Markerless Augmented Reality Applications

Fesih KESKİN

Master of Science Thesis

Electrical and Electronics Engineering Program

July 2014

JÜRİ VE ENSTİTÜ ONAYI

Fesih KESKİN'in **"Markerless Augmented Reality Applications"** başlıklı **Elektrik ve Elektronik Mühendisliği** Anabilim Dalındaki, Yüksek Lisans Tezi 23.07.2014 tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

		Adı Soyadı	**İmza**
Üye (Tez Danışmanı) :		**Prof. Dr. Ömer Nezih GEREK**	**……..**
Üye	**:**	**Doç. Dr. Serkan GÜNAL**	**……..**
Üye	**:**	**Yrd. Doç. Dr. Tansu FİLİK**	**……..**

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …….. tarih ve …….. sayılı kararıyla onaylanmıştır.

Enstitü Müdürü

ÖZET

Yüksek Lisans Tezi

İŞARETÇİSİZ EKLENMİŞ GERÇEKLİK UYGULAMALARI

FESİH KESKİN
Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Ömer Nezih GEREK
2014, 57 sayfa

Eklenmiş gerçeklik bilgisayarda oluşturulan bilgilerin gerçek zamanda gerçek dünya görünümüne eklenmesi işlemidir. Bilgisayar ortamında oluşturulan 3-boyutlu modellerin, animasyonların, videoların ve çeşitli sanal materyallerin gerçek dünyada arzu edilen hedef üzerine konumlandırılmasını sağlayan eklenmiş gerçeklik, gerçek dünyanın zenginleştirilmesini, daha faydalı ve işlevsel hale getirilmesine izin vermektedir. Bu tez de eklenmiş gerçeklik teknolojisinin nasıl çalıştığıyla ilgili bilgilendirmekte, uygulama alanlarına göre irdelemekte ve tarihten bugüne gelişimini özetlemektedir.

Tez Android mobil cihazda çalışan, tamamen işaretçisiz eklenmiş gerçeklik uygulamasının tasarım ve geliştirilmesini sunar. Bu amaç doğrultusunda OpenCV bilgisayarlı görü kütüphanesi kullanılarak öznitelik çıkartma ve tanıma, SURF (hızlandırılmış gürbüz öznitelik çıkarım), işaretçisiz takip etme (doğal öznitelik izleme) gibi birçok bilgisayarlı görü tekniği kullanılarak çalışıldı. Bunun yanı sıra OpenGL (açık grafik kütüphanesi) kullanılarak 3-Boyutlu modelleme, doku giydirme, gölgelendirme vb. ve ayrıca Blender3D ile 3-boyutlu Wavefront OBJ model dosyaları oluşturuldu.

Son olarak bu tezde, yukarıda bahsedilen yöntemlerden faydalanarak bir kaç uygulama sunulmuştur. İlk olarak eklenmiş gerçeklik mantığını anlamak için kamera kalibrasyonu, poz izleme ve işaretçi bazlı eklenmiş gerçeklik uygulamaları üzerinde çalışıldı, sonrasında algoritmanın hedef izleme kısmı için Vuforia SDK kullanılarak Android mobil cihaz üzerinde işaretçisiz eklenmiş gerçeklik uygulaması geliştirildi.

Anahtar Kelimeler: Arttırılmış Gerçeklik, Eklenmiş Gerçeklik, Bilgisayarlı Görü, Mobil Cihazlar, Android Ara yüz Yazılımı, SURF

ABSTRACT
Master of Science Thesis

MARKERLESS AUGMENTED REALİTY APPLİCATİONS
Fesih KESKİN

Anadolu University
Graduate School of Sciences
Electrical and Electronics Engineering Program

Supervisor: Prof. Dr. Ömer Nezih GEREK
2014, 57 pages

Augmented Reality (AR) is the process of adding virtual computer generated information on view of a physical real-world environment in real-time. AR, which allows adding 3-dimensional models, animations, videos and various virtual materials which are generated in computer environment to be positioned on desired targets in the real world, allows for enabling and enriching the real world more functional and beneficial. In this thesis explains how AR technologies work, expresses and explains fields of application and gives a summary from history to present time and also presents a real-life demonstration of Mobile device-based AR.

Thesis presents contributions for design and development of a fully working Markerless Augmented Reality Application that works on Android Mobile devices. For this purpose, were studied on several computer vision techniques such feature extraction-detection, SURF (Speeded Up Robust Features), Markerless tracking (Natural feature tracking) by using OpenCV Computer Vision library. As well as, were done 3D modelling, texturing, shading etc. by using OpenGL (Open Graphics Library) and also creating Wavefront OBJ 3D model files via Blender 3D.

Finally in this thesis, several applications are presented that make use of these methods which is mentioned above. Firstly camera calibration, pose tracking and marker based augmented reality applications were worked for understanding augmented reality idea, then a markerless augmented reality application were developed on android mobile device by using Vuforia SDK for target tracking part of algorithm.

Keywords: Augmented Reality, Computer Vision, Mobile Devices, Android Software Interface, SURF

ACKNOWLEDGEMENTS

First and foremost I thank my supervisor Ömer Nezih GEREK for providing to study and complete my M.Sc. I am extremely grateful for our many discussions which helped guide me in the right direction, his motivating attitude, his availability as a supervisor and his many insightful suggestions.

And most importantly I would like to thank my wife Fatmanur and my family for their encouragement and support and for keeping me honest.

Fesih KESKİN

July 2014

TABLE OF CONTENTS

ÖZET ..... **i**
ABSTRACT ..... **ii**
ACKNOWLEDGEMENTS ..... **iii**
TABLE OF CONTENTS ..... **iv**
LIST OF ABBREVIATIONS ..... **vi**
INDEX OF FIGURES ..... **viii**
INDEX OF TABLES ..... **x**

1. INTRODUCTION **1**
1.1. History of Augmented Reality 1
1.2. Motivation and Approach 3
1.3. Contributions 4
1.4. Thesis Outline 5

2. LITERATURE REVIEW AND BACKGROUND **7**
2.1. Augmented Reality 7
2.2. Augmented Reality on Handheld and Embedded Devices 8
2.3. 3D User Interfaces 10
2.3.1. Applications of 3D UIs 11
2.3.2. Mobile Applications 11
2.4. Pose Tracking 13
2.4.1. Natural Feature Tracking 13
2.4.2. Marker Tracking 13
2.5. Computer Vision 14
2.5.1. Cameras and camera calibration 15
2.5.2. Feature detection and description 16
2.6. Discussion 18

3. IMAGE FEATURES **19**
3.1. Feature Detectors and Descriptors 19
3.1.1. Feature Description 19

3.1.2. Feature extraction 21
3.2. SURF (Speeded-Up Robust Features) 22
3.3. SURF Detector Algorithm 31

4. POSE TRACKING **35**
4.1. Camera Calibration 35
4.2. Pose Tracking 40

5. RENDERING **42**
6. VUFORIA SDK **45**
6.1. Introduction 45
6.2. Architecture of Vuforia 45
6.3. Target management system 46

7. APPLICATION DEVELOPMENT **47**
7.1. Software Implementation 47

8. TEST AND RESULTS **49**
9. CONCLUSION **51**
9.1. Summary and Conclusion 51
9.2. Future Works 51

REFERENCES **52**

LIST OF ABBREVIATIONS

2D 2-dimensional.

3D 3-dimensional.

ADT Android Developer Tools.

API Application Programming Interface.

AR Augmented Reality.

ARToolKit Augmented Reality Tool Kit.

AV Augmented Virtuality.

CCD Charge Coupled Device.

CDT C/C++ Development Tools.

CMOS Complementary Metal Oxide Semiconductor.

DoF Degree of Freedom.

DoG Difference of Gaussians.

FLANN Fast Library for Approximate Nearest Neighbors.

fps frames per second.

GPS Global Positioning System.

GPU Graphical Processing Unit.

HMD Head Mounted Display.

IA Intelligence Amplification.

KF Kalman Filter.

LCD Liquid-Crystal Display

LoG Laplacian of Gaussian.

MARS Mobile Augmented Reality Systems.

MR Mixed Reality.

mtl Material Template Library

NCFM Network Coupled Feature Maps.

NDK Native Development Kit.

NN Nearest Neighbor.

OBJ Object file.

OpenCV Open source Computer Vision.

OpenGL Open Graphics Library.

PC Personal Computer.

PCA Principal Component Analysis.

PDA Personal Digital Assistant

PnP Perspective-n-Points.

RANSAC RANdom SAmple Consensus.

RGB Red, Green and Blue.

SIFT Scale Invariant Feature Transform.

SDK Software Development Kit.

SLAM Simultaneous Localization and Mapping.

SURF Speeded-Up Robust Features.

SUSAN Smallest Univalue Segment Assimilating Nucleus.

UI User Interface

UMPC Ultra-Mobile Personal Computer.

USB Universal Serial Bus.

VR Virtual Reality.

VRML Virtual Reality Modelling Language.

INDEX OF FIGURES

Figure 1.1 Milgram's Reality-Virtuality continuum [3] 2
Figure 1.2 Augmenting Real World Objects [6] 4
Figure 2.1 Different level of outsourcing to a server: a) All tasks are run natively by the client, b) Server performs tracking, c) Server performs tracking and application logic, d) All work is done by the server [19] 8
Figure 2.2 Left image: VTT's AR Scale Model application augments a virtual model of a building on top of a floor plan in the correct scale and pose using marker detection. Right image: an example of a marker (ALVAR marker number 14) [45]. (Image: VTT Augmented Reality team) [43] 14
Figure 2.3 Pinhole camera model. C is the optical center and f is the shortest distance from C to the image plane. P is a 3D point with its projection p on the image plane 16
Figure 3.1 Image gradients with a Gaussian window, indicated by the overlaid circle as shown on the left. A keypoint descriptor created by first computing the gradient magnitude and orientation as shown on the right [62] 21
Figure 3.2 Exact and approximated Gaussian kernels [64] 24
Figure 3.3 Rectangular Region for Integral Image 24
Figure 3.4 Lena image and the corresponding (normalized) integral image 25
Figure 3.5 Box filtering example 25
Figure 3.6 Calculation of an Arbitrary Rectangle Sum 26
Figure 3.7 Smallest kernel for box filtering 27
Figure 3.8 Box Filter for Second Order Partial Gaussian Derivative in y-direction on the top and in xy-direction on the bottom [64] 28
Figure 3.9 Scale Space Structure of SURF Feature Detector 29
Figure 3.10 Non-maxima suppression for candidate SURF features [63] 29
Figure 3.11 Haar Wavelets [64] 30
Figure 3.12 Calculation of SURF descriptor components [72] 31
Figure 3.13 SURF implementation with good matches 34

Figure 3.14 SURF implementation with rotated target good matches.....34
Figure 4.1 Several screen shoots of Camera Calibration application 36
Figure 4.2 Comparison of Calibrated and Uncalibrated Camera.....36
Figure 4.3 Augmentation of a Cube with calibrated camera (Left) and uncalibrated camera (Right).....40
Figure 4.4 Pose estimation experimental work.....41
Figure 5.1 A 3D virtual model with completed of rendering pipeline.....42
Figure 5.2 The OpenGL Render Pipeline 43
Figure 5.3 A spaceship Wavefront OBJ 3D model in wireframe form shown in the upper left, solid form without texture shown in the upper right. The pieces in the image texture (the lower right) are "glued" onto the spaceship, and the result is shown in the lower left.....44
Figure 6.1 Data flow diagram of the application with the Vuforia SDK.....45
Figure 7.1 3D Wavefront OBJ Roof Model with solid and textured mode 47
Figure 8.1 Handmade 3D target model for augmentation.....49
Figure 8.2 Augmentation of a handmade 3D home model 50
Figure 8.3 Unwrapped of 3D handmade home model (right) with its 2D target images 50

INDEX OF TABLES

Table 3-1 SURF Algorithm Steps 31

1. INTRODUCTION

Augmented Reality (AR) is adding virtual computer generated information on view of a physical real-world environment in real-time. AR is both interactive and registered in 3D as well as combines real and virtual objects [1]. The field of AR has very good potential to improve lives in many ways and help people learn, navigate, and search the environment. Up until now, the technology available has made development in this field not very meaningful or worthwhile. With the advancements in mobile phone technology, incorporating things like GPS data, a video camera, a compass, and an internet connection, the benefits of AR are becoming available to more and more people every day.

The recent novelty of Augmented Reality (AR) and mobile technologies has enabled the creation of new mobile AR applications. Mobile AR allows users to integrate the information of the internet with their real lives. Lately, mobile AR applications becomes more commonplace in consequence of the image processing, computer vision techniques and rapid processing capabilities of mobiles has grown in recent years.

The aim of this thesis is understanding augmented reality and create a mobile based 3D AR application. The future of augmented reality looks very promising and with the advancements in technology it will someday be an important part of many people's lives [1].

1.1. History of Augmented Reality

First Augmented Reality systems were developed in the 1960s, Augmented Reality only separated itself from virtual reality and became a research area in its own rights in the beginning of the 1990s. Today there is two main definitions that describe Augmented Reality. Because of a lack of an official agreement on the term, both are accepted. Following the definition of Azuma [2] An AR system has to accomplish the three requirements:

- Combine the Virtual world and Real world
- Registered in the real world in 3D
- Interactive in real time

The first requirement is a fundamental description of AR in that it combines the real world with virtual contents. The second requirement separates Augmented Reality from the more general concepts of mixed reality or mixed media by requiring that the virtual content must be registered in 3D within the real world. Finally “Interactive in real time” requires the system to react to the user and update in real time which differentiate AR from all off-line augmentations such as the use of computer graphics in movies.

According to the older Virtuality continuum proposed by Milgram [3] (see Figure 1.1), AR is just one possible manifestation of Mixed Reality (MR), which brings together real and virtual within a single display. The Virtuality continuum collocate AR and Augmented Virtuality (AV). AR is mostly grounded in the real world, with a limited set of virtual objects mixed in. The inverse concept, AV, is designed as a Virtual Environment with some real directions - a recurring example for AV are video-textured avatars (showing a live video feed of real people) within a Virtual Environment. The boundary between AR and AV is not strictly defined.

Figure 1.1 Milgram's Reality-Virtuality continuum [3]

Benefit of the mobile augmented reality (AR) is that mobile computing research is a natural complement because of the mobile AR system can assist at the workplace instead of requiring to stationary workstations. The advantage of mobile approaches is that hardware and software very similar to traditional non-mobile AR systems can be used. Whilst there are a lot of working systems created of a head mounted display (HMD) and a notebook, most of these setups have been designed as pure proof of concept and do not provide an utilizable form factor. Generally HMDs have all their hardware mounted to a large rucksack, including

heavy power supplies for items not designed for mobile use. While such rucksack/HMD combinations combine high performance with handsfree operation, they seriously affect ability, avoid practical use and are socially unacceptable. They are maintenance dense and lack sturdiness on account of their complex hardware setups. Most of the devices used were not designed for mobile deployment and therefore not only require heavy batteries but also use fragile connectors and cables. Additionally, the prohibitive cost of these setups prevents dispersing them in a commercial market. On the side, the development of HMD technology, which is an unavoidable part of such an approach to wearable AR, is not keeping quickness with the advances in computer and sensor technology.

Simultaneously, broad consumer interest in cell phones and handheld computers. Owing to, this is dramatically accelerating the development in this area. Therefore consideration of AR development will be shift to smaller and ergonomic devices which is smartphones [4].

1.2. Motivation and Approach

Lately, augmented reality increases as an interesting topic in various field. What is benefit of combining real and virtual objects in 3D? Augmented Reality is used several areas, such as Archaeology, Architecture, Art, Commerce, Construction, Education, Gaming, Industrial design, Medical, Military, Navigation, Television etc. While these do not cover every potential application area of this technology, they do cover the areas explored so far [2].

Augmented Reality improve perception of users with the real world. The virtual objects display information on devices that the user cannot directly detect with his own senses. The virtual objects helps a user perform real-world tasks. AR is a particular example of what Fred Brooks calls Intelligence Amplification (IA): using the computer as a tool to make a task easier for a human to perform [5].

The dream feeding and motivating the research presented in this thesis is a “Totally Markerless Mobile Architectural Augmentation Paradigm" which can be illustrated as Figure 1.2

Figure 1.2 Augmenting Real World Objects [6]

The main focus of the thesis is on finding aspect of 3D target as a 2D target image and render 3D Virtual Object model on proper world coordinates. After completion of augmentation of the 3D Virtual object, the user can look around 3D virtual object. This will show user as a 3D reconstruction of a 3D real world object.

1.3. Contributions

The contribution of this thesis is design and development of a fully working Markerless Augmented Reality Application that works on Android Mobile devices and was tested in multiple practical applications.

This thesis presents a real-life demonstration of Mobile device-based AR. The Mobile device has a number of ergonomic advantages over a notebook AR and head-mounted displays and the suitability of this device as an AR medium has been demonstrated by the development and public demonstration of a functional 3D reconstruction AR applications.

The application of this thesis try to the reconstruction of a 3D Real world object by Augmented Reality technology to the mobile phone. In order to do that, by aspect of 3D world object, it is recognized as a 2D target image. By the way,

positioning 3D Object model on the world object, then we get a reconstructed world object by augmentation of our model.

1.4. Thesis Outline

In this thesis for preliminary study, to understanding idea of augmented reality, have worked on several computer vision techniques such feature extraction-detection, SURF, Markerless tracking (Natural feature tracking) by using OpenCV [7] Computer Vision library. As well as, 3D modelling, texturing, shading etc. by using OpenGL [8] (Open Graphics Library) and creating Wavefront OBJ 3D files via Blender 3D [9]. Chapter 1 contains an introduction to augmented reality and its history by explaining some basic concepts and the purpose of this thesis. To close this introductory chapter, the organization of the remaining chapters of this thesis is outlined below.

Chapter 2 provides an overview of related work and literature review of AR consisting of six primary sections. Firstly it describes the technology typically associated with AR in order to give the reader context for 3D User interface, Pose tracking and Computer Vision elements being studied and implemented in this area. Besides, mentioned categories of AR applications.

Chapter 3 presents the recognition methods for augmentable targets. Primarily it describes features of image in order to clarify the certain properties of the image. Then studied theory and implementation of feature detection and description. Additionally examined SIFT and SURF to detect and describe local features in images and implementation of SURF on notebook by using OpenCV library.

Chapter 4 investigate an important part of augmented reality which is pose tracking. For augmentation of 3D object model with an appropriate view on camera, we implemented and tested calibration and pose of camera by notebook USB camera.

Chapter 5 presents 3D object viewing and rendering pipeline, introduced a 3D API which is OpenGL and its properties. Besides, usage of Wavefront OBJ 3D files and Blender 3D will described.

Chapter 6 then briefly introduces Vuforia SDK which used for target recognition part of mobile augmented reality application on Android software.

Chapters 7 and 8 develops the stage of markerless augmented reality application, then tests and results it.

2. LITERATURE REVIEW AND BACKGROUND

2.1. Augmented Reality

In 1968 Ivan Sutherland created the first head-mounted display (HMD) [9, 10]. Owing to restricted processing power, his application demonstrated just a simple wireframe model overlaid onto the real world. But nevertheless, it marks the first application that fulfils the definition by Azuma and Milgram (see section 1.1).

The first Augmented Reality applications developed from basic research, used very expensive hardware and last of all mostly covered research and technical problems only. In his 1995 survey paper Azuma lists six categories for AR applications: medical, manufacture and repair, visualization and annotation, military aircraft, robot path planning and entertainment. Some seminal works in these areas are given in the following.

Researchers at UNC Chapel Hill administered first trials of overlaying 3D representations of ultra-sound data onto patients [10]. In the "Knowledge-based Augmented Reality for Maintenance Assistance" (KARMA) project Feiner and the others created a laser printer maintenance application [11]. Milgram developed the ARGUS system [3] to create an easier way for robot path planning.

With the introduction of powerful portable computers and notebooks, mobile AR setups became possible. The Touring Machine [13, 14] was among the first to use this new hardware platform for mobile systems. A later project of the same research group was MARS (Mobile Augmented Reality Systems) [12]. It was one of the first indeed mobile augmented reality setups, Presented in 1999, which allowed the user to freely walk around with all necessary equipment mounted onto his back. Several similar platforms such as Studierstube [13], Tinmith [14] and BARS [15] examined in various application areas.

Due to the recent availability of Tablet PCs and UMPCs many researchers use these devices to bring existing software to smaller devices. Newman et al. use these mobile devices for experiments on wide area tracking [16]. Reitinger uses UMPCs to gather data in an urban environment [17]. After starting with backpack setups the iPERG project [18] then switched to UMPCs and Tablet PCs due to

their lower costs and hardware maintenance requirements. The AMIRE3 project used Tablet PCs for a museum guide [19].

2.2. Augmented Reality on Handheld and Embedded Devices

Many early works at least partly outsourced processing tasks to a nearby server via tethered or wireless networking. As can be seen in Figure 2.1, there are four different levels of outsourcing processing tasks to a server: In the ideal case Figure 2.1(a), all work is performed natively by the client making it independent of the server and substructure. At the other extreme, many early handheld AR applications were based on a thin client approach with a "video-in/video-out" communication mechanism for receiving assistance from a computing server, which is shown as Figure 2.1(d). Such a setup does not only require a frame-by-frame communication but also requires sending video images in both directions requiring maximum performance of the network connection [19].

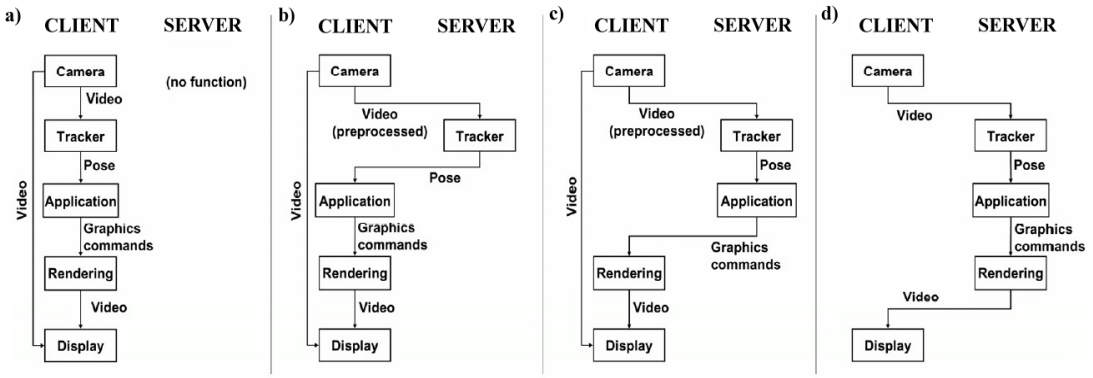

Figure 2.1 Different level of outsourcing to a server:
a) All tasks are run natively by the client, b) Server performs tracking,
c) Server performs tracking and application logic,
d) All work is done by the server [19]

On the other hand, these Figure 2.1(a) and Figure 2.1(d) solutions are just extreme examples of allocation of works among a handheld client and server. These extreme allocation may be necessary and useful depending on conditions or solutions.

Typical augmented reality system for both tracking and video see-through display uses a single video source. The main process of pipeline is: video

acquisition, tracking, application computation, rendering, display. Dispose of some these task to a computing server is an example of horizontally distributed simulation [20], and it is founded a scalable solution necessitates using of the available network bandwidth [21]. Communication of raw video flows in both directions (Figure 2.1c) does not satisfy such bandwidth limitation. A more preferable alternative seems to be flowing graphics commands back to the client such as done in the Chromium [22] framework.

The approach demonstrated in Figure 2.1(b) leaves the tracking task to computing server, which needs upstream communication of pre-processed, compressed video for visual tracking purposes, followed by downstream communication of pose information. The advantage of this approach is that a very succinct, but general and computationally pricey task is transferred to the server, the client just handles all application details, in this way dependencies between client and server are minimal. For example, while tracking of artificial fiducials can be performed in realtime on embedded clients now, natural feature tracking can benefit from the greater computational power of a server for at least several more years.

A small tethered LCD displays used for location based information by Amselem's work [23] and Fitzmaurice's Chameleon [24]. To track objects in the environment, Rekimoto's NaviCam [25] used color-coded sticker. Owing to the tethered trackers lately works, the degree of mobility was quite limited. mPARD [26] is a version of using analogue wireless video transmission to replace tethers. Sony CSL introduced the Transvision [27] project which is handheld augmented reality devices for a shared space. Researchers at HITLab later improved this concept [28] with a better user interface and an optical tracking solution re-using the camera needed for video see-through. All these works use simple tethered displays and cameras for the mobile device and are therefore extreme examples of Figure 2.1(d).

The Batportal [29] used non-mixed 3D graphics using VNC, while the AR-PDA project [30] used digital image streaming from and to an application server. Both projects again use the method describe in Figure 2.1(d). Shibata's work [31] goals to load balancing between client and server - the weaker the

client, the more tasks are outsourced to a server. It can therefore vary between all situations described in Figure 2.1. ULTRA uses PDA-based AR to support maintenance workers, but concentrates on augmenting "snapshot" still images [32]. In the absence of real-time tracking for infrastructure independence it performs all tasks natively (Figure 2.1a).

In 2003 the author ported ARToolKit [33] to the PocketPC and developed the first fully self-contained PDA AR application [4]. This platform was used in a peer to peer game in [34]. Möhring et al. were the first to successfully target a consumer smartphone for mobile AR [35]. The scarce processing power of the target platform allowed only a very coarse estimation of the object's pose on the screen. Henrysson ported ARToolKit to the Symbian platform and created the first two-player AR game [36] on current-generation smartphones.

Summarizing these developments one can conclude that there is no ideal solution for systems with scarce processing capabilities. An infrastructure independent solution, as developed in the work of this thesis is desirable, but not feasible for all situations. E.g. when artificial feature tracking is not an option, embedded devices simply do not have the processing capabilities yet. While this will certainly change in the future, new more demanding problems will emerge too [19].

2.3. 3D User Interfaces

A 3D user interface is as simply "a UI that involves 3D interaction." This simply delays the inevitable, as we now have to define 3D interaction. 3D interaction is "human-computer interaction in which the user's tasks are performed directly in a 3D spatial context [37]."

One key word in this definition is "directly." There are some interactive computer systems that display a virtual 3D space, but the user only interacts indirectly with this space—e.g., by manipulating 2D widgets, entering coordinates, or choosing items from a menu. These are not 3D UIs.

The other key idea is that of a "3D spatial context." This spatial context can be either physical or virtual, or both. The most prominent types of 3D UIs involve a physical 3D spatial context, used for input. The user provides input to

the system by making movements in physical 3D space or manipulating tools, sensors, or devices in 3D space, without regard for what this input is used to do or control. Of course, all input/interaction is in some sense in a physical 3D spatial context (a mouse and keyboard exists in 3D physical space), but the intent here is that the user is giving spatial input that involves 3D position (x, y, z) and/or orientation (yaw, pitch, roll) and that this spatial input is meaningful to the system.

Thus, the key technological enabler of 3D UIs of this sort is spatial tracking [38, 39]. The system must be able to track the user's position, orientation, and/or motion to enable this input to be used for 3D interaction. For example, the Microsoft Kinect tracks the 3D positions of multiple body parts to enable 3D UIs, while the Apple iPhone and others Mobiles tracks its own 3D orientation, allowing 3D interaction.

2.3.1. Applications of 3D UIs

Why is it important to understand and study 3D UIs? For many years, the primary application of 3D UIs was in high-end virtual reality (VR) and augmented reality (AR) systems. Since users in these systems were generally standing up, walking around, and limited in their view of the real world, traditional mouse- and keyboard-based interaction was impractical. Such systems were already using spatial tracking of the user's head the correct view of the virtual world, it was natural to also design UIs that took advantage of spatial tracking as well. As we indicated above, however, recent years have seen an explosion of spatial input in consumer-level systems such as game consoles and smartphones. Thus, the principles of good 3D UIs design are now more important to understand than ever. To further motivate the importance of 3D UI research, let's look in a bit more detail at Mobile technology areas where 3D UIs are making an impact on real-world applications.

2.3.2. Mobile Applications

Mobile devices, such as smartphones and tablets, are an interplay designer's playground, not only due to the rich design space for multi-touch input,

but also because these devices incorporate some quite powerful sensors for 3D spatial input. The combination of accelerometers, gyroscopes, and a compass give these devices the ability to track their own orientation quite accurately. Position information based on GPS and accelerometers is less accurate, but still present. These devices offer a key opportunity for 3D interaction design, however, because they are ubiquitous, they have their own display, and they can do spatial input without the need for any external tracking infrastructure (cameras, base stations, etc.).

Many mobile games are using these capabilities. Driving games, for example, use the "tilt to steer" metaphor. Music games can sense when the user is playing a virtual drum. And golf games can incorporate a player's real swing.

But "serious" applications can take advantage of 3D input for mobile devices as well. Everyone is familiar with the idea of tilting the device to change the interface from portrait to landscape mode, but this is only the tip of the iceberg. A tool for amateur astronomers can use GPS and orientation information to help the user identify stars and planets they point the device towards. Camera applications can not only record the location at which a photo was taken, but also track the movement of the camera to aid in the reconstruction of a 3D scene. Perhaps the most prominent example of mobile device 3D interaction is in mobile AR. In mobile AR, the smartphone becomes a window through which the user can see not only the real world, but virtual objects and information as well [12, 40]. Thus, the user can browse information simply by moving the device to view a different part of the real world scene. Mobile AR is being used for applications in entertainment, navigation, social networking, tourism, and many more domains. Students can learn about the history of an area; friends can find restaurants surrounding them and link to reviews; and tourists can follow a virtual path to the nearest subway station. Prominent projects like MIT's SixthSense [40] and Google's Project Glass have made mobile AR highly visible. Good 3D UI design is critical to realizing these visions.

2.4. Pose Tracking

Any Augmented Reality system requires some kind of tracking the targets or display's pose in order to register it in respect to the real world. Pose tracking is especially useful for identifying camera views in databases, video streams, video sequences, and live recordings. All of these applications require a fast pose recognition process in real-time video. For fast pose recognition it is possible to extend the materials to update the recognition system online [41]. Pose tracking must run in real-time, typically requiring solutions that estimate poses in less than 50 milliseconds. Also it must be robust under many conditions such as varying lighting. In case tracking is lost, the system must be able to recover quickly [19].

2.4.1. Natural Feature Tracking

Natural feature tracking is necessary to make markerless augmented reality applications practical on low performance mobile devices.

Markerless tracking methods are using natural features such as color and shape of the environment to be augmented for tracking. However, until recently, performance of appropriate AR methods and algorithms were not sufficient on mobile devices. Recently processing power has reached a level that allows natural feature tracking in real time. Natural feature tracking using optical flow has been successfully implemented on these devices though [42].

2.4.2. Marker Tracking

One of the fundamental components of augmented reality is tracking that calculating location and orientation of camera in real-time. A computer system detect the sign or image from a video frame by using image processing, computer vision and pattern recognition techniques (e.g. right image in figure 2.2) When detect the marker, then it defines the correct scale and pose of the camera.

Once detected, it then defines both the correct scale and pose of the camera. This method widely used in AR application [43] and it is called marker-based tracking. Marker-based systems are easy to implement and there are lots of well-know and handy toolkits (e.g. ARToolKit [44], ALVAR [45], ARTag [46]). These kind of toolkits provide a base for starting AR application development. As

well as, markers make certain that the correct scale and convenient coordinate frames as previously mentioned. In marker-based tracking, the system needs to detect the marker, identify it and then calculate the pose [43].

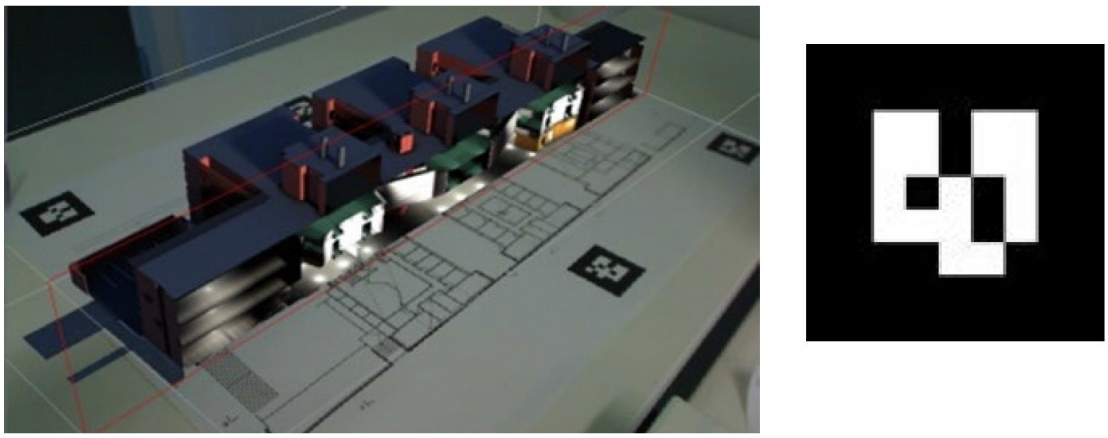

Figure 2.2 Left image: VTT's AR Scale Model application augments a virtual model of a building on top of a floor plan in the correct scale and pose using marker detection. Right image: an example of a marker (ALVAR marker number 14) [45]. (Image: VTT Augmented Reality team) [43]

2.5. Computer Vision

As humans, we sense the 3D structure of the world around us with obvious simplify. Computer vision looks for generating useful and intelligent descriptions of visual sequences and scenes, and adding 3D virtual objects by performing operations on the signals received from camera frames. Vision is an important sense for humans since it allows them to understand the structure of their environment. This process of inferring the spatial relationships (i.e. perspective order and 2-dimensional (2D) positions) between the objects in the surrounding can be described in two stages. First, the reflected light from the objects in the environment must be sensed through a sensor (the eyes), then it must be interpreted by a processing mechanism (the brain) to make sense of the surroundings.

The process becomes harder if the environment is not static i.e. constantly changing in terms of viewpoints (e.g. self-motion), dynamic content (e.g. moving objects) and lighting conditions (e.g. day/night, shadows, etc.). Fortunately, our

brains dedicate half of the cerebral cortex, the outer layer of the brain, for this processing [47] and can perform the necessary `calculations' to understand these spatial relationships instinctively. Trying to emulate the same functionality with computers instead of the human brain using cameras as sensors is harder.

Most of the applications require an understanding of the scene and finding spatial parameters for the camera, which is an involved process. Common approaches start with a camera calibration step, which aims to identify the internal parameters of the camera, and continues by finding and extracting useful bits of information called features from the images; and then calculating a signature or `descriptor' for these features that is assumed to identify them uniquely. These descriptors are then used to establish correspondences between images, after which methods for motion estimation can be used to find spatial parameters such as position and orientation. The following subsections explain some of the sensors, algorithms and methods that make such applications possible.

In computer vision, we are trying to describe the world by calculating of camera calibration, pose or coordinates etc., as well as recognition of images then reconstruct its properties such as their shape, illumination, and color distributions. Most importantly augmented the real world by 3D virtual objects.

2.5.1. Cameras and camera calibration

A digital camera can be viewed as two components, the lens and the imaging sensor. Reflected light from objects pass through the lens and is then projected onto the sensor, which can be manufactured as Charge Coupled Device (CCD) or Complementary Metal Oxide Semiconductor (CMOS) device, both comprising of an array of sensors sensitive to light. These sensors convert the light into electrical signals which can be read out digitally for storage or processing. This relatively complex imaging process is normally represented using an ideal pinhole camera model [48, 49, 50]. In this simple model, shown in Figure 2.3, the camera is modelled using a 3D position for the optical center and a 2D image plane. The focal length of the camera is the shortest distance between the optical center and the image plane. The projection of a 3D point can be

obtained by drawing a line from the optical center through the image plane to the 3D point. The projection is found as the 2D location on the image plane.

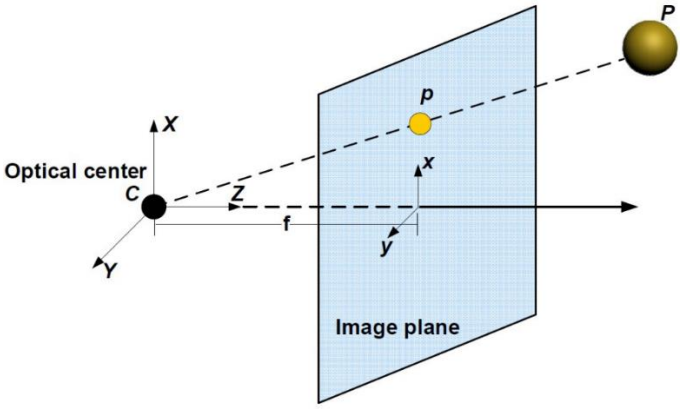

Figure 2.3 Pinhole camera model. C is the optical center and f is the shortest distance from C to the image plane. P is a 3D point with its projection p on the image plane

Unlike this theoretical representation, real-world cameras introduce distortion due to problems in the manufacturing process. For a more realistic representation, these distortion parameters should also be included in the projection model. The process for finding these parameters (as well as other internal parameters such as the focal length) is called camera calibration [50, 51]. There are dedicated toolboxes for this purpose (e.g. [51]) which can be used to find the distortion parameters as long as an image sequence acquired with that camera is provided. A camera is used for vision-based user tracking algorithm and the camera calibration is performed as described in Chapter 4.1

2.5.2. Feature detection and description

Features are often depends on the problem or what you intend to applicate. So definition is depends on purpose. In our purpose in image processing or computer vision, it can be defined as an "interesting" part of an image, and features are a starting point for many computer vision algorithms and also it is an image primal that contains valued information about the content of the image [52]. As a result, for feature detector usually desired repeatability: whether or not the same feature will be detected in two or more different images of the same scene [43].

Every feature appearing in an image shadows a real-world object. A feature can be in form of a corner [53] , an edge [54], a small region (blob) [55] or a segment [56]. Features are represented using descriptors, which are calculated using the pixel information around the feature using a variety of methods: A small patch of surrounding pixels can include the descriptor, or a more complex description like an oriented gradient histogram [57].

The literature presents many different feature detectors and descriptors. An evaluation of many feature detectors can be found in [58]. Based on the review given therein, a good feature detector should be able to detect features that are stable in terms of geometry under different viewing conditions [59, 60] should present important amount of variation in its neighborhood so that they will be prominent and provide useful information as well as presenting good localization accuracy [61]. It is also important for the detector to detect such features in a reasonable amount of time, a vital requirement for real-time applications [62].

Scale Invariant Feature Transform (SIFT) [63] works by selecting candidate key-points from locations which can be repeatedly chosen under different orientations and scales. Scale invariance is achieved by using a \scale space" which appears as a pyramid of images consisting of the octaves created by resizing the original image to its half size and then applying a Gaussian blur operation. Keypoints are found using a method called Difference of Gaussians (DoG) as an approximation of Laplacian of Gaussian (LoG). A local descriptor is then generated by calculating the magnitude and orientation of the gradient. Later, a feature vector is computed using a histogram of these orientations.

Speeded-Up Robust Features (SURF) [64] were developed as an improvement to SIFT for extracting features in a shorter time, employing integral images as an intermediate image representation and using Hessian-Laplacian to approximate LoG. For the description, Haar wavelet responses inside a circular window are summed to obtain the orientation vector of the feature. SURF is also claimed to be more invariant to affine transformations such as translations or rotations than SIFT by its authors.

It is known that scale invariance on its own is not enough to show robustness against changes in viewpoint, which result in affine transformations in

the image [65]. For this reason, a number of affine-invariant feature detectors have been proposed.

2.6. Discussion

This chapter presented a large portion of Augmented Reality from different research topics with the aim of promoting the miscellaneous different solutions developed in this thesis. The next five chapter's present technology developed for mobile phones based AR and several applications.

3. IMAGE FEATURES

The interrogation for images are mostly to compare images directly. That is, the pixel values of the image itself or a scaled version of image are compared directly to the corresponding values of other images. However, this method is not suitable for lots of application, since it is not clear which pixels are correspond to pixels in the other image [66].

Additionally taking the pixel values themselves several extensions are possible. For certain properties of the image, filters and transformations can be applied to the image, e.g. discrete cosine transformation or PCA transformation to give a more compact representation and Sobel filters are applied to emphasize edges [66]. Besides, image patches between images with significantly different viewpoints or image landmarks such as their (x, y) position, scale and orientation can be identified as image features.

3.1. Feature Detectors and Descriptors

Feature tracking and detection algorithms are widely used for different purposes in computer vision applications. They are applied in image matching, tracking, mosaicing, 3D modelling, motion detection, object recognition and panorama stitching. In this instance, tracking was considered as a means for detecting the relative pose of the camera.

We can expose localized features into three categories: feature points (e.g. corners), feature descriptors (e.g. SURF, SIFT) and edges. A feature point (an interest point or keypoint) which has a clear definition and a well-defined position is a small area of an image.

A detector is used to create the descriptor and it needs to be repeatable, meaning the same feature needs to be detected in two or more different images of the same scene accounting for lighting and/or viewpoint changes.

3.1.1. Feature Description

A descriptor is a description of the specific point from the image stored in the database, application, or service. For a good descriptor, clearance and invariance are two main requirements. By meaning clearance is that feature points

corresponding to two different physical points result in different descriptors. As for invariance is to changes in view points and directions, image noise and illumination [67]. Steady detectors are selected in the image from the detection step. In the description step, each interest point is represented by a feature vector, which is a description of the point. To get image information, image gradients are used. Image gradients give details on the directional change of the intensity or color in an image.

The computation of the keypoint descriptor is shown in Figure 3.1. At first orientations and gradient magnitudes of the image are sampled around the keypoint location. Then, to select the level of Gaussian blur for the image, used the scale of the keypoint. The orientation invariance is achieved by rotating the gradient orientations and the coordinates of the descriptor to relative the keypoint orientation. Which are shown in left side of Figure 3.1 with small arrows at each sample location. By means of Gaussian weighting function σ window, with a circular window which is shown in left side of Figure 3.1 is used for allocation of a weight to the magnitude of each sample point [63].

The keypoint descriptor is shown in right side of Figure 3.1. By creating orientation histograms over 4x4 sample regions, this allows for important shift in gradient position. For each orientation histogram the figure shows eight directions with the length of each arrow corresponding to the magnitude of that histogram input. A gradient sample on the left can shift up to 4 sample positions while still contributing to the same histogram on the right, thereby achieving the objective of allowing for larger local positional shifts [63].

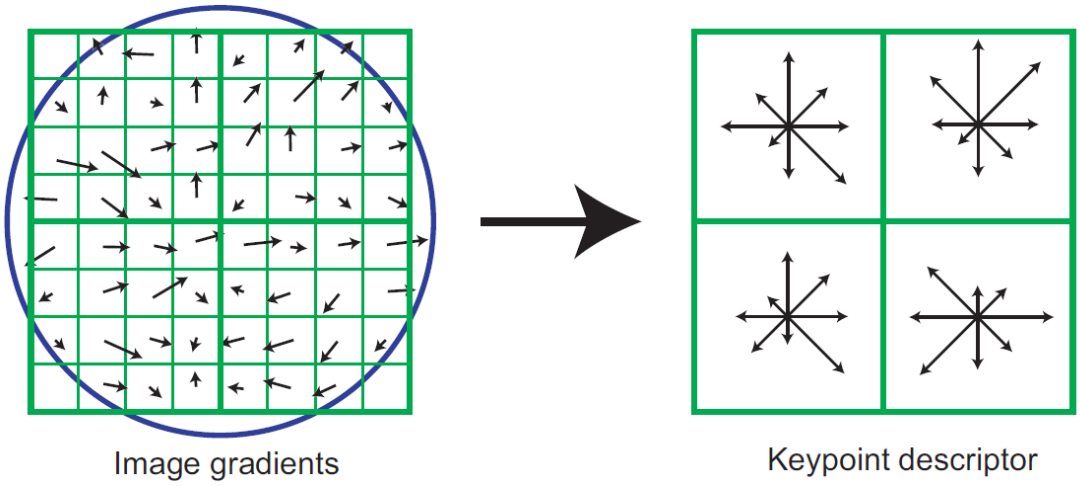

Figure 3.1 Image gradients with a Gaussian window, indicated by the overlaid circle as shown on the left. A keypoint descriptor created by first computing the gradient magnitude and orientation as shown on the right [62]

3.1.2. Feature extraction

Feature-detection algorithms, which search for corners, edges or blobs. In our case we are interested in corner detection. The corner detection is based on an analysis of the edges in the image. A corner-based edge detection algorithm searches for quick changes in the image gradient. Usually it's done by looking for extremism of the first derivative of the image gradients in the X and Y directions. Feature-point orientation is usually computed as a direction of dominant image gradient in a particular area. When the image is scaled or rotated, the orientation of dominant gradient is recomputed by the feature-detection algorithm. This means that regardless of image rotation, the orientation of feature points will not change. Such features are called rotation invariant. Also, I have to touch on the size feature point. Some of the feature-detection algorithms use fixed-size features, while others calculate the optimal size for each keypoint separately. Knowing the feature size allows us to find the same feature points on scaled images. This makes features scale invariant [68].

For understanding feature extraction we have work on OpenCV feature-detection algorithms, and we used SURF method for markerless tracking. OpenCV has several feature-detection algorithms. All of them are derived from the base class cv::FeatureDetector. To use SURF feature-detection algorithm;

```
cv::Ptr<cv::FeatureDetector> detector =
cv::Ptr<cv::FeatureDetector>(new cv::SurfFeatureDetector());
```

To detect feature points, we call the *detect* method:

```
std::vector<cv::KeyPoint> keypoints; detector->detect (image, keypoints);
```

The detected feature points are placed in the *keypoints* container. Each keypoint contains its center, radius, angle, and score, and has some correlation with the "quality" or "strength" of the feature point.

The best results in pattern detection are achieved if the detector computes keypoint orientation and size. This makes keypoints invariant to rotation and scale. The most famous and robust keypoint detection algorithms are well known, they are used in SIFT and SURF feature detection / description extraction.

If we deal with images, which usually have a color depth of 24 bits per pixel, for a resolution of 640 x 480, we have 912 KB of data. How do we find our pattern image in the real world? Pixel-to-pixel matching takes too long and we will have to deal with rotation and scaling too. And this is not an option to achieve. By using feature points this problem can be solved. By detecting keypoints, we can be sure that returned features describe parts of the image that contains lot of information (that's because corner based detectors return corners, edges and other sharp figures). Therefore to find correspondences between two frames, we only have to match keypoints.

From the patch defined by the keypoint, we extract a vector called descriptor. It's a form of representation of the feature point [68].

3.2. SURF (Speeded-Up Robust Features)

SURF (Speeded-Up Robust Features) introduced in 2006 [64]. SURF is a speeded-up version of SIFT (Scale-invariant feature transform). In SIFT, David G. Lowe for finding scale-space approximated Laplacian of Gaussian with Difference of Gaussian. SURF is a robust image descriptor, published by Herbert Bay [64], which can be used in computer vision areas like 3D Reconstruction, Object Recognition or AR applications. SURF detects Hessian blob like structures

and fundamentally it is based on determinant of the Hessian Matrix [69, 70], SURF features are scale, rotation and translation invariant.

In order to searching of extraction image point feature, SURF has two main steps. First, SURF interest points are selected at distinctive locations in the image, such as blobs, corners and T-junctions. Next, the neighborhood of every interest point is represented by a feature vector.

SURF interest points can be found by calculating an interest point criteria $R(x, y)$ which is the blobness value of a pixel in the image. The blobness value R can be formulated with blobness function f as follows with input image I

$$R(x, y) = f\big(I(x, y)\big) \tag{3.1}$$

For robustness to scale changes, a collection of the input image in different scales is considered. Therefore $I(x, y)$ becomes a 3D data which is $I(x, y, \sigma)$ "image pyramid". Here σ is scale parameter. So interest point criteria becomes

$$R(x, y, \sigma) = f\big(I(x, y, \sigma)\big) \tag{3.2}$$

The blobness value of a pixel in the image is the determinant of the Hessian Matrix which is equal to interest point criteria. Thus, Hessian Matrix can be define as follows

$$H(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{yx}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix} \tag{3.3}$$

Here $L_{xx}(x, y, \sigma)$ is the convolution of $\frac{\vartheta^2}{\vartheta x^2} g(\sigma)$ (second order derivative Gaussian) with input image I at point(x, y). The blobness value $R(x, y, \sigma)$ finally becomes as follows

$$R(x, y, \sigma) = det\big(H(x, y, \sigma)\big) \tag{3.4}$$

In order to save time Bay [64] suggest an approximation for the second order Gaussian derivative kernel which proper box filter kernels. Instead of a discretized Gaussian kernel, this mostly affect the performance of the algorithm. In order to clarify the advantage of box filter more clearly, integral of the input image is obtained. Integral images developed by Viola and Jones [71]. Discretized Gaussian Kernels and related box filters are shown in Figure 3.2.

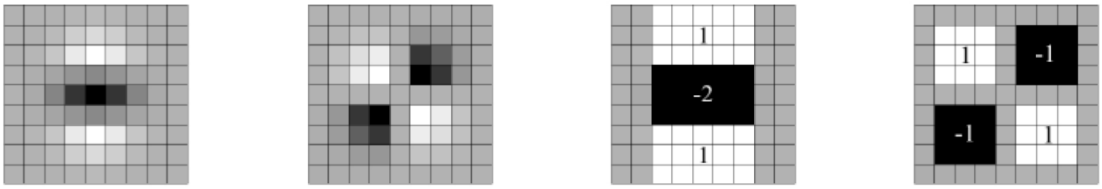

Figure 3.2 Exact and approximated Gaussian kernels [64]

Integral image $I_{\Sigma}(x, y)$ of an image $I(x, y)$ is defined as follows [64]

$$I_{\Sigma}(x, y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(x, y) \tag{3.4}$$

The intensity value at (x, y) in the integral image $I_{\Sigma}(x, y)$ is the sum of the pixel values above and to the left of (x, y) included shown in Figure 3.3.

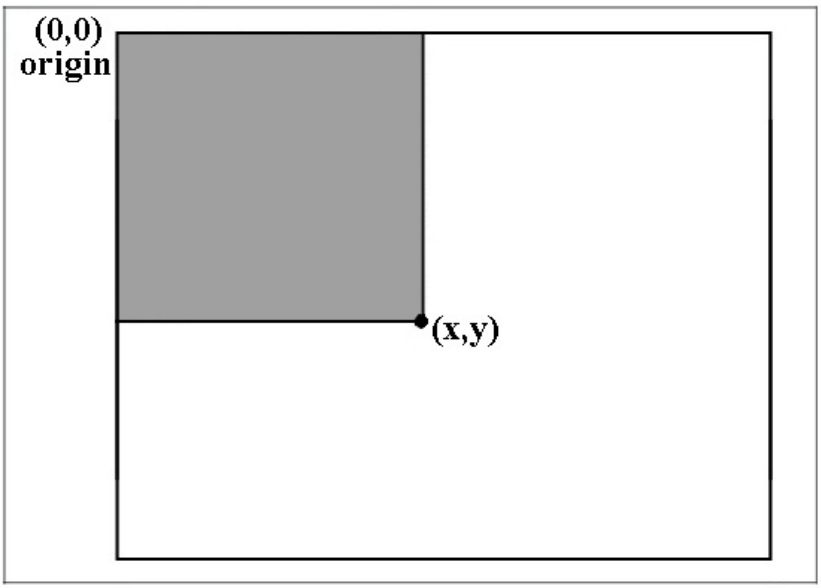

Figure 3.3 Rectangular Region for Integral Image

Also demonstration of integral image (on the left) of Lena image (on the right) is shown in Figure 3.4.

Figure 3.4 Lena image and the corresponding (normalized) integral image

The integral image concept is making easier calculating the summation of the pixel intensities in a rectangular area on the image as can be seen in Figure 3.5.

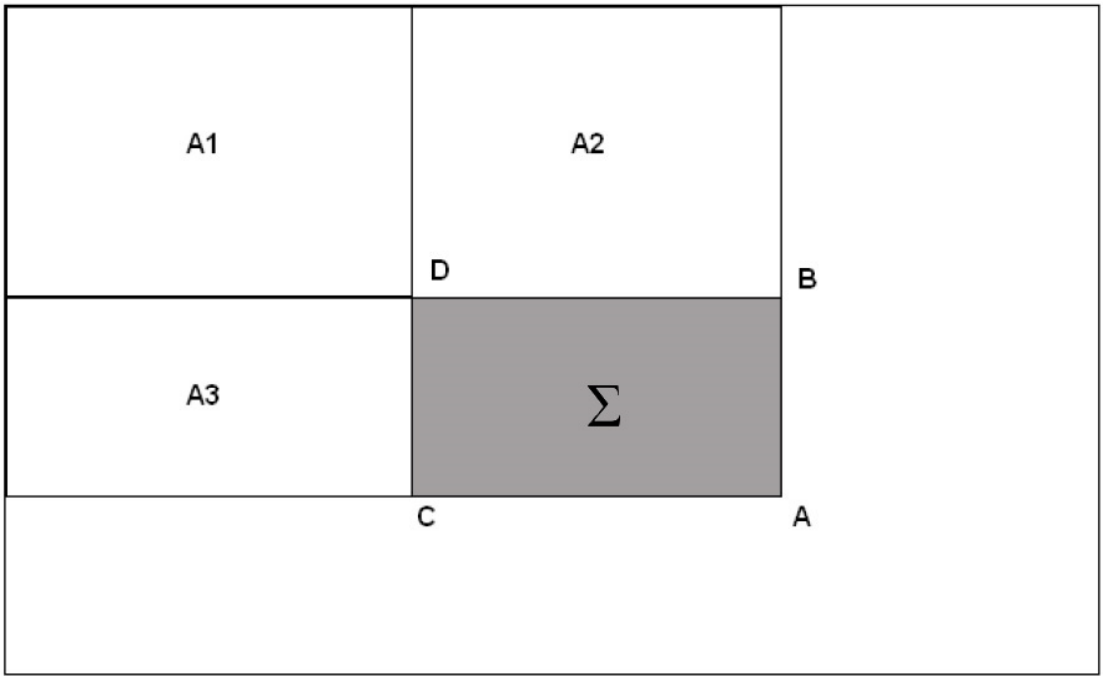

Figure 3.5 Box filtering example

Let's consider an image $I(x, y)$ in the Figure 3.5. Number of summations $(B - D)x(A - B)$ which the normal operation is to calculate summation of the pixels in the region Σ. Assume the integral image $I_\Sigma(x, y)$ corresponding to the image $I(x, y)$;

$$I_\Sigma(A) = A_1 + A_2 + A_3 + \Sigma \tag{3.5}$$

$$I_\Sigma(B) = A_1 + A_2 \tag{3.6}$$

$$I_\Sigma(C) = A_1 + A_3 \tag{3.7}$$

$$I_\Sigma(D) = A_1 \tag{3.8}$$

After mathematical operations we get integral image value (rectangle sum) for an arbitrary rectangular region Σ inside the image is as follows;

$$\Sigma = I_\Sigma(A) + I_\Sigma(D) - I_\Sigma(B) - I_\Sigma(C) \tag{3.8}$$

The formula (3.8) can be demonstrate as shown in Figure 3.6.

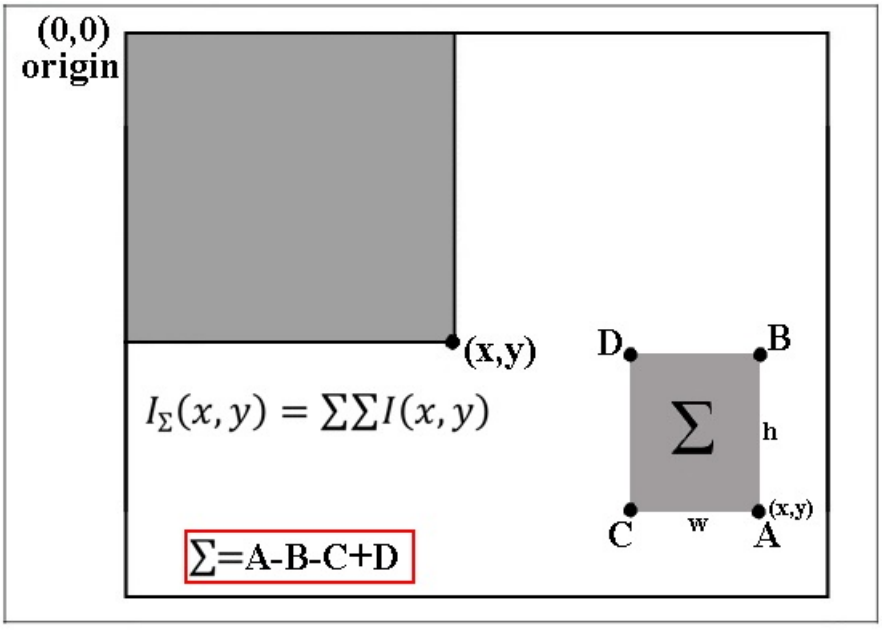

Figure 3.6 Calculation of an Arbitrary Rectangle Sum

SURF feature detector is based on determinant of Hessian matrix [69, 70] for both scale and location. Therefore second order partial Gaussian differentials are established in scale space. In order to approximate Hessian matrix determinant calculation, considered the smallest kernel (9x9) of box filter in Figure 3.7.

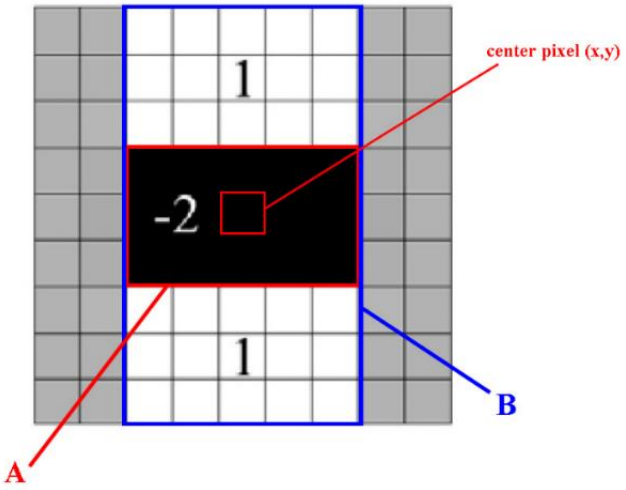

Figure 3.7 Smallest kernel for box filtering

The determinant of the approximated Hessian matrix is define in (3.9). Constant multiplier is 0.9 in order to normalize the error caused by the approximation. The second order partial Gaussian derivative filter in x-direction L_{xx}, y-direction L_{yy} and xy-direction L_{xy} shown in Figure 3.8. Approximation for the second order partial Gaussian derivative filter in x-direction D_{xx}, y-direction D_{yy} and xy-direction D_{xy}.

$$det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \quad (3.9)$$

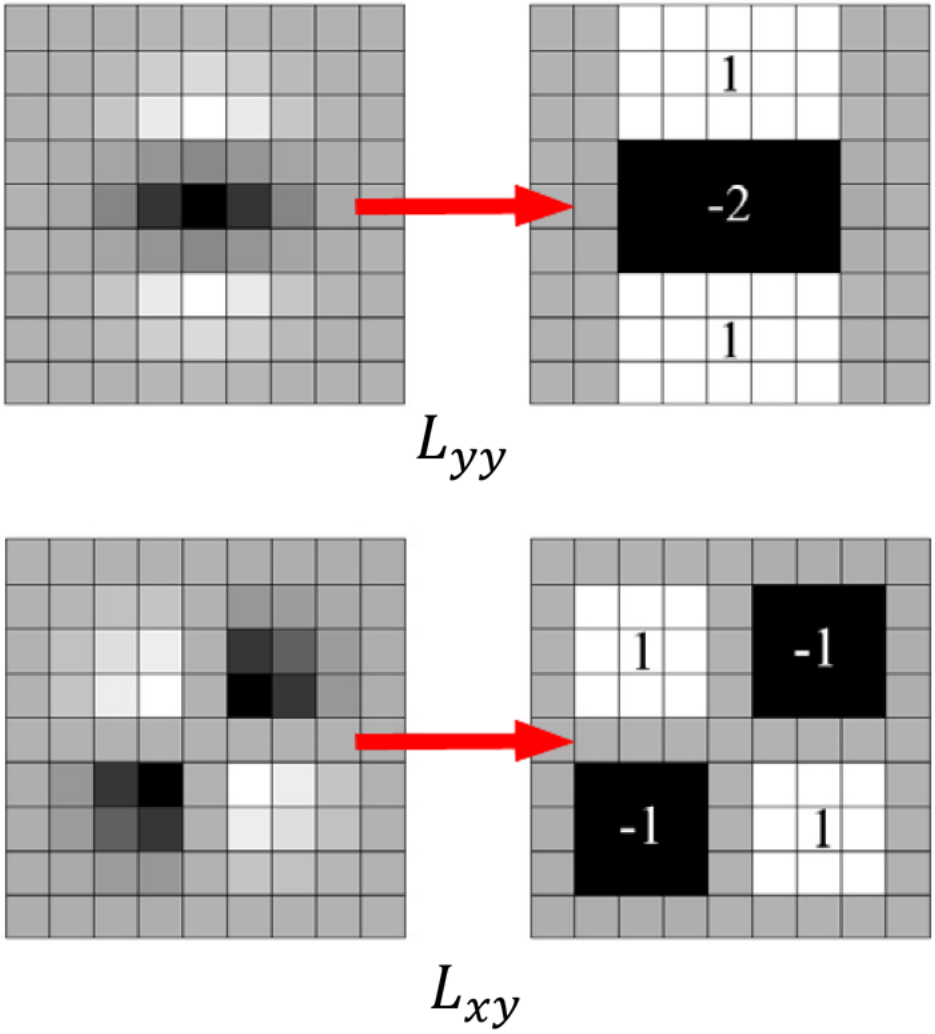

Figure 3.8 Box Filter for Second Order Partial Gaussian Derivative in y-direction on the top and in xy-direction on the bottom [64]

Scale space of SURF feature detector is implemented by using image pyramids. The initial filter kernel size is 9x9 and the image is filtered by filter kernels of 15x15, 21x21, 27x27 and so on for the next scale levels of the first octave. The filter size increase 12 for the next octave, beginning from 39x39 filter kernel. The scale space structure of the SURF detector is shown in Figure 3.9 below.

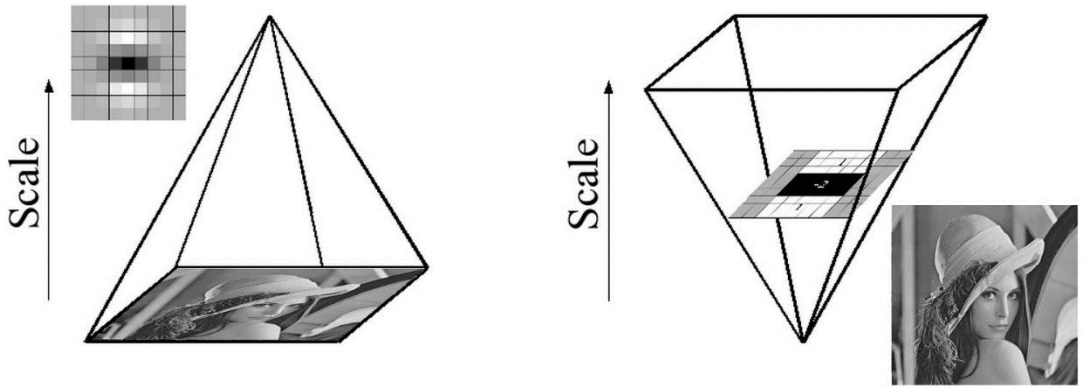

Figure 3.9 Scale Space Structure of SURF Feature Detector

After finding approximated hessian determinant values in all scales and candidate, obtained final step is “Non-maxima Suppression”.

Approximated hessian determinant values through the image in all scales and candidate interest features are found. Final step to obtain SURF features is “Non-maxima Suppression”. A blob on image may give blobness response on more than one scale or more than one point on the coordinate plane. So, a candidate point is chosen as SURF feature if its blobness response is greater than its entire 3x3x3 neighborhood in x, y, σ dimensions. The visualization is shown in Figure 3.10.

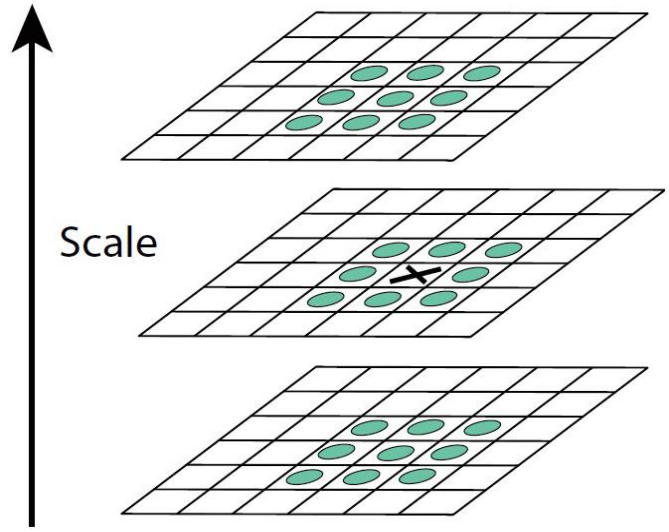

Figure 3.10 Non-maxima suppression for candidate SURF features [63]

So far, SURF interest points on an image are found. For each features, descriptor calculation needs. Descriptors are used for the matching step. Haar wavelets are operated during descriptor calculation steps. 2D Haar wavelets responses makes an efficient use of integral images. Haar wavelet in Figure 3.11 are simple filters for gradients calculations. The left filter in the x- direction and the right filter in the y-direction computes the response. Weights of black region is 1 and -1 for white region [72].

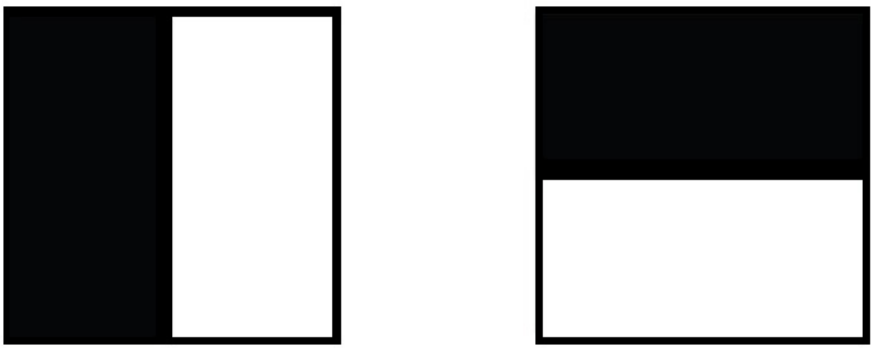

Figure 3.11 Haar Wavelets [64]

Haar wavelet's mother wavelet function $\psi(t)$ is shown below.

$$\psi(t) = \begin{cases} 1 & 0 \leq t < 1/2, \\ -1 & 1/2 \leq t < 1, \\ 0 & otherwise \end{cases} \tag{3.10}$$

Scaling function $\varphi(t)$ of Haar wavelet is also shown below.

$$\varphi(t) = \begin{cases} 1 & 0 \leq t < 1, \\ 0 & otherwise \end{cases} \tag{3.11}$$

SURF descriptor calculation occurs with two main steps which are orientation assignment and calculation of descriptor components. In the first step a robust and repeatable orientation is assigned for each SURF feature. In the second step calculated descriptor components. Based on the orientation which is calculated before. Finally applying these two procedure for each SURF feature, a

descriptor array of size 64 (16x4) is constructed. Similarity of two features determined by calculating the Euclidean distance between their descriptors. Figure 3.12 visualizes the descriptor concept as well as the descriptor formulation. One of the 16 subregions which is the green square and blue circle inside its represents the sample points at which computed the wavelet responses.

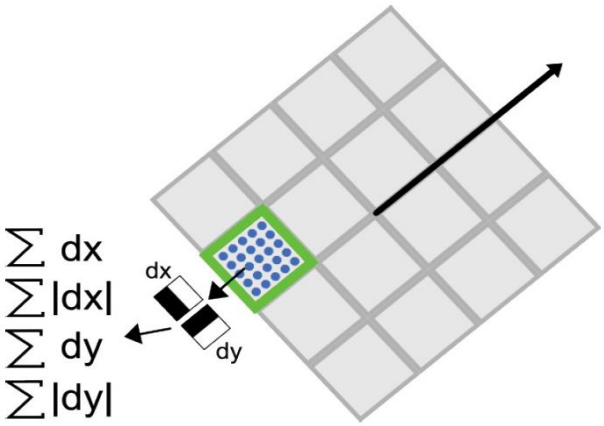

Figure 3.12 Calculation of SURF descriptor components [72]

3.3. SURF Detector Algorithm

Table 3-1 SURF Algorithm Steps

SURF Detector Algorithm

1. Finding integral image of the input image
   - ✓ Integral image value $I_\Sigma(x, y)$ at pixel (x, y) is calculated in a single pass.

$$I_\Sigma(x, y) = I(x, y) + I_\Sigma(x, y-1) + I_\Sigma(x, y)+I_\Sigma(x-1, y) - I_\Sigma(x-1, y-1)$$

$$\Sigma = I_\Sigma(A) + I_\Sigma(D) - I_\Sigma(B) - I_\Sigma(C)$$

   - ✓ The integral image values are stored as a look-up table for later use in construction of differential scale space.
   - ✓ End
2. Differential SURF Scale Space of input image is constructed.

- ✓ Integral image look-up table is initialized.
- ✓ Box filter of size 9x9 is initialized based on the following second order differential filter kernels.

$$h_{xx} = [1 \quad -2 \quad 1] \qquad h_{yy} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

$$h_{xy} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

- ✓ Integral image values corresponding to the inner rectangular regions of size 9x9 is calculated.

$I_\Sigma(x, y, h, w) = I_\Sigma(x, y) + I_\Sigma(x + w, y + h) - I_\Sigma(x, y + h) - I_\Sigma(x + w, y)$

height (h) and width (w) of the rectangular regions are 9.

- ✓ Second order partial Gaussian derivatives of input image D_{xx}, D_{yy} and D_{xy} are calculated by using box filters.
- ✓ Box filters of sizes 15x15, 21x21 and 27x27 with an increase of 6 units are initialized for next scale levels of the first octave.

SURF Detector Algorithm(Continued)

- ✓ Second order partial Gaussian derivatives of input image D_{xx}, D_{yy} and D_{xy} are calculated for next scales.
- ✓ Box filter size is doubled.
- ✓ Box filters of sizes 39x39, 51x51, 63x63, and so on with an increase of 12 units are initialized for next scale levels of the second octave.
- ✓ Continue smoothing until number of octaves corresponding to the size of box filters is less than the size of the input image.
- ✓ End, differential SURF scale spaceD_{xx}, D_{yy} and D_{xy} is obtained.

3. Calculation of determinant of Hessian matrices
   - ✓ Components of approximate Hessian matrices are initialized for all image coordinates at all scale levels.
   - ✓ Weights of the derivatives are balanced by Frobenius norm of the second order Gaussian partial derivatives.

$$\frac{|L_{xy}(1.2)|_F |D_{xx}(9)|_F}{|L_{xy}(1.2)|_F |D_{xy}(9)|_F} \cong 0.9$$

- ✓ Determinant of Hessian matrices are calculated.

  $$det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2$$

- ✓ End

4. Finding local maxima of determinant measure
   - ✓ Local maxima of determinant values are found in a region around each point of the image.
   - ✓ Local maximum points are stored whose values are greater than some local maximum threshold.
   - ✓ Candidate SURF feature points are obtained.
   - ✓ End
5. Subpixel localization of initial SURF feature points
   - ✓ Initial feature points are initialized.
   - ✓ Non-maximum suppression is applied in a 3x3x3 neighborhood of each initial feature point.
   - ✓ Initial feature points are localized in space and scale.
   - ✓ The value of the maximum of the determinant is interpolated in space and scale.
   - ✓ End, final SURF feature points are obtained.

In this thesis for the extraction of SURF feature points mainly used SURF detector implementation of Bay et.al. The algorithm is developed in C++ environment and OpenCV library used. The purpose of this chapter was natural feature tracking for the markerless augmented reality tracker part. However, for purpose of mobile augmented reality SURF algorithm by using Mobile version of OpenCV was not efficient and robust. So this approaches are unsuitable for low-end embedded platforms such as phones. Because of that we used Vuforia SDK for tracking part in mobile application.

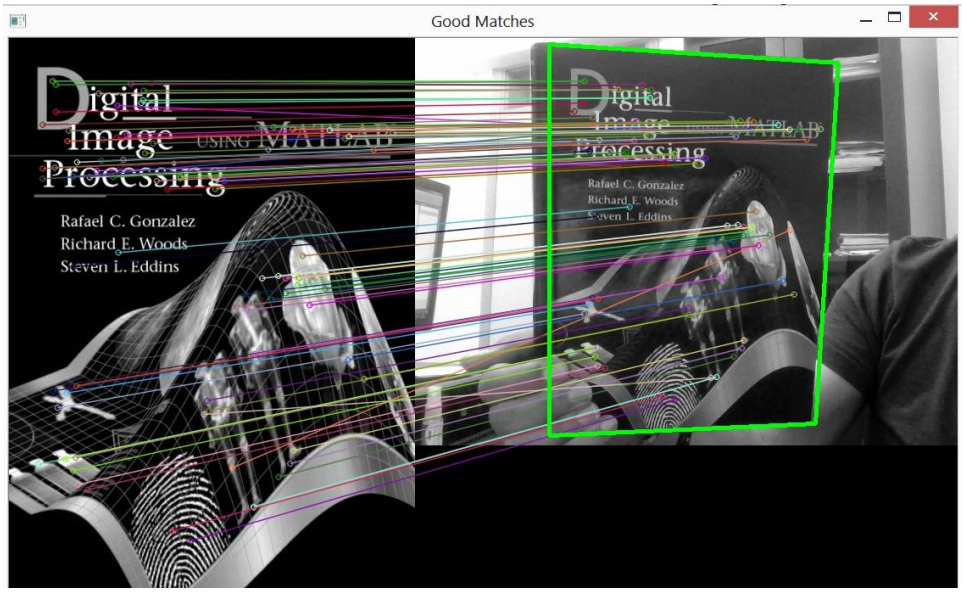

Figure 3.13 SURF implementation with good matches

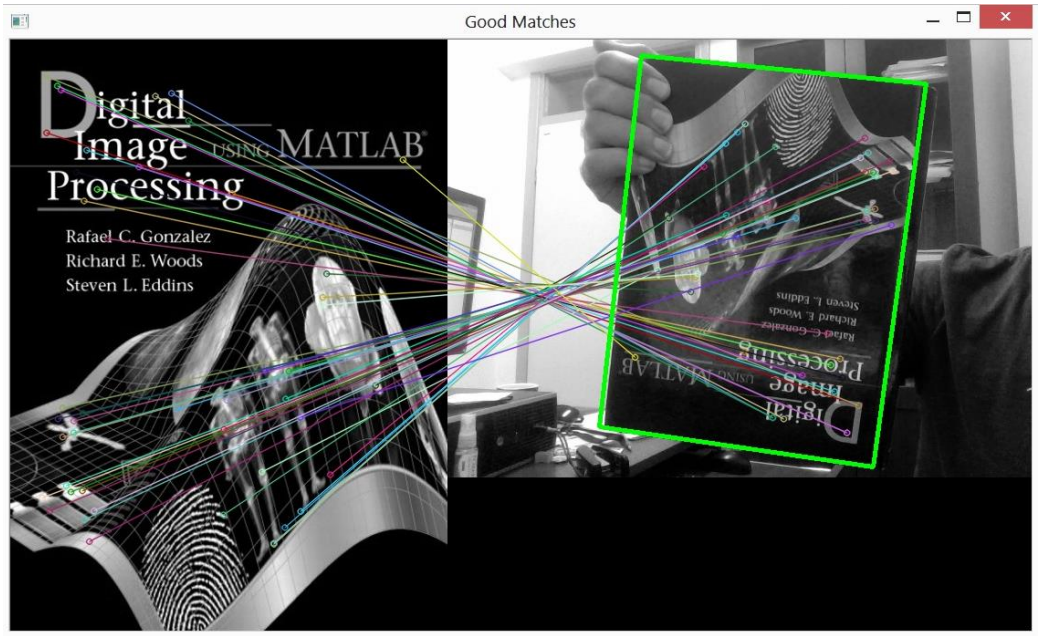

Figure 3.14 SURF implementation with rotated target good matches

4. POSE TRACKING

This chapter introduces the concepts of pose estimation and presents Vuforia SDK solution for markerless target tracking on mobile phones. It gives details on the phone specific features. Augmented Reality (AR) and Virtual Reality (VR) require real-time and proper 6DOF pose tracking of devices. Pose tracking should be cheap, work robustly in changing environmental conditions, provide automatic localization in global coordinates and support a large working area.

4.1. Camera Calibration

Camera calibration is an important issue in computer vision. With calibrated camera, the system can render 3D virtual objects on the target in the correct place. Camera parameters consist of the intrinsic and extrinsic parameters. The intrinsic camera parameters are the vertical and horizontal focal lengths and the principal point of the camera and the skew. For satisfy distortion of the camera, also the tangential and radial distortion coefficients up to second order need to be computed. To calibrate these, an enough amount of 2D–3D correspondences has to be created, this is typically done using a known target like a chessboard. For the calibration of the low resolution camera, in this thesis used the calibration method from Zhang [73] from OpenCV [7].

Here captured inner corners of the chessboard from live camera, then by updating the extrinsic and intrinsic parameters of the camera, iteratively minimized the squared distance of the reprojection of the chessboard corners to the detected corners [1]. Experimental work shown in Figure 4.1 with good results.

Figure 4.1 Several screen shoots of Camera Calibration application

If enough capturing inner corner didn't, the camera calibration get wrong parameters and the view would be erratic as shown comparison of with calibrated camera in Figure 4.2.

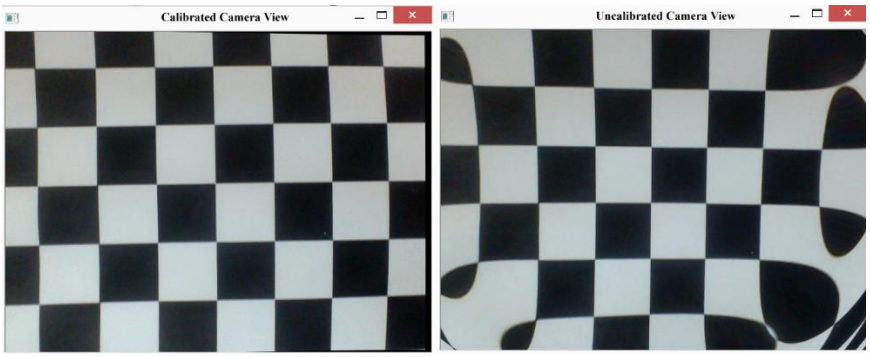

Figure 4.2 Comparison of Calibrated and Uncalibrated Camera

After calibration, the application will create two XML (Extensible Markup Language) files which are include Distortion and Intrinsic parameters. Intrinsic parameters are as follows:

```
<?xml version="1.0"?>
<opencv_storage>
<Intrinsics type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>f</dt>
  <data>
    6.15828369e+002 0. 3.71354889e+002 0. 6.15828369e+002
    2.26973648e+002 0. 0. 1.</data></Intrinsics>
</opencv_storage>
```

In this thesis used OpenCV library for experimental work and an A4 Tech webcam so-called pinhole camera model calibrated. Camera calibration theoretical steps shown as following equations. In this model, a scene view is formed 3D points into the image plane using a perspective transformation.

$$s\mathrm{m}' = \mathrm{A}[R|t]M' \tag{4.1}$$

Or

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & \alpha \end{bmatrix} \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{array}\right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.2}$$

Where:

- (u,v) are the coordinates of the projection point in pixels
- A is a camera matrix which matrix of intrinsic parameters

- (c_x, c_y) is parameters of intrinsic matrix which is principal point that usually at image center.
- f_x, f_y are the focal lengths expressed in pixel units.
- α is aspect ratio usually 1. Note that, $f_y = f_x * \alpha$
- r_{ij} and t_i are the joint Rotation and Translation matrix elements.
- (X,Y,Z) are the coordinates of a 3D point in the world coordinate space.

Thus, if an image from the camera is scaled by a factor, all of these parameters should be scaled (multiplied/divided, respectively) by the same factor. The matrix of intrinsic parameters does not depend on the scene viewed. So, once estimated, it can be re-used as long as the focal length is fixed (in case of zoom lens). The joint rotation-translation matrix $[R|t]$ is called a matrix of extrinsic parameters. It is used to describe the camera motion around a static scene, or vice versa, rigid motion of an object in front of a still camera. That is, $[R|t]$ translates coordinates of a point (X,Y,Z) to a coordinate system, fixed with respect to the camera. The transformation above is equivalent to the following (when ≠ 0):

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \tag{4.3}$$

$$x' = x/z \tag{4.4}$$

$$y' = y/z \tag{4.5}$$

$$u = f_x * x' + c_x \tag{4.6}$$

$$v = f_y * y' + c_y \tag{4.7}$$

Real lenses usually have some distortion, mostly radial distortion and slight tangential distortion. So, the above model is extended as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \tag{4.8}$$

$$x' = x/z \tag{4.9}$$

$$y' = y/z \tag{4.10}$$

$$x'' = x' \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + 2p_1 x' y' + p_2\left(r^2 + 2x'^2\right) \quad (4.11)$$

$$y'' = y' \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + p_1\left(r^2 + 2y'^2\right) + 2p_2 x' y' \quad (4.12)$$

Where

$$r^2 = x'^2 + y'^2 \quad (4.13)$$

$$u = f_x * x'' + c_x \quad (4.14)$$

$$v = f_y * y'' + c_y \quad (4.15)$$

k_1, k_2, k_3, k_4, k_5 *and* k_6 are radial distortion coefficients. p_1 *and* p_2 are tangential distortion coefficients. In the functions below the coefficients are passed or returned as:

$$\left(k_1, k_2, p_1, p_2, \left[k_3, [k_4, k_5, k_6]\right]\right)$$

vector. That is, if the vector contains four elements, it means that $k_3 = 0$. The distortion coefficients do not depend on the scene viewed. Thus, they also belong to the intrinsic camera parameters. And they remain the same regardless of the captured image resolution. If, for example, a camera has been calibrated on images of 320 x 240 resolution, absolutely the same distortion coefficients can be used for 640 x 480 images from the same camera while f_x, f_y, c_x *and* c_y need to be scaled appropriately.

Camera calibration is needed for property augmentation of 3D model on our target, Augmentation of a Cube on the target by using calibrated and uncalibrated camera is shown in Figure 4.1. In this thesis we have worked on desktop application to understand camera calibration features and needed approach for mobile 3D reconstruction.

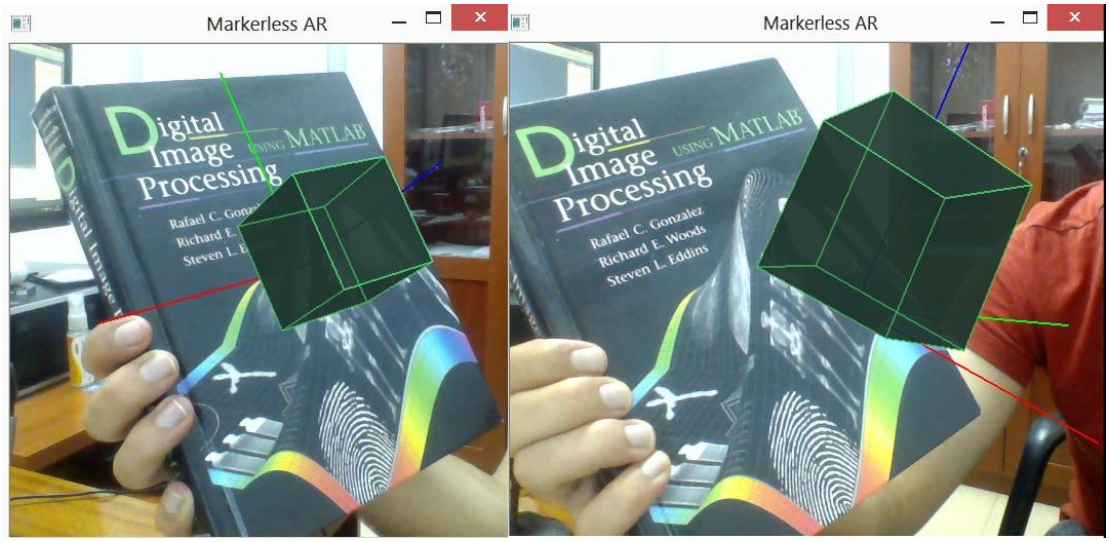

Figure 4.3 Augmentation of a Cube with calibrated camera (Left) and uncalibrated camera (Right)

4.2. Pose Tracking

Pose is a position and orientation of an object (6DOF) and description of pose estimation is that getting the pose of an object from a 2D image. In order to get convenient augmentation to my thesis, we have to deal with the pose estimation. Therefore, we did some theoretical researches and experimental work on this.

Previous subsection on camera calibration, we have found the camera matrix, distortion coefficients etc. By given target image, we can use this information to calculate its pose, or how the object is situated in space, like how it is displaced, how it is rotated etc. We assume that Z=0 for a 2D object, so, the problem now becomes how camera is placed in space to see our target image. Therefore, if we know object location in the space, we can draw some 2D diagrams in it to simulate the 3D effect.

Problem is that we want to draw our 3D coordinate axis (X, Y, Z axes) on our chessboard's first corner. X axis in blue color, Y axis in green color and Z axis in red color. Therefore, Z axis should feel like it is vertical to our chessboard target.

Initially, we load the camera matrix and distortion coefficients from the camera calibration result. Then, we take the corners in the chessboard and axis points to draw a 3D axis.

Then, we create object points (3D points of corners in chessboard) and axis points. Axis points are points in 3D space for drawing the axis. After that, searches for 8x5 grid (chessboard size). If found, we depurate it with subcorner pixels. Then calculates the rotation and translation [7]. Finally, we draw it on chessboard target. The results shown in Figure 4.4.

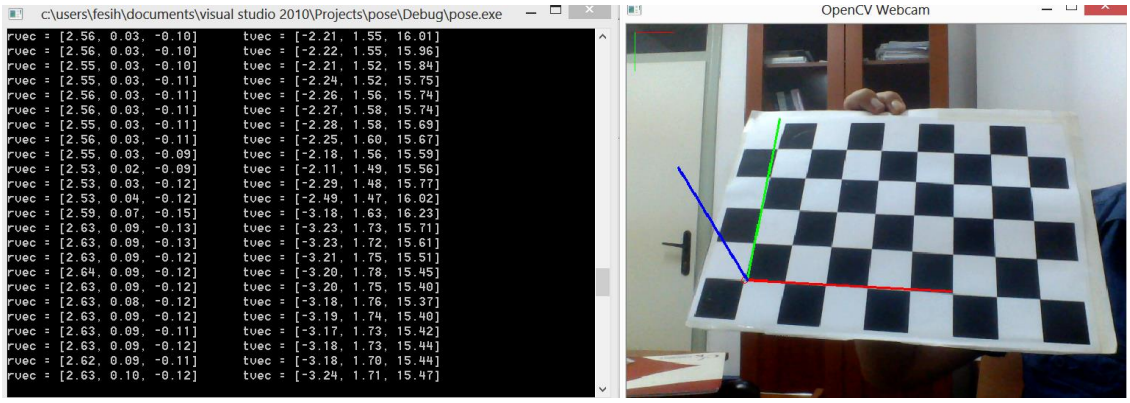

Figure 4.4 Pose estimation experimental work

5. RENDERING

Real time rendering is related with generating images quickly on the camera scene. This is totally interactive on computer vision, computer graphics and augmented reality applications etc. The rendering pipeline occurs with a 2D image, a virtual camera, 3D object, light sources, shading equations, textures, and more. The process of using the pipeline is depicted in Figure 5.1. The location and shape of the object on the camera are determined by their geometry [74].

Figure 5.1 A 3D virtual model with completed of rendering pipeline

In mobile applications prompt rendering is needed. Therefore, we use a 3D API which is OpenGL for 3D rendering in my augmented reality application [8]. OpenGL is a common tool which is represented in research and professional applications [4]. OpenGL render pipeline is shown in Figure 5.2. OpenGL has two stages; first stage is geometry stage which consists model and view transform, vertex shading, projection, clipping and screen mapping. Second stage is rasterizer stage which consists triangle setup, triangle traversal, pixel shading and merging [74].

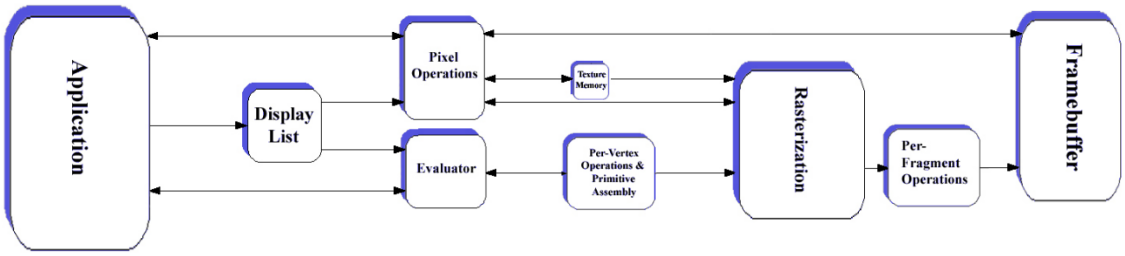

Figure 5.2 The OpenGL Render Pipeline

Creating a 3D model has crucial importance for augmented reality applications. In our application we chose Wavefront OBJ 3D files. Reason of choosing this file format is a simple data-format that represents 3D geometry alone namely, the position of each vertex, the UV position of each texture coordinate vertex and texture vertices etc. Also it is a simple text-based format that is supported by many 3D packages. In order to generate 3D model and its texture we used Blender 3D software [9]. Exporting 3D Wavefront OBJ Model, Blender creates two files with .obj (Object) and .mtl (Material Template Library) extensions. The .obj extension file contains geometry information of 3D model. The .mtl extension file contains the visual aspects of the polygons and texture image file path.

Texturing is one of the most important steps for pixel shading on 3D object model. Texturing an object is simply means gluing an image onto that object. This process is depicted in Figure 5.3.

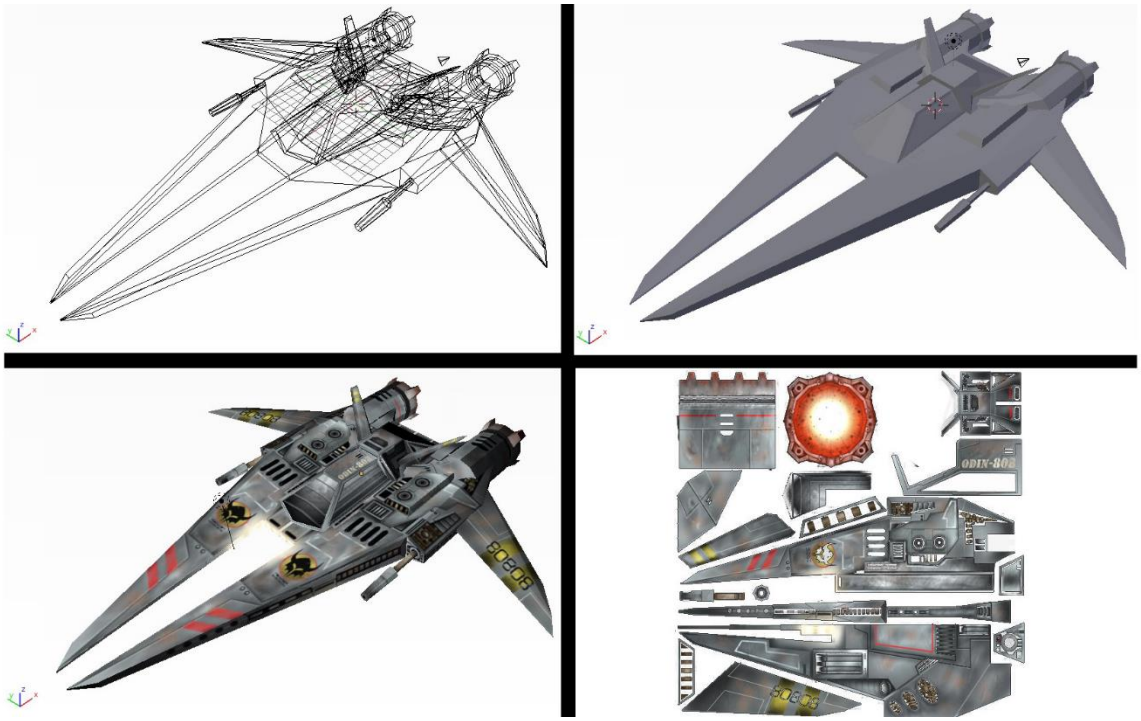

Figure 5.3 A spaceship Wavefront OBJ 3D model in wireframe form shown in the upper left, solid form without texture shown in the upper right. The pieces in the image texture (the lower right) are “glued” onto the spaceship, and the result is shown in the lower left

6. VUFORIA SDK

6.1. Introduction

Vuforia is an augmented reality SDK (Software Development Kit) which is supported by Android, iOS and Unity 3d that allows actualize real time AR applications on smartphone devices. This software development kit uses computer vision technology to recognize target image and get pose of objects by camera in real time. In this project the used Vuforia SDK 2.8.8 version for android software environment [75].

6.2. Architecture of Vuforia

Vuforia is capture and pass the frame of camera efficiently to the tracker. The frame of camera is automatically arranges image format and size dependent devices. Then converts pixel format from the camera format (e.g., YUV12) to suitable format for OpenGL ES 3D model rendering (e.g., RGB565) and for tracking internally. Finally Vuforia detects and tracks real-world objects in camera video frames by computer vision algorithms.

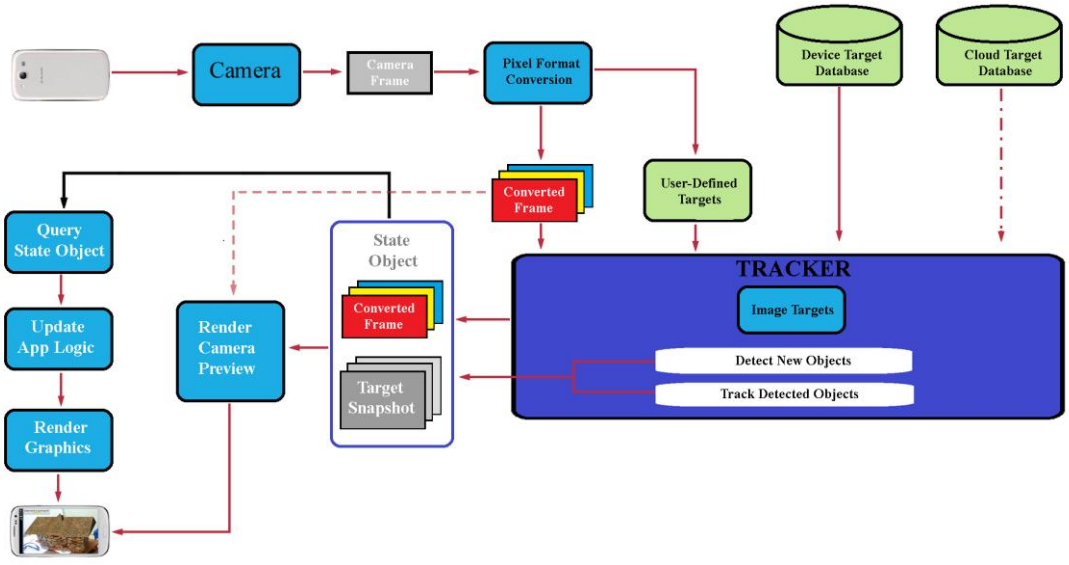

Figure 6.1 Data flow diagram of the application with the Vuforia SDK

6.3. Target management system

Vuforia has two type of target database. One of these device databases which is uploading image on target manager website, then downloading the target database to recognize. In this way the device doesn't need internet connection. The second type of target database is cloud database which is creating a cluster of image and suitable for uploading several images. In this way device needs to internet connection and it is creating an Id and password to be able to use in application. In this thesis we used both of them.

7. APPLICATION DEVELOPMENT

The aim of thesis is creating markerless augmented reality application on Android platform. In order to achieve, we need Eclipse IDE (Integrated Development Environment) and its plugins; Android SDK, ADT (Android Developer Tools), Eclipse CDT (C/C++ Development Tools) and Android NDK (Native Development Kit).

Eclipse IDE is a project which aiming to provide a universal tool set for software development. Open Source IDE, mostly provided in Java, but the development language is independent [76]. Our application implemented and tested on this environment.

Blender is a 3D computer graphics software is used for creating a Wavefront OBJ 3D model for augmentation. For this application, we created a 3D roof model as shown in Figure 7.1.

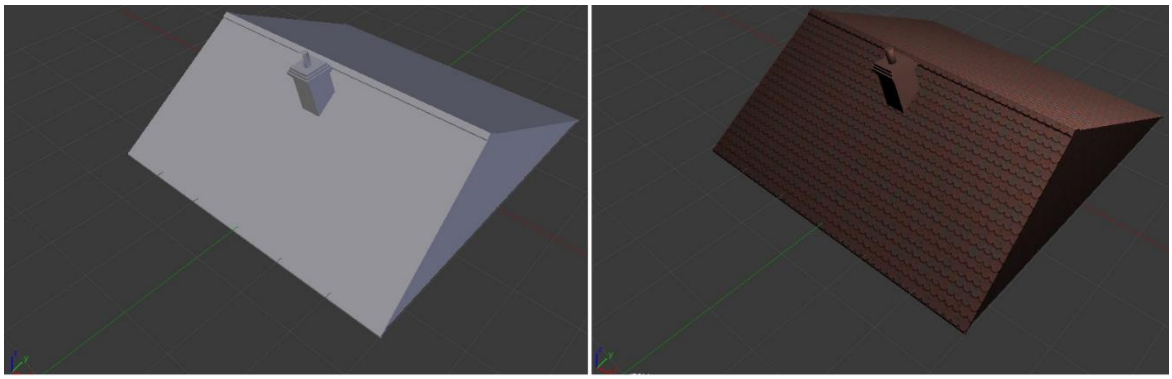

Figure 7.1 3D Wavefront OBJ Roof Model with solid and textured mode

7.1. Software Implementation

In this thesis OpenCV 2.4 version is used on Microsoft Visual Studio 10.0 to implementation of camera calibration, pose tracking and markerless augmentation of a cube. Project properties must be included Additional Include Directories and Libraries for usage of OpenCV on Visual Studio environment.

OpenGL is used in Markerless augmented reality desktop application for drawing cube on the target image. This also must be included additional directories on Visual Studio similarly with OpenCV,

OpenGL is also used in Mobile Markerless application for rendering of 3D Wavefront OBJ model.

Vuforia was introduced in chapter 6. In our application we used Vuforia 2.8.8 version. This SDK only used for target recognition, other steps are covered by developer section.

8. TEST AND RESULTS

The experimental works has been done on Visual Studio by using OpenCV library. Then the final version of Markerless Augmented Reality Application which is successfully working on Android Mobile devices has been implemented. Some screen shoots of the application are shown in Figure 8.1 and Figure 8.2. In figure 8.1 handmade 3D target that has covered with 2D target images as in figure 8.3 is shown.

Figure 8.1 Handmade 3D target model for augmentation

As shown in figure 8.2 the screen shoot of Samsung Galaxy SIII mobile devices is showing that Augmentation of a 3D roof model on proper position.

Figure 8.2 Augmentation of a handmade 3D home model

After augmentation of 3D roof model, even if user looks around to 3D target home model, the 3D roof model will track view of camera. So the aim of thesis which was 3D reconstruction of an architectural building was achieved.

Figure 8.3 Unwrapped of 3D handmade home model (right) with its 2D target images

9. CONCLUSION

9.1. Summary and Conclusion

As a conclusion we analyzed how the totally working Markerless Augmented Reality Application that works on Android Mobile devices. We discussed how each requirement was met; gave guidelines on creating handheld AR applications and finished with an outlook to future work.

Application of AR to cultural heritage is an enchanting research topic. It allows maintaining the original building structures, already subject to wear and tear, and provides a way of learning their history by seeing the original building structures instead of ruins.

To achieve AR of cultural heritage necessary experimental studies are completed and results are obtained. In this thesis firstly the basic parts of Markerless AR was developed. It was aimed to develop a mobile based solution which is capable real time working. In order to ensure real time performance, C++ native language and Vuforia SDK were used for algorithm development. Similar applications in the literature were analyzed.

According to the initial goal of the thesis, 3D target recognition and its augmentation was developed on mobile platform. The developed application can work approximately at 30 fps. The application was tested with several targets and appropriate results were obtained.

9.2. Future Works

The developed application in this thesis is using the Vuforia SDK target recognition system. Therefore for future work, this part of algorithm desired to made by client and achieve robust recognition system as good as Vuforia's performance. After all, next targeted work is using architectural 3D model data for augmentation of historical places in Turkey for touristic guidance purpose.

Another future work is developing of the same Markerless AR application on iOS platform. Moreover not only target based but also GPS based application can be developed.

REFERENCES

[1] B. Furht, "Augmented Reality: An Overview," in *Handbook of Augmented*, Florida Atlantic University, Florida, USA, Springer Science+Business Media, LLC, 2011, p. 3.

[2] R. T. Azuma, "A Survey of Augmented Reality," *Computer Graphics (SIGGRAPH '95 Proceedings, Course Notes #9: Developing Advanced Virtual Reality Applications),* pp. 1-38, 1995.

[3] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual displays," *EICE Transactions on Information Systems,* Vols. E77-D, no. 12, December 1994.

[4] D. Wagner and D. Schmalstieg, "First Steps Towards Handheld Augmented Reality," in *Proceedings of the 7th International Conference on Wearable Computers,* IEEE Computer Society Press,2003.

[5] F. P. Brooks, "The Computer Scientist as Toolsmith II," *Communications of the ACM,* vol. 39, no. 3, pp. 61-68, March 1996.

[6] ARmedia, "Augmenting Real World Objects with ARmedia 3D Tracker," http://arblog.inglobetechnologies.com/?p=1188, Last Access: 01/ 07/ 2014.

[7] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.

[8] OpenGL, "OpenGL," http://www.opengl.org/about/#12, Last Access: 20 June 2014.

[9] Blender 3D, "http://wiki.blender.org/index.php/Doc:2.6/Manual," http://wiki.blender.org/index.php/Doc:2.6/Manual, Last access: 15 July 2014.

[10] M. Bajura, H. Fuchs and R. Ohbuchi, "Merging Virtual Reality with the Real World: Seeing Ultrasound Imagery Within the Patient," *ACM SIGGRAPH Computer Graphics,* vol. 26, no. 2, pp. 203-210, 1992, USA.

[11] S. Feiner, B. MacIntyre and D. Seligmann, "Knowledge-based augmented reality," *Communications of the ACM,* vol. 36, no. 7, pp. 52-62, 1993.

[12] T. Höllerer, S. Feiner, T. Terauchi and D. Gus Rashid, "Exploring MARS: Developing Indoor and Outdoor User Interfaces to a Mobile Augmented Reality System," *Computers and Graphics,* vol. 23, no. 6, pp. 779-785, Dec. 1999.

[13] M. Kalkusch, T. Lidy, M. Knapp, G. Reitmayr, H. Kaufmann and D. Schmalst, "Structured Visual Markers for Indoor Pathfinding," in *Proceedings of the First IEEE International Workshop on ARToolKit (ART02),* 2002.

[14] W. Piekarski and B. . H. Thomas, "Tinmith evo5 - An Architecture for Supporting Mobile," *2nd International Symposium on Augmented Reality (ISAR),* pp. 177-178, 2001, USA.

[15] D. G. Brown, S. Julier, Y. Baillot, M. A. Livingston and L. J. Rosenblum, "Event-Based Data Distribution for Mobile Augmented Reality and Virtual Environments," *Presence - Teleoperators and Virtual Environments,* vol. 13, no. 2, pp. 211-221, 2004.

[16] J. Newman, G. Schall, I. Barakonyi, A. Schürzinger and D. Schmalstieg, "Wide-Area Tracking Tools for Augmented Reality," in *In Proceedings of the 4th International Conference on Pervasive Computing*, UK, 2006.

[17] B. Reitinger, C. Zach, . K. Karner and D. Schmalstieg, "Automated Model Acquisition using 3D Reconstruction for Urban Planning," in *Demo at the ISMAR 2006 symposium*, USA, 2006.

[18] I. Lindt, J. Ohlenburg, U. Pankoke-Babatz and S. Ghellal, "A report on the crossmedia game epidemic menace," *Computers in Entertainment (CIE)*, vol. 5, no. 1, Section on Pervasive gaming, ACM Press,2007.

[19] D. Wagner, "Handheld Augmented Reality," in *Dissertation*, Graz, Austria, October 1st, 2007, pp. 15-18.

[20] B. MacIntyre and S. Feiner, "A Distributed 3D Graphics Library," *Annual Conference Series*, pp. 361-370, 1998.

[21] M. R. Macedonia and M. J. Zyda, "A Taxonomy for Networked Virtual Environments," *Networked Virtual Environments, In Proceedings of the Virtual Reality Annual International Symposium,VRAIS '95,* pp. 230-231, 1995.

[22] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner and J. T. Klosowsk, "Chromium: a stream-processing framework for interactive rendering on clusters," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 693-702, 2002.

[23] D. Amselem, "A window on shared virtual environments," *Presence: Teleoperators and Virtual Environments*, vol. 4, no. 2, pp. 130-145, 1995.

[24] G. W. Fitzmaurice, "Situated Information Spaces and Spatially Aware Palmtop," *Communications of the ACM,*, vol. 36, no. 7, pp. 38-49, 1993.

[25] J. Rekimoto, "The World through the Computer: Computer Augmented Interaction with Real World Environments," *User Interface Software and Technology (UIST '95)*, pp. 29-38, 1995.

[26] H. Regenbrecht and R. .. Specht, "A Mobile Passive Augmented Reality Device - mPARD," *Proceedings of ISAR*, pp. 81-84, 2000, Germany.

[27] J. Rekimoto, "A Hand-held Augmented Reality System for Collaborative," *Proceedings of Virtual Systems and Multi-Media (VSMM '96),*, pp. 18-20, Gifu, Japan, 1996.

[28] D. Mogilev, K. Kiyokawa, M. Billinghurst and J. Pair, "AR Pad: An Interface for Face-to-Face AR Collaboration," *Conference on Human Factors in Computing Systems (CHI'02) Extended abstracts on Human factors in computer systems*, pp. 654-655, 2002, USA.

[29] J. Newman, D. Ingram and A. Hopper, "Augmented Reality in a Wide Area Sentient Environment," *Proceedings of the 2nd IEEE and ACM International Symposium on Augmented Reality (ISAR 2001)*, pp. 77-86, 2001, USA.

[30] J. Gausemeier, J. Fruend, C. Matysczok, B. Bruederlin and D. Beier, "Development of a real time image based object recognition method for mobile AR-devices," *Proceedings of the 2nd International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa (Afrigraph 2003)*, pp. 133-139, 2003, Africa.

[31] F. Shibata, "Mobile Computing Laboratory," Department of Computer Science, Ritsumeikan University, http://www.mclab.ics.ritsumei.ac.jp/research.html, Japan.

[32] A. Makri, . D. Arsenijevic, J. Weidenhausen, P. Eschler, D. Stricker, O. Machui, C. Fernandes, S. Maria, G. Voss and . N. Loannidis, "ULTRA: An Augmented Reality System for Handheld Platforms," Targeting Industrial Maintenance Applications, Proceedings of 11th International Conference on Virtual Systems and Multimedia (VSMM'05), Belgium, 2005.

[33] H. Kato and M. Billinghurst, "Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System," in *Proceedings of the 2nd International*, USA, 1999, pp. 85-94.

[34] D. Wagner, T. Pintaric, F. Ledermann and D. Schmalstieg, "Towards Massively Multi-User Augmented Reality on Handheld Devices," *Proceedings of the 3rd International Conference on Pervasive Computing (PERVASIVE 2005),* pp. 208-219, 2005, Germany.

[35] M. Möhring, C. Lessig and O. Bimber, "Video See-Through AR on Consumer Cell-Phones," *Proceedings of International Symposium on Augmented and Mixed Reality (ISMAR'04),* pp. 252-253, 2004, USA.

[36] A. Henrysson , M. Billinghurst and M. Ollila, "Face to Face Collaborative AR on Mobile Phones," *Proceedings International Symposium on Augmented and Mixed Reality (ISMAR'05),* pp. 80-89, 2005, Austria.

[37] D. A. Bowman, E. Kruijff, J. J. LaViola and I. Poupyrev, "What Are 3D User Interfaces?," in *3D User Interfaces: Theory and Practice*, USA, Pearson Education, Inc., 2005, pp. 3-4.

[38] K. Meyer, H. L. Applewhite and F. A. Biocca, "A Survey of Position Trackers," *Teleoperators and Virtual Environments,* vol. 1, no. 2, pp. 173-200, Spring 1992 .

[39] G. Welch and E. Foxlin, "Motion Tracking: No Silver Bullet, but a Respectable Arsenal," *IEEE Computer Graphics and Applications,* vol. 22, no. 6, pp. 24-38, 2002.

[40] P. Mistry and P. Maes, "SixthSense: a wearable gestural interface," in *International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH ASIA*, Yokohama, Japan, December , 2009.

[41] M. Felsberg and J. Hedborg, "Real-time view-based pose recognition and interpolation for tracking initialization," *Journal of Real-Time Image Processing*, vol. 2, no. 2-3, pp. 103-115, November 2007.

[42] OpenCV: Open Source Computer Vision Library Reference Manual, Intel, 2000.

[43] S. Siltanen, "Marker-Based Tracking," in *Theory and applications of marker-based augmented reality*, Finland, Copyright © VTT , 2012, p. 39.

[44] ARToolKit , "ARToolkit," http://www.hitl.washington.edu/artoolkit, Last Access: 15 May 2014.

[45] ALVAR, "A Library for Virtual and Augmented Reality," www.vtt.fi/multimedia/ alvar.html, 01 May 2013.

[46] ARTag, "Augmeneted Reality system," http://www.artag.net , 01 May 2013.

[47] D. J. Felleman and D. C. Van Essen, "Distributed hierarchical processing in the primate cerebral cortex," *Cereb Cortex*, pp. 1-47, 1991.

[48] Y. Ma, S. Soatto, J. Kosecka and S. S. Sastr, in *An Invitation to 3-D Vision: From Images to Geometric Models*, Springer, 2004.

[49] R. Hartley and A. Zisserman, in *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2003.

[50] D. A. Forsyth and J. Ponce, in *Computer Vision: A Modern Approach*, Pearson Education, Limited, 2011.

[51] J.-Y. Bouguet, "Camera Calibration Toolbox for Matlab," http://www.vision.caltech.edu/bouguetj/calib_doc/index.html, 2008., Last Access: 24 June 2014.

[52] "Feature detection (computer vision)," http://en.wikipedia.org/wiki/Feature_detection_(computer_vision), Last access: 24 June 2014.

[53] M. Zuliani, C. Kenney and B. S. Manjunath, "A Mathematical Comparison of Point Detectors," in *IEEE Computer Vision and Pattern Recognition*, Washington, DC, 2004.

[54] M. O. SHNEIER, "Extracting Linear Features from Images Using Pyramids," *IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS*, vol. 12, no. 4, pp. 569-572, July/August, 1982.

[55] A. K. Jain and G. R. Cross, "Markov random field texture models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI, no. 1, pp. 25-39, 1983.

[56] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *Computer Vision and Pattern Recognition,Proceedings,IEEE Computer Society Conference on*, pp. 731-737, 1997.

[57] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Computer Vision and Pattern Recognition, CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886-893, 2005.

[58] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177-280, 2008.

[59] C. Schmid, R. Mohr and C. Bauckhage, "Evaluation of interest point detectors," *International Journal of Computer Vision*, vol. 37, no. 2, pp. 151-172, 2000.

[60] O. M. Mozos, A. Gil, M. Ballesta and O. Reinoso, "Interest Point Detectors for Visual SLAM," *Current Topics in Arti cial Intelligence*, vol. 4788, pp. 170-179, 2007.

[61] S. M. Smith and J. Brady, "Susan: A new approach to low level image processing," *International Journal of Computer Vision*, vol. 23, no. 1, pp. 45-78, 1997.

[62] G. E. Bostancı, "User Tracking Methods for Augmented Reality Applications in Cultural Heritage," in *School of Computer Science and Electronic Engineering University of Essex*, Colchester, England, November, 2013.

[63] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[64] H. Bay, T. Tuytelaars and L. V. Gool, "SURF: Speeded Up Robust Features," *EECV, LNCS:3951,* pp. 404-417, 2006.

[65] K. MIKOLAJCZYK and C. SCHMID, "Scale & Affine Invariant Interest Point Detectors," *International Journal of Computer Vision,* vol. 60, no. 1, pp. 63-86, 2004.

[66] H. Ney, in *Features for Image Retrieval*, Aachen, Germany, 2003, p. 17.

[67] D. Kurz and S. B. Himane, "Inertial sensor-aligned visual feature descriptors," *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 161-166, 2011.

[68] D. L. Baggio, S. Emami, D. M. Escriva, K. Ievgen, N. Mahmood, J. Saragih and R. Shilkrot, "Markerless Augmented Reality," in *Mastering OpenCV with Practical Computer Vision Projects*, Packt Publishing , December 3, 2012, pp. 95-120.

[69] K. MIKOLAJCZYK and C. SCHMID, "Indexing based on scale invariant interest points," *ICCV,* vol. 1, pp. 525-531, 2001.

[70] T. Lindeberg, "Feature detection with automatic scale selection," *IJCV,* vol. 30, no. 2, pp. 79-116, 1998.

[71] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," *IEEE Conference on Computer Vision and Pattern Recognition,* vol. 1, p. 511, 2001.

[72] C. Evans, "Notes on the OpenSURF Library," *University of Bristol Tech. Rep. CSTR,* vol. 9, no. 1, p. 7, January 18,2009.

[73] Z. Zhang, "A Flexible New Technique for Camera," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 22, no. 11, pp. 1330-1334, 2000.

[74] T. Akenine-Moller, E. Haines and N. Hoffman, Real-Time Rendering, Wellesley, Massachusetts, USA: A K Peters/CRC Press, July 25, 2008.

[75] A. S. Ibañez and J. P. Figueras, "Vuforia v1.5 SDK: Analysis and evaluation of capabilities," Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels, Barcelona, İspanya, 19 March 2013.

[76] Eclipse IDE, "http://www.eclipse.org/org/," Eclipse Foundation, Inc., Ontario, Canada, Last access: 15 July 2014.

[77] O. Bimber and R. Raskar, "Modern Approaches to Augmented Reality," in *ACM SIGGRAPH*, New York, USA, 2006.

[78] I. Sutherland, "The Ultimate Display," in *Proceedings of International Federation of Information Processing*, Spartan Books, 1965, pp. 506-508.

[79] I. Sutherland, "A Head-Mounted Three Dimensional Display," in *Proceedings of Fall Joint Computer Conference*, USA, 1968, pp. 758-765.

[80] S. Feiner, B. MacIntyre and T. Höllerer, "First steps toward mobile augmented reality systems," *In Proceedings of ISMR,* pp. 363-377, 1999, Japan.

[81] S. Feiner, B. MacIntyre and T. Höllerer, "A touring machine: Prototyping 3d

mobile augmented reality systems for exploring the urban environment," *Proceedings of the First International Symposium on Wearable Computers (ISWC)*, pp. 74-81, 1997,USA.

[82] J. Rekimoto, "Multiple-Computer User Interfaces: A Cooperative Environment Consisting of Multiple Digital Devices," *Cooperative Buildings: Integrating Information, Organization, and Architecture Lecture Notes in Computer Science* , vol. 1370, pp. 33-40, 1998.

[83] S. Ashley, "Annotating the Real World: Augmented Reality Makes Commercial Headway," *Scientific American*, vol. 299, no. 4, pp. 27-28, 2008.

[84] R. Szeliski, in *Computer Vision: Algorithms and Applications*, September 3, 2010.

[85] J. Weng, P. Cohen and M. Herniou, "Camera calibration with distortion models and accuracy evaluation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, no. 10, pp. 965-980, 1992.

[86] I. Guyon and . A. Elisseeff, "An Introduction to Feature Extraction," *Spring Berlin Heidelberg*, pp. 1-25, 2006.

[87] M. Zhao and S. Qin, "Socket connector recognition based on SVM with speeded up robust feature (SURF)," *International Conference on Electronic Measurement & Instruments*, vol. 9, no. 4, pp. 827-831, 2009.