
Electronic Thesis and Dissertation Repository

12-12-2013 12:00 AM

A Modular Approach to the Development of Interactive Augmented Reality Applications.

Nelson J. Andre
The University of Western Ontario

Supervisor
Dr. Michael James Katchabaw
The University of Western Ontario

Graduate Program in Computer Science
A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science
© Nelson J. Andre 2013

Follow this and additional works at: <https://ir.lib.uwo.ca/etd>



Part of the [Graphics and Human Computer Interfaces Commons](#), and the [Software Engineering Commons](#)

Recommended Citation

Andre, Nelson J., "A Modular Approach to the Development of Interactive Augmented Reality Applications." (2013). *Electronic Thesis and Dissertation Repository*. 1800.
<https://ir.lib.uwo.ca/etd/1800>

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlsadmin@uwo.ca.

A MODULAR APPROACH TO THE DEVELOPMENT OF INTERACTIVE
AUGMENTED REALITY APPLICATIONS.

(Thesis format: Monograph)

by

Nelson Andre

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Masters of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

© Nelson Andre 2013

Abstract

Augmented reality (AR) technologies are becoming increasingly popular as a result of the increase in the power of mobile computing devices. Emerging AR applications have the potential to have an enormous impact on industries such as education, healthcare, research, training and entertainment. There are currently a number of augmented reality toolkits and libraries available for the development of these applications; however, there is currently no standard tool for development. In this thesis we propose a modular approach to the organization and development of AR systems in order to enable the creation novel AR experiences. We also investigate the incorporation of the framework that resulted from our approach into game engines to enable the creation and visualization of immersive virtual reality experiences. We address issues in the development process of AR systems and provide a solution for reducing the time, cost and barrier of entry for development while simultaneously providing a framework in which researchers can test and apply advanced augmented reality technologies.

Keywords: Augmented Reality, Mixed Reality, Virtual Reality, Computer Vision, Image Processing, Game Engines

Contents

Abstract	ii
List of Figures	v
1 Introduction	2
1.1 Problem Statement	3
1.2 Research Questions	3
1.3 Contributions	4
1.4 Roadmap	4
2 Related Work	5
2.1 Visual Tracking Methods	5
2.2 Simultaneous Localization and Mapping	6
2.3 Natural User Interfaces	7
2.4 Existing Toolkits and Frameworks	8
2.4.1 Augmented Reality Authoring Tools	9
2.5 Displays	10
2.5.1 Handheld Displays	10
2.5.2 Head Mounted Displays	10
2.5.3 Surround Screen Projection Based Displays	10
2.6 Photorealistic Rendering	11
2.6.1 Lighting and Shadows	11
2.6.2 Accounting for Captured Image Artefacts	12
2.7 Diminished Reality	13
2.8 Summary	13
3 Approach and Solution	14
3.1 Navigational Augmented-Virtual Reality Framework	14
3.2 Providing an Interface Between Layers	15
3.2.1 Trackable Detection Interface	16
3.2.2 Feature Point Mapping Interface	17
3.2.3 Image Processing Interface	17
3.3 Video Capture Input	18
3.4 Rendering Layer Manager	19
3.5 World State Manager	20
3.5.1 Feature Point Cloud Map	21

3.5.2	Egocentric Pose Determination	21
3.5.3	Trackable Groups	22
3.6	System Architecture	23
3.7	Graphics Engine Integration	24
3.8	Summary	26
4	Evaluation and Validation	28
4.1	Description of Previous Framework	28
4.1.1	Physical Setup	29
4.1.2	Software Architecture	29
4.2	Development with NAVR	30
4.2.1	Defining Rendering Layers	31
4.2.2	Creating Virtual Environments	32
4.2.3	Interaction Between Trackable Groups	33
4.2.4	Image Effects for Photorealistic Rendering	34
4.3	Summary	34
5	Discussion	36
5.1	Disadvantages of Our Approach	38
6	Conclusion	39
6.1	Contributions	39
6.2	Future Work	39
6.3	Final Comments	40
	Bibliography	40
	Curriculum Vitae	45

List of Figures

1.1	Reality-virtuality continuum [18]	3
2.1	Microsoft Kinect TM (a) is a natural user interface device used for tracking skeletal pose in real-time (b).	7
2.2	Leap Motion TM (a) is a natural user interface device that tracks hand and finger positions (b).	8
2.3	Augmented reality authoring tools can be used to extend existing content creation software.	9
2.4	Apple iPad TM device augmenting a fiducial marker with a virtual object.	10
2.5	Head mounted displays are commonly used in immersive augmented reality systems.	11
2.6	Adding shadows to virtual objects and an example of lighting acquisition procedure using a diffuse sphere (circled in green).	12
2.7	Virtual hamburger model generated in AR with simulated noise [9].	13
2.8	Diminished reality example, images from left to right: original, augmentation over existing object, diminished image, augmentation over diminished image. [25]	13
3.1	NAVR Framework Layer Diagram	16
3.2	Camera input is captured and processed and an interface is provided to modules within the NAVR framework.	18
3.3	Rendering Manager entity relationship diagram.	19
3.4	An example of how rendering layers are used to determine the systems classification on the Reality-Virtuality Continuum.	20
3.5	World State Manager entity relationship diagram.	20
3.6	Egocentric pose estimation using module providing an interface to the Parallel Tracking and Mapping implementation by Klein et al. [14]	22
3.7	Related trackables objects are classified into trackable groups that relate related physical bodies. In this example there are two trackable groups, a cereal box and a table.	23
3.8	Representing trackables as game objects within Unity3D and assigning virtual objects. Single marker target (Left), multi marker target (Middle), planar image target (Right).	25
3.9	Interface to NAVR within Unity3D for creating for creating single trackable marker in Unity3D.	25
3.10	By extending the NAVR framework, we provide an interface for defining, training and visualizing planar image targets within Unity3D editor.	26

3.11	Visualizing trackables within a graphical engine allows for organization of augmentable objects in the real world in relation to virtual objects.	27
4.1	The VUZIX iWear 920VR head mounted display.	29
4.2	Virtual scenes rendered in previous framework developed by Garcia [11]. . . .	29
4.3	Physical space in which fiducial markers were setup in order to create a predefined multi-marker system.	30
4.4	Comparison of layer diagrams representing software architecture of previous system vs NAVR framework.	31
4.5	Rendering layers are created by the image processing modules within NAVR and by the rendering of virtual objects in the application.	32
4.6	Street environment created using the NAVR framework in Unity3D (a) and then a rendering layer defined by a skin classification module was overlaid (b) to create an immersive experience.	33
4.7	An interaction event is triggered when trackable groups come within a threshold distance.	34
4.8	Comparison of augmented objects rendered with and without image effects. Applied effects include shadows, noise and grain, and a focusing blur.	35

Chapter 1

Introduction

The rise in the availability of powerful camera equipped mobile devices is paving the way for visual Augmented Reality (AR) technologies. The field of AR revolves around overlaying virtual objects or information over ones field of view in real time [3]. Milgram and Kishino [18] have proposed that all realities lie at a point along the Virtuality Continuum (VC), ranging from completely real to completely virtual environments. A subclass of virtual reality, known as Mixed reality (MR), refers to the merging of real and virtual worlds [18], in which both real and virtual objects can coexist and interact in realtime.

Another concept explored by Milgram and Kishino [18] is the Extent of the World Knowledge (EWK) scale, which can range from completely unmodelled to completely modelled environments. It is not necessary to have a completely accurate description of the real world, but instead a simple approximation that will allow for augmentation of objects of interest to a specific application [10]. Once a language is developed for defining knowledge of objects within the real world, augmentation can occur and a virtual environment can be created. The knowledge of one's physical surroundings can be used in different ways to create environments and hence different user experiences in the same real world environments can exist. By having an approximate representation of the structure of a real world environment, this can enable the development of complex augmented reality simulations and allow for relationships between features within the real world and the virtual to be clearly defined.

The main focus of research in augmented reality technology is the techniques used for the tracking and localization of trackable objects within a scene. Information about a scene must first be obtained to allow for mixed reality environments to be created and thus reliable tracking is important. Many augmented reality systems use mechanisms for combining object recognition, fiduciary marker tracking, and other localization methods to create hybrid AR systems. In this thesis we discuss the evolution of tracking techniques in Section 2.1 and how they can be used to create AR experiences.

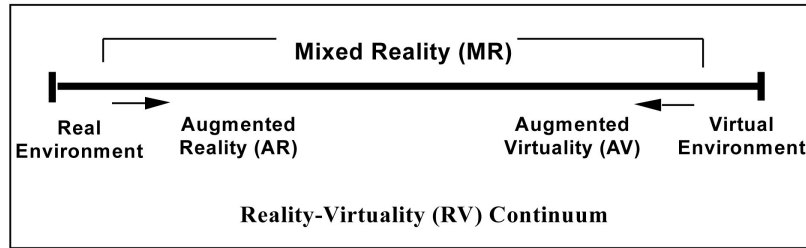


Figure 1.1: Reality-virtuality continuum [18]

In fully immersive virtual reality environments, applications often make use of a head-mounted display (HMD) that allows users to experience a more natural field of view. The display hardware used in the setup for an augmented reality system can greatly impact the degree of immersion experienced. Wearable technologies like Google Glass TM and Oculus Rift TM serve as a more natural way of experiencing a virtual world and interacting with information. However, providing a virtual interface on mobile devices has the benefit of being a convenient and portable means of display. We will further discuss the different types of available displays used in AR applications in Section 2.4.

1.1 Problem Statement

Currently, there is no standard tool for the development of augmented reality applications. There are numerous toolkits and libraries available for tracking objects in AR environments, however a comprehensive system for using these tools in the creation of AR environments does not exist. A flexible framework for creating portable cross-platform AR applications is needed to enable the development of dynamic experiences. Augmented reality technology currently has many applications in entertainment, education, healthcare and training [12] and such tools would be helpful in allowing developers to create reliable believable systems with minimal knowledge of computer vision.

1.2 Research Questions

In this thesis we will address the following research questions:

1. How can we improve the integration of AR software technologies to facilitate and ease the development of AR applications?

2. Can we address the first question while at the same time making it easier to deploy and use AR applications?
3. How do we accomplish the above while maintaining or improving the quality of the AR experience being created?

By answering these questions we hope to allow developers and researchers to create and use virtual reality applications at a lower develop cost, time and barrier of entry.

1.3 Contributions

In this thesis, we have developed a high level approach for the creation of mixed reality environments. We have provided a modular interface for linking computer vision and augmented reality libraries with 3D graphical and physics engines in order to facilitate the creation of interactive, realistic mixed reality environments. This interface is to be extendable, highly configurable and cross platform allowing for augmented reality setups and feature databases to be shared across virtual reality applications. We have developed an approach that allows developers to incorporate existing commercial toolkits as well as open source solutions in order to create augmented reality experiences. As well, we provide an interface that allows for the integration of this framework into game engines enabling the creation and organization of immersive virtual reality experiences.

1.4 Roadmap

In this thesis we will provide a review of the current state of AR technologies and tracking methods (Section 2.1), and how they can be used to create interactive AR applications. We will then provide an overview of existing development tools and libraries and explain how they can be used for the creation of augmented reality applications (Section 2.1). We then provide a brief overview of displays commonly used in augmented reality applications (Section 2.4). Next we explain how rendering techniques can be used to convincingly blend real and virtual environments, creating immersive and believable experiences (Section 2.5). Subsequently, we describe our approach for solving the outlined problems in the development of augmented reality systems and provide details of implementation (Section 3.1). We then provide an evaluation of our framework and compare it to previous developments of augmented reality applications (Section 4). We will then outline the development process with this framework and discuss the benefits and drawbacks of using our approach and the future direction of development.

Chapter 2

Related Work

2.1 Visual Tracking Methods

One of the main topics of study in research in augmented reality is tracking [32] because it provides some knowledge of the orientations of objects within a physical environment and allows for augmentation of these trackables. Tracking techniques can generally be grouped into visionbased, sensor-based and hybrid techniques. The majority of this thesis will focus on techniques for vision based tracking, which can be used for determining camera pose relative to real world objects. In computer vision, most available techniques are divided into two classes, feature based and model-based [22]. Feature-based methods focus on the tracking of 2D features such as geometrical primitives, object contours, and regions of interest. Model based approaches explicitly use a 3D CAD model of tracked objects and can provide a robust solution. Relative camera pose can then be calculated by projecting the 3D coordinates of the features into the observed 2D image coordinates and minimizing the distance to their corresponding 2D features [30].

Computer vision techniques for efficiently calculating camera pose in real time from artificial markers is a common approach to tracking virtual objects. The ARToolKit library was one of the first development kits to describe this approach[13], while other research at the time focused on line finding [27] and minimizing marker error. Square markers were the dominant technique studied in the early 2000s and several techniques for marker tracking have been published and source code has been made freely available.

In feature-based tracking, camera pose is determined from naturally occurring scene features such as points, lines, edges or textures [31]. After camera pose had been calculated from known visual features, additional natural features can be acquired then subsequently used to update the pose calculation. This allows for robust tracking even when original fiducial markers are no longer in view. Other techniques use an explicit model of tracked objects such as

a CAD model or 2D template of the object based on distinguishable features. Modelbased trackers generally create their models based on lines or edges in the model. Edges are used most frequently because they are the most efficient to find and the most robust to changes in lighting. Vachettit et al. [29] combined edge information and feature points to let the tracker handle both textured and untextured objects which resulted in a system with increased tracking stability. Similarly, Presigout and Marchand proposed a model-based hybrid monocular vision system [23], combining edge extraction and texture analysis to obtain a more robust and accurate pose computation.

In many augmented reality applications, hybrid-tracking techniques are used in the development of a robust tracking solution. Azuma et al [4] have proposed the development of outdoor AR systems, requiring the use of a tracking system based on GPS as well as inertial and computer vision sensing mechanisms. Since then there has been a consensus among the augmented reality community to combine both inertial and computer vision technologies. Vision based tracking approaches have low jitter and no drift, but they are slow and outliers may occur. As well, sudden changes in motion and direction lead to tracking failure with a large recovery time. Inertial based tracking methods are fast and robust and can be used for motion prediction when rapid changes occur. This is especially useful in mobile devices as object pose position can be recovered based on accelerometer and gyroscope data [15]. Support for a hybrid based pose tracking system and concepts of sensor fusion are explored in our approach in Section 3.5.2.

2.2 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is an important problem that is widely studied in the robotics community. This problem is based on the question of whether it is possible for a mobile robot to be placed in an undefined location and dynamically build a representation of this environment while concurrently estimating its motion. Research in vision based SLAM often concentrates on reconstruction problems from small image sets, and makes up a field known as Structure from Motion (SFM). These algorithms generally work on longer image sequences and can produce a reconstruction of the camera trajectory and scene structure. To obtain globally consistent estimates over a sequence, local motion estimates from frame-to-frame matching are refined by a technique known as bundle adjustment. These methods are suited for automatic analysis of short image sequences obtained from arbitrary sources but do not scale to consistent localization over long sequences in time.

For mobile devices with a single camera a monocular SLAM approach must be used. An implementation of an Extended-Kalman Filter (EKF) based monocular SLAM system was first

presented by Davison [7]. The quadratic computational complexities of EKF-SLAM make it beneficial for smaller maps of reliable features to be used instead of mapping an entire scene. Klein and Murray have presented a method for estimating camera pose in an unknown scene that is designed to be suitable specifically for tracking a hand held camera in small workspaces [14]. In their approach, which they refer to as PTAM, they split tracking and mapping into two separate tasks processed in parallel. One task deals with the robust tracking of erratic handheld motion, while the other provides a three dimensional map of features from observed frames. This allows for the use of processor intensive bundle adjustment operations that are not suitable for use in real time calculations and must be performed on a lower priority thread. This results in a system that can produce detailed maps with thousands of landmarks that can be tracked in real time.

2.3 Natural User Interfaces

Interaction of users with virtual environments can be enhanced through the use of natural user interface (NUI) devices such as the Microsoft KinectTM, Leap MotionTM and other hardware devices (Figure 2.1). These devices provide information on the pose of a users body parts, which can then be calculated in relation to the device. If the devices pose relative to the user is known then it can be used in virtual environments to allow for real time interaction with virtual objects. Hybrid user interfaces that make use of multiple forms NUI devices can be combined in a manner such that the techniques will complement each other. By doing so, you can provide a more robust mechanism of interaction with the virtual world.

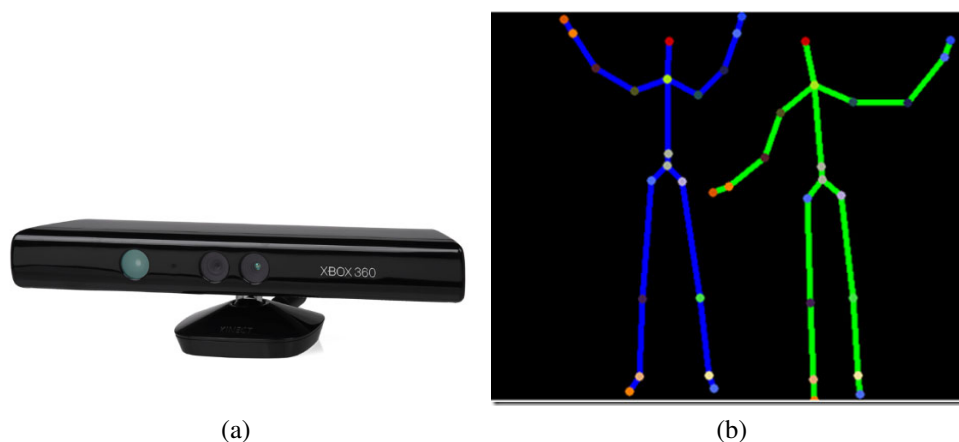


Figure 2.1: Microsoft KinectTM(a) is a natural user interface device used for tracking skeletal pose in real-time (b).



Figure 2.2: Leap Motion TM(a) is a natural user interface device that tracks hand and finger positions (b).

2.4 Existing Toolkits and Frameworks

Several tools for creating augmented reality applications are currently available for use. These tools contain libraries, toolkits and SDKs that are used for AR application development. They contain methods for tracking objects and may often provide their own graphical interface. There are several open source augmented reality libraries available that are based on the OpenCV libraries. An example of such library is the ALVAR toolkit developed by VTT [2] that contains methods for marker and natural feature based tracking. Many augmented reality toolkits also use additional libraries for lower levels tasks and mathematical operations, such as Eigen or LAPACK for linear algebra which may be optimized for certain processors and make use of hardware acceleration. Most currently available AR technologies support iOS and Android devices, however there is currently no standard library for use in mobile AR applications. OpenCV is rapidly becoming the accepted standard for computer vision application development and it can be used rapidly expandable AR frameworks.

In the past, AR applications relied on fiducial marker tracking, however due to advancements in computer vision research and hardware, AR applications are tending to rely on markerless technologies based on tracking and mapping of natural features, allowing for virtual content creation to be performed dynamically [16]. Currently the most popularly used computer vision libraries provide support for SLAM methods in order to augment a users environment without the use of predefined trackables. Some of these libraries include Metaio [21] and Vuforia by Qualcomm TM[26]. These libraries all contain some form of SLAM implementation allowing the applications that leverage these toolkits to gain physical knowledge of a scene and store a 3D representation which can be used for augmentation.

2.4.1 Augmented Reality Authoring Tools

There are many systems, which allow for content authoring in augmented reality that will extend existing 3D modeling software. For example, DART (The Designers Augmented Reality Toolkit) [17] was introduced in 2003 and provided an interface for creating augmented reality environments at a higher level (Figure 2.2). This software is based on top of Macromedia Director, which was popular at the time and allowed for a large number of designers to create VR experiences and applications with a low barrier of entry. It advertised itself as having the ability to create rapid content creation for experience testing and design early on in augmented reality creation process.

Currently, many toolkits provide multiple ways for developers to use their software within applications. Generally, these kits provide an API for using their SDK within a development environment. The majority of toolkits now support some sort of interface with the popular game engine Unity3D. This allows for rapid development of high quality augmented reality applications with a low development barrier for developers. For example, the Vuforia SDK offers complete support for Unity3D integration, allowing for rapid cross-platform development. By integrating augmented reality toolkits within existing editors and graphical engines, developers can easily and quickly create AR applications with minimal development time and cost.

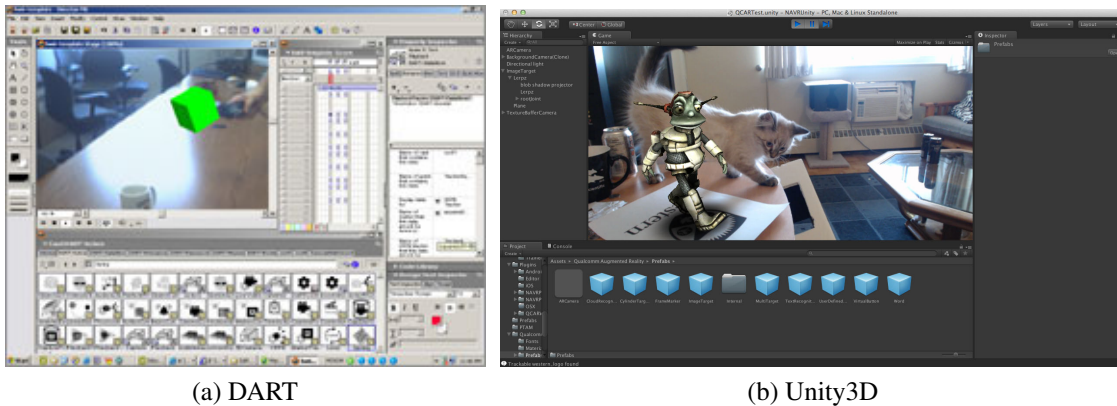


Figure 2.3: Augmented reality authoring tools can be used to extend existing content creation software.

2.5 Displays

2.5.1 Handheld Displays

As the inclusion of cameras in handheld devices is becoming ubiquitous, they are becoming an increasingly popular platform for augmented reality applications. Handheld displays have the ability to experience applications without needing an extra device and allows for a platform which users are comfortable using. Handheld displays are often used as a means to engage audiences in content in the form of advertising, education and training.



Figure 2.4: Apple iPad TMdevice augmenting a fiducial marker with a virtual object.

2.5.2 Head Mounted Displays

Using augmented reality technologies in combination with a head mounted display (HMD) can provide a fully immersive mixed reality experience. The availability of low cost displays has brought upon the possibility of using these devices in everyday applications. The Oculus RiftTM(Figure 2.5) has received a large amount of support from the gaming community due to its ability to provide a high feeling of immersion with a high-resolution screen with a large field of view. The amount of support for the development of augmented reality applications with this hardware is lacking. Applications have been demonstrated which use a camera attached to a head mounted display to augment the virtual world. This idea is also seen in Google GlassTM where information is overlaid on objects in the virtual world.

2.5.3 Surround Screen Projection Based Displays

Another approach to creating an immersive virtual reality experience is the CAVE (CAVE Automatic Virtual Environment) virtual reality/scientific visualization system first described by Cruz et al [6]. Room-sized projection based surround virtual reality (VR) system have



(a) Oculus Rift™

Figure 2.5: Head mounted displays are commonly used in immersive augmented reality systems.

been in development since 1991. There are many notable designs of the CAVE displays and advancements in visual acuity are constantly increasing.

2.6 Photorealistic Rendering

An important aspect of creating immersive and believable virtual environments is defining how to augment objects in a way that the virtual objects seem like they belong in the real world. The type of visualization technique generally depends on the purpose of the application and in some cases it may be better for the augmented objects to have non-photorealistic rendering. This type of rendering is useful in cases in which the application wants the object to contrast from the background, and can be more effective at conveying information [8]. However, techniques relating to photorealistic rendering, attempt to render the virtual object in such a way that they are indistinguishable from real objects. In the following sections we will describe some of these techniques and how they can increase the feeling of realism of virtual objects.

2.6.1 Lighting and Shadows

In order to have a virtual object seem indistinguishable from the real world objects, it should be lit and shaded correctly and cast accurate shadows in correspondence with the real world [28]. Without the use of shadows, virtual objects lack depth cues, causing the virtual objects to appear as if it is floating on top of a real surface. Sugano has determined that shadows of a virtual object provide two effects in AR environments, presence and spatial cues.

Lighting and shadows are very important techniques in achieving realistic environments. Using real time lighting allows for the implementation of realistic object surfaces. By adjusting the direction of virtual lights based on physical light sources realism of the scene is increased.

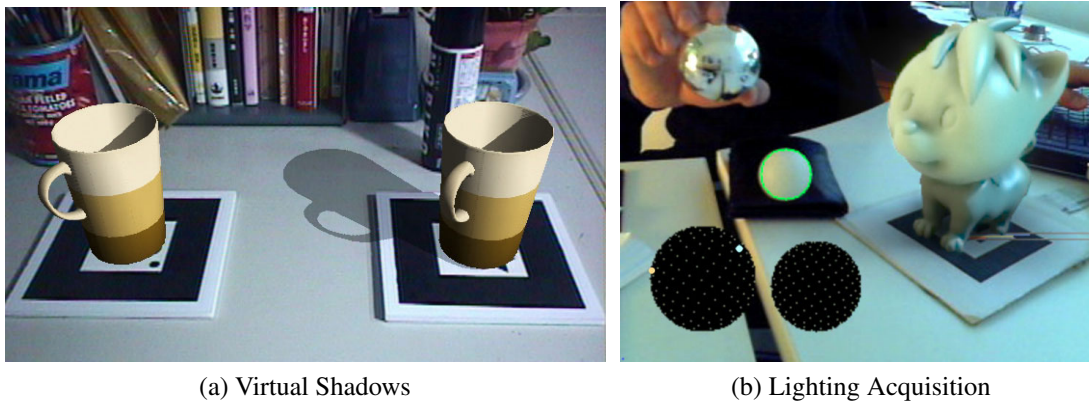


Figure 2.6: Adding shadows to virtual objects and an example of lighting acquisition procedure using a diffuse sphere (circled in green).

Virtual lights can imitate natural lights of the scene in such a way that the virtual objects seem to be illuminated naturally.

Users can define virtual lights manually or it is also possible for measurements to be carried out using special devices such as photometers or computer vision methods. Aittala has proposed a method of detecting lighting conditions using a white object [1], after which an inverse lighting method can be used to determine the light source (Figure 2.6). Other approaches have been used to improve the realism of lighting in augmented reality, including tracing [24] as well as detecting light sources and illumination with a sphere mirror [20].

2.6.2 Accounting for Captured Image Artefacts

It is necessary for virtual objects to appear similar in quality and appearance to achieve believability. A system must adapt the focus and motion blur of an augmented object to that of the virtual imagery. Many AR applications can also compensate for noise, lens distortion or other capture properties of the input image. Most commonly, AR applications render virtual objects sharp and in focus, which looks unnatural when combined with a fuzzy image. Adapting the augmented objects to have the properties of the captured image requires the detection of the attributes of the captured image and rendering the augmented layer appropriately with this knowledge.

Detecting the motion blur and defocusing in a captured image has been studied by image processing and computer vision researchers. The blur is generally modelled in a captured image as a convolution of the ideal image and a *point spread function* (PSF). Since both these artefacts commonly appear together it is convenient to use a PSF that expresses both [19]. An example

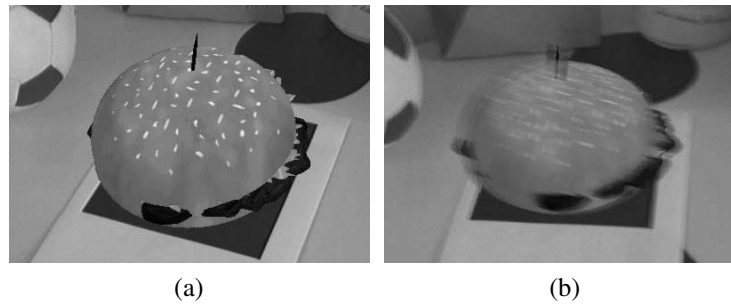


Figure 2.7: Virtual hamburger model generated in AR with simulated noise [9].

of the effects of a motion blur rendering method is seen in Figure 2.7.

2.7 Diminished Reality

In augmented reality systems, existing objects often disturb the feeling of augmentation. For example, if a fiducial marker is used for pose tracking, it may seem unnatural to the scene. Using image processing techniques, these can be virtually removed from the input image in a process referred to as diminished reality [25].



Figure 2.8: Diminished reality example, images from left to right: original, augmentation over existing object, diminished image, augmentation over diminished image. [25]

2.8 Summary

In this section we provided an overview of augmented reality technology and how it can be used to create believable augmentations of the real world. In the next section we describe our approach for the development of these applications and how these methods can be combined into a single framework.

Chapter 3

Approach and Solution

In our approach we describe a framework for the organization and development of an augmented reality system. Our framework aims to decrease the development time of visual based augmented reality systems by providing a simple cross-platform interface for application development. It allows for visual features to be organized, grouped and stored within an application, a task that is necessary for the augmentation of real world objects. Our framework supports the hybridization of numerous tracking methods, and provides a context for storing the knowledge of a user's physical environment.

We also provide a structure for which we can interface with contextual information derived from existing augmented reality toolkits. This allows us to provide a representation of trackables and features within our framework that can be used by applications to define virtual environments. In the following sections we describe the structure of the framework developed and how it interfaces with these toolkits as well as providing an open source platform for the development of extended functionality within the framework. By doing so we aim to reduce the development time, cost and barrier to entry while simultaneously providing a platform for which researchers can test and apply new augmented reality technologies.

3.1 Navigational Augmented-Virtual Reality Framework

The framework we describe in this thesis is intended to allow for the development a broad range of AR applications. For example, it could be used to simplify the development of a simple application that may place a virtual character on a trackable marker, or it could also be extended to create a fully immersive virtual reality training simulation. This framework organizes information to provide a general representation of the real world, such that a developer can define how a user will navigate and interact within the defined virtual environment. The Navigational Augmented-Virtual Reality (NAVR) framework described in this thesis provides a context for

the physical state of the real world as well as tools that support the processing of visual input to be rendered within augmented reality applications. This framework accomplishes this by providing an interface to multiple toolkits and modules, which contain functionality useful for the creation of AR applications. In developing and publishing this framework we hope to extend the lifecycle of AR applications by providing an adaptable development environment, which can incorporate advancements in AR related technologies.

Many of the modules that NAVR provides an interface with perform similar tasks and thus it is necessary to provide a layer of abstraction to translate and contextualize the information provided by each module. This is beneficial for developers because it allows for experimentation with several toolkits and tracking strategies within a minimal effect on the development of the application itself. For example, a user may know that they wish to use planar image targets as trackables within their augmented reality application. However, there are several toolkits and methods that exist for this task, which may have varying platform support and may also lose support in the future. In this case, a developer would define a generic trackable target within the NAVR framework and could swap the method used to perform this tracking supporting changes in the development lifecycle of the AR software.

Another benefit of this system is the ability to combine functionalities of augmented reality modules. For instance, a developer may wish to incorporate a SLAM implementation to map out a user's environment, but they may also wish to also include support for trackable detection. By using our framework, the developer would be able to experiment with several SLAM implementations in combination with trackable detection to determine which suited the needs of their application.

Developers can build AR applications based on the interface provided by the NAVR framework layer (Figure 3.1). This framework contains bindings and interfaces for both open source and closed source modules, which can be selectively incorporated within the application layer. Details of how NAVR interfaces with these modules and contextualizes the information they provide is described in Section 3.2.

3.2 Providing an Interface Between Layers

Different modules provide support for independent frameworks and toolkits that will have their own internal representations of information that can be used within an AR application. They may output information in different data structures or formats for trackable pose, but within NAVR these are all translated to a common representation. The NAVR framework must provide a wrapper for these modules that allows for translation of their outputted values to the format used and outputted by the NAVR framework. In a visual AR system, the functionality of the

system relies upon the detection of trackables within a scene as well as processing a video capture input for rendering in an application.

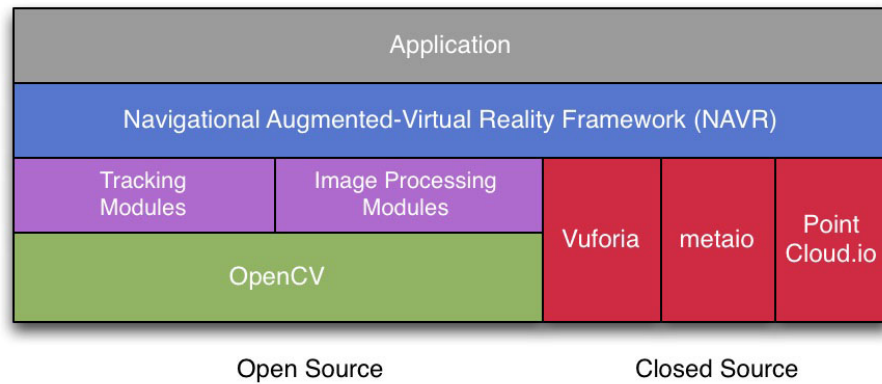


Figure 3.1: NAVR Framework Layer Diagram

In our framework the functionalities of these modules can be classified as one of the following:

- Trackable Detection
- Feature Point Mapping
- Input Frame Processing

For each module that is included in the NAVR framework, an interface must be provided to allow the user to activate and initialize each module. In the following sections (3.2.1, 3.2.2, 3.2.3) we describe the interface for translating the information that each of these modules provide. We also explain how this information is stored and contextualized for use by AR applications. Another function of this framework is updating each active module with each new captured frame. The NAVR framework must provide an interface that translates the captured video frame to the appropriate a format compatible with the module. A more detailed description for the interface for video captured input is provided in Section 3.3.

3.2.1 Trackable Detection Interface

Prior to the initialization of a trackable, the application must first specify the module it wishes for detection. The application must then initialize all specific trackables for which it wishes to detect within the application. To construct a trackable representation within our framework, the application must specify the trackable type as well the module it is to use for detection.

Additionally, the parameters specific to that trackable must be set by the application such as physical size, tracking error thresholds and classifier information.

The application must also specify a trackable group identifier, which is used to provide a reference to a virtual object or state. For instance, an application may create a trackable group called "Hero". The application layer would then need to define a virtual hero character object in their application and specify that they wish for this object to be displayed on screen relative to this trackable group.

Each trackable also has its own unique identifier generated by the NAVR framework at runtime. This identifier is used to link the trackable object in NAVR to the representation within the module performing trackable detection. The properties of trackables and trackable groups will be discussed further in Section 3.5.3.

Each detected trackable provides knowledge of 3D points and their corresponding view coordinates within the world. If a trackable is found the framework will update a feature map representation within framework with the feature points derived from the trackable pose. It will also store the corresponding normalized view coordinates within the current frame. The reason for this is that the framework can use correspondences stored in the feature map to determine a camera pose estimate for calculation of a users egocentric pose (Section 3.5.2).

3.2.2 Feature Point Mapping Interface

The framework stores a point cloud map that contains information on the states of all tracked features within a users environment. Modules that support an implementation of SLAM will provide an interface that will update the point cloud with mapped features. It is important for each feature point to have a unique identifier as well as a label that specifies the module being used to detect the feature as well as other properties, such as an error value representing the confidence of detection. The features that are tracked by SLAM are representative of the surrounding static environment of a user and can be used to determine their egocentric motion relative to a defined origin point. Information on how the egocentric pose is calculated as well as defining the origin point of a scene is described in Section 3.5.2.

3.2.3 Image Processing Interface

Modules that perform an image-processing task will take the captured input frame and output a layer that can be rendered within the application. The module must provide an interface that converts the input frame to a format compatible with the NAVR framework. The NAVR framework will then organize the rendering layers into a map, which can be accessed in a

variety of formats by the application layer for display. More information on how these modules are developed and their output is managed is described in Section 3.4.

3.3 Video Capture Input

The NAVR framework stores and processes video capture input via the interface provided by OpenCV. On each updated input frame, NAVR triggers an update of all active modules within the framework. It is important to note that each module handles the processing of captured input frames independently and possibly in a separate thread. Some modules may process the input frame in realtime on the update call from the AR application, but they may also add the input frame to a processing cue to be processed or discarded in a separate thread. This technique is heavily used in modules that have support for SLAM implementations in which expensive operations like bundle adjustment are used. In this case, these operations can be run in parallel on a low priority thread to avoid performance issues within the application.

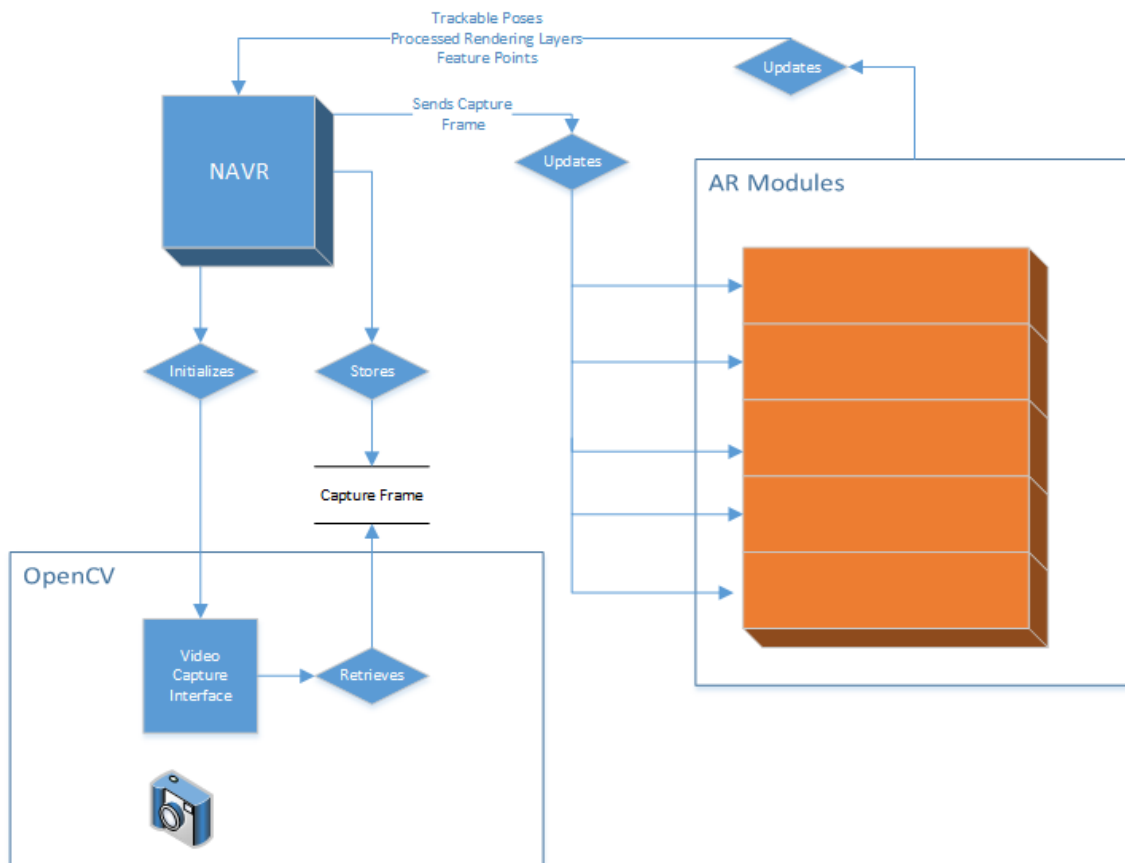


Figure 3.2: Camera input is captured and processed and an interface is provided to modules within the NAVR framework.

3.4 Rendering Layer Manager

The rendering layer manager provides an interface for the application to access the output of the image processing modules that are being used. The application can access these images in a format compatible with their application, and the application can choose to display the rendering layers stored as necessary. The manner in which the application chooses to display the rendering layers defines the classification of the system on the Reality-Virtuality continuum (Figure 3.4) described by Milgram and Kishino [18].

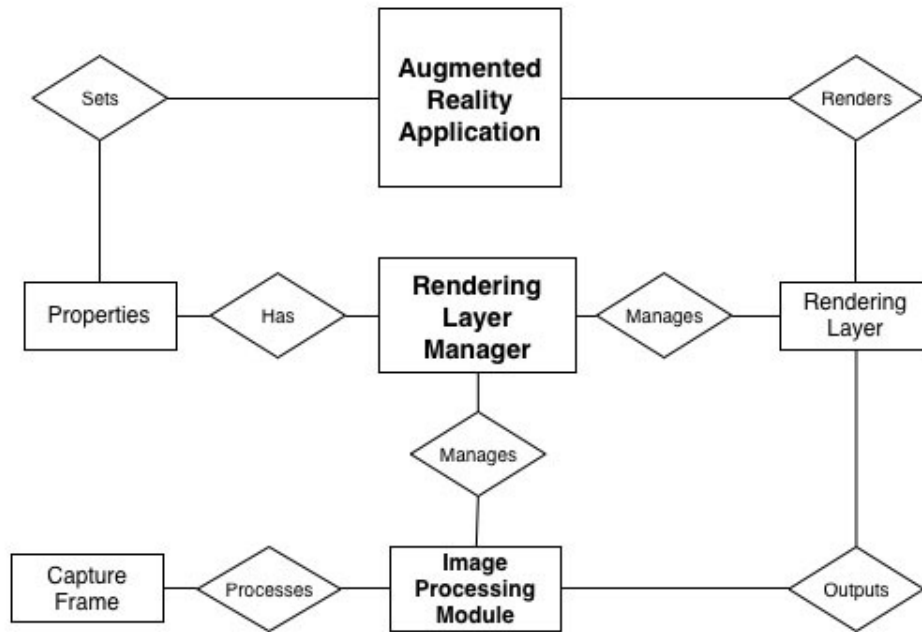


Figure 3.3: Rendering Manager entity relationship diagram.

For instance, if an application renders a virtual layer as the background and a user's hands and body in the foreground, this places the user in a predominantly virtual space. In this case, this would be an intermediate case with the Virtuality Continuum referred to as mixed reality. On the other hand, if an application were overlaying a virtual layer over the video capture frame this would be representative of augmented reality. If no rendering layers are used within the application everything that the user will see will be virtual. In this case the setup of the system will be defined as a completely virtual reality.



Figure 3.4: An example of how rendering layers are used to determine the systems classification on the Reality-Virtuality Continuum.

3.5 World State Manager

In AR applications, it is necessary to obtain knowledge of the state of surrounding augmentable objects in a user's physical environment. The NAVR framework manages information regarding the orientations and states of trackables and feature points within a users field of view. In the following sections, we discuss how the framework stores trackables such that they can be accessed by the application layer (Section 3.5.3), as well as organizing all known features into a point cloud that can be used to provide an estimate of structure for virtualization. We also discuss how the egocentric pose can be estimated by using correspondences between points stored in the point cloud and their respective positions in the viewport (Section 3.5.2).

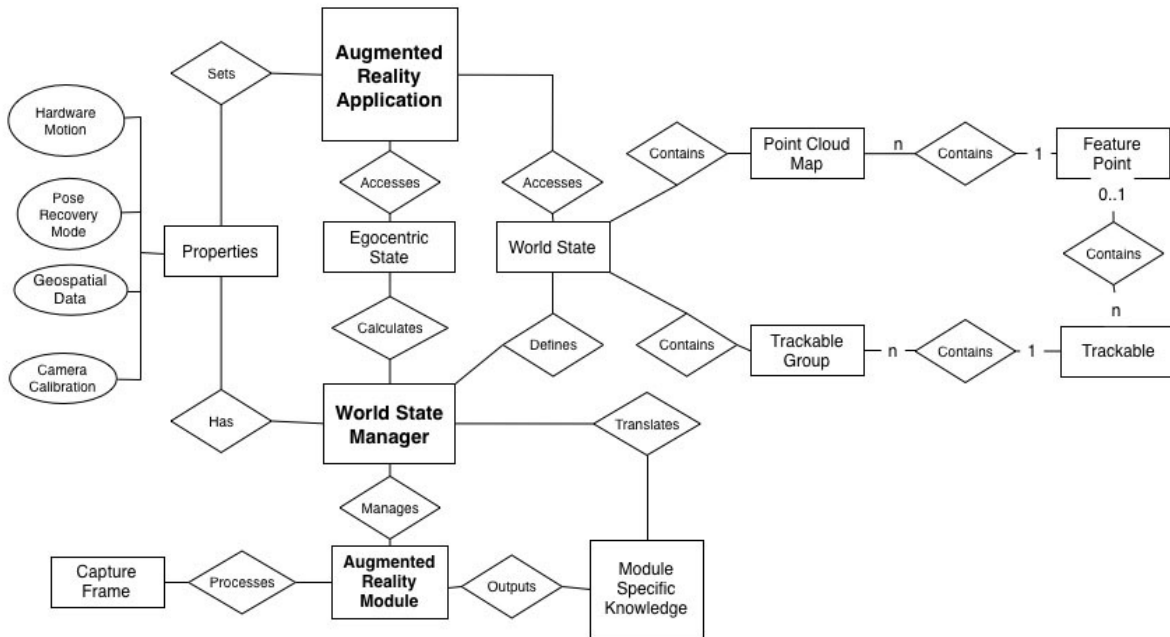


Figure 3.5: World State Manager entity relationship diagram.

3.5.1 Feature Point Cloud Map

The states and pose of all known features contributed from by trackables and mapping procedures are stored within the framework. From this a point cloud of mapped features is created and used to infer the physical structure of a users scene. A feature contains information on its estimated world position, the last time it was tracked, and the confidence value of its current position. Also, if a feature point is currently being tracked in an input frame, the normalized screen coordinates are stored. This feature point map contains information from trackables as well as mapping procedures.

This map can be defined by a mapping procedure implemented in a module, but this module must provide an interface that allows the NAVR library to interpret the information provided. With access to the current map data being used by the application, a virtual environment can be built to augment the associated physical surroundings. This allows for an application to create a mixed reality experience in which virtual objects and characters can interact with real world structures based on information received from various modules.

3.5.2 Egocentric Pose Determination

In this section we describe how the framework can be used to determine the three dimensional location of a user with respect to a defined origin. By providing the ability for users to navigate through the virtual environment the framework allows for the interaction with fully immersive virtual worlds.

In order for the camera pose to be calculated from the feature point correspondences stored within the framework (Section 3.5.1) it is first necessary for the intrinsic parameters to be determined using a calibration procedure. An interface to OpenCVs calibration methods is provided by the NAVR framework and can be used. Alternatively, modules could be used in order to provide an alternate calibration method and would be able to update the intrinsic parameter representation stored in the framework.

The NAVR framework supports the use of inertial measurement input (IMU) sensors as well geospatial coordinates within the application to perform a data fusion for pose estimation. This technique was proposed by You et al. [31], in which a framework was created for stable static and dynamic egocentric pose calculation. By using sensor fusion within the framework can account for tracking failures caused by rapid camera motion or occlusion. The framework does not currently provide an explicit implementation, but provides support for several extensions of Kalman filters.

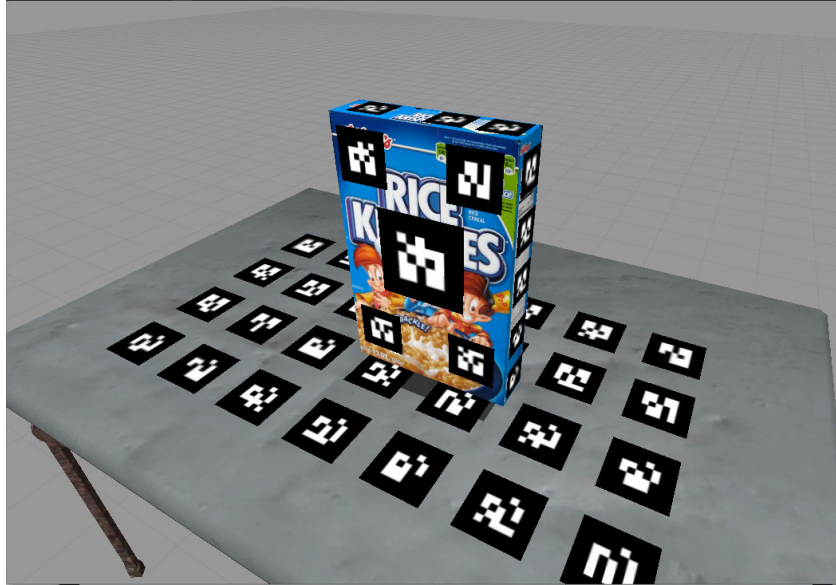


Figure 3.7: Related trackables objects are classified into trackable groups that relate related physical bodies. In this example there are two trackable groups, a cereal box and a table.

allows for the determined world position of a trackable group to be based on the most reliable trackable pose information. Conditional hierarchies can be defined, in which the estimated pose of certain trackable is used preferentially under certain conditions set by the user. This is used by the application to selectively use trackables that are most reliable in various lighting conditions or hardware setups.

In some immersive virtual reality applications developers may choose to segment out real objects such as users hands or objects that the user wishes to interact with and have visible within the virtual world. As well, in augmented reality setups it is also possible to define areas which act as windows to the virtual world. This is commonly referred to as diminished reality. In our approach, reality segmentation regions are anchored to trackable groups. Segmentation regions are used by an image processing module to determine which regions should be segmented out into separate rendering layers (See Section 3.4). In this implementation only rectangular shapes are only supported; however, trackable groups could be extended to use vector-shape based segmentation regions in the future.

3.6 System Architecture

Augmented reality tools and technologies are continually changing and it is necessary for the architecture of the system to support and adapt these changes. Thus, our framework is designed to be modular and cross platform such that it can be extended to make use of various hardware

and display setups. This framework was developed in C++ and is based on the open source computer vision library, OpenCV. Input from capture devices is stored in an OpenCV image format, however wrappers and support for other image formats are provided. In the following section (3.7) we will describe how the framework can be incorporated into graphical engines to provide an interface for creating virtual environments.

3.7 Graphics Engine Integration

This section details an example of how this framework can be incorporated as a plugin into a graphical engine, providing an intuitive interface for development. The framework currently provides support for integration into the Unity3D game engine. This engine was chosen because of its cross-platform support, large developer community base and its overall popularity. It is currently standard for commercial augmented reality toolkits to provide support for Unity3D; however, there is currently little support for open source augmented reality toolkits.

In order to use the NAVR framework within this game engine a plugin had to be developed to interact with the NAVR libraries. The Unity3D Engine is built on Mono 2.6, an open source implementation of the .NET framework. Programmers may use C# to write scripts to assign behaviours to game objects. In order to integrate the NAVR framework it was necessary to write a C# wrapper within Unity3D to integrate with the NAVR plugin which has a C++ interface. It is our hope that by providing this framework we will be able to help developers integrate open source computer vision libraries and AR tools into game engines.

Within the Unity3D game engine the base class for all entities in a scene is the Game Object. A developer will attach scripts to game objects to provide functionality. Trackable groups are stored as game objects within the scene and their properties and parameters can be set in the editor. Child trackables can also be attached to other trackables and this configuration can be done within the Unity Editor (Figure 3.6). If these trackables are defined within the scene they will be initialized within the NAVR framework at runtime.

A user interface is provided by extending the Unity Editor, allowing for modification of trackable parameters within the NAVR framework. Being able to provide a 3D visual representation of trackables and features allows developers to effectively organize and link virtual objects to real world objects, allowing for augmentation of the users physical environment. As well, game objects can react to changes in pose as well as states of trackable groups. This allows developers to build scripts and behaviours based on information queried from the NAVR framework to create immersive AR environments.

The NAVR interface within Unity3D allows developers to create trackables and set their detection parameters within the Unity Editor. In Figure 3.7 an interface for creating fiduciary

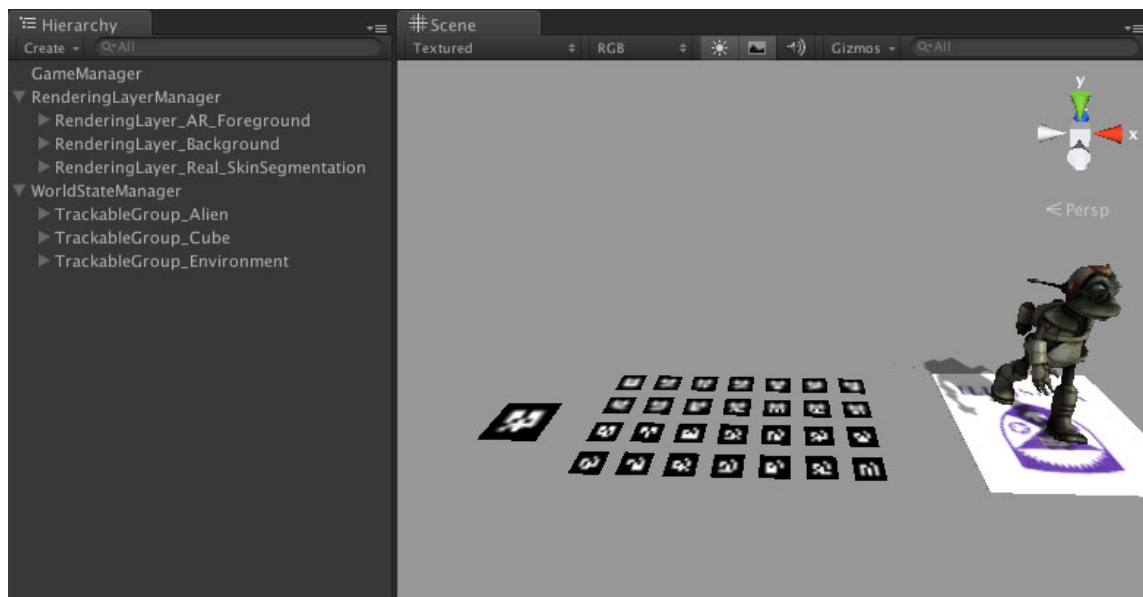


Figure 3.8: Representing trackables as game objects within Unity3D and assigning virtual objects. Single marker target (Left), multi marker target (Middle), planar image target (Right).

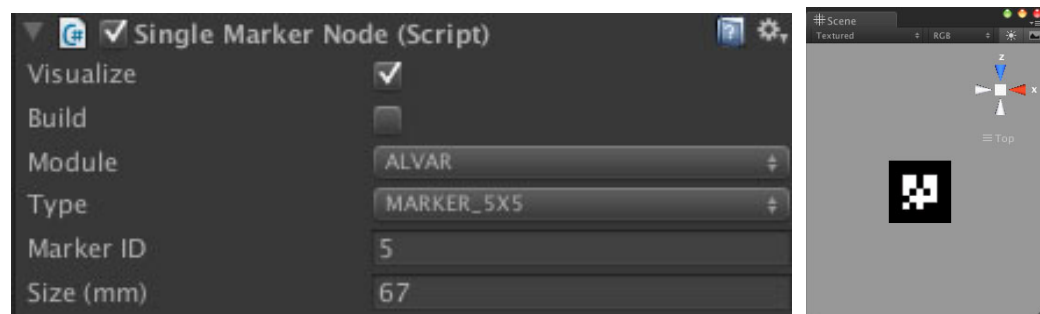


Figure 3.9: Interface to NAVR within Unity3D for creating for creating single trackable marker in Unity3D.

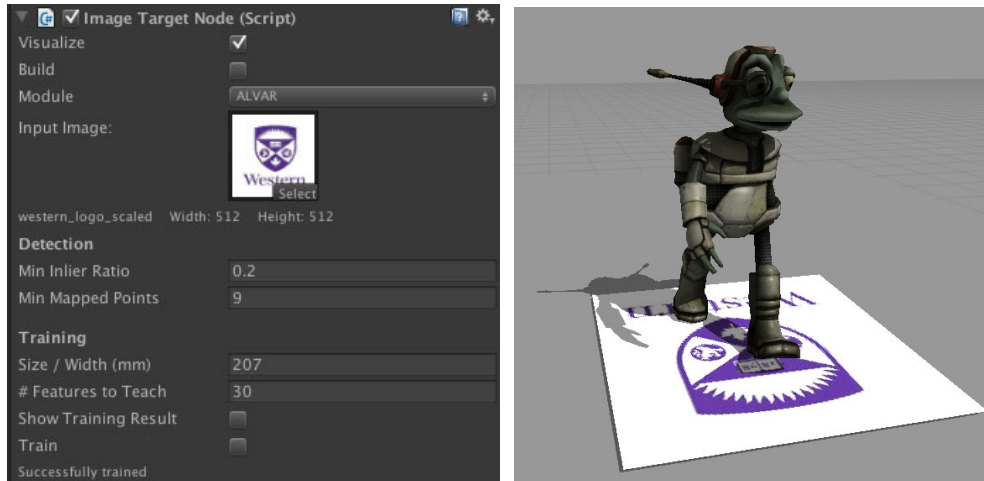


Figure 3.10: By extending the NAVR framework, we provide an interface for defining, training and visualizing planar image targets within Unity3D editor.

markers can be seen in which a user specifies the module to be used for detection and the relevant parameters. The current implementation of allows for detection and training of planar image targets. The module that is being used in Figure 3.8 is the ALVAR augmented reality toolkit. ALVAR provides functions that are can be used to train image recognition classifiers for planar image target tracking. Our interface allows for developers to perform training from directly within the editor. This lets developers configure image training and detection parameters within an editor, whereas alternatively they would have previously needed to be performed programmatically.

The ability to visualize feature points and trackables that are to be used within the framework can allow developers to place virtual objects directly in the scene that the user will be navigating through. These viewpoints could potentially be mapped out prior to development using a complex SLAM implementation or they could be inferred from a trackable setup (As seen in Figure 3.9).

3.8 Summary

In this section we outlined our approach for the development of AR applications. We discussed how our framework provides a modular interface for the integration of toolkits that provide functionalities useful in the development of an AR system. Our framework provides a representation of the world state that can be used to define virtual environments, characters and objects. We also described how our framework could be integrated into game engines to

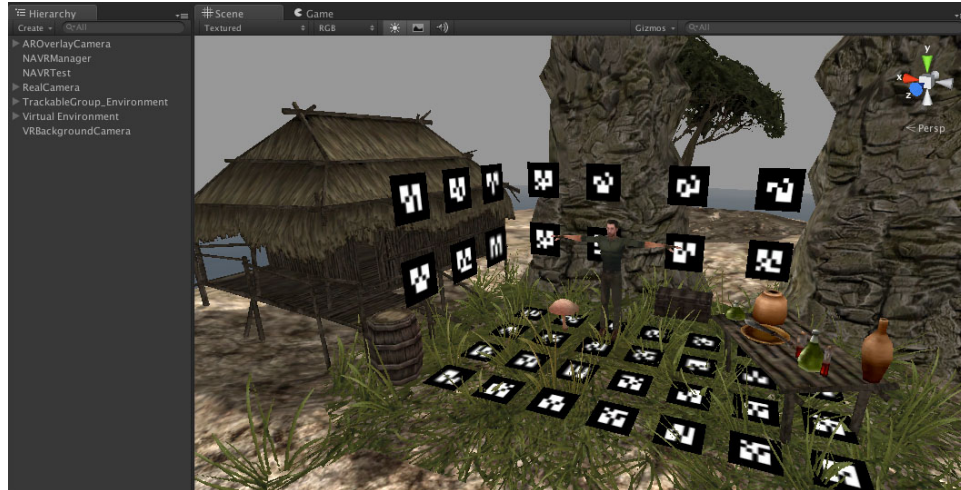


Figure 3.11: Visualizing trackables within a graphical engine allows for organization of augmentable objects in the real world in relation to virtual objects.

provide an intuitive interface for development. In Chapter 4 we will provide an evaluation of the development process using our framework, and compare it to previous development efforts.

Chapter 4

Evaluation and Validation

In this Chapter, we first describe the system that was previously used for creating navigational augmented reality systems that was the inspiration for this research. We will describe the hardware used in this system as well as the software architecture. The NAVR framework described in this thesis is designed to be completely compatible with ALVAR, the augmented reality libraries used in the previous system, as well as the formats of the 3D models and environments that were created. In Section 4.2 we describe the outline the development process with NAVR and its benefits versus the original system. We also compare it to development with other commercial SDKs and outline the advantages and disadvantages of developing with our framework.

4.1 Description of Previous Framework

The previous framework developed by Garcia [11] used a head mounted display for presenting virtual environments to the user. This system was designed for use as a potential rehabilitative tool and there was a requirement for it to be completely wearable, allowing for navigation throughout the virtual environment. The main hardware considerations were the weight of the system, as well as processing power and connectivity of the devices. The system used a lightweight portable computer that could be placed within a backpack to provide minimal hindrance to the user. This device also needed to have significant 3D graphical and computation processing power. The aim of this system was to develop a system that could be used to provide a high sense of immersion with low equipment cost. This system used the VUZIX iWear 920VRTM, a head mounted display with an attached camera (Figure 4.1), allowing for users to visualize virtual environments from a first-person perspective.



Figure 4.1: The VUZIX iWear 920VR head mounted display.



Figure 4.2: Virtual scenes rendered in previous framework developed by Garcia [11].

4.1.1 Physical Setup

This setup relied on a physical space with a predefined fiducial marker system that defined a point cloud in the virtual world. In order to setup the environment a room that measured $6.68m \times 4.92m$ was enclosed with four white vinyl walls, which were then covered with printed markers (Figure 4.3). These black and white markers were chosen because they were ideal for detection in varying light sources due to their high contrast. This setup consisted of markers of varying sizes to support detection from various distances. Markers were placed in such a manner that at least one marker could be seen from all viewpoints within the room. Overall a total of 110 fiduciary markers were used. The setup of this room required precise measurement and this required a large amount of time to accurately determine the coordinates of each fiducial marker.

4.1.2 Software Architecture

In this section we provide an overview of software architecture of the previous framework developed by Garcia [11]. This framework was built using C++ using OpenCV for image processing and OpenGL for graphical rendering. It used a module, referred to as the AR Driver, which built upon the ALVAR toolkit to compute the 3D transformations of the trackables within the scene. The application would then feed this information into a Scenario Manager module which would handle rendering of the virtual scene. The system also had an image-processing



Figure 4.3: Physical space in which fiducial markers were setup in order to create a predefined multi-marker system.

module built with OpenCV that performed skin segmentation. The result of this segmentation was overlaid over the 3D graphical layer. In this framework users were able to interact with real objects within the virtual world by the strategic use of rendering layers. The development time and cost of this system was high due to the lack of graphical engine and scene management tools.

4.2 Development with NAVR

The creation of the NAVR framework was inspired by the need to quickly create a navigational virtual reality system in which immersive virtual environments could be explored. The original framework required the development of a graphical engine, augmented reality integration, and system for defining behaviours of virtual objects. The physical setup of the scenes was also quite time consuming and took weeks to setup and accurately measure the coordinates of each of the fiducial markers placed within the room. With our framework we address the development issues of the previous framework and provide a novel way of organizing the development of augmented reality applications.

To develop an augmented application using the NAVR framework it is ideal for a developer to work within a game engine. Developers can alternatively call the NAVR layer explicitly or develop their own extension to work within an engine. To integrate the augmented reality functionality into their application, the developer must specify which modules and trackables

within those modules it wishes to use. Then it must update the NAVR framework at a suitable timestep determined by their application. After each update frame the application can query the NAVR framework to retrieve the states and poses of all of defined trackables. The developer can then use this information within their game or simulation to provide augmentation of the real world by defining virtual objects.

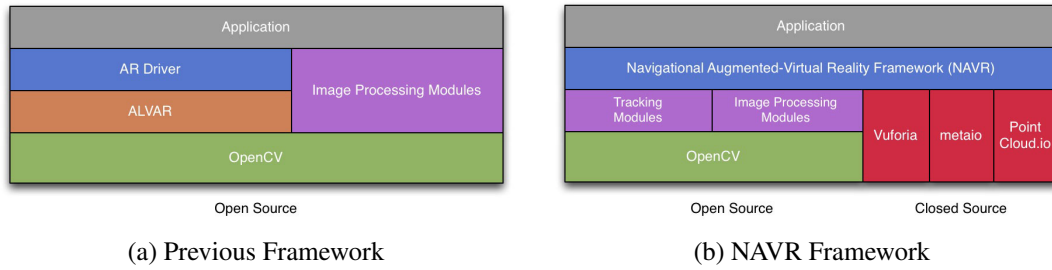


Figure 4.4: Comparison of layer diagrams representing software architecture of previous system vs NAVR framework.

As seen in Section 3.7, providing a graphical user interface for the configuration of trackables and scene features, allows for quick and intuitive definition of trackables within a physical scene. This allows the developer to visualize the relation of the markers in the real world to virtual objects while developing the AR application. In the previous framework it became difficult to decide the distance a wall or an obstacle should have been placed in the original framework due to the inability to visualize this information. By using a game engine to design these simulations one is able to experiment with the placement of objects. As well, it is much simpler in terms of development to create behaviours and events within a game engine because game engines are designed for the purpose of creating highly interactive and responsive virtual worlds.

4.2.1 Defining Rendering Layers

The NAVR framework supports the development of virtual reality applications that fall under several various classifications of the virtual reality continuum. This system supports traditional augmented reality configurations in which objects and information can be overlaid on top of real camera input. It also supports an immersive mixed reality environments in which the user can only see virtual objects within field of view with certain real world objects, including hands and interactable objects were segmented from the video input frame and overlaid in the user's field of view. Defining a virtual reality system is achieved by on controlling which image processing modules are incorporated and the order in which layers are rendered within

the system. For example, in a standard augmented setup, the application would display the captured input frame in the background. The application would need to define which trackables and are present are to be searched for. It will then send the frame capture information to NAVR which will then in turn use the specified AR modules to calculate and store the pose of each trackable within the input frame.

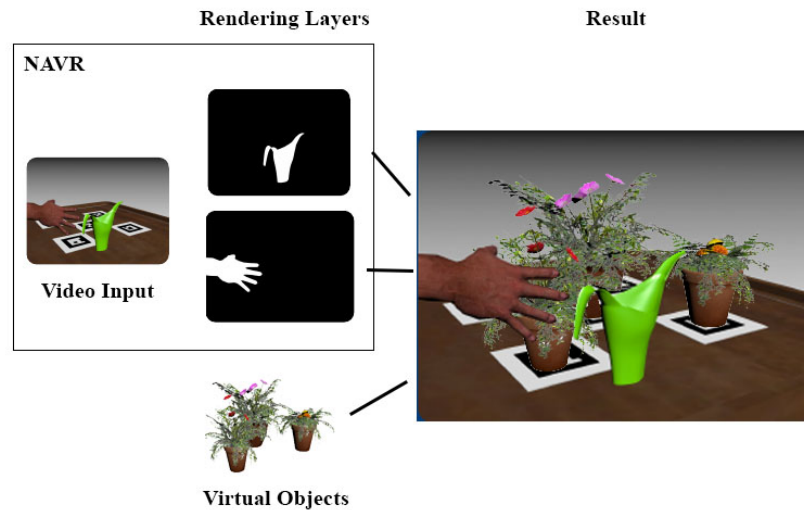


Figure 4.5: Rendering layers are created by the image processing modules within NAVR and by the rendering of virtual objects in the application.

As a result of the modular design of the NAVR framework, developers can choose which image-processing module the application uses and can then access the resulting rendering layer within the application being developed. For each rendering layer the developer must select a depth value within the graphical engine that will determine the order in which the rendering layer will be displayed within the scene. By providing this interface for development we are able to experiment with several variations of image processing algorithms. In the previous framework Garcia [11] used a skin classification algorithm that was used to overlay a users hands over a virtual world in the system. By doing this, the problem of unnatural occlusion of virtual objects was overcome by placing the hand rendering layer in front of the virtual objects within the scene.

4.2.2 Creating Virtual Environments

This section outlines the process for the creation of virtual reality environments in a game engine. In our implementation we use Unity3D, however a similar interface could be created for other common engines such as Unreal Engine or Ogre3D. By using a game engine we were

able to add more advanced characters states, and behaviours to the virtual worlds that we created. When developing with our framework, developers can assign scripts in the game engines, which can respond to changes in state of trackable groups. Game developers can choose a development environment which they are already familiar with to create their applications and can use existing assets and environments. This drastically decreases development time and allows for more complex simulations to be created.

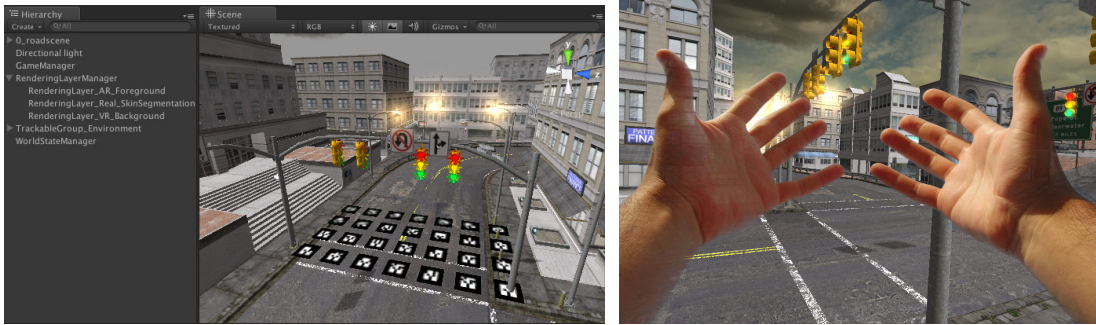


Figure 4.6: Street environment created using the NAVR framework in Unity3D (a) and then a rendering layer defined by a skin classification module was overlaid (b) to create an immersive experience.

A rendering of one of the virtual environments created by Garcia [11] can be seen in Figure 4.2. Development of the 3D models and rendering code took months of development time. In the new system using NAVR the time necessary to create a similar fully functional application could be decreased to just days. We created the environment scene in Figure 4.6 in approximately an hour by using the our the interface for NAVR created within Unity, as described in Section 3.7 with a predefined multi-marker setup.

4.2.3 Interaction Between Trackable Groups

An integral part of believable virtual environments is creating characters and objects that can react to changes in the real world. To allow for interaction between virtual objects within the application, events can be triggered upon conditional changes in trackable positions and states. For example, if two trackable groups are defined and are linked to virtual characters in a game scene, interaction events can be triggered when the two groups come within a certain distance (Figure 4.7). For example, a humanoid character could be anchored to one trackable group and a dog to another. When the interaction event is triggered, the human reaches out and pets the dog and the dog sits on its hind legs. By using trackable groups, we can define these interactions independent of the tracking mechanism. This allows developers to change

the trackables within the group and the modules being used to perform the tracking while still having the behaviour associated with the group itself.



Figure 4.7: An interaction event is triggered when trackable groups come within a threshold distance.

4.2.4 Image Effects for Photorealistic Rendering

Creating realistic environments is key in the construction believable virtual reality experiences. In Section 2.5 we discussed techniques for photorealistic rendering. Producing these effects was quite difficult in the previous system due to the lack of built in support since the software was not built upon a rendering engine. Most game engines provide support for image effects that can be used to merge real and virtual world objects. These effects can easily be integrated and help to increase the photorealism of augmented objects within the scene, as can be seen in Figure 4.8. In this example, we enabled real-time lighting and shadows, and shadows were cast onto a plane defined below the character. By providing shadowing effects we enhanced the perception of the characters depth within the scene. We also used a noise and grain filter effect to account for the image artefacts that were present within the camera input source. As well, we applied a focusing blur, such that objects within the periphery of the field of view were blurred.

4.3 Summary

In this chapter we discussed the development process within our framework and compared it to previous development efforts. We also described how virtual environments could be created within game engines, and how by using our proposed approach we could visualize and organize

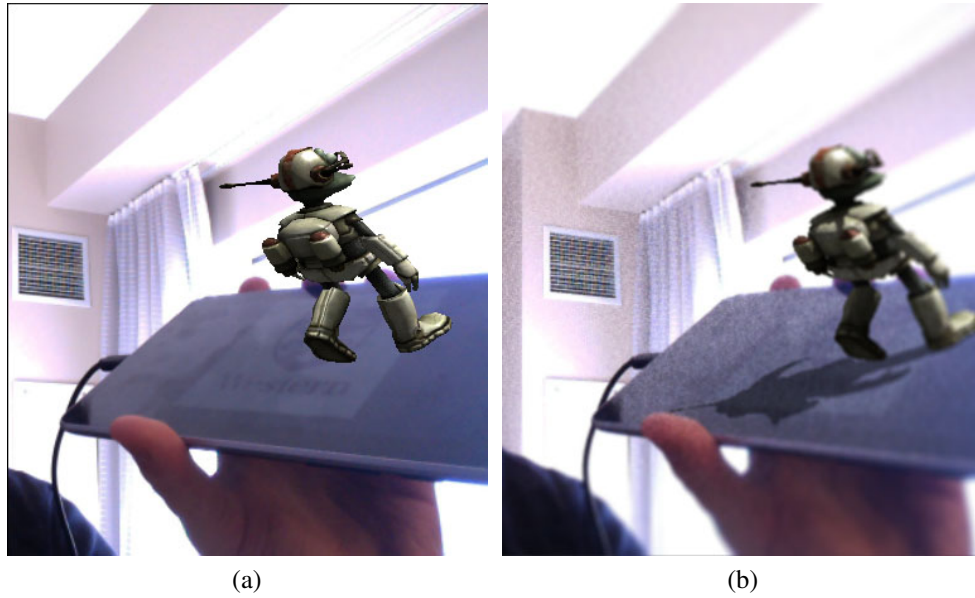


Figure 4.8: Comparison of augmented objects rendered with and without image effects. Applied effects include shadows, noise and grain, and a focusing blur.

trackable groups. In Chapter 5 we will outline the benefits of using this approach and also discuss the development issues that this system addresses.

Chapter 5

Discussion

The development of our framework was intended to address the shortcomings of the previous system as well as other similarly structured AR applications. In this section we discuss some of the development issues faced when developing AR systems and our approach for addressing them. These issues include:

- Development time and cost
- Physical setup time and cost
- Tracking stability
- Game engine integration
 - Graphical quality
 - Physics support
- Portability
 - Ability to deploy on mobile devices
- Cross platform support
- Software extensibility

It is important to note that many of these issues are not addressed by the system directly, but instead by providing an interface for enabling the incorporation of AR systems into game engines. In Section 3.7 we described how we integrated the NAVR framework into the Unity3D engine. By doing so we have reduced development time for future AR systems. The overall quality of the system was enhanced due to the provided support for rendering environments

and other graphical effects. These engines also contain support for complex physics simulations which enables realistic environments and simulations to be created. They also allow for developers to incorporate realistic characters, behaviours and animations. By doing so we can increase the overall feeling of immersion by simulating realistic interactions with characters within games such as in [5].

One issue of the system that we wanted to address was the issue of portability in order to provide support for a system that allows for navigation through an environment. Previously we used a laptop that was carried by users as they walked through our virtual environment. We developed the new system to be cross platform in order to potentially allow for mobile devices running on Android or iOS platforms to be used.

The previous system used a large multi-marker system to create a point cloud for egocentric pose calculation. It is advantageous to reduce the number of markers used in the system to reduce setup cost and time. To address this issue we incorporated support for SLAM implementations enabling for AR systems to be created that can track natural features in addition to trackable markers. We also added interfaces for sensor fusion to make use of gyroscope, magnetometer, accelerometer and geospatial data to be incorporated into AR systems. Through the use of filters and marker-less technologies we were able to also address the issue of tracking stability by implementing pose recovery mechanisms.

In this thesis we have described how our modular framework can be used for the integration AR technologies into software applications. By allowing for developers to apply new and emerging tracking methods we can potentially enable AR applications to adapt to rapid changes in technology therefore enhancing the extensibility of software developed with this framework. One of the main benefits of providing a framework that interfaces with augmented reality libraries is that developers can focus on creating virtual experiences that are not necessarily tied to a specific toolkit. As a result, the developer does not need to worry about updating the computer vision toolkits used and can experiment with different configurations that suit the needs of the application. For instance, some SLAM implementations are ideal in small workspaces whereas others may be better equipped to deal with larger environments.

Currently, development kits and tools that are readily available AR system creation are expensive, and are not always expandable to suit the needs of specific applications. Our solution is based on OpenCV and other open source libraries that will allow for AR technologies to be developed and incorporated into modules resulting in lower development cost by avoiding the use of commercial software. Development of OpenCV is currently working towards full support for all popular mobile platforms. Efficient use of the CPU and GPU is an important factor in real-time image processing. The development of methods that make use of hardware acceleration is currently one of the focuses of OpenCV and will likely be available in future

versions.

Future development of the NAVR framework will allow for the inclusion of modules that allow for more advanced AR functionalities that are currently only included in commercial toolkits. By providing an open source solution, our framework can support variations of popular tracking methods as well as the ability to efficiently hybridize augmented reality techniques.

The development of virtual reality experiences within game engines seems to be a logical relationship. By using augmented reality libraries within game engines it enables developers to visualize and define how virtual characters and environments will interact with the real world. It is clear that game engines will play a pivotal role in the development of augmented and virtual applications due to their popularity and the support of emerging augmented reality toolkits.

5.1 Disadvantages of Our Approach

One of the major downsides of adding a framework for the organization is that there is an extra layer in the software that could potentially reduce efficiency as opposed to just interfacing directly with a toolkit. This extra layer may also store extra information that is not necessary for specific applications and in some cases it may actually be a better to use a specific toolkit directly. The NAVR framework is meant to be an adaptable cross-platform solution that can adapt quickly to new technologies. Thus, this framework is especially useful for projects that are intended to have a longer development life, and will need to the adapt to rapid changes in augmented reality technology. This framework is built in anticipation of wearable technology and devices and is a potential solution for developers.

Chapter 6

Conclusion

6.1 Contributions

In this thesis we described our approach for creating a framework for the development of augmented reality applications. Our approach addressed how the creation of a modular system could improve the ability of developers to harness and adapt to augmented reality related software technologies. Our proposed framework provides a means for reducing the cost and time involved in the development of these systems, thereby making them easier to deploy.

One of the most useful features of our proposed framework is the ability for developers to create software that allows for a user to navigate freely through immersive environments. By providing an interface to dynamic feature mapping implementations we allow augmented reality applications to be created that are independent of a predefined physical environment. Also described in this thesis is the ability to incorporate and provide an interface for our framework within game engines. As virtual reality technologies become increasingly popular the integration of computer vision and gaming technology will become an important area of research. We hope that our framework will be an important tool for demonstrating how these technologies can be combined to create novel applications.

6.2 Future Work

In order to provide a quantitative analysis of the effectiveness of this framework it will be necessary to evaluate the development cycle of AR applications using our framework. We will need to provide a comparison of applications developed with and without our approach. From this we can provide an analysis of how well the research questions proposed in Section 1.2. It will need to assess the factors that we earlier discussed such as development time, cost and also

lifetime of the software itself. It will be necessary to provide some sort of measure of the ability for applications using our approach to adapt to changes in technology and the availability of AR toolkits versus those using a standard approach to the development of AR applications. Some of the benefits of this approach could be tested with simple user testing; however, factors such as long term flexibility of this framework will need to be tested over an extended period of time.

Future developments will focus on the ability for developers of AR technologies to create modules that can interface with our framework. By allowing developers of AR applications to quickly add and modify existing modules within our framework we will be able to enable the development of virtual reality experiences. It will also be necessary to provide a more complete interface for the storage of features within our framework. It will be necessary to provide a means for the storage of the features within a database for sharing and retrieving throughout applications. It will also be necessary to further investigate the integration of sensor fusion methods within our system to allow for the use of geospatial data and motion data to be applied to the egocentric motion model of users.

6.3 Final Comments

As augmented reality technologies become increasingly popular, new ways of interacting with the world around us will arise. It is our hope that by using the approach described in this thesis we can enable developers to use emerging augmented reality technologies in fields such as education, training, entertainment and healthcare. With advancements in virtual reality display technology and the processing power of mobile devices we expect to see a rise in the popularity of augmented reality experiences. It is our hope that in the future this framework will provide a basis for the creation of unique virtual experiences that will impact how we interact with and experience the world around us.

Bibliography

- [1] Miika Aittala. Inverse lighting and photorealistic rendering for augmented reality. *The visual computer*, 26(6-8):669–678, 2010.
- [2] Antti Ajanki, Mark Billinghurst, Hannes Gamper, Toni Järvenpää, Melih Kandemir, Samuel Kaski, Markus Koskela, Mikko Kurimo, Jorma Laaksonen, Kai Puolamäki, et al. An augmented reality interface to contextual information. *Virtual reality*, 15(2-3):161–173, 2011.
- [3] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent advances in augmented reality. *Computer Graphics and Applications, IEEE*, 21(6):34–47, 2001.
- [4] Ronald T Azuma, Bruce R Hoff, Howard E Neely III, Ronald Sarfaty, Michael J Daily, Gary Bishop, Vern Chi, Greg Welch, Ulrich Neumann, Suyu You, et al. Making augmented reality work outdoors requires hybrid tracking. In *Proceedings of the First International Workshop on Augmented Reality*, volume 1. Citeseer, 1998.
- [5] Christine Bailey and Michael Katchabaw. An emergent framework for realistic psychosocial behaviour in non player characters. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, pages 17–24. ACM, 2008.
- [6] Carolina Cruz-Neira, Daniel J Sandin, and Thomas A DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 135–142. ACM, 1993.
- [7] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

- [8] Jan Fischer, Dirk Bartz, and Wolfgang Straßer. Stylized augmented reality for improved immersion. In *Virtual Reality, 2005. Proceedings. VR 2005. IEEE*, pages 195–202. IEEE, 2005.
- [9] Jan Fischer, Dirk Bartz, and Wolfgang Straßer. Enhanced visual realism by incorporating camera image effects. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 205–208. IEEE Computer Society, 2006.
- [10] Russell Freeman, Anthony Steed, and Bin Zhou. Rapid scene modelling, registration and specification for mixed reality systems. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 147–150. ACM, 2005.
- [11] Andres Ayala Garcia. Interactive augmented reality as a support tool for Parkinson’s Disease rehabilitation programs. Master’s thesis, The University of Western Ontario, London, ON, 2012.
- [12] Charles E Hughes, Christopher B Stapleton, Darin E Hughes, and Eileen M Smith. Mixed reality in education, entertainment, and training. *Computer Graphics and Applications, IEEE*, 25(6):24–30, 2005.
- [13] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Augmented Reality, 1999.(IWAR’99) Proceedings. 2nd IEEE and ACM International Workshop on*, pages 85–94. IEEE, 1999.
- [14] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR ’07*, pages 1–10, Washington, DC, USA, 2007. IEEE Computer Society.
- [15] Peter Lang, Albert Kusej, Axel Pinz, and Georg Brasseur. Inertial tracking for mobile augmented reality. In *Instrumentation and Measurement Technology Conference, 2002. IMTC/2002. Proceedings of the 19th IEEE*, volume 2, pages 1583–1587. IEEE, 2002.
- [16] Tobias Langlotz, Stefan Mooslechner, Stefanie Zollmann, Claus Degendorfer, Gerhard Reitmayr, and Dieter Schmalstieg. Sketching up the world: in situ authoring for mobile augmented reality. 16(6):623–630, 2012.
- [17] Blair Macintyre, Maribeth G, Steven Dow, and Jay David Bolter. Dart: A toolkit for rapid design exploration of augmented reality experiences. In *In ACM Symp. on User Interface Software and Technology (UIST’04*, pages 197–206, 2004.

- [18] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321–1329, 1994.
- [19] Bunyo Okumura, Masayuki Kanbara, and Naokazu Yokoya. Precise geometric registration by blur estimation for vision-based augmented reality. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–4. IEEE Computer Society, 2007.
- [20] Saulo Pessoa, Guilherme Moura, JPSM Lima, Veronica Teichrieb, and Judith Kelner. Photorealistic rendering for augmented reality: A global illumination and brdf solution. In *Virtual Reality Conference (VR), 2010 IEEE*, pages 3–10. IEEE, 2010.
- [21] Juri Platonov, Hauke Heibel, Peter Meier, and Bert Grollmann. A mobile markerless ar system for maintenance and repair. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 105–108. IEEE Computer Society, 2006.
- [22] Muriel Pressigout and Eric Marchand. Hybrid tracking algorithms for planar and non-planar structures subject to illumination changes. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 52–55. IEEE Computer Society, 2006.
- [23] Gerhard Reitmayr, Ethan Eade, and Tom Drummond. Localisation and interaction for augmented maps. In *Proceedings of the 4th IEEE/ACM international Symposium on Mixed and Augmented Reality*, pages 120–129. IEEE Computer Society, 2005.
- [24] Fabian Scheer, Oliver Abert, and Stefan Müller. Towards using realistic ray tracing in augmented reality applications with natural lighting. In *4. Workshop Virtuelle und Erweiterte Realität der GI-Fachgruppe VR/AR*, 2007.
- [25] Sanni Siltanen. *Theory and applications of marker-based augmented reality*. 2012.
- [26] Alexandro Simonetti Ibañez, Josep Paredes Figueras, et al. Vuforia v1. 5 sdk: Analysis and evaluation of capabilities. 2013.
- [27] Didier Stricker, Gundrun Klinker, and Dirk Reiners. A fast and robust line-based optical tracker for augmented reality applications. In *Proceedings of the international workshop on Augmented reality: placing artificial objects in real scenes: placing artificial objects in real scenes*, pages 129–145. AK Peters, Ltd., 1999.

- [28] Natsuki Sugano, Hirokazu Kato, and Keihachiro Tachibana. The effects of shadow representation of virtual objects in augmented reality. In *Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium on*, pages 76–83. IEEE, 2003.
- [29] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Combining edge and texture information for real-time accurate 3d camera tracking. In *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*, pages 48–56. IEEE, 2004.
- [30] Harald Wuest, Florent Vial, and D Stricker. Adaptive line tracking with multiple hypotheses for augmented reality. In *Mixed and Augmented Reality, 2005. Proceedings. Fourth IEEE and ACM International Symposium on*, pages 62–69. IEEE, 2005.
- [31] Suyu You and Ulrich Neumann. Fusion of vision and gyro tracking for robust augmented reality registration. In *Virtual Reality, 2001. Proceedings. IEEE*, pages 71–78. IEEE, 2001.
- [32] Feng Zhou, Henry Been-Lirn Duh, and Mark Billinghurst. Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 193–202. IEEE Computer Society, 2008.

Curriculum Vitae

Name: Nelson Andre

Post-Secondary Education and Degrees: The University of Western Ontario
Honors Bachelor of Science
Double Major - Computer Science, Microbiology and Immunology
2008- 2012

Honours and Awards: Western Gold Medal
Major in Microbiology and Immunology Module

Julian Davis Silver Medal
Computer Science

Related Work Experience: Teaching Assistant
The University of Western Ontario
2012 - 2013