A Qualitative Evaluation of Visual Localization and the Application to Robotics using ARToolKit and the Lego™ NXT

Submitted in partial fulfilment
of the requirements of the degree of

Bachelor of Science (Honours)

of Rhodes University

Ryan James Davey

Grahamstown, South Africa

November 2009

"My robot collects information about its enviroment,
then discards it and drives into walls"
ANONYMOUS

Abstract

The research presented by this dissertation describes an investigation into the application of visual localization through stationary cameras and subsequent evaluation and analysis on the concept and technologies involved. Proposed was a system that consisted of several modules all preforming different actions. A visual module using ARToolKit+ to capture and scan camera feeds for fiduccial markers. A Localizer that uses geometrical mathematics to localize and determine the position of those markers and a robotic module that facilitates control to a small robot.

A implementation was attempted and while incomplete provided many opportunities to assess visual localization and the applications robotics. What was created was a visual scanning module that uses the ARToolKit+ to scan for the position of fiducial markers. The position of the markers was trigonometrically determined using established mathematics. DSS as included in Microsoft Robotics Studio was also assessed and evaluated for creation og the robotic module.

The project itself proved to be far more complex than the problem statement may have suggested and the scope of the project was large, despite the complications, a concept was tested and a different approach to the problem of visual localization and robotics evaluated. The results were qualitative and it was found that the visual recognition has its advantages and while is effective is not reliable.

ACM Classification

I.2.9 Robotics

I.2.10 Vision and Scene Understanding

I.3.5 Computational Geometry and Object Modeling

I.4.1 Digitization and Image Capture

Acknowledgments

I would firstly like to thank my supervisor, Dr Karen Bradshaw for her guidance and support throughout the year. Thank you for always putting me back in the right direction. I would also like to thank Richard John Barnett for his help and advice, I could not have managed without that support.

I would also like to thank my parents who have provided unconditional support all these years as well as all my friends, who have been great pillars of strength this year.

Lastly I would like to acknowledge the financial and support provided by the Department of Computer Science at Rhodes University, Telkom South Africa, Comverse, Tellabs, STortech, OpenVoice, Mars Technologies, Amatole Telecommunication Services, Bright Ideas Project 39 and the Department of Trade and Industry, all partners and sponsors through the Telkom Centre of Excellence at Rhodes University.

Table of Contents

List of Figures

Chapter 1

Introduction

In recent years mobile computational devices have become common technology and with the release of reasonably priced robotic kits there has been increased design and research into mobile robotics and devices. Since these devices are becoming more common there is pressure to change the perceived conceptions regarding robotics and the development of robotic systems.

Previously robotics systems were inaccessible, or complicated, being implemented in lower languages close to the hardware level. Recently it has become possible to develop on robotic platforms using modern object orientated or scripting languages.

Since many robotic systems rely on a multitude of human-computer interaction modalities, across many different human-machine interfaces, including tangible remotes, voice commands and pointing visual interfaces.

Thus in this project a different approach is taken and we will specifically explore a model of vision based robotic direction using the Lego^{TM} NXT.

1.1 Problem Statement

Relaying of commands to robotic systems is often a complex and unintuitive task for users to understand, often requiring extensive training or tutorials. One such complication of commands for robotic systems is that such commands are often positionally context sensitive. The controller is required to know the location of any robotic system and be capable of issuing commands to the robotic system to achieve the desired task. Recent research has been directed at the

development of autonomous systems capable of navigation of autonomous vehicles in different environments. One of the major problems in the development of self navigating autonomous systems is the establishment of positional information for the said system [17]. This constraint has lead to stunted growth of development in this area. Thus the demand is for dynamic methods of detection of positional information relative to a mobile device while both stationary and in motion as well as for software that is capable of interpreting and using such information as data for the generation of commands and instructions for a robotic system.

1.2 Research Objectives and Outcomes

The empirical goal of the project is to develop a system capable of controlling a small robot by combining localization and direction into a single stream-lined step that removes such concerns from the user. In a traditional "remote control" based system the user would have to maneuver the robot to the position where an intended action needs to be preformed. The aim of the project is to provide an interaction modality where a user can convey the action to be performed as well as where it is to be performed in one step. The system itself provides the methods and algorithms as well as interfacing with the robot which will lead to the evaluation of the technologies involved and the assesment of visual localization in the field of robotics.

1.3 Document Structure

This dissertation begins in Chapter two by looking at previous works completed in the areas of visual technologies and robotics as well as introducing concepts that are often used in the area. The chapter introduces the challenges faced by visual systems as well as methods employed to reduce the interferences and weaknesses experienced by the technology. Also explored is the concept of three dimensional localization, or the representation of actual position computationally and subsequent use of such technology. The mathematical methods triangulation and trilatigation are explored and briefly explained. Past work into combining robotics and visual technologies are explored and the robot-vision paradigm is deconstructed and challenged. Tools and concepts for robotic control are also discussed.

Chapter three introduces and deconstructs the high level conceptual design of the system. The actions performed by the system are broken into different modules that approach the required objective independently and provide the information needed by the next module to perform its

function.

Chapter four discusses the implementation of the project in addition to platform decisions as well as challenges experienced. The use of the ARToolKit+ and frame capture is discussed, the configurations needed for such a system and the challenge of integration of legacy technology into a modern system implemented on a more recent development platform. The mathematical methods needed for the localization module are explained and implemented. The chapter then discusses the communication between the camera modules and the localization module. DSS services are explained and the implementation of such service based systems is explained. The code and implementation decisions for each module are provided and explained.

Chapter five evaluates the system and discusses the implications and considerations of the technolgies involved.

Chapter six summerises the project and presents conclusions and suggest future advancments. The chapter also revisits the project problem statement.

Chapter 2

Background Reading

2.1 Introduction

There has been a definite movement in recent research towards systems that are capable of interaction and interpretation of their environment, the focus then falls on the interaction techniques that are necessary.

Technology is moving away from the traditional interfaces that have been proposed in the past. In a robotic system there is a definite focus and need for the integration of the interface in to the environment. It is necessary to try and design interaction techniques that make it easy to interact and specify the position in the environment where a task needs to take place.

To achieve this new technologies need to be developed and explored particularly in the area's of three dimensional position management and objects and device detection. The focus of this project is the integration of these technologies.

2.2 Interpretive Vision Based Technologies

With the development of high resolution cameras capable of high frame rate capture and the increase in standard CPU power to process image frames in real time has really encouraged the development of systems capable of interpreting visual information as input and generating responses to such inputs.

The use visual camera tracking technology can been seen in many compelling systems like Fails' light widget [8] as well as the "eyePliances" that were suggested by Shell [31], Appliances that respond to visual information gathers from user attention. Work by Starner [32] has been on a inferred "gesture pendant" that was designed for control of an automated home and medical monitoring. The advances in visual recognition technology and motion capture have been applied to computer gaming with Sony EyeToy^{TM}[19]

One of the major challenges of visual capture implementations is the real time tracking of moving objects. Work by Daniilidis [7] showed that by the implementation of Kalman estimators and linear Quadratic regulators, active sensing can be achieved and a recognised pattern tracked from real time monocular camera feeds.

Demonstrated by Wang [33] was DROID analysis that uses visual motion to capture corner features of the image to achieve three-dimensional reconstruction. Harris postulated by the use of Kalman filters to track and for locations for image features, DROID could determine both the camera motion and the three-dimensional position of the image features. This, labelled as "ergo-motion" was considered a very accurate method of extracting positional information of images. But due to the limitation of the DROID system being monocular there were still a self-consistent error in the perceived three-dimensional structure.

Kato and Billinghurst [16] developed a video conferencing technique that utilized fiducial markers to track position of visual outputs, the vital tool being the generation of a transformation matrix relating camera position and marker position, thus allowing the camera itself to be used as reference for visualization of augmented reality [16]. The work signified the importance of visual tracking techniques as a step in the translation of visual input information as a interaction technique. Work of this nature later contributed to development of light detection interaction techniques as well as tracking for the marker being important for the handling of occlusion between the generated reality and actual reality [10] .

The take home point being that there has been a definite move towards systems that use and base their operation on visual inputs and tracking of visual elements. Thus supporting the proposed system of a visual marker based robotic system using visual elements to support it's operation and facilitate communication.

2.2.1 Visual Recognition

Vision based systems that require recognition of visual elements need to make a distinction between appearance based and pattern matching [24]. The importance being the trade off between performance of generating statistical probabilities that the visual element is indeed in the video frame(or image) being analysed as well as algorithmic complexity of matching to a stored pattern the latter being more likely to generate a false negative.

Pattern Matching

Pattern Matching emphasizes the use of shape and grid models to recognize three and two dimensional objects. A grid model or "pattern" is stored in memory and algorithmically searched for. Advantages to this approach is a lesser chance of a false positive, but motion tracking becomes challenging as changes in orientation of the object can make recognition improbable [5, 6].

Appearance Based Detection

Appearance based visual recognition is a method where the internal processing of the brain is attempted to be modelled computationally. An object is assigned properties (like shape, colour, lighting and texture) that can be visually determined. The properties are then searched for and factored statistically into a probability that the element appears. Appearance based has an advantage in that the system becomes more resilient to dynamic changes to the environment, but smart computer appearance based recognition routines are yet to be developed [24].

2.2.2 Stereo-vision

Stereo-vision is the most common method of attaining three dimensional information of an environment through a visual interface. Marrs and Poggio proposed a algorithm for visual localization where every image is filtered at different orintations [21]. This approach is supposed to mirror the human neuro-physical adaptations for stereo-vision and depth perception which has become the convention when designing a system that relies on vision and computational interpretation of visual elements.

Building on the work by Marr and Poggio [21] Grimson [13] proposed matching and computer

vision algorithms that use the stateful position of the "eyes" in respect to the environment as well as the preform transformation steps while attempting to match the detected images.

2.2.3 Computer Range Determination

The important implication of stereo vision techniques is the ability to add computer depth and distance determination to applications and systems. Jarvis [15] makes the distinction between instrument based and image based passive range determination. The assessment of different range determination methods are outside of the scope of this project except one of interest being the simple triangulation range finder [15] which utilizes light from points of interest to calculate position. This method has been easily extended for other computer vision applications particularly mobile robots using multiple cameras like Matthies' planetary rover [22].

Another less discussed and explored method of computer range determination is called dead reckoning, which works by the employing devices like inertial sensors or accelerometers to determine the displacement from a starting point. The model always has some inaccuracies over long distances as devices do have inaccuracies and the are subject to external factors manipulating the the system in methods not observable to the sensors. Thus this method of localization is often depreciated to preforming corrections in highly error prone localizations, like experienced on GPS devices or in virtual environments like multiplayer games over a network [28].

2.3 Computational Device Localization

The concept of computing devices being aware of position in a environment is commonly being referred to as computer localization. Which is starting to be utilized in many different areas of computing. The idea that a computer can be context aware as to it's current position either relative to other computing devices as seen in the creation of mobile ad hoc networks [23] or in the mobile phone and telephony industry [3] where the real time tracking of device location is a vital concern.

2.3.1 Mathematics of Localization

With a definite movement towards real time tracking of devices, several classical mathematical methods have been established and popularized utilizing geometrical properties that can be

applied to reality. The methods discussed below are Trilatigation and Trianglation.

Trilatigation/Lateration

Trilatigation is a method for finding the intersection point of 3 or more circles, given the centres of the circles and the radii of these 3 spheres [25, 35].

$$(x-a)^2+(y-b)^2=r^2 \tag{2.1}$$

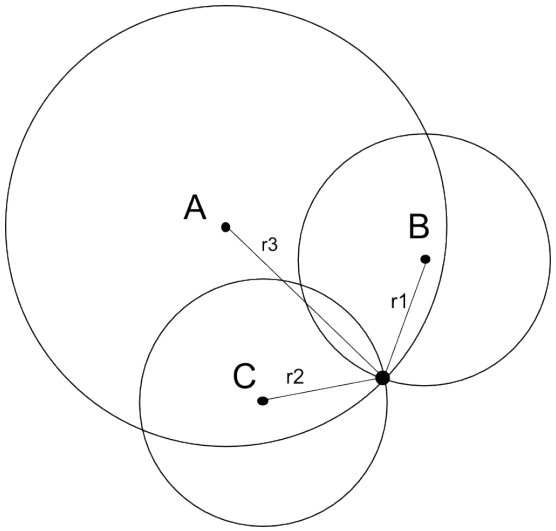

Figure 2.1: Simple example of mathematical trilatigation where a point is determined by solving for the point of intersection of three circles

The mathematical solution for a trilatigation problem is to take the formula (shown in equation 2.1) for three different circles and solve them as equal to each other, and providing they are not overlapping there is mathematically only one point of intersection as shown in simple example figure 2.1 [35].

Triangulation

Triangulation is the mathematical method of finding the position of a point of interest give the angles between the desired point and two points of reference of which the distance between them is known [35].

Equation 2.2, which employs the use of tangent to calculate the result.

$$d = \frac{l}{\left(\frac{1}{\tan\alpha} + \frac{1}{\tan\beta}\right)} \tag{2.2}$$

Equation 2.3 demonstrates the law of sines that can also be utilized for the determination of the result.

$$d = \frac{l \bullet \sin\alpha \bullet \sin\beta}{\sin(\alpha + \beta)} \tag{2.3}$$

Figure 2.2 shows a simple example of mathematical triangulation to which equations 2.2 and 2.3 can easily be applied.

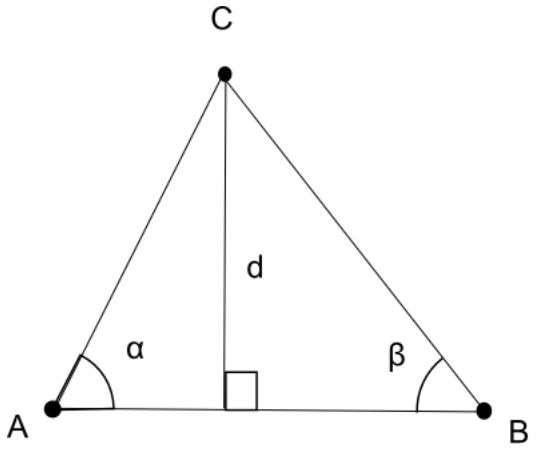

Figure 2.2: Simple example of mathematical triangulation where a point is determined using two known angles and a known side length

2.3.2 Inertial Systems

Inertial systems often follow two modes of implementation, Mechanized-Platform systems and strap-down systems [9] The purpose of an inertial system is to add additional information like velocity, displacement or acceleration to a position determination and usually supports GPS.

In mechanised-platform systems the platform maintains alignment with the navigation coordinate system by maintaining gyro outputs in an attempt for the platform to resist rotation relative to the frame, in spite of vessel motion. Applying scaling vectors and integrating can provide acceleration and velocity vectors as well as the altitude of the vessel relative to the navigation frame by measurements of the relative angles [9].

Strap-Down Systems have the inertial sensors attached to the moving vessel thus experiencing the full dynamic displacement of the vessel. This has the advantage of being a simpler implementation but more has to achieved computationally as the gyros need a greater dynamic rage and are prone to greater error [9].

2.3.3 Wireless Localization

Since there are many application that now require locational information of wireless sensor devices there has been much advancement in the algorithms and technology needed to develop and manage wireless devices that are capable of positional determination [20]. Much of this is done through signal strength analysis by taking readings from 3 different wireless receivers to locate a forth [35]. Or in a case where the sensing doesn't need to be as sensitive and the proximity to a certain point is more important (often referred to as location based services).

Global Position System

The GPS satellites and the receivers that use them are based on the the model problem based in figure 2.1 except the more complex case of 3 dimensions. The Global Positioning system utilizes the signal strength from the satellite and estimates the distance from at least 3 nearest satellites then solves for the point of intersection of the spheres [34, 9].

The system uses the simplification that the receiver is aware that it itself exists on the sphere plane of the earth to aid the localization [9] but modern GPS receivers often use inertial (described below) and altitude sensing to achieve greater resolution of positional information and increase effectiveness of the device.

Ad-Hoc Sensor Networks

Ad hoc networking is wireless networking with wireless devices communicating over possibly multiple hops without the aid of base stations or external coordination. Ad Hoc networks use distance vector routing, shortest path determination and signal strength determination to efficiently route packets through a network [14].

The challenge is to choose close nodes for routing of packets that provide excellent serviceability but keep the numbers of hops between nodes to a minimum [18].

Monte Carlo Localization

Monte Carlo Localization is a statistical uncertainty function where a set of samples are maintained and compared to a map in an effort to locate a mobile device. The method uses probability densities in a effort to represent arbitrary distributions [11]. While statistical methods are necessary for the accurate localization of a device, they are beyond the scope of this project.

Robotic Wireless Localization

One of the major challenges in modern robotics is determining position of the device [17]. Thus many have found implementations using various wireless technologies to approach the problem [12].

Most robotic implementations are designed for the outdoor environment where inexpensive GPS is employed with inertial sensing as demonstrated by Panzeiri [29] where the GPS is used for large scale localization and the inertial sensor accounts for small scale motions using dead reckoning (which is a method employing geometrical equations to determine position from a starting point).

Other implementations like those employed by Ladd [17] where 802.11 wireless Ethernet is used to preform wireless localization on a small robot using 3 reference "beacons" . This lead to consideration of a project to achieve the same result with the Bluetooth standard, but the impartibility of such an undertaking will be discussed in Chapter 4.

In Ladds implementation the weaknesses of wireless systems were explored and the need for the application of probabilistic inference algorithms to reduce the noise of the receivers, as the 802.11 standard is not entirely accurate and does experience external interference from the environment.

2.3.4 Visual Localization

With the advent of high resolution and high frame rate cameras the trend has been towards the use of visual methods of localization for mobile devices. Thus it has become necessary to develop robust methods of algorithmic cognitive visual recognition and tracking, with the purpose of determining an objects position based on the objects position relative to the environment at large and do this accurately in a domain of research where results from the visual sensors are

"fuzzy" at best.

One of the methods of localization is landmark based, in that a the visual receptors are programmed to remember different visual elements like appearance based objects or visual features, the problem with this implementation is the need for mapping as the focus is on systems where the frame of vision regularly changes[30].

Nakazato suggested that markers could potentially be used to help mobile devices localize users dependent on the detection of the marker by a infra-red camera. This implementation was more robust than landmark sensing as the markers can be easily pre-programmed and defined in the system. Markers are easier for a system to detect with certainty [26, 27, 16].

2.4 ARToolkit

Tracking of fiduccial markers and the pattern matching techniques and algorithms that are needed are complex and difficult to implement.

The ARToolKit was designed as a extendable multiplatform framework for the tracking and identification of fidduccal markers on frames captured from visual feeds [16, 1]. Thus the aim was to abstract away from the complexity of tracking the user viewpoint and the recognition of the marker.

The ARToolKit supports multiple patterns for different markers and is built on a simple graphics library which is based on GLUT. This allows for graphics support in augmented reality applications.

2.5 LEGO^{TM} NXT

The Lego ^{TM} NXT is a programmable robotics kit created by Lego corporation. The kit itself supplies all the parts and components needed to build fully functional robots with sensor that simplistically mirror the human senses.

Supplied in the kit is the programmable *Brick* which is the central processing unit that contains a 32-bit AT91SAM7S256 main microprocessor at 48 MHz and a 8bit ATmega48 at 4MHz. The brick itself has 3 output ports and 4 input ports for different sensors or motor devices. The brick

also contains a class 2 Bluetooth device for communication [2].

2.6 Chapter Summary

In this chapter past work and reaserch was explored with the intention of gathering the knowlege needed to approach the problem domain as well as assess the the need for work in the area in general. The mathermathematics that are often used in localization (Triangulation and Trilateration) were discussed and explained and it was found that for visual localization triangulation was most appropriate. The chapter then explores visual technologioes and the appropriateness to robotics. The next chapter will propose a design to meet the problem presented.

Chapter 3

Design

3.1 Requirements of System

The proposed system was designed with the following considerations in mind.

3.1.1 Intuitiveness

It is a requirement that any interaction system be intuitive and easy to use. Should a system become complicated to use the scope of the device's user base becomes narrow. The goal of any user focused system should be to reduce task time as well as allow new users to learn the modality of the systems input rapidly and adapt to using it, whether through experimentation or streamlined instruction [4].

3.1.2 Robust and Responsive

The system needs to be robust and unbreakable. The preface being that an error state is a problem in a system in which contextual interaction with an environment is expected. Robotic systems that are unresponsive can be dangerous and unreliable, as well as add to user frustration. The system needs to react quickly to allow the user to know that the human system communication was successful.

3.1.3 Resolution, Precision and Accuracy

The system needs to be accurate and have decent resolution, precision and accuracy to be effective.

Resolution

Resolution is the smallest difference between two measurements that can be detected. Sometimes processing truncates the measurement. Sampling is also proven to improve resolution. Any system that relies on any form of detection and filtering is affected by detection resolution.

Precision

Precision is difficult to singularly define, taking a different meaning based on context. For the purpose of this study, precision will be considered the range of possible output values for one input.

Accuracy

Accuracy is an independent measure from resolution and precision. The ability to distinguish between two different values does not make a system accurate. Accuracy is a conclusion drawn from the system's determined value and the actual value.

3.2 Overview of the System

The proposed system that addresses the problem presented in Chapter 1 is depicted in Fig. 3.1 and described below.

The camera's are placed above the scene to provide stereo vision of the environment and locate both the markers and the robot.

The use case depicted in Fig. 3.2 shows an overview of the proposed system action. Empirically the system follows the following algorithmic procedure.

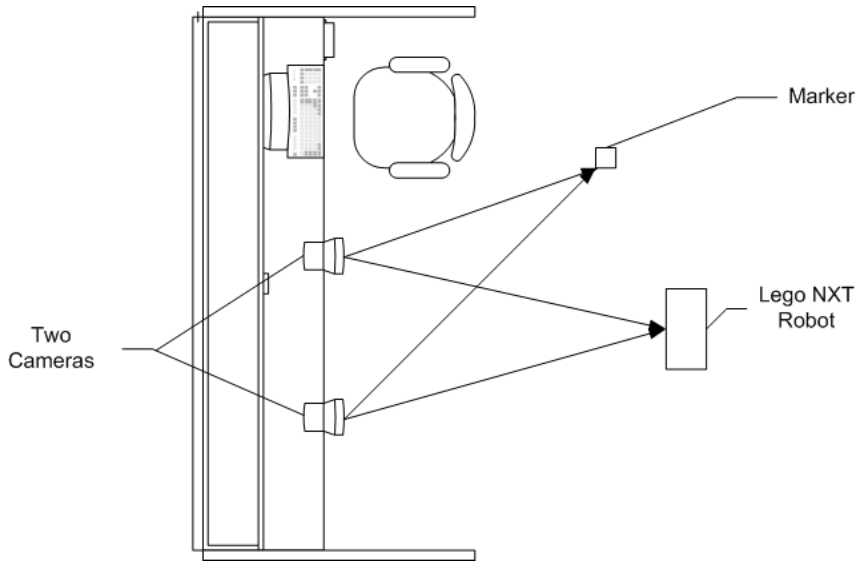

Figure 3.1: System design and set up, displaying general hardware requirements.

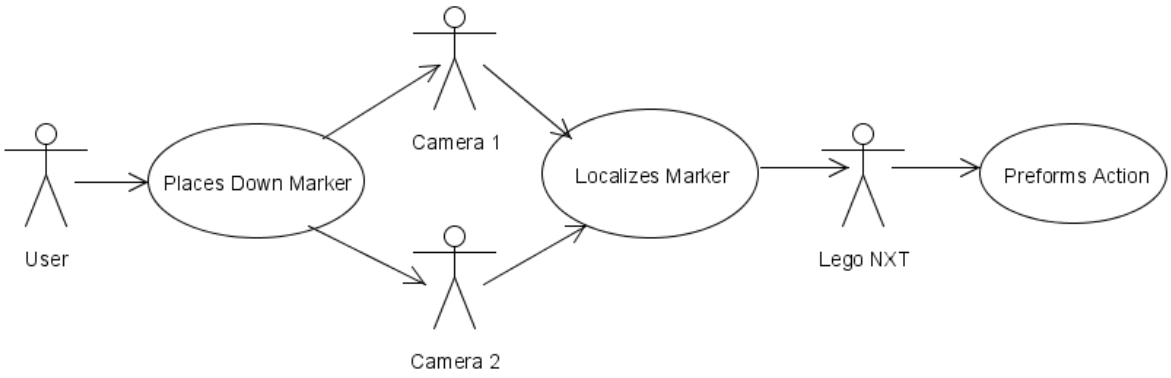

Figure 3.2: General use case of the proposed system.

1. The system uses camera feeds to track and assess the position of a small robot in an environment.
2. The system then evaluates the location of the markers on the robot in the environment.
3. The system then enumerates the markers and the intended system side “meaning” thereof, potentially contextual commands on the robot in question.
4. The user can then place a marker (example being a GoTo marker) on the environment.
5. The system detects the placement of the marker.
6. The placed marker position is evaluated and localized.

7. The system then selects robotic methods (from available robotic libraries) to execute action determined by marker.

8. The robot moves to the marker and executes the action in question.

In the next few sections the design and requirements of the different subsystems of the project will be deconstructed. The larger system is itself divided into smaller modules, where a module in this project is considered to be a collection of functionality or methods performing a service for another module.

3.3 Visual Capturing

The visual capturing forms the first component of the larger system.

The requirements of the visual tracking module are as follows:

- Capture and manage multiple camera feeds.
- Experience no slow down with the evaluation of multiple camera feeds.
- Provide accurate visual recognition.
- Evaluate visual elements on the captured feeds.
- Provide a suitable output for the next component.

Fig. 3.3 displays the use case of the visual module. The design specification requires that a module be designed that can evaluate multiple camera feeds for markers in the frames captured in real time by the camera. The challenge of such a system is to achieve real time performance that can provide robust accurate visual recognition.

3.3.1 Appearance-Based versus Pattern Matching

Any system that relies on visual inputs needs to make a high level design decision on the type of visual recognition that is required. In this system the design calls for visual cues that are user based, and thus a pattern matching algorithm can be employed. The advantages of a pattern

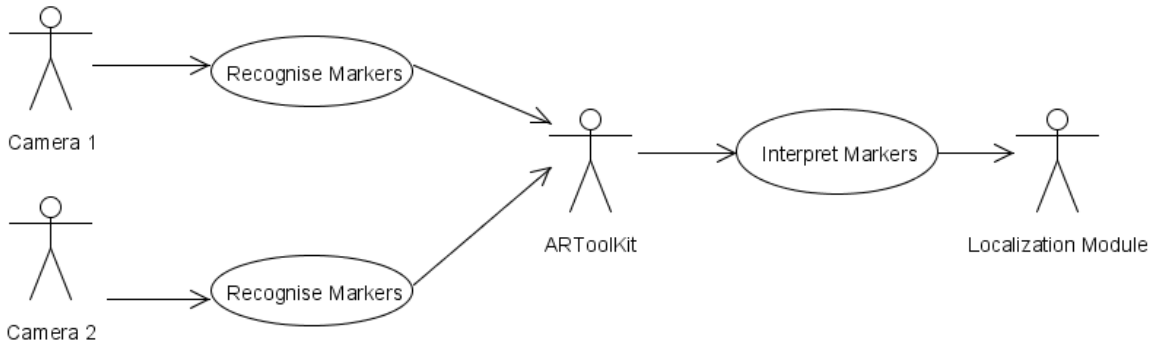

Figure 3.3: Use case of the visual capturing module, showing the intended actions and the acting localization module.

matching solution are given below; the differences between the approaches are discussed in Chapter 2.

- The recognition is fast and robust.
- The artificial intelligence algorithms involved in searching for and comparing a stored visual element are less complex and faster than the appearance based algorithms that rely primarily on statistical probability.
- Having a pre-generated or determined pattern allows for more robust control in the programming and implementation phase. This is explored in greater detail in Chapter 4.

Now all that remains is to explore the design concept of a "marker" in the context of this system

3.3.2 Concept of a "Marker"

A marker in this system can be deconstructed as a computationally recognisable element that can be recognised from a captured frame from a camera feed. The chosen design specification is to adopt markers that are black and white fiducial markers (as depicted in Fig. 3.4). The consideration for this design choice is that the black and white fiducial markers are easy to create, and easy to replace or redefine. Another consideration is that there are software libraries that manage the recognition and computational interpretation of these markers, such as the ARToolKit, which became the conceptual basis for the design of the ARToolKit+ that was employed in this project. This toolkit effectively extends the original toolkit.

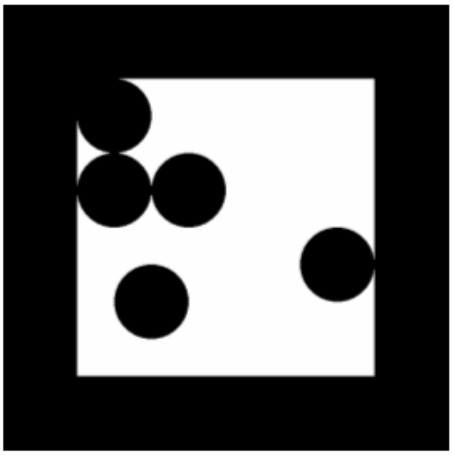

Figure 3.4: Example fiducial marker that is designed to be recognized visually for use in an application employing computer vision

3.3.3 Final Requirements of the Visual Capturing Module

The visual capturing module needs to capture camera feeds and assess them for fiducial markers. The module then needs to compile information about the detected markers into a format or form that can be used by the next module in the system that will be responsible for localizing the markers. The localization module or subsystem is discussed in the following section.

3.4 Two-Dimensional Localization

The localization module forms the correlation between the actual system and the environment. The function of this module is to provide a 3-dimensional position within the environment based on the output produced by the visual capturing module.

Fig. 3.5 shows the high level action of the localization module.

To provide accurate localization the following requirements need to be met.

- The localization needs to be accurate.
- The module needs to produce output that can be interpreted by other components in the system.
- The module needs to take into account the nature of current visual technologies.

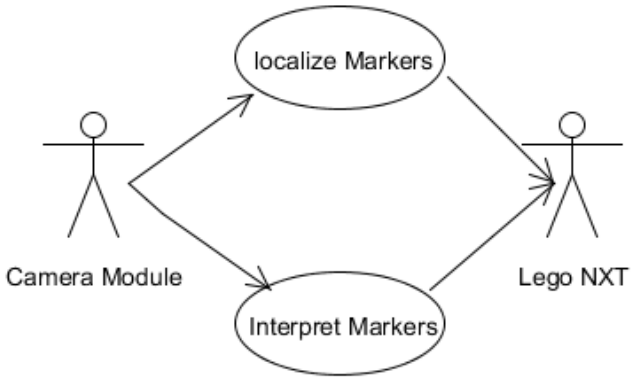

Figure 3.5: Use Case diagram depicting the high level action of the localization Module

- Mathematical methods need to be selected that support the technology employed.

3.4.1 Application of Mathematical Methods

In Chapter 2 the mathematics of localization was briefly discussed. While the deeper mathematical theory of applying understood methods to visual processing forms an in depth study in its own right, such evaluation is beyond the scope of this problem domain and for purpose of proof of concept the simple case will be considered as discussed in Chapter 2 and 2.2.

Thus since the design employs two visual feeds from different perspectives the simple triangulation model is appropriate as depicted in the diagram in Fig. 2.2 where Points **A** and **B** are the sources (cameras) for the visual capture.

3.4.2 Final Requirements of Localization Module

Algorithms need to be produced that can be called to localize the position of markers in the visual feeds. The module also needs to interpret the detected markers and communicate that information to the rest of the system.

3.5 Robotic Direction Module

The robotic control module is the final component in the system. The module provides the basic services (for the purpose of the project, additional services would be preferred in a project of larger scope) depicted in Fig. 3.6. The purpose of the module is to provide connectivity to the robot and provide an abstracted interface for providing commands and communication to the robot itself.

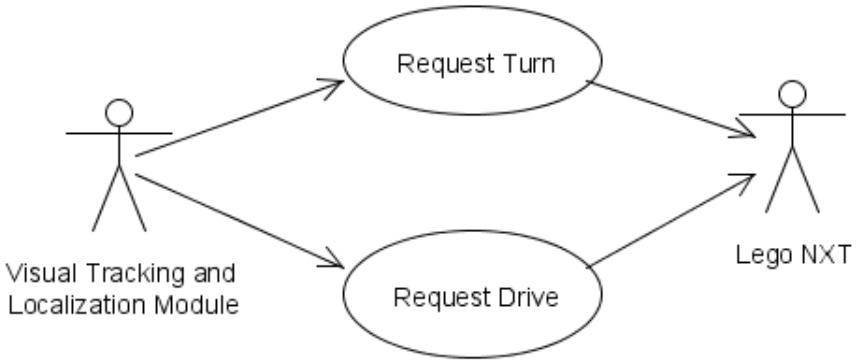

Figure 3.6: Use Case of the Robotic Control Module displaying Robot Action Cases.

3.5.1 Required Methods

In initial evaluation of the functionality needed by the system several robotic methods (essentially services) are required by the system at large.

Turning Algorithm

For the functionality of the system a turning algorithm is needed that can accept a degree of rotation argument that instructs the robot to turn to face a specified direction.

Drive Algorithm

A drive method is also fundamental. This method needs to accept an argument that instructs the robot to move forward a certain distance.

3.6 Final System Design and Module Relationships

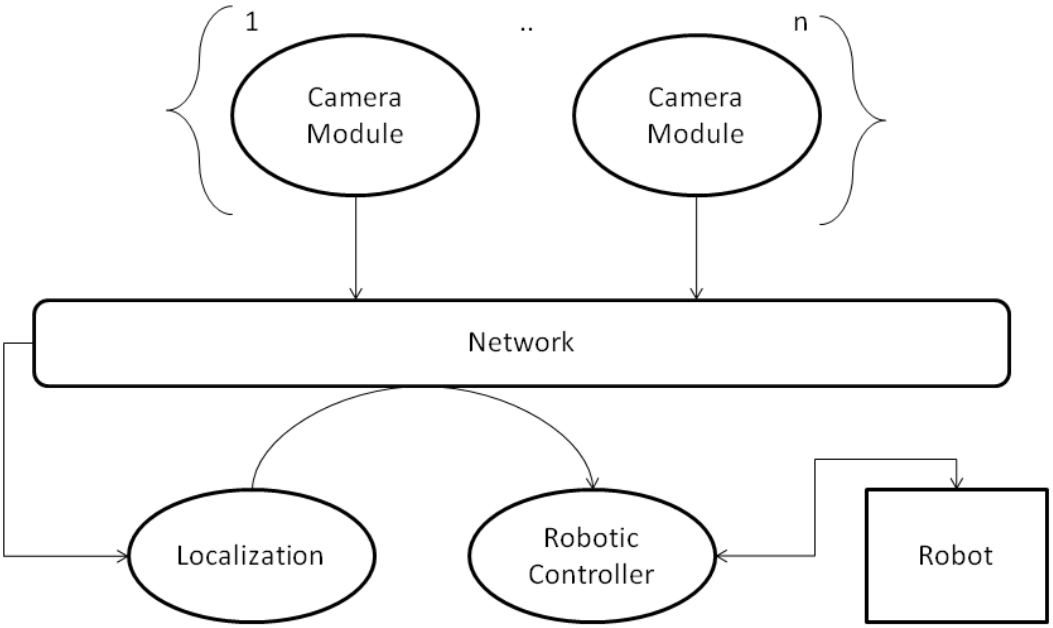

Figure 3.7: System design implementing networking to facilitate interprocess communication.

Conceptually the design of the system has been divided into multiple domains of functionality, thus the system can be constructed as a sequence of processes that embody methods or provide services to the system as a whole.

These services can therefore, be offered over the network as shown in Fig. 3.7.

3.7 Chapter Summary

In this chapter the requirements and design of the system was discussed and the problem broken apart into three different systems. The implimentation of these systems will be discussed in the next chapter.

Chapter 4

Implementation

4.1 Introduction

The previous chapter outlined the design of the system, describing the modules and the roles they play in the larger design. This chapter will delve into the implementation of the system and work completed towards reaching the proposed design goals. In this chapter we look at each individual module that makes up the project and its implementation.

4.2 Development Platform and Software Libraries

For the project the chosen platform was Microsoft technologies using C# with a direct focus on the intercommunication of the modules using socket based networking. The Microsoft .net 3.5 C# framework has all the tools needed to complete the project.

Various other languages that contained defined robotic API's, were also considered, such as Java, Python and C/C++ (RobotC and pure). These languages were considered to be unsuited as the entire scope of the project would either be impossible or ultimately too complex to implement in the rejected languages. Another factor of consideration was the Microsoft Robotic Studio R1 that was released in October 2008. This framework and it's features will be discussed in a later section in this chapter.

Figure 4.1: The Logitech QuickCam E3500 Plus (picture from logitech website)

4.3 Hardware and System Recommendations

This section covers the specific hardware unique to this project. The project itself involved interfacing with a multitude of hardware devices.

4.3.1 Logitech QuickCam E3500

The cameras used are Logitech QuickCam E3500 which are affordable standard USB web cams capable of capturing up to 960 x 720 pixels at up to 30 frames a second.

The cameras were set up as depicted in Fig. 4.2 with a distance of 15 cm between them to provide a suitable angle for the localization.

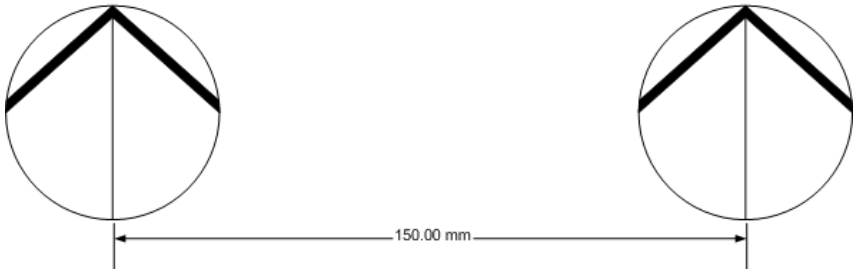

Figure 4.2: Two cameras placed 150 mm apart.

4.3.2 Lego NXT TriBot

The Lego NXT robotic kit, described in Chapter 2, was converted into a slightly modified TriBot design. The original Tribot as illustrated in Fig. 4.3.

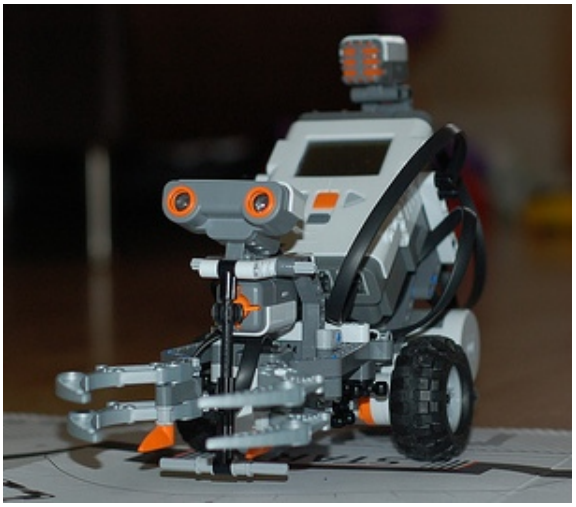

Figure 4.3: The standard Lego Tribot (picture from the net)

The design used in the project is depicted in fig. 4.4. The claw apparatus was removed and the ultrasonic sensor replaced with a touch sensor that operates as a button, the use of which will be explained in the discussion on the robotics DSS service.

Figure 4.4: The modified Tribot design used in the project

4.3.3 D-Link 122 Bluetooth Dongle

The final piece of hardware to be introduced is the D-Link DBT-122 Bluetooth dongle shown in Fig. 4.5), which is a standard Bluetooth V1.2 device that works with the Lego NXT robot. After investigating many Bluetooth V2.0 devices, the D-Link device wasfound to be the most compatible of the available devices that used the standard Windows Bluetooth stack.

Figure 4.5: The D-Link DBT-122 Bluetooth Dongle

4.4 ARToolKit+ and Visual Interface

In the previous chapter we discussed the ARToolKit and the need for tracking markers. In this section, we discuss the implementation and configuration of the ARToolkit.

4.4.1 C# Wrapping of ARToolKit+ and DirectShow

The ARToolKit+ is a C++ library. Since we have chosen to use a more modern programming language(C#) it was necessary to employ a wrapper for the legacy C++ system. Such a wrapper was found, credited to Mr. Casey Chesnut (`http://www.brains-n-brawn.com`). The wrapper itself takes a C++ library compiled into a DLL and then pinvokes methods that are offered by that DLL, The wrapper created by Mr. Casey was found to be sufficient as not all methods offered by the ARToolKit+ were needed in this project.

```
1 /// <summary>
2 /// ARToolKitPlus multi sample
3 /// </summary>
4 [DllImport("ARToolKitPlus.dll")]
5 public static extern int fnARTKPWrapperMulti();
```

The above code listing extracted from *ArManWrap.cs*, shows the simple *pinvoke* wrapper, to allow access to the method in the DLL, in this case, for multi marker tracking. The `extern` keyword is a C# reserved word indicating the method exists in an external DLL file that is imported using the `[DllImport()]` statement.

```
1 /// <summary>
2 /// Loads a camera calibration file and stores data internally.
3 /// To prevent memory leaks, this method internally deletes any existing cameras.
4 /// If you want to use more than one camera, retrieve the existing camera using getCamera
      () and call setCamera(NULL);
5 /// before loading another camera file. On destruction, ARToolKitPlus will only destroy
      the currently set camera.
```

```
6  /// All other cameras have to be destroyed manually.
7  /// </summary>
8  /// <param name="tracker"></param>
9  /// <param name="nCamParamFile"></param>
10 /// <param name="nNearClip"></param>
11 /// <param name="nFarClip"></param>
12 /// <returns></returns>
13 [DllImport("ARToolKitPlus.dll", SetLastError = true)]
14 //ARTKPWRAPPER_API bool ARTKPLoadCameraFile(ARToolKitPlus::Tracker* tracker, const char*
       nCamParamFile, float nNearClip, float nFarClip)
15 public static extern bool ARTKPLoadCameraFile(IntPtr tracker, string nCamParamFile, float
       nNearClip, float nFarClip);
```

The above code segment enables the ARTooLKit to take in the parameters and configurations for the camera. We can see this is wrapped in a very similar way, despite being a more complicated method.

Disadvantages of Code Wrapping

Although using algorithms written in older languages may seem attractive for the reuse of code and are considered to be a valuable "time saver", there are a few negative issues. Other than the small performance hits in the overhead needed for method invocations through the wrapper and the underlying DLL calls, the major disadvantage is noticeable in the event that something goes wrong. C# wrapped DLL libraries are very difficult to debug, due to the abstraction layer. It was found that in C#, a language with more complex types than C++ (which was the development platform for the toolkit), type inconsistencies were reported when an exception was thrown in one of the DLL's inner methods. In fact, many methods had to be cast to more complex types for compatibility with the rest of the system. Essentially wrapper classes are written for use with error free code, which in a development cycle is highly unlikely.

4.4.2 Marker Selection

The ARToolKit provides support for several different marker types and sizes. The main distinction being whether the markers are trained or hard coded into the system itself.

Trained Markers

The ARToolKit supports a trained marker mode, where simple to complex high contrast markers can be visually trained for a system. The markers are trained using third party software and a *markers.cfg* file is generated containing the results of the training and the properties of the markers. This method allows for unlimited marker selection but does pose a problem in that often the training fails and such a failure is not immediately apparent. As the markers need to be trained in conditions (lighting and orientation) similar to those in which the system is ultimately going to be used, simple ID coded markers, offered by the ARToolKit, were favored.

Simple ID coded Markers

The ARToolKit+ provides new implementations of the types of markers that are accepted by the ARToolKit. Originally the markers accepted by the system had to be trained.

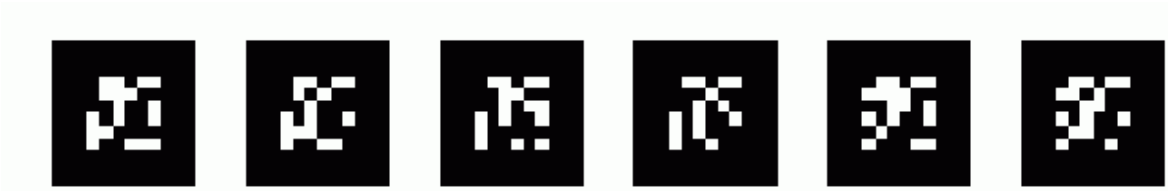

Figure 4.6: Five Enumerated Markers Used in the ARToolKit+, each is associated with a market ID in the ARToolKit+.

The toolkit offers five hundred and twelve preprogrammed markers, such as those depicted in Fig. 4.6, that have enumerated ids. There are two advantages of this approach; the training phase is eliminated and the markers can easily be represented electronically. This makes the identification of different markers much simpler for the programmer.

4.4.3 ARToolKit+ Initialization

Before the toolkit can be used a series of initialization steps needs to be taken, two of which are the importing of the camera configuration and the marker definition files.

Below is the camera configuration file, that contains all the calibrated resolutions and settings needed for the ARToolKit+ to use the supplied cameras.

```
1 ARToolKitPlus_CamCal_Rev02
2 640 480 330.27758683214108 228.10613100912309 891.32055276878066 888.496623076345940
      -0.049480033893318 0.215863227944058 -0.001997733892605 -0.003151872552003 0.0 0.0 10
```

Below is a sample marker definition file. The file defines the markers used by the system from a possible five hundred and twelve hard coded markers. The file also contains each marker's relative position to the adjacent marker (in the form of a transformation matrix). This allows the ARToolkit+ to define markers that may possibly appear out of view, or at an angle where the marker is difficult to determine, such as on the far side of the robot.

```
1  # multimarker definition file for ARToolKit (format defined by ARToolKit)
2
3  # number of markers
4  2
5
6  # marker 0
7  480
8  40.0
9  0.0 0.0
10  1.0000  0.0000 0.0000 -100.0
11  0.0000  1.0000 0.0000   75.0
12  0.0000  0.0000 1.0000    0.0
13
14 # marker 1
15 481
16 40.0
17 0.0 0.0
18  1.0000  0.0000 0.0000  -50.0
19  0.0000  1.0000 0.0000   75.0
20  0.0000  0.0000 1.0000    0.0
```

The ARToolKit+ is invoked in the code example below using a tracker object (from the AR-ToolKit+ Class) constructed in line 1 and `ArManWrap.ARTKPInitMulti()` is called to initialize the toolkit in line 3. The tracker object is unique to the cameras and the frame being captured.

```
1 tracker = ArManWrap.ARTKPConstructTrackerMulti(-1, dShowHelper.Info.BmiHeader.Width,
      dShowHelper.Info.BmiHeader.Height);
2
3 int initAR = ArManWrap.ARTKPInitMulti(tracker, Properties.Settings.Default.
      CameraParameterFile, Properties.Settings.Default.MarkerSetConfigFile,Properties.
      Settings.Default.MarkerWidth * 0.01f, Properties.Settings.Default.MarkerWidth * 100.0
      f);
```

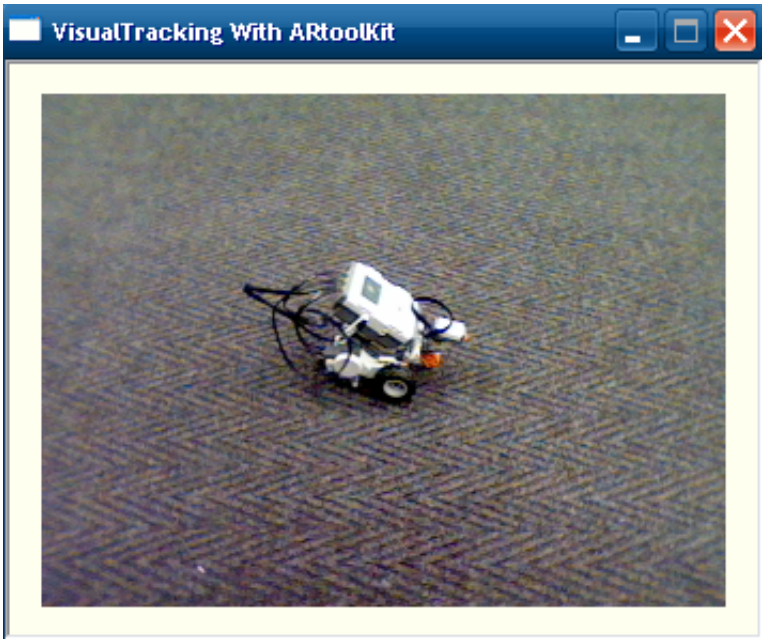

Figure 4.7: Windows Presentation Foundation Window Displaying Camera Feed

4.4.4 Capturing of Camera Feeds

Fig. 4.7 depicts a screen based view finder that also embodies the ARToolKit+ implementation. The live camera feed is intended to help the user position the cameras for use with the system.

Windows Presentation Foundation (WPF) viewfinder

The code segment below is taken from the .xaml file that forms the basic view finder. Lines 1 to 17 deal with setting all the parameters and configurations for the canvas, used to display the camera feed. The canvas is then mapped to a `MatrixCamera` object which is subsequently used for the Augmented Reality processing.

```
1  <Canvas Margin="12,13,12,12" Name="Canvas1" Grid.RowSpan="2">
2      <Viewport3D Name="Canvas13D" IsHitTestVisible="False" ClipToBounds="True">
3          <Viewport3D.Width>
4              <Binding Path="ActualWidth">
5                  <Binding.RelativeSource>
6                      <RelativeSource Mode="FindAncestor" AncestorType="{x:Type␣Canvas}"
                           AncestorLevel="1"></RelativeSource>
7                  </Binding.RelativeSource>
8              </Binding>
9          </Viewport3D.Width>
10
11         <Viewport3D.Height>
12             <Binding Path="ActualHeight">
```

```
13                  <Binding.RelativeSource>
14                      <RelativeSource Mode="FindAncestor"  AncestorType="{x:Type␣Canvas}"
                            AncestorLevel="1"></RelativeSource>
15                  </Binding.RelativeSource>
16              </Binding>
17          </Viewport3D.Height>
18          <Viewport3D.Camera>
19              <MatrixCamera x:Name="matrixCamera" />
20          </Viewport3D.Camera>
21      </Viewport3D>
22  </Canvas>
```

Windows Presentation Foundation is built on an XML styled expression language designed for rapid creation of graphical user interfaces.

4.4.5 Marker Detection and ARToolKit+ Output

Now that the system and ARToolKit+ have been initialized, we can look at the detection and interpretation of markers.

```
1  //detects markers from the current frame
2  int numMarkers = ArManWrap.ARTKPCalcMulti(tracker, imageBytes);
3
4  //obtains projection matrix!
5  float[] projMatrix = new float[16];
6  ArManWrap.ARTKPGetProjectionMatrix(tracker, projMatrix);
7  Matrix3D wpfProjMatrix = ArManWrap.GetWpfMatrixFromOpenGl(projMatrix);
```

The above code excerpt requests from the ARToolKit+ the detected markers and obtains the projection matrices from the camera.

```
1  IntPtr[] markerInfoPtr = new IntPtr[maxMarkers];
2
3  for (int i = 0; i < numMarkers; i++)
4  {
5          markerInfoPtr[i] = ArManWrap.ARTKPGetDetectedMarker(tracker, i);
6  }
```

ARToolKit+ composes the marker into a list that can be enumerated and accessed by the system. Thus certain markers can be selected and interpreted through a switch or for loop statement. Line 3shows the method call for returning a pointer to a certain marker. This can be used to access properties of the marker itself.

The information is then serialised into string form and can be sent to the next component of the system.

4.4.6 Threading versus Separate Processes

Since the proposed application requires the use of two cameras both using the ARToolKit+, a threaded solution was attempted, but due to the unsafe nature of the ARToolKit+ with respect to synchronization and threading, this was not possible. An alternative implementation resulted from the fact that the solution relies on a fixed number of cameras. We chose to package the visual tracking module as an executable that could be run the appropriate number of times (depending on the number of cameras), with a different camera selected on each execution.

It was also discovered that the .NET enviroment has very limited support for process management, leading developers to have to resort to coding TCP or UDP networking packet solutions for processes communication rather than the high level communication libraries available in other languages, this did hamper the development of an elegant solution.

4.5 Localization

Since augmented reality relies on graphics libraries to superimpose three dimensional textures onto captured camera frames and since graphics libraries operate on three dimensional virtual environments, there is a possibility of applying graphics libraries to actual environments. The first step in completing the localization is to provide a level of simplification but taking the determined vector and extracting the x and y components, thus turning a three dimensional problem into a planar problem, thus simplifying the localization (by making the assumption of a level environment and the robot will remain on the floor).

4.5.1 Angle determination

The mathematical formula for the determination of an angle between 2 vectors is:

$$u \bullet v = \cos(\theta) \, \|u\| \, \|v\| \tag{4.1}$$

which by taking the arccos of the dot product of the two vectors over the product of their respective distances it is possible to determine the smallest angle between two vectors. This

allows us to find the angle produced by a vector drawn through the camera center point and the vector drawn to the marker detected (which is easily determined by transforming a point from (or near) the camera (the grid origin) to the marker using the ARToolKit provided transformation matrix).

Algorithmically this happens as shown below (using a slightly simplified stepped C#-like psuedocode example and libraries from the XNA gaming framework).

```
1 float CalculateAngle(vector markerVector)
2 {
3         //unit vector along the x-axis
4         vector = = {1,0};
5         int dotprod = Vector2.dot(vector, markerVector);
6         int lengthvecs = Vector2.Length(vector)*Vector2.Length(markerVector);
7         angle = arccos(dotprod/lengthvecs);
8         return angle;
9 }
```

The above algorithm could be normalized to use unit vectors for the calculation thus eliminating the need for length calculations. But since this is not a high powered rendering engine the need for such a simplification is unnecessary. Of course, should you wish to deploy this system on smaller devices, normalizing the vectors would be a sensible step to reduce the complexitvity of the resulting equation, unit vectors are equal to a single unit, thus the denominator in the above angle determination becomes equal to one, thus eliminating several processing steps, but the increase of performance is debatable due to the added task of normalizing the vectors, but this is discussion that is outside the scope of the project.

Based on which camera the detected image comes from, an adjustment may be needed to ensure that the inner angle is used in the triangulation methods, this can be determined based on the result of the dot product of the vectors. If the dot product is equal to zero the vectors are at ninety degrees to each other. An angle less than ninety degrees produces a positive dot product while an angle greater than ninety degrees has a negative dot product of the vectors.

4.5.2 Distance Determination

Once the 2 angles have been determined, the triangulation can be done.

```
1 float triangulationDistance (float angle1, float angle2, float length)
2 {
3         return length / (1/tan(angle1) + 1/tan(angle2));
4 }
```

The triangulation algorithm uses the length and two angles to calculate the distance from the vector drawn through the two cameras.

4.5.3 Robot Turning Angle

The proposed solution included mounting a number of markers on the robot itself to determine robot localization and orientation. Unfortunately this was beyond what could actually be achieved in the time frame allocated to the project. Thus we were unable to implement this part of the system.

4.6 Robotics Studio DSS Services

For the implementation of the robotic methods, we adopted a platform consisting of the distributed software services (DSS), C#, and Microsoft Visual Programming Language (VPL), which are new technologies provided by Microsoft in the Microsoft Robotics Studio 2008 R2. This framework designed for the rapid development of robotic applications in both home and industry, providing an extensible simulation environment for the testing of the produced DSS solutions.

The aim of the Robotics studio is to make asynchronous programming easy with a REST based model of parallel software services.

4.7 Lego NXT Control Algorithms

The Lego NXT was implemented according to the specification depicted in Fig. 4.8.

Robotics Studio works primarily on XML files known as manifests, that contain all the configurations for the system. Given below are the NXT brick and motor manifests.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <NxtBrickState xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:wsa="http://
      schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:d="http://schemas.microsoft.com/xw
      /2004/10/dssp.html" xmlns="http://schemas.microsoft.com/robotics/2007/07/lego/nxt/
      brick.html">
3         <Configuration>
4                 <SerialPort>4</SerialPort>
5                 <BaudRate>115200</BaudRate>
```

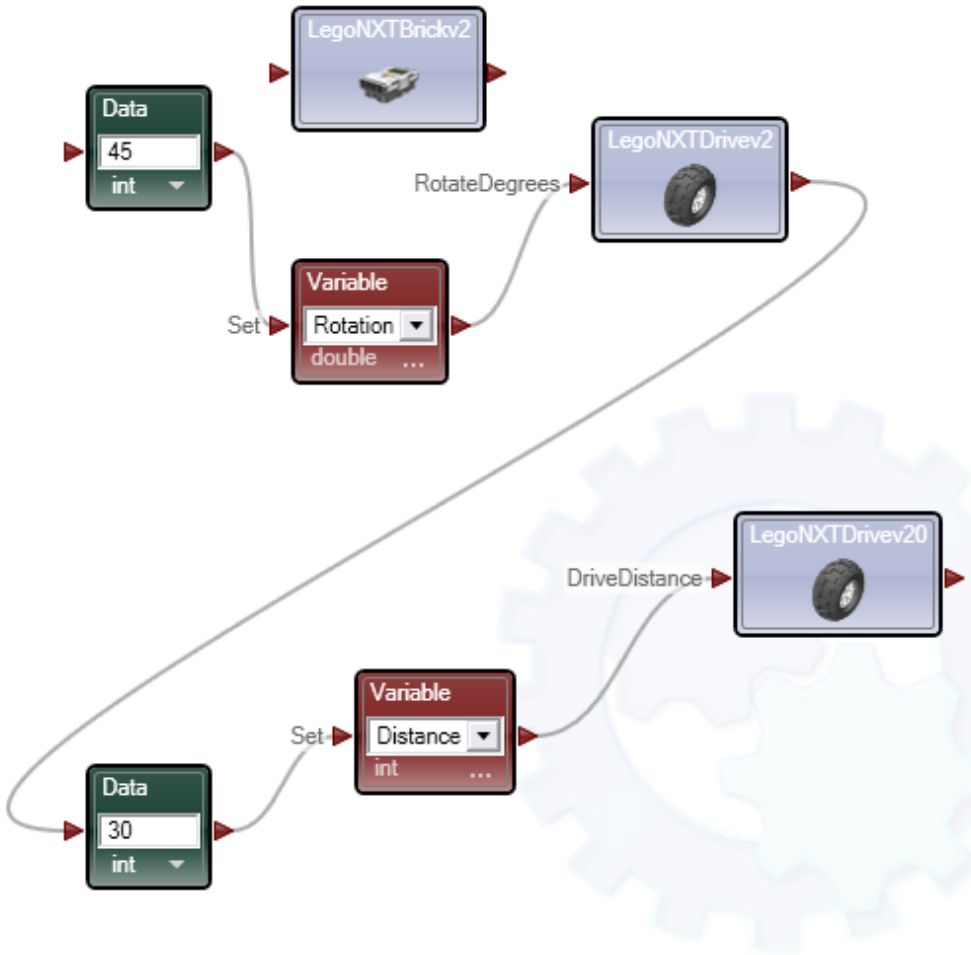

Figure 4.8: Visual Programming Language Diagram showing connections between different distributed software services

```
6                   <ConnectionType>Bluetooth</ConnectionType>
7                   <ShowInBrowser>true</ShowInBrowser>
8           </Configuration>
9   </NxtBrickState>
```

This Brick manifest contains all the configurations needed to connect via a Bluetooth connection (on this system abstracted to serial COM port 4) as well as a permission line allowing the system to be displayed in a browser.

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <DriveState xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org
        /2001/XMLSchema-instance" xmlns="http://schemas.microsoft.com/robotics/2007/07/lego/
        nxt/drive.html">
3           <DistanceBetweenWheels>0.112</DistanceBetweenWheels>
4           <LeftWheel>
5                   <MotorPort>MotorB</MotorPort>
```

```
6                   <ReversePolarity>false</ReversePolarity>
7                   <WheelDiameter>0.055</WheelDiameter>
8           </LeftWheel>
9           <RightWheel>
10                  <MotorPort>MotorC</MotorPort>
11                  <ReversePolarity>false</ReversePolarity>
12                  <WheelDiameter>0.055</WheelDiameter>
13          </RightWheel>
14          <PollingFrequencyMs>0</PollingFrequencyMs>
15  </DriveState>
```

The motor manifest sets all the hardware configurations for a 2 motor differential drive using the NXT. The wheel parameters are set and the ports for communication established.

The purpose of a manifest is to allow rapid configuration of the system. The system also supports generic robotic programming, where an appropriate manifest will be selected based on the detected hardware.

It is also possible, but far more complicated to program the implementation by hand using C#.NET or VB.NET.

```
1  // Partner: LegoNXTDrivev2, Contract: http://schemas.microsoft.com/robotics/2006/05/drive
       .html
2  [dssa.Partner("LegoNXTDrivev2", Contract = drive.Contract.Identifier, CreationPolicy =
       dssa.PartnerCreationPolicy.UsePartnerListEntry)]
3  drive.DriveOperations _legoNXTDrivev2Port = new drive.DriveOperations();
```

The above code details the pairing of a service with another service. The partner invocation sets up ports of communication and creates an abstracted object embodying the services that the method provides.

```
1  DiagramState a = new DiagramState();
2  State.Rotation = (double)45;
3  a.Rotation = State.Rotation;
4  drive.RotateDegreesRequest request = new drive.RotateDegreesRequest();
5  request.Power = 0.5D;
6  request.Degrees = a.Rotation;
7
8  Increment();
9  Activate(
10     ccr.Arbiter.Choice(
11         LegoNXTDrivev2Port.RotateDegrees(request),
12         OnRotateDegreesSuccess,
13         delegate(soap.Fault fault)
14         {
15             base.FaultHandler(fault, @"LegoNXTDrivev2Port.RotateDegrees(request)");
16             Decrement();
17          }
18     )
19 );
```

Methods provided by the motors are accessed as demonstrated in the above method. A request object is created in line 4 and is sent to the service defined in Activate() on line 9, where error checking takes place and the service then waits for an indication of success of the action.

The platform was chosen for the implementation of the proposed system as it is modern and abstracts away from memory address programming. Furthermore, it can be compiled to act distributively. The task of creating a method to move forward a certain distance, and a method to rotate on the spot rotation was also simplifies. While DSS services were explored and potential solutions determined, a working system that could communicate with the rest of the system was not achieved.

4.8 Networking and Communication

The proposed solution relies on a great deal of networking on the TPC or UDP layer. The idea being that all the modules communicate across the network allowing for distributivity. For this purpose serializable types were used at critical stages. Unfortunately as the system was large the networking and communication was not fully implemented. Thus the syncronization and communication of multiple real time parallel processess and Distrubuted Software Services is a possbile avenue of further research.

4.9 Chapter Summary

In this chapter the implementation of the design was explored. The wrapping, initialization and implementation of the ARToolKit was explored and the localization methods explained. The chapter also discusses DSS and the applications to robotics. The problems experienced while implementing the solution were also discussed as well us the incomplete implementation of several due to time constraints and the project itself being of a scope that was difficult to approach. Chapter 5 evaluates the technologies used in this project and the implications those technologies can have for modern robotics.

Chapter 5

Evaluation

5.1 Introduction

In this project a variety of technologies and concepts were explored; ranging from different visual marker recognition technologies to different methods of robotics programming. In this chapter we assess some of the more core technologies and the effectiveness of such technologies. We also discuss some of the strengths and weaknesses of such technologies from a developer's perspective.

5.2 Visual Technologies

Earlier in this project we assessed visual technologies and parsing visual feeds. We now assess the experience and challenges using vision as an input modality.

Vision is a complicated modality that is effective only after careful consideration of the challenges faced by the camera modality. The first challenge is the lighting of the environment. It has been documented that in highly variable and inconsistent lighting conditions, the accuracy of the detection can be compromised. Cameras in changing conditions often have to be recalibrated or need to adjust the focus and the exposure. This slows down frame rates and thus decreases the chance of detection. In all effect this often makes many visual techniques unreliable in all but a closed environment. For the concept of robotic vision based systems it was found that in an indoor environment the detection can be quite robust.

The focus of this project was on the detection of fiduccial markers, the implication of which is discussed in the next section.

5.2.1 ARToolKit+

The ARToolKit+ is a relatively undocumented library (in contrast to the ARToolKit which has very detailed documentation) designed to detect fiduccial markers and easily provide such a service to applications. The ARToolKit+ itself is difficult to configure, relying on complicated configuration files and camera parameters and often has to be compiled directly on the system on which it is going to be used. The toolkit has very strict dependencies on OpenGL graphics and for first experiences has a steep learning curve.

The detection rate is good except in the case where a marker occurs at a particular orientation to the camera. At extreme angles the marker escapes detection as discussed in the next section, which covers the disadvantages of marker based detection.

An important consideration with the ARToolKit is that it is very specific, and although very technical, it allows rapid integration of detection methods without having to develop the visual recognition analysis methods from scratch despite the configurations being complex.

Apparent Disadvantages of Marker Based Detection

Marker based detection in the context of this project, is a detection method where distinct markers are used in the environment and are "easily" detected by the system.

Our experience with such a system highlighted a few disadvantages with using markers. First, markers at particular orientations to the camera could possibly escape detection. Thus if a system requires robust detection, a method for adding marker redundancy needs to be devised. In the ARToolKit+ there exists the capacity in situations where markers are obscured that the other markers can be determined based on their orientation or relative position to the detected markers.

Another disadvantage of marker based detection systems is the markers themselves. In visual based interaction systems much emphasis is placed on the idea of Ubiquitous computing, which allows a user to use a computer without being aware that the computer was involved. In this case the markers are the fulcrum of the interaction technique and break the illusion. The solution for

this would be to create visually aesthetically pleasing markers.

5.2.2 Reliance on Graphics Libraries

The ARToolKit+ and its predecessors rely heavily on the graphics libraries. The focus of augmented reality is the application of on the fly graphics on a visual feed or headset feed. Thus the focus of the development of the ARToolKit was on providing methods and functionality that supports the graphics platform. This makes it challenging to apply the ARToolKit to other applications where the focus is not graphics based as the programmer has to have a detailed knowledge of OpenGL or DirectX as well as matrix mathematics to understand the interpretation of the ARToolKit and its markers.

5.3 Lego ^{NXT} as a Research Platform

The Lego ^{TM} NXT proved to be an effective platform for the research objective. The main strength of the device being its simplicity. The port and plug architecture of the NXT allows the developer to assemble in a short space of time a functioning robot that potentially meets the user requirements. The NXT also has the advantage that many possible programming platforms are available. These range from C to simple visual programming languages.

The only disadvantages of the Lego ^{TM} NXT, is the possible lack of compatible sensors and motors that meet the requirements of a research project. The sensors available are potentially limited and fairly inaccurate. However, the drivers are open source, and thus 3rd party sensors could be created.

A serious challenge at the start of the project was finding a compatible Bluetooth device for the unit. The Lego NXT uses a very specific Bluetooth stack and standard and the only available compatible device was the D-Link DBT-122 which is detailed in Chapter 4.

The NXT proved itself to be an excellent rapid prototyping device for the development of robotic based applications.

5.3.1 Usefulness of Rapid Prototyping Devices

The benefit of rapid prototyping devices lies in their potential for speeding up the development cycle of experimental devices such that a concept can be tested and enhanced without the need for expensive custom devices or hardware. The Lego ^{TM} NXT fits this category as it allows the construction of a custom robot quickly. This allowed us to test different techniques and software, such as the Microsoft Robotics Studio.

5.4 Microsoft Robotics Studio

The Microsoft Robotics Studio, introduced in Chapter 4, was the implementation platform for the robotics methods and connection for the NXT.

5.4.1 Distributed Software Services

The Microsoft Robotics Studio relies on a very interesting concept known as distributed software services (DSS). This is a method of programming for robotic systems as it essentially embodies each robotic component as a potential asynchronous parallel process that provides services over REST web services. The Microsoft Concurrency and Coordination Runtime (CCR) provides the framework that provides DSS all the synchronisation it needs.

What is interesting about this system is the design for implementation on both distributed and clustered platforms as well as a single machine. This makes it an excellent platform for the development of industral robotic systems that often have many different components.

DSS itself is very difficult for a beginner to comprehend as it breaks the sequential programming style that many are very familiar with and presents a programming architecture where objects and hardware have to be represented as services. What is interesting is the analogy that can be drawn to the Communicating Sequential Processes (CSP) model. Often in DSS, the services synchronise at the point of communication or message passing between the services, alghtothis "wait" is explicit requiring a "wait for completion" added on to the service request.

The major advantage associated with DSS is that once a method is written it can potentially be reused oftenand the functionality exists to add entire services to an application. Microsoft has advanced this functionality with the inclusion of generic services that don't specifically

interface with any sort of hardware but potentially work with any hardware provided the correct XML manifest. This makes it useful for use with rapid prototyping devices like the NXT. The NXT manifest just needs to be swapped with another manifest and the other device can easily use the same methods with little alterations.

5.4.2 Visual Programming Language

Included with the Microsoft Robotics Studio is the Microsoft Visual Programming language. A new concept for the integrated development environment has been proposed, and while not a new concept the VPL environment is perfect for the development of distributed software services as it is capable of visually depicting the services, the inputs and outputs from the services and the connections between services. VPL was found to be a perfect basis for the development of any DSS based application based on the fact that standard services can be planned out and then C# code generated that matches that VPL planned system. The only disadvantage is that development of custom VPL objects and DSS objects is difficult and not well documented.

5.4.3 Robotics Studio and the Lego ^{TM} NXT

The Lego ^{TM} NXT is one of the supported robotic platforms with included manifests in the Microsoft Robotics Studio. Despite a few issues setting up the initial configuration with respect to the Bluetooth ports and pairing the robot with the system, development for the NXT platform was quite simple. Once the manifests had been set out and configured for the custom nxt build, programming simple methods was not difficult at all. In fact from experimentation it was found that robotics studio provided much smoother control than is offered by other programming platforms, particularly the NXT-G that was shipped with the NXT itself.

5.5 Chapter Summary

In this chapter we examined the technologies, hardware and software used in this project and evaluated the appropriateness of each technology together with the thoughts and criticisms from a developer's perspective of the technologies involved. Each core component was evaluated and apparent problems or successes stated.

Chapter 6

Conclusion

With a new focus on incorporating vision into robotics and developing systems that actively use computer vision to drive and control robotic systems, the approach used in the project was to develop a prototype system that incoporates all the problem domains when computer vision is used in the field of robotics.

6.1 Summary

Chapter two discussed previous work and concepts that need to be considered in the fields of visual technologies, recognition and computation using these technologies and how such advancements can be applied to modern robotics. Simple triangulation was determined to be important for the project as it allows the calculation of the relative position of an object by the position of two other objects (possibly cameras) provided the angles of incidence between the line between the two reference objects and the distance between them are known. An overview of the visual technologies in the field of robotics was also explored.

In chapter three the design requirements and approach were discussed. We assessed the expectancies of any visual based system and proposed a solution breaking the solution into several different smaller problem domains. The first of these dealt with the problem of visual recognition by proposing a marker based system with pattern based fidducial markers powering the visual detection. The localization design was then proposed supported by the triangulation method presented in chapter two. It was also assessed that for the basic requirements of the system the robot needed two "methods" of functionality, namely a drive and a turn method.

Chapter four detailed the attempt to create a system that meets the design requirements proposed in Chapter 3. The development was performed primarily on the Microsoft visual C# platform. A C++ library, called the ARToolKit was compiled into a DLL and accessed through a C# wrapper class. The localization could be performed using graphics libraries and the triangulation formulae presented in Chapter two. Then the implementation of DSS was discussed and the implications of developing using such a architecture as well as the advantages of using DSS. The uncompleted sections of the implementation were also discussed with regard to the networking and connectivity of the system and the functionlity of the unimplemented methods.

Chapter five gave a final assessment of the system developed, proposed recommendations and qualitatively assessed the technologies. Each problem domain was approached from the perspective of a developer and evaluated and the appropriateness for the problem was discussed. The weaknesses of marker based detection were discussed together with the implications for the localization. DSS was discussed as a programming architecture for distributed and robotics systems and was shown to be very effective.

6.2 Problem Statement Revisited

A system was proposed to meet the project goals of developing a vision based robotic system that uses cameras to both control and localize the robot. Upon implementation however it was found that the problem was much larger than it first appeared. The project involved many different technologies that were often undocumented. While the system could not be completed in the course of a year a large part of the system was implemented. This implemetation allowed several assessments to be made after the experience obtained in creating such a system and the use of several different technologies that were useful in isolation but fell victim to incompatibilities when applied to the larger system.

6.3 Future Extensions

The project suggests several avenues for future study, most being perhaps enhancements to the system itself. The largest extension would be the implementation of an appearance based system that does not require fiduccial markers to power the visual detection, but insteads stores a mesh of the robot and localizes via scanning for objects fitting the mesh, but at a cost of higher processing requirements. In the area of localization, wireless localization using 802.11 wireless

Ethernet could possibly be a worthwile pursuit with respect to the field of localizing moving robots. There is also the avenue of adding additional interaction modalities to the robot, like speech or remote control, to provide additional options for the user, and testing how accurately these systems can cooperate.

References

[1] ABAWI, D. F., BIENWALD, J., AND DORNER, R. Accuracy in Optical Tracking with Fiducial Markers: An Accuracy Function for ARToolKit. In *ISMAR '04: Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 260–261.

[2] BAGNALL, B. *Maximum Lego NXT: Building Robots with Java Brains*. Variant Press, 2007.

[3] BARKUUS, L., AND DEY, A. Location-Based Services for Mobile Telephony: a Study of User's Privacy C. In *INTERACT 2003,9th IFIP TC13 International Confererence on Human-Computer Interaction* (2003).

[4] BASS, L., KASABACH, C., MARTIN, R., SIEWIOREK, D., SMAILAGIC, A., AND STIVORIC, J. The Design of a Wearable Computer. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1997), ACM, pp. 139–146.

[5] BESL, P. J., AND JAIN, R. C. Three-Dimensional Object Recognition. *ACM Comput. Surv. 17*, 1 (1985), 75–145.

[6] CHIN, R. T., AND DYER, C. R. Model-based Recognition in Robot Vision. *ACM Comput. Surv. 18*, 1 (1986), 67–108.

[7] DANIILIDIS, K., KRAUSS, C. H., M., H., AND G., S. Real Time Tracking of Moving Objects with an Active Camera. *Real Time Imaging* (1997).

[8] FAILS, J. A., AND OLSEN, D. R. Light Widgets: Interacting in Every-day Spaces. In *Proceedings of IUI'02* (2002).

[9] FARRELL, J., AND BARTH, M. *The Global Positioning System and Inertial Navigation*. McGraw-Hill Proffesional, 1999.

[10] FISCHER, J., REGENBRECHT, H., AND BARATOFF, G. Detecting Dynamic Occlusion in Front of Static Backgrounds for AR Scenes. In *EGVE '03: Proceedings of the workshop on Virtual environments 2003* (New York, NY, USA, 2003), ACM, pp. 153–161.

[11] FOX, D., BURGARD, W., DELLAERT, F., AND THRUN, S. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence* (Menlo Park, CA, USA, 1999), American Association for Artificial Intelligence, pp. 343–349.

[12] GOEL, P., ROUMELIOTIS, S. I., AND SUKHATME, G. S. Robust Localization Using Relative and Absolute Position Estimates. In *Intelligent Robots and Systems* (1999), vol. 2, pp. 1134–1140.

[13] GRIMSON, W. E. L. A Computer Implementation of a Theory of Human Stereo Vision. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 292*, 1058 (1981), 217–253.

[14] HU, Y., JOHNSON, D., AND PERRIG, A. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. *Ad Hoc Networks 1*, 1 (2003), 175 – 192.

[15] JARVIS, R. A. A Perspective on Range Finding Techniques for Computer Vision. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1983).

[16] KATO, H., AND BILLINGHURST, M. Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System. In *IWAR '99: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality* (Washington, DC, USA, 1999), IEEE Computer Society, p. 85.

[17] LADD, A. M., BERKIS, K. E., RUDYS, A., KAVRAKI, L. E., AND WALLACH, D. S. Robotics-based Location Sensing using Wireless Ethernet. *Wirel. Netw. 11*, 1-2 (2005), 189–204.

[18] LANGENDOEN, K., AND REIJERS, N. Distributed Localization in Wireless Sensor Networks: a Quantitative Comparison. *Computer Networks 43*, 4 (2003), 499 – 518. Wireless Sensor Networks.

[19] LARSSEN, A. T., LOKE, L., ROBERTSON, T., AND EDWARDS, J. Understanding Movement as Input for Interaction-A Study of Two Eyetoy Games. In *Proceedings of OZCHI 2004* (2004).

[20] LI, Z., TRAPPE, W., ZHANG, Y., AND NATH, B. Robust Statistical Methods for Securing Wireless Localization in Sensor Networks. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks* (Piscataway, NJ, USA, 2005), IEEE Press, p. 12.

[21] MARR, D., AND POGGIO, T. A Computational Theory of Human Stereo Vision. *Proceedings of the Royal Society of London. Series B, Biological Sciences 204*, 1156 (1979), 301–328.

[22] MATTIES, L. Stereo Vision for Planetary Rovers: Stochastic Modeling to Near Real-Time Implimentation. *International Journal of Computer Vision 8* (1992), 71–91.

[23] MAUVE, M., WIDMER, A., AND HARTENSTEIN, H. A survey on Position-Based Routing in Mobile Ad Hoc Networks. In *Network, IEEE* (2001).

[24] MURASE, H., AND NAYAR, S. K. Visual Learning of 3-D Objects from Appearance. *International Journal of Computer Vision 14* (1995), 5–14.

[25] MURPHY, W., AND HEREMAN, W. Determination of a Position in Three Di-mensions using Trilateration and Approximate Distances. *Colorado School of Mines* (1999).

[26] NAKAZATO, Y., KANBARA, M., AND YOKOYA, N. Discreet Markers for User Localization. vol. 0. IEEE Computer Society, Los Alamitos, CA, USA, 2004, pp. 172–173.

[27] NAKAZATO, Y., KANBARA, M., AND YOKOYA, N. An Initialization Tool for Installing Visual Markers in Wearable Augmented Reality. *Advances in Artificial Reality and Tele-Existence 4282/2006* (2006), 228–238.

[28] PANTEL, L., AND WOLF, L. C. On the Suitability of Dead Reckoning Schemes for Games. In *NetGames '02: Proceedings of the 1st workshop on Network and system support for games* (New York, NY, USA, 2002), ACM, pp. 79–84.

[29] PANZIERI, S., PASCUCCI, F., AND ULIVI, G. An Outdoor Navigation System Using GPS and Inertial Platform. *Mechatronics 7* (2002), 134–142.

[30] SALICHS, M. A., ARMINGOL, J. M., MORENO, L. E., AND DE LA ESCALERA, A. Localization System for Mobile Robots in Indoor Environments. *Integr. Comput.-Aided Eng. 6*, 4 (1999), 303–318.

[31] SHELL, J. S., VERTEGAAL, R., AND SKABURSKIS, A. W. EyePliances: Attention-Seeking Devices that Respond to Visual Attention. In *Conference on Human Factors in Computing Systems* (2003), pp. 770–771.

[32] STARNER, T., AUXIER, J., ASHBROOK, D., AND M., G. The Gesture Pendant: A Self-Illuminating, Wearable, Infrared Computer Vision System for Home Automation Control and Medical Monitoring. In *International Symposium on Wearable Computing* (2000).

[33] WANG, H., BOWMAN, C., BRADY, M., AND C., H. A Parallel Implimentation of a Structure-From-Motion Algorithm. *Lecture Notes in Computer Science 588* (1999), 272–276.

[34] WARD, P., SCOTT, H., HOLMES, J., AND LAPADULA, L. GPS System and Method for Deriving Pointing or Altitude from a Single GPS Reciever, 1993.

[35] WELLS, D. Ieee 802.11 Signal Source Mapping using Low Cost Spectrum Analysers. Master's thesis, Rhodes University, 2007.

Appendix Included with CD Deliverable

The contents of the accompanying CD is as follows:

- Final Thesis (PDF and TEX files)
- Final Poster
- Final Presentation
- VC# project of ARToolKit+ implimentation (Not trained with markers)
- Microsoft Robotics Studio VPL Service for Modified TriBot