple on-site and off-site users and enabling them to communicate and collaborate over a global map is necessary to support bigger teams of users (as in collaborative product design). In CSI domain, the first step towards that goal would be merging multiple maps that are created when multiple users roam in the same or different scene (such as the rooms of a house). Automatic merging of created maps can help multiple users to augment the same map and communicate with each other over a global map during operation.

Appendix A

Demonstration of the System

The developed system was demonstrated in various events, such as 'The Night of the Nerds, Amsterdam', 'The opening day of CSI labs, Den Haag' and 'Pauw & Witteman Show, Amsterdam' as illustrated in the following sections and mentioned in various newspapers such as 'De Telegraaf', 'De Pers', 'Metro' and magazines such as the 'New Scientist' and 'De Ingenieur'.

A.1 The night of the nerds

A.2 The opening of CSI Lab

A.3 Pauw and Witteman show

Figure A.1: *The night of the nerds in Amsterdam.*

Figure A.2: *The opening of Crime Scene Investigation Lab in Den Haag.*

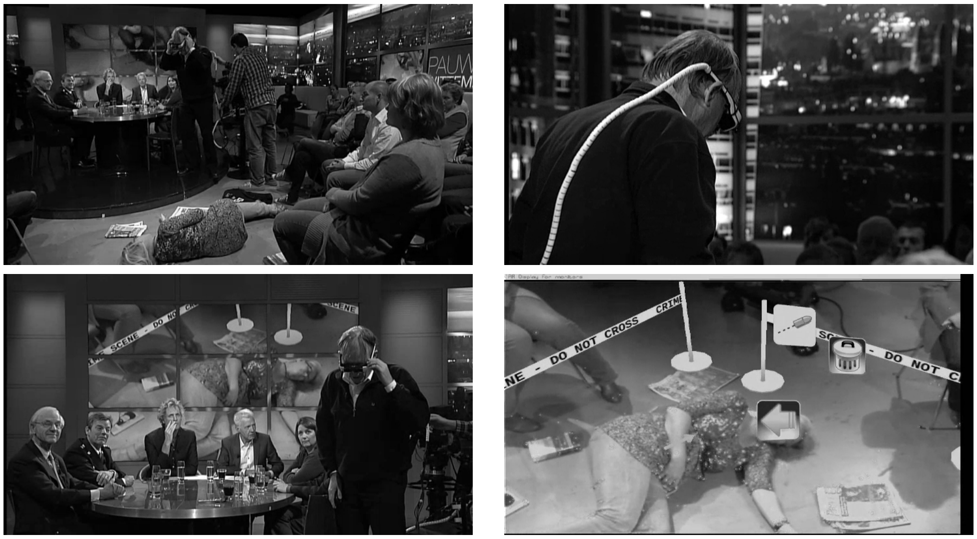


Figure A.3: *Pauw and Witteman show on TV.*

Bibliography

[1] A. Akbarzadeh, J.M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nister, and M. Pollefeys. Towards urban 3d reconstruction from video. In *3rd International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 1–8, June 2006.

[2] O. Akman. *Automation in Warehouse Development*, chapter 11 Object Recognition and Localisation for Item Picking, pages 153–161. Springer, 2011.

[3] O. Akman. *Automation in Warehouse Development*, chapter 13 Self-localisation and Map Building for Collision-Free Robot Motion, pages 177–189. Springer, 2011.

[4] O. Akman and P. Jonker. Computing saliency map from spatial information in point cloud data. In *Advanced Concepts for Intelligent Vision Systems*, volume 6474 of *Lecture Notes in Computer Science*, pages 290–299. Springer Berlin / Heidelberg, 2010.

[5] H. Alvarez, I. Aguinaga, and D. Borro. Providing guidance for maintenance operations using automatic markerless augmented reality system. In *Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 181–190, 2011.

[6] J. Appenrodt, A. Al-Hamadi, M. Elmezain, and B. Michaelis. Data gathering for gesture recognition systems based on mono color-, stereo color- and thermal cameras. In *Future Generation Information Technology*, Lecture Notes in Computer Science, pages 78–86. Springer Berlin / Heidelberg, 2009.

[7] A. A. Argyros and M. I. A. Lourakis. Real-time tracking of multiple skin-colored objects with a possibly moving camera. In *Proceedings of the European Conference on Computer Vision, ECCV '04*, pages 368–379. Springer, 2004.

[8] R. Azuma. Tracking requirements for augmented reality. *Communications ACM*, 36:50–51, Jul 1993.

[9] R. Azuma. A survey of augmented reality. *Presence*, 6:355–385, 1997.

[10] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, Nov 2001.

[11] R. Azuma and G. Bishop. Improving static and dynamic registration in an optical see-through HMD. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 197–204. ACM, 1994.

[12] C. Baber, P. Smith, J. Cross, D. Zasikowsk, and J. Hunter. Wearable technology for crime scene investigation. In *9th IEEE International Symposium on WearableComputers*, ISWC '05, pages 138–143, 2005.

[13] M. Bajura and U. Neumann. Dynamic registration correction in video-based augmented reality systems. *IEEE Computer Graphics and Applications*, 15:52–60, 1995.

[14] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '01*, volume 1, pages 1090–1097, 2001.

[15] S. Baraldi, A. D. Bimbo, L. Landucci, and A. Valli. wikitable: finger driven interaction for collaborative knowledge-building workspaces. In *Conference on Computer Vision and Pattern Recognition Workshop, CVPRW '06*, page 144, june 2006.

[16] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[17] M. Billinghurst, H. Kato, and I. Poupyrev. The Magicbook: a transitional AR interface. *Computers and Graphics*, 25(5):745–753, 2001.

[18] G. Bleser, M. Becker, and D. Stricker. Real-time vision-based tracking and reconstruction. *Journal of Real-Time Image Processing*, 2:161–175, 2007.

[19] G. Bleser and D. Stricker. Advanced tracking through efficient image processing and visual-inertial sensor fusion. In *IEEE Virtual Reality Conference, VR '08*, pages 137–144, March 2008.

[20] G. Bleser, H. Wuest, and D. Strieker. Online camera pose estimation in partially known and dynamic scenes. In *IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR '06*, pages 56 –65, Oct. 2006.

[21] J. Y. Bouguet. Pyramidal implementation of the Lucas Kanade feature tracker. Technical report, Intel, 2000.

[22] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '98, pages 648–655, 1998.

[23] G. R. Bradski. Computer video face tracking for use in a perceptual user interface. Technical report, Intel, 1998.

[24] D. C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971.

[25] A. M. Burton, D. Schofield, and L. M. Goodwin. Gates of global perception: forensic graphics for evidence presentation. In *13th annual ACM international Conference on Multimedia*, Multimedia '05, pages 103–111. ACM, 2005.

[26] J. Caarls. *Pose estimation for mobile devices and augmented reality.* PhD thesis, TU Delft, 2009.

[27] J. Caarls, P. Jonker, Y. Kolstee, J. Rotteveel, and W. van Eck. Augmented reality for art, design and cultural heritage: system design and evaluation. *EURASIP Journal on Image and Video Processing*, 2009:5:2–5:2, February 2009.

[28] M. B. Caglar and N. Lobo. Open hand detection in a cluttered single image using finger primitives. In *Conference on Computer Vision and Pattern Recognition Workshop, CVPRW '06*, page 148, June 2006.

[29] J. F. Canny. Finding edges and lines in images. Technical report, MIT Artificial Intelligence Laboratory, 1983.

[30] R. O. Castle, D. J. Gawley, G. Klein, and D. W. Murray. Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In *IEEE International Conference on Robotics and Automation, ICRA '07*, pages 4102–4107, April 2007.

[31] R. O. Castle, G. Klein, and D. W. Murray. Combining monoSLAM with object recognition for scene augmentation using a wearable camera. *Image and Vision Computing*, 28:1548–1556, November 2010.

[32] R.O. Castle and D.W. Murray. Object recognition and localization while tracking and mapping. In *8th IEEE International Symposium on Mixed and Augmented Reality, ISMAR '09*, pages 179–180, Oct. 2009.

[33] A. P. Chatzimichali, W. H. Gijselaers, M. S. R. Segers, P. van den Bossche, H. van Emmerik, F. E. Smulders, P. P. Jonker, and J. C. Verlinden. Bridging the multiple reality gap: Application of augmented reality in new product development. In *IEEE International Conference on Systems, Man, and Cybernetics, SMC '11*, pages 1914–1919, Oct. 2011.

[34] D. Chekhlov, M. Pupilli, W. Mayol-Cuevas, and A. Calway. Real-time and robust monocular SLAM using predictive multi-resolution descriptors. In *Proceedings of the Second International Conference on Advances in Visual Computing*, ISVC'06, pages 276–285. Springer Berlin / Heidelberg, 2006.

[35] Y. Cho and U. Neumann. Multi-ring color fiducial systems for scalable fiducial tracking augmented reality. In *IEEE Annual International Symposium on Virtual Reality*, page 212, 1998.

[36] H. H. Clark and D. Wilkes-Gibbs. Referring as a collaborative process. *Cognition*, 22:1–39, 1986.

[37] D. Claus and A. W. Fitzgibbon. Reliable fiducial detection in natural scenes. In *Proceedings of the European Conference on Computer Vision, ECCV'04*, volume 3024 of *Lecture Notes in Computer Science*, pages 469–480. Springer, 2004.

[38] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[39] A. I. Comport, E. Marchand, M. Pressigout, and F. Chaumette. Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):615–628, July 2006.

[40] T. Coogan, G. Awad, J. Han, and A. Sutherland. Real time hand gesture recognition including hand segmentation and tracking. In *Advances in Visual Computing*, Lecture Notes in Computer Science, pages 495–504. Springer Berlin / Heidelberg, 2006.

[41] J. Cross, C. Baber, and P. Smith. Multi-platform crime scene investigation field tool. In *11th IEEE International Symposium on Wearable Computers*, pages 55–62, Oct. 2007.

[42] Y. Cui, S. Schuon, D. Chan, S. Thrun, and C. Theobalt. 3D shape scanning with a time-of-flight camera. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR '10*, pages 1173–1180, June 2010.

[43] M. Cummins and P. Newman. Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, Nov. 2010.

[44] Cybermind. http://www.cybermindnl.com, (last visited 28.04.2012).

[45] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. *International Journal of Computer Vision*, 37:175–185, June 2000.

[46] A. J. Davison, W. W. Mayol, and D. W. Murray. Real-time localization and mapping with wearable active vision. In *2nd IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR '03*, pages 8–27, 2003.

[47] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[48] F. De Crescenzio, M. Fantini, F. Persiani, L. Di Stefano, P. Azzari, and S. Salti. Augmented reality for aircraft maintenance training and operations support. *IEEE Computer Graphics and Applications*, 31(1):96–101, Jan 2011.

[49] M. de La Gorce, N. Paragios, and D.J. Fleet. Model-based hand tracking with texture, shading and self-occlusions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR '08*, pages 1–8, June 2008.

[50] Q. Delamarre and O. Faugeras. Finding pose of hand in video images: a stereo-based approach. In *3rd IEEE International Conference on Automatic Face and Gesture Recognition*, pages 585–590, Apr 1998.

[51] T. Deselaers, A. Criminisi, J. Winn, and A. Agarwal. Incorporating on-demand stereo for real time recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR '07*, pages 1–8, June 2007.

[52] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):932 –946, Jul 2002.

[53] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.

[54] E. Eade and T. Drummond. Edge landmarks in monocular SLAM. In *British Machine Vision Conference, BMVC '06*, September 2006.

[55] E. Eade and T. Drummond. Scalable monocular SLAM. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '06*, volume 1, pages 469–476, june 2006.

[56] E. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. In *Photogrammetric Computer Vision, PCV '06*. ISPRS, 2006.

[57] S. Feiner, B. Macintyre, and T. Hollerer. Wearing it out: First steps toward mobile augmented reality systems. In *1st International Symposium on Mixed Reality, ISMR'99*, pages 363–377. Ohmsha (Tokyo)-Springer Verlag, 1999.

[58] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, October 2006.

[59] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications ACM*, 24(6):381–395, June 1981.

[60] E. Foxlin, Y. Altshuler, L. Naimark, and M. Harrington. Flighttracker: A novel optical/inertial tracker for cockpit enhanced vision. In *3rd IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 212–221, 2004.

[61] J. M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building Rome on a cloudless day. In *Proceedings of the 11th European Conference on Computer vision*, ECCV'10, pages 368–381, Berlin, Heidelberg, 2010. Springer-Verlag.

[62] P. Fua and V. Lepetit. Vision based 3D tracking and pose estimation for mixed reality. *In Emerging Technologies of Augmented Reality: Interfaces and Design*, 2007.

[63] S. R. Fussell, L. D. Setlock, and R. E. Kraut. Effects of head-mounted and scene-oriented video systems on remote collaboration on physical tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 513–520. ACM, 2003.

[64] D. Gallup, J. M. Frahm, and M. Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR '10*, pages 1418–1425, 2010.

[65] A. P. Gee, P. J. Escamilla-Ambrosio, M. Webb, W. Mayol-Cuevas, and A. Calway. Augmented crime scenes: virtual annotation of physical environments for forensic investigation. In *Proceedings of the 2nd ACM workshop on Multimedia in forensics, security and intelligence*, MiFor '10, pages 105–110. ACM, 2010.

[66] A. Geiger, M. Roser, and R. Urtasun. Efficient large-scale stereo matching. In *Asian Conference on Computer Vision*, Nov. 2010.

[67] A. Geiger, J. Ziegler, and C. Stiller. StereoScan: Dense 3D reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium, IV '11*, pages 963–968, June 2011.

[68] Y. Genc, S. Riedel, F. Souvannavong, C. Akinlar, and N. Navab. Marker-less tracking for AR: A learning-based approach. In *1st International Symposium on Mixed and Augmented Reality*, ISMAR '02, page 295, 2002.

[69] S. Ghobadi, O. Loepprich, F. Ahmadov, J. Bernshausen, K. Hartmann, and O. Loffeld. Real time hand based robot control using 2D/3D images. In *Advances in Visual Computing*, Lecture Notes in Computer Science, pages 307–316. Springer Berlin / Heidelberg, 2008.

[70] H. P. Graf, E. Cosatto, D. Gibbon, M. Kocheisen, and E. Petaja. Multimodal system for locating heads and faces. In *IEEE International Conference on Automatic Face and Gesture Recognition*, page 88, 1996.

[71] R. Grasset, P. Lamb, and M. Billinghurst. Evaluation of mixed-space collaboration. In *4th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '05, pages 90–99, 2005.

[72] R. Grzeszczuk, G. Bradski, M. H. Chu, and J.Y. Bouguet. Stereo based gesture recognition invariant to 3D pose and lighting. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR '00*, volume 1, pages 826–833, 2000.

[73] T. Guan, L. Duan, J. Yu, Y. Chen, and X. Zhang. Real-time camera pose estimation for wide-area augmented reality applications. *IEEE Computer Graphics and Applications*, 31:56–68, 2011.

[74] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: image understanding using qualitative geometry and mechanics. In *Proceedings of the 11th European Conference on Computer Vision*, ECCV'10, pages 482–496, Berlin, Heidelberg, 2010. Springer-Verlag.

[75] M. Habbecke and L. Kobbelt. A surface-growing approach to multi-view stereo reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR '07*, pages 1–8, 2007.

[76] R. M. Haralick, H. Joo, C. Lee, X. Zhuang, V. G. Vaidya, and M.B. Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1426–1446, Nov. 1989.

[77] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[78] R. I. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68:146–157, November 1997.

[79] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, second edition, 2004.

[80] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '97*, page 1106, 1997.

[81] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Proceedings of the 12th International Symposium on Experimental Robotics*, 2010.

[82] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:328–341, 2008.

[83] P. J. Huber. *Robust Statistics*. Wiley-Interscience, 1981.

[84] S. Irawati, S. Green, M. Billinghurst, A. Duenser, and H. Ko. "Move the couch where?" : developing an augmented reality multimodal interface. In *IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR '06*, pages 183–186, 2006.

[85] B. Jenkins. Laser scanning for forensic investigation. *Business and technology trends in capturing and managing existing-conditions data for engineering/construction/operations, SparView*, 3(21), 2005.

[86] B. Jiang, U. Neumann, and S. You. A robust hybrid tracking system for outdoor augmented reality. In *IEEE Virtual Reality*, pages 3–275, March 2004.

[87] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46:81–96, Jan 2002.

[88] C.M. Juan, E. Llop, F. Abad, and J. Lluch. Learning words using augmented reality. In *IEEE 10th International Conference on Advanced Learning Technologies, ICALT '10*, pages 422–426, July 2010.

[89] H. Kato and M. Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *2nd IEEE and ACM International Workshop on Augmented Reality*, page 85, 1999.

[90] L. Kerawalla, R. Luckin, S. Seljeflot, and A. Woolard. "Making it real": exploring the potential of augmented reality for teaching primary school science. *Virtual Reality*, 10:163–174, 2006.

[91] F. Kerger. *OGRE 3D 1.7 Beginner's Guide*. Packt Publishing, 2010.

[92] G. Klein and T. Drummond. Robust visual tracking for non-instrumental augmented reality. In *2nd IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 113–122, Oct. 2003.

[93] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. *IEEE / ACM International Symposium on Mixed and Augmented Reality*, 0:1–10, 2007.

[94] G. Klein and D. Murray. Improving the agility of keyframe-based slam. In *Proceedings of the 10th European Conference on Computer Vision*, ECCV '08, pages 802–815, Berlin, Heidelberg, 2008. Springer-Verlag.

[95] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *8th IEEE International Symposium on Mixed and Augmented Reality, ISMAR '09*, pages 83–86, Oct. 2009.

[96] T. M. Koller, G. Gerig, G. Szekely, and D. Dettwiler. Multiscale detection of curvilinear structures in 2-D and 3-D image data. In *5th International Conference on Computer Vision, ICCV '95*, pages 864 –869, Jun 1995.

[97] M. Kolsch and M. Turk. Fast 2D hand tracking with flocks of features and multi-cue integration. In *Conference on Computer Vision and Pattern Recognition Workshop, CVPRW '06*, volume 10, page 158, 2004.

[98] Y. Kolstee and W. van Eck. The augmented Van Gogh's: Augmented reality experiences for museum visitors. In *IEEE International Symposium on Mixed and Augmented Reality–Arts, Media, and Humanities*, volume 0, pages 49–52, 2011.

[99] Kopin. http://www.kopin.com, (last visited 28.04.2012).

[100] R. Kumar and A. R. Hanson. Robust methods for estimating pose and a sensitivity analysis. *CVGIP: Image Understanding*, 60:313–342, November 1994.

[101] T. Kurata, T. Oyabu, N. Sakata, M. Kourogi, and H. Kuzuoka. Tangible tabletop interface for an expert to collaborate with remote field workers. In *1st International Conference on Collaboration Technology*, pages 58–63, Tokyo, Japan, July 2005.

[102] T. Kurata, N. Sakata, M. Kourogi, H. Kuzuoka, and M. Billinghurst. Remote collaboration using a shoulder-worn active camera/laser. In *8th International Symposium on Wearable Computers, ISWC '04.*, volume 1, pages 62–69, Oct. 2004.

[103] D. Kurz and S. Benhimane. Gravity-aware handheld augmented reality. In *10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 111–120, 2011.

[104] H. Kuzuoka. Spatial workspace collaboration: a sharedview video support system for remote collaboration capability. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '92, pages 533–540. ACM, 1992.

[105] AR Lab. http://www.arlab.nl, (last visited 28.04.2012).

[106] M. Lee, R. Green, and M. Billinghurst. 3D natural hand interaction for AR applications. In *23rd International Conference on Image and Vision Computing New Zealand, IVCNZ '08*, pages 1–6, 2008.

[107] S. H. Lee, Y. I. Yoon, J. H. Choi, C. W. Lee, J. T. Kim, and J. S. Choi. AR squash game. In *IEEE International Conference on Information Reuse and Integration*, pages 579–584, 2006.

[108] T. Lee and T. Hollerer. Handy AR: Markerless inspection of augmented reality objects using fingertip tracking. *11th International Symposium on Wearable Computers*, 2007.

[109] Y. J. Lee and J. B. Song. Visual SLAM in indoor environments using autonomous detection and registration of objects. In *Multisensor Fusion and Integration for Intelligent Systems*, pages 301–314, 2009.

[110] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*, 81(2):155–166, February 2009.

[111] S. Lieberknecht, A. Huber, S. Ilic, and S. Benhimane. RGB-D camera-based parallel tracking and meshing. In *10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR '11*, pages 147–155, Oct. 2011.

[112] T. Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, pages 224–270, 1994.

[113] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[114] S. Lu, D. Metaxas, D. Samaras, and J. Oliensis. Using multiple cues for hand tracking and model refinement. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '03*, volume 2, pages 443–450, june 2003.

[115] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *7th International Joint Conference on Artificial Intelligence*, IJCAI '81, pages 674–679, 1981.

[116] S. G. Lukosch, R. Poelman, O. Akman, and P. Jonker. A novel gesture-based interface for crime scene investigation in mediated reality. In *Proceedings of the CSCW workshop on Exploring collaboration in challenging environments*, February 2012.

[117] Lumus. http://www.lumus-optical.com, (last visited 028.04.2012).

[118] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3D Vision: From Images to Geometric Models*. Springer Verlag, 2003.

[119] J. MacLean, R. Herpers, C. Pantofaru, L. Wood, K. Derpanis, D. Topalovic, and J. Tsotsos. Fast hand gesture recognition for real-time teleconferencing applications. In *IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 133–140, 2001.

[120] E. Mair, G. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *Proceedings of the 11th European Conference on Computer Vision*, volume 6312, pages 183–196. Springer Berlin / Heidelberg, 2010.

[121] S. Malassiotis, F. Tsalakanidou, N. Mavridis, V. Giagourta, N. Grammalidis, and M. G. Strintzis. A face and gesture recognition system based on an active stereo sensor. In *International Conference on Image Processing, ICIP '01*, pages 7–10, 2001.

[122] C. Manders, F. Farbiz, J. H. Chong, K. Y. Tang, G. G. Chua, M. H. Loke, and M. L. Yuan. Robust hand tracking using a skin tone and depth joint probability model. In *8th IEEE International Conference on Automatic Face Gesture Recognition, FG '08*, pages 1–6, 2008.

[123] S. Mann and W. Barfield. Introduction to mediated reality. *International Journal of Human Computer Interaction*, 15(2):205–208, 2003.

[124] E. Marchand and P. Bouthemy. A 2D-3D model-based approach to real-time visual tracking. *Image and Vision Computing*, 19:941–955, 2001.

[125] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761 – 767, 2004.

[126] B. Matei and P. Meer. A general method for errors-in-variables problems in computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR '00*, volume 2, pages 18–25, 2000.

[127] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. RSLAM: A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision*, 94:198–214, September 2011.

[128] C. Mei, G. Sibley, M. Cummins, P. M. Newman, and I. D. Reid. A constant-time efficient stereo SLAM system. In *British Machine Vision Conference, BMVC '09*, 2009.

[129] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J. M. Frahm, R. Yang, D. Nister, and M. Pollefeys. Real-time visibility-based fusion of depth maps. In *IEEE 11th International Conference on Computer Vision, ICCV '07*, pages 1–8, 2007.

[130] D. Merrill and P. Maes. Augmenting looking, pointing and reaching gestures to enhance the searching and browsing of physical objects. In *Pervasive Computing*, Lecture Notes in Computer Science, pages 1–18. Springer Berlin / Heidelberg, 2007.

[131] T. Moons, M. Vergauwen, and L. Van Gool. *3D reconstruction from multiple images.* 2008.

[132] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3D reconstruction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '06*, volume 1, pages 363–370, 2006.

[133] L. Naimark and E. Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *International Symposium on Mixed and Augmented Reality, ISMAR '02*, pages 27–36, 2002.

[134] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR '11*, pages 127–136, Oct. 2011.

[135] R. A. Newcombe and Davison A. J. Live dense reconstruction with a single moving camera. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '10*, pages 1498–1505, 2010.

[136] J. Newman, D. Ingram, and A. Hopper. Augmented reality in a wide area sentient environment. In *IEEE and ACM International Symposium on Augmented Reality*, pages 77–86, 2001.

[137] J. Newman, M. Wagner, M. Bauer, A. Macwilliams, T. Pintaric, D. Beyer, D. Pustka, F. Strasser, D. Schmalstieg, and G. Klinker. Ubiquitous tracking for augmented reality. In *International Symposium on Mixed and Augmented Reality, ISMAR '04*, pages 192–201, 2004.

[138] C. W. Ng and S. Ranganath. Real-time gesture recognition system and application. *Image and Vision Computing*, 20(13-14):993 – 1007, 2002.

[139] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '04*, volume 1, pages 652–659, June 2004.

[140] K. Oka, Y. Sato, and H. Koike. Real-time tracking of multiple fingertips and gesture recognition for augmented desk interface systems. In *5th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 429–434, May 2002.

[141] T. Olsson and M. Salo. Online user survey on current mobile augmented reality applications. In *10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR '11*, pages 75–84, Oct. 2011.

[142] OpenCV. http://opencv.willowgarage.com/wiki/, (last visited 25.07.2012).

[143] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):448–461, March 2010.

[144] J. Park, B. Jiang, and U. Neumann. Vision-based pose computation: robust and accurate augmented reality tracking. In *2nd IEEE and ACMInternational Workshop on Augmented Reality, IWAR '99*, pages 3–12, 1999.

[145] J. Park and Y. L. Yoon. LED-Glove based interactions in multi-modal displays for teleconferencing. In *16th International Conference on Artificial Reality and Telexistence–Workshops, ICAT '06*, pages 395–399, 2006.

[146] J Park, S. You, and U. Neumann. Natural feature tracking for extendible robust augmented realities. In *International Workshop on Augmented reality: placing artificial objects in real scenes*, IWAR '98, pages 209–217, 1999.

[147] Y Park, V Lepetit, and W. Woo. Multiple 3D object tracking for augmented reality. In *7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 117–120, 2008.

[148] Y. Park, V. Lepetit, and W. Woo. Texture-less object tracking with online training using an RGB-D camera. In *10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 121–126, 2011.

[149] S. Persa and P. Jonker. Human-computer interaction using real time 3D hand tracking. In *21st Symposium on Information Theory in the Benelux*, pages 25–26, 2000.

[150] S. Persa and P. P. Jonker. On positioning for augmented reality systems. In *1st International Symposium on Handheld and Ubiquitous Computing*, 1999.

[151] S. F. Persa. *Sensor Fusion in Head Pose Tracking for Augmented Reality*. PhD thesis, Delft University of Technology, 2006.

[152] N. Petersen and D. Stricker. Fast hand detection using posture invariant constraints. In *KI 2009: Advances in Artificial Intelligence*, volume 5803 of *Lecture Notes in Computer Science*, pages 106–113. Springer Berlin / Heidelberg, 2009.

[153] W. Piekarski and B. H. Thomas. ThumbsUp: Integrated command and pointer interactions for mobile outdoor augmented reality systems. In *10th International Conference on Human-Computer Interaction Symposium on Human Interface*, volume 1, pages 1223–1227, 2003.

[154] K. Pirker, M. Ruther, H. Bischof, and G. Schweighofer. Fast and accurate environment modeling using three-dimensional occupancy grids. In *IEEE International Conference on Computer Vision Workshops*, pages 1134–1140, Nov. 2011.

[155] R. Poelman, O. Akman, S. Lukosch, and P. Jonker. As if being there: Mediated reality for crime scene investigation. In *ACM Conference on Computer Supported Cooperative Work, CSCW '12*, 2012.

[156] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, September 2004.

[157] M. Pressigout and E. Marchand. Hybrid tracking algorithms for planar and non-planar structures subject to illumination changes. In *IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR '06*, pages 52–55, Oct. 2006.

[158] M. Pressigout and E. Marchand. Real-time 3D model-based tracking: combining edge and texture information. In *IEEE International Conference on Robotics and Automation, ICRA '06*, pages 2726–2731, May 2006.

[159] M. Pupilli and A. Calway. Real-time camera tracking using a particle filter. In *Proceedings of British Machine Vision Conference, BMVC '05*, pages 519–528, 2005.

[160] J. Quarles, S. Lampotang, I. Fischler, P. Fishwick, and B. Lok. A mixed reality approach for merging abstract and concrete knowledge. In *IEEE Virtual Reality Conference, VR '08*, pages 27–34, March 2008.

[161] G. Reitmayr and T.W. Drummond. Going out: robust model-based tracking for outdoor augmented reality. In *IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR '06*, pages 109–118, Oct. 2006.

[162] Gartner Research. http://www.gartner.com/hc/images/215650-0001.gif, (last visited 07.04.2012).

[163] J. P. Rolland, Y. Baillot, and A. A. Goon. *A Survey of Tracking Technology for Virtual Environments*. 2001.

[164] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision*, volume 1, pages 430–443. Springer Berlin / Heidelberg, May 2006.

[165] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:105–119, 2010.

[166] E. Royer, M. Lhuillier, M. Dhome, and T. Chateau. Localization in urban environments: monocular vision compared to a differential GPS sensor. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '05*, volume 2, pages 114–121, June 2005.

[167] M. Sarkis, W. Zia, and K. Diepold. Fast depth map compression and meshing with compressed tritree. In *9th Asian Conference on Computer Vision*, volume 5995 of *Lecture Notes in Computer Science*, pages 44–55. Springer Berlin / Heidelberg, 2010.

[168] D. Saxe and R. Foulds. Toward robust skin identification in video images. In *2nd International Conference on Automatic Face and Gesture Recognition*, pages 379–384, October 1996.

[169] A. Saxena, M. Sun, and A.Y. Ng. Make3D: Learning 3D scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):824 –840, May 2009.

[170] G. Schall, E. Mendez, and D. Schmalstieg. Virtual redlining for civil engineering in real environments. In *7th IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR '08*, pages 95–98, Sept. 2008.

[171] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, April 2002.

[172] M. Schlattman and R. Klein. Simultaneous 4 gestures 6 DOF real-time two-hand tracking without any markers. In *ACM symposium on Virtual realitysoftware and technology, VRST '07*, pages 39–42. ACM, 2007.

[173] D. Schofield. Animating and interacting with graphical evidence : Bringing courtrooms to life with virtual reconstructions. In *International Conference on Computer Graphics, Imaging and Visualisation, CGIV '07*, pages 321 –328, Aug. 2007.

[174] S. Se and P. Jasiobedzki. Instant scene modeler for crime scene reconstruction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR '05*, CVPR '05, page 123, 2005.

[175] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The International Journal of Robotics Research*, 21(8):735–758, 2002.

[176] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '06*, volume 1, pages 519–528, June 2006.

[177] J. R. Shewchuk. Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. In *1st ACM Workshop on Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996.

[178] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, CVPR '94*, 1994.

[179] G. Simon, A.W. Fitzgibbon, and A. Zisserman. Markerless tracking using planar structures in the scene. In *IEEE and ACM International Symposium on Augmented Reality, ISAR '00*, pages 120–128, 2000.

[180] I. Skrypnyk and D. G. Lowe. Scene modelling, recognition and tracking with invariant image features. In *3rd IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR '04*, pages 110–119, Nov. 2004.

[181] P. Smith, I. Reid, and A. Davison. Real-time monocular SLAM with straight lines. In *British Machine Vision Conference*, volume 1, pages 17–26, September 2006.

[182] L. Soler, S. Nicolau, J. Schmid, C. Koehl, J. Marescaux, X. Pennec, and N. Ayache. Virtual reality and augmented reality in digestive surgery. In *3rd IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR '04*, pages 278–279, 2004.

[183] Sony. http://www.sony.co.uk/hub/personal-3d-viewer, (last visited 28.04.2012).

[184] S. Soutschek, J. Penne, J. Hornegger, and J. Kornhuber. 3-D gesture-based scene navigation in medical imaging applications using time-of-flight cameras. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPRW '08*, pages 1–6, 2008.

[185] A. Stafford, W. Piekarski, and B. H. Thomas. Implementation of god-like interaction techniques for supporting collaboration between outdoor ar and indoor tabletop users. In *IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR '06.*, pages 165–172, Oct. 2006.

[186] K. M. Stanney and R. S. Kennedy. Aftereffects from virtual environment exposure: How long do they last? In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 1998.

[187] D. Stricker, G. Klinker, and D. Reiners. A fast and robust line-based optical tracker for augmented reality applications. In *International Workshop on Augmented Reality:placing artificial objects in real scenes*, IWAR '98, pages 129–145, 1999.

[188] J. Stuhmer, S. Gumhold, and D. Cremers. Real-time dense geometry from a handheld camera. In *32nd DAGM conference on Pattern recognition*, pages 11–20, Berlin, Heidelberg, 2010. Springer-Verlag.

[189] R. Subbarao, P. Meer, and Y. Gene. A balanced approach to 3D tracking from image streams. In *4th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR '05*, pages 70–78, Oct. 2005.

[190] W. Sun and J. Cooperstock. An empirical evaluation of factors influencing camera calibration accuracy using three publicly available techniques. *Machine Vision and Applications*, 17:51–67, 2006.

[191] R. Szeliski. *Computer Vision: Algorithms and Applications.* Springer, 2010.

[192] CSI the Haag. http://www.csithehague.com, (last visited 28.04.2012).

[193] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Vision Algorithms: Theory and Practice, LNCS*, pages 298–375. Springer Verlag, 2000.

[194] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, 1987.

[195] M. Tuceryan, D. S. Greer, R. T. Whitaker, D. E. Breen, C. Crampton, E. Rose, and K. H. Ahlers. Calibration requirements and procedures for a monitor-based augmented reality system. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):255–273, Sep. 1995.

[196] L. Vacchetti, V. Lepetit, and P. Fua. Combining edge and texture information for real-time accurate 3D camera tracking. In *3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '04, pages 48–57, 2004.

[197] S. Vogt, A. Khamene, F. Sauer, and H. Niemann. Single camera tracking of marker clusters: Multiparameter cluster optimization and experimental verification. In *1st International Symposium on Mixed and Augmented Reality*, ISMAR '02, page 127, 2002.

[198] Vuzix. http://www.vuzix.com, (last visited 28.04.2012).

[199] D. Wagner, T. Pintaric, F. Ledermann, and D. Schmalstieg. Towards massively multi-user augmented reality on handheld devices. In *3rd International Conference on Pervasive Computing*, Pervasive'05, pages 208–219, Berlin, Heidelberg, 2005. Springer-Verlag.

[200] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *7th IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR '08.*, pages 125–134, Sept. 2008.

[201] Q. Wang, X. Chen, and W. Gao. Skin color weighted disparity competition for hand segmentation from stereo camera. In *Proceedings of the British Machine Vision Conference*, pages 1–11, 2010.

[202] R. Y. Wang and J. Popovic. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics*, 28(3), 2009.

[203] G. Q. Wei and S. D. Ma. Implicit and explicit camera calibration: Theory and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:469–480, 1994.

[204] T. Weise, T. Wismer, B. Leibe, and L. Van Gool. In-hand scanning with online loop closure. In *IEEE 12th International Conference on Computer Vision Workshops*, pages 1630–1637, Oct. 2009.

[205] G Welch and G. Bishop. An introduction to the Kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[206] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965–980, Oct. 1992.

[207] H. Wuest, F. Vial, and D. Strieker. Adaptive line tracking with multiple hypotheses for augmented reality. In *4th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 62–69, Oct. 2005.

[208] M. H. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, January 2002.

[209] G. Ye, J. J. Corso, and G. D. Hager. Gesture recognition using 3D appearance and motion features. In *Conference on Computer Vision and Pattern Recognition Workshop, CVPRW '04*, page 160, 2004.

[210] Zeiss. http://cinemizer.zeiss.com, (last visited 28.04.2012).

[211] X. Zhang, S. Fronz, and N. Navab. Visual marker detection and decoding in AR systems: a comparative study. In *International Symposium on Mixed and Augmented Reality, ISMAR '02*, pages 97–106, 2002.

[212] Z. Zhang. Parameter estimation techniques: a tutorial with application to conic fitting. *Image and Vision Computing*, 15(1):59 – 76, 1997.

[213] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov. 2000.

[214] Z. Zhang, R. Deriche, O. Faugeras, and Q. T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78:87–119, October 1995.

[215] F. Zhou, H. B. L. Duh, and M. Billinghurst. Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. In *7th IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR '08*, pages 193–202, Sept. 2008.

[216] Y. Zhu, G. Xu, and D. J. Kriegman. A real-time approach to the spotting, representation, and recognition of hand gestures for human-computer interaction. *Computer Vision and Image Understanding*, 85(3):189 – 208, 2002.

Summary

Robust Augmented Reality

The field of Computer Vision is concerned with problems that involve interfacing computers with their surrounding environment through cameras. Consequently artificial vision systems can replace human perception in many tasks. Recent advances in technology, such as increase in computational power, good quality low cost CMOS cameras, improvement in peripherals and decreasing form-factor, allow the vision systems to be carried on roaming platforms such as tablet PCs and mobile phones. More generally, it leads to the possibility of wearable visual computing that can assist its carrier such as humans or robots in executing various perception-action tasks.

Augmented Reality (AR) technologies already have a history in the field of computer vision and many attempts have been made to use AR to create meaningful, immersive experiences incorporating humans and computers. It is appealing for many applications such as in entertainment and gaming to improve and enrich perception, cognition and interaction by providing extra information and guidance that is not available in the immediate surroundings.

This thesis presents the application of computer vision algorithms to create a marker-less, mobile and wearable AR system that can provide assistive services to its users. The overall system design includes definition and implementation of the necessary software modules as well as its combination with digital hardware. In the wearable mobile set-up we discuss in this thesis, the user can perceive 3D virtual moving objects augmenting the real world perceived at the same time.

The Delft University of Technology initiated their AR research in 1999 with outdoor head-mounted optical see-through AR, fusing data from a GPS, a natural feature tracking camera, and an inertia tracker, using a desktop PC in a backpack. Soon a switch back was made to indoor AR based on markers and inertia tracker data in order to improve the real-time performance and static and dynamic accuracy of the head-pose estimation by the visual odometry system. For applications we collaborated since 2006 with the Royal Academy of Art in The Hague, which has implemented many projects in art and design with our systems. In this thesis, our focus has shifted from art and design onto the design and implementation of a system for multi-user collaboration and spatial analysis using multiple AR systems; more specifically we selected Crime Scene Investigation (CSI) as our application domain. For this we collaborated with the Systems Engineering Section (SES-TPM) of TU Delft and the National Forensic Institute in The Hague. We also switched from optical see-through head-mounted displays (HMD) to cheaper video see-through head mounted displays a.k.a eye wear.

In this thesis, we introduce the challenges of mobile AR and CSI, and derive the system requirements that can meet these challenges. We further discuss how a remote collaborator can work with on-site users by decoupling him/her self from the on-site user's view while assisting the investigation. Then we present the real-time and on-line modules that satisfy these requirements:

- Robust, marker-less, extensible auto-motion tracking based on the tracking of 3D key-points observed by a low cost stereo camera pair
- Coarse 3D map-building to be able to augment a real scene with virtual objects, while roaming around in e.g. crime scenes
- On-line and on-site scene structure capturing by reconstructing a metric, dense 3D map of the scene in real time, with the aim to let off-scene experts guide the CSIs
- Human-Computer Interaction (HCI) software that exploits the user's hand motions. This acts as an interaction device for user interface operation instead of other auxiliary equipment, such as keyboard and mouse. With this HCI the user - e.g. a CSI - can place virtual objects in the scene, such as tags indicating the position of found evidence
- Software for remote connection to and on-line collaboration with off-scene experts, a lightweight and affordable HMD and a wearable computer

The realized head-mounted AR system allows interaction and collaboration between two or more parties and their environment, while it provides tools, guidance and information to on-site and off-site users to perform their tasks both independently and in collaboration with each other. According to our knowledge, this is one of the first examples of a complete 3D stereo AR system that integrates 3D marker-less AR capabilities with dense reconstruction, remote collaboration and HCI in a carefully engineered way that can be applied in the CSI domain and many other applications in which on-scene and off-scene experts work together.

Oytun Akman

Samenvatting

Robuuste Augmented Reality

Binnen de computer vision (computerwaarneming) wordt onderzocht hoe computers kunnen reageren op hun omgeving door middel van camera's. Hierdoor kunnen ze in veel gevallen menselijk toezicht vervangen. Vanwege nieuwe ontwikkelingen op het gebied van snellere computers, goede en goedkope camera's, betere randapparatuur en kleinere formaten kunnen dit soort systemen nu worden gebruikt in mobiele en draagbare apparaten zoals tablet PCs en mobiele telefoons. Draagbare, ziende systemen kunnen zo de drager – zowel mens als robot – helpen in het uitvoeren van diverse taken.

Augmented Reality (AR; toegevoegde werkelijkheid) wordt al lang onderzocht binnen de computer vision, en er is al vaak geprobeerd om via AR de gebruikservaring van computers te verbeteren. Veel toepassingen, zoals gaming, kunnen baat hebben bij de rijkere perceptie, cognitie en interactie die mogelijk worden door de toegevoeging van informatie en sturing die niet in de directe omgeving beschikbaar zijn.

Dit proefschrift beschrijft de toepassing van computer vision algoritmen om een mobiel en draagbaar AR systeem te maken dat de gebruiker kan ondersteunen in dagelijkse omgevingen. Het werk omvat het ontwerp en de implementatie van de benodigde softwarecomponenten en de combinatie met camera's en displays. Dragers van het systeem kunnen tegelijk 3D virtuele bewegende objecten en de echte wereld waarnemen.

De TU Delft is in 1999 begonnen met onderzoek naar AR, met een deels transparant scherm in een bril. Dat voor buitengebruik bedoelde systeem combineerde GPS, een inertiasensor en een op natuurlijke kenmerken gebaseerde camera, en werd gedragen als een rugzak. Al snel werd dat vervangen door een binnenshuissysteem met vaste patronen om de snelheid en naukeurigheid van de positiebepaling te verbeteren. Voor toepassingen werd samengewerkt met de

Koninklijke Academie van Beeldende Kunsten in Den Haag, dat veel projecten met dit systeem heeft verwezenlijkt. In dit proefschrift is de focus verschoven van kunst naar de samenwerking tussen meerdere gebruikers, en wel op het gebied van sporenonderzoek. Hiervoor hebben we samengewerkt met de Sectie Systeemkunde van de Faculteit Techniek, Bestuur en Management van de TU Delft en het Nederlands Forensisch Instituut in Den Haag. Ook maken we nu gebruik van goedkopere videobrillen zonder transparante schermen, waarop de gebruiker een actuele video van de omgeving ziet.

In dit proefschrift beschrijven we de uitdagingen voor mobiele AR en sporenonderzoek, en leiden de voorwaarden af waaraan een system voor deze toepassing moet voldoen. Ook beschrijven we hoe onderzoekers op afstand en op lokatie via AR samen kunnen werken, waarbij het standpunt van de coordinator op afstand losgekoppeld kan worden van de onderzoekers ter plekke. Daarna presenteren we onze software, die aan de gestelde eisen voldoet zonder daarbij het getoonde beeld te vertragen:

- Robuuste, uitbreidbare hoofdpositiebepaling met een goedkope stereocamera door het volgen van 3D punten met natuurlijke kenmerken
- Het bouwen van een ruwe 3D kaart van de omgeving om virtuele objecten te kunnen plaatsen terwijl de gebruiker rondloopt
- Het zonder vertraging modelleren van de omgeving door middel van een fijne, metrische 3D kaart, zodat experts op afstand de onderzoekers kunnen sturen.
- Mens-machine interactie (MMI) met handbewegingen, zodat de gebruikers geen toetsenbord of muis hoeven te gebruiken. Hiermee kunnen virtuele objecten worden geplaatst, zoals kaartjes bij gevonden bewijsmateriaal.
- Software om op afstand samen te kunnen werken met experts, een lichte en betaalbare headset, en een draagbare computer.

Met het door ons gebouwde AR systeem kunnen meerdere personen werken binnen een gedeelde, deels virtuele omgeving. Het biedt de informatie en sturing die nodig zijn om mensen op lokatie en op afstand hun taken te laten uitvoeren en samen te laten werken. Voorzover wij weten is dit een van de eerste voorbeelden van een compleet 3D stereo AR systeem dat 3D AR met natuurlijke kenmerken combineert met een dichte reconstructie, samenwerken op afstand en MMI, op een manier dat het gebruikt kan worden in sporenonderzoek en vele andere toepassingen waarin experts op verschillende lokaties samenwerken.

Oytun Akman

Acknowledgements

It was quite an experience! Over the last five years, it has been my good fortune to encounter many great people who have given me support in professional, academic and social life.

Pieter, thank you very much for all your supervision and support that a graduate student can expect from his professor. You also gave me a lot of freedom that is necessary to learn how to be an independent researcher. But still your door was always wide open to me, you were there when I was struggling. I enjoyed our travels to international conferences and gained insight through our conversations about life.

I also would like to thank all the committee members for accepting being a member of my PhD committee and for their valuable comments.

Ronald, I remember how we met and instantly started to talk about computer vision. I could not have wished for a more thorough discussion partner and thanks for giving me the motivation and insight I needed during my PhD and fortunately afterwards. You thought me a lot on being focused, precise and to the point, and this thesis would not have been possible without you.

Loubna, it was my pleasure to be a friend and a colleague of yours. I have learned and enjoyed a lot from our discussions which are not just about research but also about life.

My gratitude also goes to Robot Vision Group members, Xin, Jurjen, Boris and Robert. It was extremely fun working with you all. Xin, my crazy colleague, I will miss our mis-communications; we had a lot of fun even without understanding each other. I also want to thank all the DBL group members that I have worked, talked and lunched over the past years. Martijn Wisse, I always enjoyed our conversations and learned a lot from your few but invaluable advices. Diones, Anouk and Dinneke, I want to thank you for helping me with every matter. Also, I want to thank AR Lab for our insightful discussions and collaboration.

Baboliler, Wouter and Erik, the lab would have been very boring without you two. Wouter Caarls, thanks for enduring our pointless conversations with Berk, and even sometimes being a part of them. For me, you are the voice of logic and I know when I am confused about something, you will be there to give the right answer. Erik Schuitema, I am grateful for your high information density talks. We suffered but also had fun together during our PhDs and now we are almost done. Guys, I will miss our traditional dinners at the Argentinian place. But next time, lets try something else.

Our collaboration with TPM-SES group gave me the great opportunity to work with Stephan Lukosch and Alexander Verbraeck. Alexander, I really enjoyed our discussions on computer vision. Also I want to thank you for showing me that being a full-professor doesn't necessarily mean that you are too busy for doing hands-on research. Stephan, I had a lot of fun during the CSI demo days and our travel to Seattle. It was my pleasure to work with you. Maybe next time we can enter that national park.

A special word of gratitude goes to all the members of the Falcon Project, especially ESI and Vanderlande Industries. I also want to thank Jacques, Roelof and Jan for providing a stimulating research environment during the first years of my PhD.

A. Aydin Alatan and Neslihan Bayramoglu, I want to thank you for our valuable collaboration. Neslihan, I already miss our long conversations/complaints about PhD.

Now, the great Turks that I have met during these 5 years. I think you are the reason that we managed to endure this challenging and long process. I wish I could thank you all for everything but it would take another 100 pages.

Alper, Olgu, Caner and Ilhan, thanks for being on the other end of the telephone when I needed friends, because of you guys I never felt alone. Our coffee breaks, dinners and basketball games were invaluable. Alper, special thanks to you for being there when we most needed a helpful hand.

Alex, Umut, Bilge, Yunus, Namik, Emre, Sinan and Rogier, I wish we had more time together. We were distributed sparsely but it never stopped us gathering and having fun. Alex, I have already prepared your room. Rogier, I know you have some Turkish genes in your DNA. Yunus, I am still waiting for the promised barbecue.

Sevinc (and Deniz pasa), Merve, Nesen, Ozge, Gozde, Esra, Nazli, Umut, Tsolin and Nihal I will miss all the social events mainly organized by you and thanks for keeping Melis busy from time to time. I really appreciate that.

Ugur and Taha, sharing similar faiths I feel like you will always be close even if we are inhabiting three different continents.

Onur, thank you very much for designing my artistic cover page. It is awesome!

Narin, you were always there when I needed help. You are more than a neighbor; a traveling mate, an eating partner, a voice of common sense and above all a real friend. Thanks for everything.

Nihal, now when I am away I understand how precious our time together was.

You (and of course Birol) are one of most important things that the Netherlands introduced to us. Thanks for always being there and understanding me no matter what.

Berk, my brother! You know me better than anybody. I would have never imagined to have such a friendship in my 30s. Our time together was priceless and will be the most missed. Together with Nihal, you made leaving the Netherlands so difficult for us. Even now, at a beautiful place, we wish you would be there to fill the two empty seats on a square table.

My gratitude also goes to Beril, Cetin baba and Demet anne for their support. Thank you for encouraging us not to give up. Beril, there will always be a place for you in our home.

I am, of course, particularly indebted to my parents who have truly always been there for me, and without them none of this would have been even remotely possible. Dad and Mom, I cannot thank you enough for your endless support in every way. Your presence always make me feel safe and strong, and always appreciated. It is so special to be able to talk to you in every aspect of life. My brother Mehmet, aunt Ayla, uncle Adem and grandma Emine, thank you all for always supporting me no matter what.

Finally, Melis, my beloved wife and my best friend, this thesis has happened because of your encouragement and unwavering support on every matter in my life. You taught me how to endure challenges and keep trying even if you are tired and exhausted. You are the most precious and special thing that happened to me. As we promised each other, we will always be together all around the world.

Oytun Akman
San Francisco, October 2012

Curriculum Vitae

Oytun Akman was born in Ankara, Turkey on July 10, 1981. In 1999 he obtained his high-school diploma from Ankara Milli Piyango Anatolian High School in Ankara, Turkey. In the same year, he was accepted to the Electrical and Electronics Engineering (EEE) Department at Middle East Technical University (METU), Ankara, Turkey. After completing the language preparatory course in 2000, he started his bachelor of science (B.Sc.) education in EEE. After completing his B.Sc. studies in 2004, he continued his M.Sc. studies under the supervision of Prof. dr. A.A. Alatan in the same department and received the M.Sc. degree on the subject of multi-camera surveillance systems in 2007 with his thesis titled "Multi-camera Video Surveillance: Detection, Occlusion Handling, Tracking and Event Recognition". During his master studies he also worked as a hardware design engineer for ASELSAN Inc. on embedded system design for airplane internal communication networks and a graphical user interface for thermal binoculars.

After finishing his MSc studies, he started his PhD study in 2007 on designing a robust Augmented Reality (AR) system under the supervision of Prof.dr.ir P.P. Jonker at the Delft Biorobotics Lab, Delft University of Technology (TU Delft), Delft, The Netherlands. His PhD research involved tracking, self-pose estimation, mapping and Human-Computer-Interaction. During his PhD, he worked in close collaboration with the TPM of TU Delft and artists from Dutch Royal Academy of Art (KABK ARLab) focusing on the utilization of AR systems in art. Also he participated in the joint research project FALCON with the partners: Vanderlande Industries, Embedded Systems Institute, TU Delft, TU Eindhoven and TU Twente. Following his PhD work, he worked as an Postdoctoral Researcher on the same research field with his PhD for 7 months.

In May 2012, Oytun joined Autodesk Inc. as a principal engineer and still working for Autodesk in San Francisco, USA. His work focuses on point cloud processing and AR applications.