DESIGN AND DEVELOPMENT OF FRAMEWORK FOR ROBOTICS USING CLOUD COMPUTING

By

Omar Rahman

UsamaYaseen

KhurramJaved

Submitted to the Faculty of Computer Software Engineering, Military College of SignalsNational University of Sciences and Technology, Islamabad in partial fulfillment for the requirements of a BE in Computer Software Engineering

JUNE 2012

Abstract

This thesis reports a cloud robotics framework composed of mobile phone based robots and a cloud computing architecture, facilitating complex and heavily computational tasks to aid the user in the completion of difficult robotics related problems. The Cellbot based framework will complement the group of networks that the user may already be a part of, contributing as a connection bridge between real and virtual worlds.

The objective of this endeavor marks the implementation of a small test/demo cloud based robotics service framework that permits distant groups of cellbots to share learned skills while simultaneously improving collaboration with non-technical human users. This cloud facility allows for robots to share new and used knowledge by downloading and uploading information as it arrives, without needing onboard storage, thus risking information freshness.

In this scenario the context aware skill arranges the system to give services such as maze traversal and path following. The small movement Cellbot used in the experiments is able to navigate its way through both the scenarios without relative difficulty after receiving commands from the Cloud server with negligible amounts of delay times. The needed input for the system to make the Cellbot move was entirely done using a single 2D camera of the Android smartphone placed on the Cellbot, whereas all storage and processing was done on the Cloud.

Declaration

We, NC Omar Rahman, PC UsamaYaseenand NCKhurramJaved, do hereby solemnly declare that the work presented in this report is our own, and has not been presented previously to any other institution.

Signature

_____

(Syndicate Leader)

_____

(Member)

_____

(Member)

Certified:

_____

Supervisor

Dedication

We would like to dedicate this project to our wonderful parents, who made every day easy for us, so that we could focus our energies on this project, while they took care of the rest.

Acknowledgements

First of all we would like to thank Allah Almighty. Whatever we have achieved, we owe it to Him totally.

We are really grateful to our parents, family and well-wishers for their admirable support not only during the course of the project, but also throughout our lives, as without them, this would never have been possible. They have always been behind our every success and are the major force that facilitates us in achieving what we aspire.

We would like to thank our project supervisor Lt. Col. Dr. Asif Masood for all the inspiration and technical corrections. Col. Dr. Fahim Arif has been very helpful in directing us to do the right thing in the right manner. Together, both of them provided us with the opportunity to polish our technical skills. This project has been brought to shape by their consistent assistance.

There are other people who played their part in various ways. Dr. Hammad Afzal was always there, whenever we needed him, to solve various problems. Dr. AwaisMajeed helped in providing documentation guidance. Lec. Bilal Rauf helped with lab facilities and technical matters. We are highly grateful to all of you in making this project a success.

Preface

Cloud Robotics is a recently envisioned concept that aims to bring together the ideas of “Remote brained robots” and Cloud computing into one niche. The common man has limited reach to robots for his everyday needs such as household work, car driving, and help for the elderly and so on. Using methods employed in the domain of cloud robotics, everyone can utilize some sort of benefit with a simple robot that takes its commands from an ordinary user and performs functions as any high end robot would.

In this project, it was aimed to develop a small demo framework that supports the idea using a small private cloud, attached to an Android based robot that fulfills basic navigational tasks. Using this set up, we demonstrate how a dumb terminal robot uses the computational and storage capabilities of a remote cloud for its own purposes. This is done when a user selects a simple navigation task on the Android robot, which then connects to the cloud, requesting for help on how to perform that task.

In real world scenarios, this framework is extensible to large scale robotic environments, where either labor costs are high or skills are less available. Quick deployment of untrained robots in a field is possible whereby one requires only a simple internet connection to the cloud. Upon connection establishment, algorithms could be downloaded for a wide array of applications. These applications range from object recognition and

autonomous navigation to automatic vehicle driving and human-less military forces.

The focus has been on a simple navigation plan, in which it has determined movements of the robot beforehand. The robot, referred to as the “Cellbot” in our document, has no prior information of the navigational area, and thus seeks information from the cloud on first use. This small scale demo partially fulfills the requirement for a full blown cloud robotics suite. Yet, even in our simple demo environment, it was shown how robots could be made simpler and yet more productive, by offloading their storage and computational needs to a remote server.

Our private cloud is an alternative to the traditional web server in the manner in which it is used. While a traditional web server may require a user to run and maintain an application on the client end, the cloud server takes all the hosting and maintenance responsibility on itself, providing a one-window interface to the user, as the user only provides the data and requests. Here too, the user shall only give the requests for the available navigation services, regardless of how the application is running to provide the service to the user.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1. Introduction

This document is the documented deliverable of the Final Year Project group of the group syndicate including Omar Rahman, UsamaYaseen, and KhurramJaved of BESE 14 at the Department of Computer Software Engineering. The title of the project is "**Design and Development of Framework for Robotics using Cloud Computing**". The project is based on the concept of cloud robotics, a new and innovative application of cloud computing. This document contains the software requirements, software design, implementation and testing, as well as the user manual for the project.

Cloud computing is a word for computation, software, data access, and storage services, not requiring end user knowledge of location nor configuration for mass distribution and usage. It is an IT service model for enabling convenient, on-demand network access to a shared pool of computing resources to users who are connected simply through the internet.

Cloud Robotics is the amalgamation of cloud computing with robotics to make robots easy to understand and build, while at the same time reduce costs and machine size requirements. It has been a challenge thus far to build robots for large scale applications, however with the help of cloud computing, robot machines now need just a connection to the cloud for processing and storage capabilities.

1.1. Document Conventions

This section describes the standards followed while writing this document.

1.1.1. Headings:

Heading are prioritized in a numbered fashion, the highest priority heading having a single digit and subsequent headings having more numbers, according to their level. All of the main headings are titled as follows: single digit number followed by a dot and the name of the section (All bold Arial, size 16, e.g.[**2.Overall Description**]).

All second level sub headings for every sub section have the same number as their respective main heading, followed by one dot and subsequent sub heading number followed by name of the sub section (All bold Arial, size 14, e.g.[**2.2 Product Features**]).

Further sub headings, i.e. level three and above, follow the same rules as above for numbering and naming, but have font sizes gradually decreasing.

1.1.2. Bullets and numbering:

Bullets have been given where there is no need for prioritizing a list. For example the list of use cases, where uses cases may appear in any order. The bullet kind used in this document is [•].

Numbered lists are normally used for prioritizing purposes. Prioritizing purposes arise when the customer has specified a specific order for the requirements or when a need for prioritizing arises due to business needs.

1.1.3. Figures and tables:

All figures have captions and numbers below them while tables have captions above them with numbering. Unless specified, all use case, context, flow and other diagrams are based on UML standards.

1.1.4. References:

All references to text in this document are provided primarily as footnotes, however where not present, the superscripts are present and indicate a passage in the references section at the end. All ambiguous terms have been clarified in the glossary at the end of this document. Reference and citation standard is IEEE:

<Author(s) Last name, Author(s) First Name>, “<Name of the work>”, <Publication/Journal/Book Title>, <Publication City>, <Publication Year>, <Pages cited>.

1.1.5. Links to web pages:

All links have been provided with underlined font, the title of the web page or e-book is written at the top of the link and the title may be searched on Google to pinpoint to the exact address.

1.1.6. Acronyms:

All acronyms have been explained at the glossary at the end of this document, unless an acronym needs expansion for explanation purposes.

1.1.7. Basic Text:

All other basic text appears in regular, size 12 Arialfonts. Every paragraph explains one type of idea.

1.2. Intended Audience and Reading Suggestions

This section describes the intended audience for the document including the stakeholders and non-stakeholders of project.

1.2.1. Project Deputy Supervisor and faculty members:

Project DS (Who must approve the submission of the SRS at department of CSE):

Maj. Dr. AsifMasood, HoD, Department of IS, MCS.

Faculty Members:

Department of CSE project evaluation panel, who shall evaluate the developed product and judge it according to given criteria.

Head of Department, Col. Dr. Fahim Arif, who shall evaluate the project separately,

Project Evaluation Coordinator, Dr. Hammad Afzal.

Any other faculty members who wish to read the SRS for non-evaluation purposes.

BESE 14-A group (developers, testers, and documentation writers):

Omar Rahman (Project Lead)

UsamaYaseen

KhurramJaved

Others at MCS:

- R&D Committee Members.
- Faculty members of other departments.
- Students of UG and PG courses.

Other Audiences:

- Students, faculty members of other universities and institutes
- People interested in Android application development, cloud computing, or both
- People interested in cloud robotics, especially in open source platforms.

1.2.2. Reading suggestions:

Readers interested in cloud computing, Android development and robotics (either or all three of them), could directly move to functional requirements. Readers interested in how cloud computing leverage and advantage may see the nonfunctional requirements for numbers on cloud computing justification. It issuggested that all readers start at the top and read till the end, because the concept of cloud robotics is fairly new and most readers would appreciate the technology more if they read the document in its entirety.

1.3. Problem Domain and Problems Addressed

The area of robotics theory has been limited to large research organizations and universities due to the cost of building a real world robot. Normal robots cost around $2000- 1 million depending on their size and complexity. Usually a humanoid robot is not available for commercial use and those that are, have very limited functionalities and high battery consumption rates.

Another problem is of the size of the machinery and task capability ratio. The complexity of a robot is not only dependent on how well programmed it is, but also how much hardware is laden on it. This machinery not only includes

mechanical parts for movement, but CPUs, sensors, memory, hard disks and more that take up more than 90% of the size of the full robot. It leaves very little space for a battery and thus robots are not able to work for a long duration under full processing load. However a lot of hardware is needed for the robot to be able to accomplish its tasks.

Connecting robots to a distant server offloads some of the hardware requirements including the large CPUs and hard disks, leaving the robot with more space for battery and mechanical parts. It increases the movement capabilities of a robot due to increased battery life and decreased space limits.

In the past, adding new functions to a robot usually meant adding more CPU and memory, plus extra sensors, effectors and servos for work (and battery with it all). This meant a whole lot of problems for the robot designers, as they had to work out an efficient design for the robot based on the software interfaces available plus physical design constraints. Cloud robotics takes out the hurdle of adding compute and store hardware (it is now shifted to the cloud), while software interfaces are also handled as the other machinery such as sensors, actuators and effectors sends their requests to a common cloud interface.

Cloud robotics represents a very large field of possible applications like robot navigation, search based AI, autonomous grasp and manipulation, and path planning to name a few. Most of these applications are difficult to work on for us and it was decided that in the given time and resources, the

selecteddomain should be path planning and experiments be done with the various dimensions that it offers to developers.

Path planning or motion planning is a term used in robotics for the process of detailing a task into discrete motions. A path is taken and the robot divides that path into bits and then covers those paths one by one according to mathematical algorithms that are calculated by it. Usually a test environment for path planning consists of a set of objects placed in the robots path that define boundaries and obstacles. These paths and obstacles are then used as variables for determining the path of the robot.

Robotics has always remained a problem for financially constrained groups due to the high costs of the hardware, which is directly related to the size of the hardware. This problem could be taken care of with the concept of Cellbots. With the arrival of the Android OS, smartphones have been loaded with GPS sensors and everything else required for basic robot building and usage. Android phones have their SDKs freely available for developers to use. Using simple Bluetooth connections to a microcontroller, an Android phone is more than capable of becoming the nerve center of a robot and moves it around, just like any Xeon CPU planted on a high end robot. The possible applications that could be made for an Android phone is endless and so are the robotics applications made for Cellbots.

Cellbots on their own have very limited functionality, again due to the small size they have and limited CPU and storage. These days, a big budget Android handset has around 2 GHz of CPU, 800 MBs of RAM, and an expandable memory of around 32 GBs. Some have GPUs that enable further

processing capabilities and image processing, but in very limited amounts. This does give a certain power to the Android as a robot controller, yet when one speaks of applications such as autonomous robot environment navigation, one requires much more resources (and not to mention battery time – average batteries on an Android run for 2.2 hours on full resource consumption).

By combining Cellbots and cloud computing, a whole new world of solutions comes into perspective. One sees many applications being done that were not possible before with such small hardware. It lets the power of processing and storage on a cloud be leveraged by the small hardware of an Android phone for moving around large servos and actuators. Since all the functionalities are outsourced to the cloud, only basic hardware is used for the robot (i.e. for movement help). A robot does not need to be trained or tested anymore to see if it carries the software perfectly, as almost most of its software does not lie on it anymore. Many robots could be handled through the same cloud and given the same exact orders while they carry them out at the same exact times. Robot maintenance also becomes much easier along with reduced prices and increased battery times.

Cellbots could be used by many researchers, practitioners, doctors, soldiers, engineers and scientists for small to large tasks, depending on the robot assembly underneath the Android phone. Robot replacement is not an issue anymore and neither is the fear of damaging one.

1.4. Objectives

Our objectives with this project are listed below:

To develop a framework that demonstrates the usage of cloud robotics and its applicability on a small scale. A small indoor environment shall be developed that shall have a limited number of objects in it to help the robot move around. The sub field of path planning shall be used to implement the framework, while using a small private cloud at the backend.

To apply the theoretical knowledge of programming and data structures, networks, database systems, web engineering, and computer architecture to practical perspective. The syndicate members have learned many subjects over the course of three years and now shall be applying their theory to real world problems.

To design a fully open source framework for cloud robotics that is helpful for academia and other students. Most of the robotics platforms available today are commercial and only a limited number of cloud robotics platforms exist. Those that do have very limited documentation and they are difficult to work with. Aim is to make a system that is easy to understand and build on. With our framework students could experiment on very small scale cloud robotics problems and appreciate the power of robotics.

To increase the development of robotics in Pakistan. Our country lacks in robotics and its applications at almost all levels. One needs platforms that enable cheap, smart and rapid robot construction and cloud robotics is an answer to all these queries. With computers easily available and broadband

internet everywhere, it is a matter of time before cloud computing enables robotics to take off in Pakistan.

1.5. Deliverables

The syndicate members shall be delivering some documents at the end of the project phases namely:

The SRS document that explains in detail the functional and non-functional requirements of the system and includes user interface specimens, the Analysis document that contains architectural level models of the system explaining the overall working and states, the Design document that explains in detail the structure of the system, the classes and data flow methodologies using models, test case document that shall describe what tests were carried out in what fashion and how were the results of each and every one of them, user manuals and troubleshooting guides, source code and installation packages, a Cellbot, a Wi-Fi router and a private cloud installed on desktop PCs in college laboratories.

1.6. Technological requirements

Software required:

The team requires only open source software for the entire duration of our project. All of the software listed below is available for free on the Internet (Genuine Microsoft software is available with MSDN subscriptions):

Windows Server 2008 R2, Windows Storage Server 2008 R2, Android SDK r13, Android OS 3.2, Eclipse 3.7 Indigo IDE, Apache Hadoop 0.20.203.0, JRE

6 with JDK 1.4, Netbeans IDE 7, Visual Studio 2008 with SP1, and MySQL standard edition.

Hardware Required:

The following hardware would be required for the project:

Three desktop computers with following specifications (64 bit Intel i3processors, 3 GB RAM, 520 GB HDD, 2 LAN cards, and other I/O peripherals), laptop computer with 64 bit architecture, TP Link Wi-Fi router, Samsung Galaxy S handset, iRobot ® Create ™ robot base, and BAM Bluetooth Accessory.

1.7. Project Plan

The project plan outlines who exactly is to work on this project and what their responsibilities would be in contribution to this endeavor. It also lists down the milestones to be achieved in a set frame of time. The team structure is shown in table 1.1 and the milestones are shown in table 1.2.

1.7.1. Team:

Name	Responsibilities
Omar Rahman	Team Lead and Designer
UsamaYaseen	Development
KhurramJaved	Testing and Deployment

Table 1.1 Project Team

1.7.2. Milestones:

Milestones		Description	Milestone Criteria	Planned Date<yyyy-mm-dd>
M0		Start Project		<2011-06-11>
		To read on cloud robotics documentation	Stakeholders identified Vision Document Reviewed	
M1		Start Planning		<2011-07-28>
		Scheduling, budget/resource allocation	Feasibility Report Validated, Requirements gathering	
M2		Start Execution of project		<2011-08-11>
		Requirement analysis	Requirements agreed, project plan reviewed, resources committed	
M3		Confirm Execution		<2011-10-19>
		Requirement Validation, Paper based Use case Models Reviewed	Requirements finalized ,baseline developed	
M4		Design and architecture		<2011-11-23>
		Developing Models using Rational Rose and MS Visio	Behavioral ,FUCTIONAL , data design models mapping from use case models finalized	
M5		Implementation		<2012-01-11>
		Programming using OOP techniques in specified development environment.	Functional Requirements implemented	
M6		Testing		<2012-04-01>

		White box and Black Box testing, Code walkthroughs	Product system tested, documentation reviewed, UAT completed, BAT completed successfully	
M6		Documentation and closing		<2012-04-12>
		Presentation given to Supervisor at our end, Documentation and Project Package sent to college	Deployment at college	

Table 1.2 (cont'd) Project milestones

1.8. Summary:

The chapter introduced the project and explained its various stages that are to be undertaken by the syndicate members in order to bring the project to a successful completion within in the given resources and time constraints. The chapter closed with a milestone and Gantt chart explaining the entire project lifetime and major obstacles to be overcome.

Chapter 2

Literature Review

2. Introduction

The syndicate members had to study very extensive material before embarking on this project. Most of the literature available on cloud computing and its sub branch, cloud robotics was either written on experimental works, works in progress, or beta phase software like Robo Apps (www.myrobots.com). Since so much work was in its early stages, reliance had to be made on a wider study domain and start from the ground up, by studying basics of cloud computing, SOA, service robots and cellbots. In the following paragraphs, summaries are given of the books, and papers that the syndicate members studied.Among the keywords searched on various online databases were: *Cloud computing, Robotics, cloud robotics, service robots, cluster computing, grid computing, hyper v, virtualized servers, assisted living.*

2.1. Works Read

In *Binocular Stereo Vision Based Obstacle Avoidance Algorithm for Autonomous Mobile Robots,*[1]Saurav Kumar described an algorithm for obstacle avoidance by building a stochastic representation of the Environment Navigation Map. This algorithm worked by dividing the test environment in a grid and assigning each cell a value of filled or unfilled.

In *The experimental humanoid robot H7: a research platform for autonomous behavior*[2], the researchers give an overview of the humanoid robot 'H7', which was developed over several years as an experimental platform for walking, autonomous behavior and human interaction research at the University of Tokyo. H7 was designed to be a human-sized robot capable of

operating autonomously in indoor environments designed for humans. The hardware is relatively simple to operate and conduct research on, particularly with respect to the hierarchical design of its control architecture.

“DAvinCi: A cloud computing framework for service robots.”[3]proposes a software framework that provides the scalability and parallelism advantages of cloud computing for service robots in large environments. The researchers implemented such a system around the Hadoop cluster with ROS (Robotic Operating system) as the messaging framework for our robotic ecosystem. They explored the possibilities of parallelizing some of the robotics algorithms as Map/Reduce tasks in Hadoop. The implemented the FastSLAM algorithm in Map/Reduce and showed how significant performance gains in execution times to build a map of a large area could be achieved with even a very small eight-node Hadoop cluster. The global map can later be shared with other robots introduced in the environment via Software as a Service (SaaS) Model.

In “*Web Services Based Robot Control Platform for Ubiquitous Functions*” [4]the researchers employ Web services, programmable application logic accessible using standard Internet protocol, to enable a robot to access the distributed application logic based on the recent network technologies like XML, SOAP, WSDL, UDDI. Applications can communicate with each other in a platform and programming language independent manner. They then explain the fundamental ubiquitous functions management framework for a robot. In this framework, the applications were constructed from multiple Web services that worked together to provide data and services for the application.

“A SemanticWeb Services Driven Application on Humanoid Robots.”[5]introduces a combined application of two different domains that are semantic Web services and robotics. It first presents semantic Web and Web services, and then describes the robotics development and their current limitation. The paper's core describes how semantic Web and Web services can be applied on robotics in order to facilitate cooperation between robots for joint tasks execution.

“*Integration of action and language knowledge: A roadmap for developmental robotics*” [6]proposes that the study of embodied cognitive agents, such as humanoid robots, can advance our understanding of the cognitive development of complex sensorimotor, linguistic, and social learning skills. This in turn will benefit the design of cognitive robots capable of learning to handle and manipulate objects and tools autonomously, to cooperate and communicate with other robots and humans, and to adapt their abilities to changing internal, environmental, and social conditions. Four key areas of research challenges are discussed, specifically for the issues related to the understanding of: 1) how agents learn and represent compositional actions; 2) how agents learn and represent compositional lexica; 3) the dynamics of social interaction and learning; and 4) how compositional action and language representations are integrated to bootstrap the cognitive system.

In “*An Architecture for Distributed High Performance Video Processing in the Cloud.”*[7]proposes the Split and Merge architecture for high performance video processing, a generalization of the Map/Reduce paradigm that rationalizes the use of resources by exploring on demand computing. To

illustrate the approach, they discuss an implementation of the Split and Merge architecture that reduces video encoding times to fixed duration, independently of the input size of the video file, by using dynamic resource provisioning in the Cloud.

"*Market-OrientedCloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities*." [8]presents a 21st century vision of computing. It identifies various computing paradigms promising to deliver the vision of computing utilities. it defines Cloud computing and provides the architecture for creating market-oriented Clouds by leveraging technologies such as VMs. It provides thoughts on market-based resource management strategies that encompass both customer-driven service management and computational risk management to sustain SLA-oriented resource allocation. It then presents some representative Cloud platforms especially those developed in industries along with our current work towards realizing market-oriented resource allocation of Clouds by leveraging the 3rd generation Aneka enterprise Grid technology. Finally reveals our early thoughts on interconnecting Clouds for dynamically creating an atmospheric computing environment along with pointers to future community research and concludes with the need for convergence of competing IT paradigms for delivering our 21st century vision.

"*Robot as a Service in Cloud Computing*." [9]reports research on service-oriented robotics computing and design, implementation, and evaluation of Robot as a Service (RaaS). To fully qualify the RaaS as a cloud computing unit, they have kept their design to comply with the common service

standards, development platforms, and execution infrastructure. They also keep the source code open and allow the community to configure the RaaS following the Web 2.0 principles of participation. Developers can add, remove, and modify the RaaS of their own. For this purpose, they have implemented our RaaS on Windows and Linux operating systems running on Atom and Core 2 Duo architectures. RaaS supports programming languages commonly used for service-oriented computing such as Java and C#. Special efforts have been made to support Microsoft Visual Programming Language (VPL) for graphic composition.

2.2. Worksand literature on Cloud Robotics

The following work has been going on with regards to cloud robotics in the world; however no work has been done on cloud robotics in Pakistan.

SOA (the cloud computing model) is replacing client server model in large companies like Google and businesses like Amazon. SOA has been here in the past also but it was never used with virtualization and scaling. Businesses have been saving billions of dollars per month after this model was introduced.[10]

In April 2011, UC Berkeley College of Engineering used a cloud with its robot to decrease latency from 50X to 10X of human speeds. The robot used was a $400,000 PR2.Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding, UCB, 2011 [11]

Android Cellbots are being made in major companies across the world to help old people and as toys for children. These cellbots are inexpensive, user friendly and maintained remotely via the internet. [12]

RoboEarth is a global project to train a cloud with 3D images of all things in the world. Anyone could contribute using RoboEarth's software and 3D sensors[13]

Hasbro has made a small Cellbot football team that has the player substitutions feature available that replaces a damaged Cellbot within 30 seconds. The replaced Cellbot has all the information of the damaged Cellbot downloaded to it.[14]

US Department of Defense is working with DARPA on creating an all-terrain rapid deployment robot army that could be trained within seconds and deployed to field within hours. [15]

Liquid Robotics is using small underwater cloud robots with only sensors and movement hardware to capture deep sea and deep mine data for military and environmental purposes.[16]

2.3. Summary:

The project has been undertaken after several careful readings of the cloud computing and robotics literature. The team began their research on the topic before carrying out the project proposal and in this chapter all those works have been mentioned that have been worked on by scientists and engineers around the world. Since the field of cloud robotics is relatively new, the

literature review is also subsequently small; however, it describes some very major efforts done in order to make cloud robotics a reality.

Chapter 3

System Requirements Specifications

3. Introduction

This chapter introduces the software requirements specifications for the project, including the functional, nonfunctional and external/internal requirements of the system.

3.1. Purpose

The product in this document is referred to either as*The System* or as *The Framework* for convenience and reading ease. Although the full name for the to-be developed software is “Design and Development of Framework for Robotics using Cloud Computing”, as slated in the title of the document, it was however revised to a shorter and subtler name for the length of the document.

The Framework’s purpose is to allow an end user to send/receive and execute command messages on remotely deployed robots on the Android mobile platform. A subset of the overall product shall include a cloud service connecting all robots for efficient communication and operation. The cloud shall also be the reservoir for all resources including multimedia, data and algorithms. Along with the Android OS, the cloud used shall be based on an open source platform to allow a complete cost free essence throughout the project. This SRS covers the *Cellbot Controller, CellbotApplication*, the *Cloud Communicator*, *Cloud Service,* and *User Agent Admin*.

Following are descriptions of each in the following pages:

Cellbot Controller: This is the software required to communicate with the hardware of the robot, to translate commands from the user and mobile into simple robot language.

Android Application: Application running on the mobile device atop the robot, capturing images and processing them, while also communicating with the Cellbot Controller.

Cloud Communicator: An application that is continuously in contact with the cloud services above and the Android Application described above. It glues together both the layers via the Internet.

Cloud Application: The cloud host at the backend of all the cellbots, doing all major processing and storage as well as information uploads/downloads upon referrals from the cellbots.

User Agent Admin: The administrator of the system, who is connected to the cloud and may interact directly with cellbots.

3.2. Scope

This section describes what constraints, limits, and other similar bounds the project is to be developed in. These bounds are resource, time, or external factor based.

3.2.1. Final year project limitations:

The mobile cloud robotics framework is being developed for the final year project of BESE 14 at School of Telecommunications. The scope is thus very limited in terms of functional requirements and only baseline functionalities of

cloud robotics are being targeted, since the syndicate members did not own a fully made R&D setup and neither had the finances. The basic purpose of the development of this project is to give the team members a chance to apply all learned theories to practice. The science of software engineering taught during the last three years is to be put into concrete form. The development of the project itself is a secondary objective after the first objective – i.e. hands on learning.

Furthermore, the absence of abundant cloud computing and cloud robotics further so has provided motivation for developing a system to fill this need. Early adopters of any new technology have to face a chasm of problems, mitigating which is a challenge in itself.

3.2.2. New Technologies in their early development stages:

In addition to cloud computing, another benefit of this system is to expose the team to mobile computing. The Android platform has been chosen as the mobile development standard, mainly for the libraries available as open source that interface easily with almost all operating systems and vendors. However, the syndicate members shall not be developing a full blown application on the Android phone. Our first priority would be to make an application that runs perfectly on our own phone, and then work shall be done on making it compatible with other Android handsets. Later work shall be adding networking components to the application on the phone, so that it could connect via a Wi-Fi hotspot to an available cloud, but internet or Bluetooth connectivity features will not be put in. The application would be primarily developed with a touch based GUI.

3.2.3. Limitations on the robot:

Robot used would be of minimal functionality i.e. forward, backward, left, right movement and with minimal I/O processing and storage related to movement only. The cloud would not support functions other than those for robot commands, video processing, and path finding, path following. The cloud would handle no more than four robot nodes and two PC nodes at any given time. Inter robot communication modules shall neither be put in at this time. Current navigations would be done on a purpose built indoor environments and objects in them shall be very limited.

Most of the work would be done on the cloud, not on the robot, therefore any high level robotics would not be considered, e.g. object detection/recognition or object tracking. The cameras used for the robot would be basically the on-phone camera, thus any special EM spectra would not be catered for, only visible light.

3.2.4. Cloud size limitations:

Finally, our private cloud would be very small in comparison to actual commercial clouds; therefore it is not expected to work in the exact same manner as clouds built on commercial data centers that span hundreds of acres with thousands of servers. Our cloud shall be based on just four dual core processors that shall only serve as a demonstrative cloud setup with minimal horizontal (adding more systems) and vertical (adding more CPU) scaling capabilities. Although this much power would suffice to work out our image and video algorithms, it would at times give the impression of a non-cloud environment.

The team however, shall be looking at scaling from a miniature perspective, i.e. of a mobile application needs, so that the power of cloud computing within minimal resources could be displayed. Had the team decided to use desktop applications as node clients to the cloud, it would not have achieved much visible scaling, as almost all of the available resources would have been used, leaving none for peak use times.

3.3. Overall Description

The Framework to be developed is to bring robotics to the mobile world. To this day, robotics development requires a large amount of work to be done, not only by the hardware designers, but by the software engineering team as well. Not to mention that even the most menial robotics tasks require very complex hardware that both expensive and hard to design. Adding extra functionalities to a robot means putting on extra hardware. In most cases the hardware is quite large which requires a lot of battery power as well, thus adding surplus weight. Many people choose to stay away from robotics for the very reason of complexity and its direct costs (labor and financial).

3.3.1. Product Perspective

Until recently, some work has been done in companies like Google on using smartphones with cloud computing to achieve a similar robotics complexity as any high end robot. Researchers there have used smartphones, attached to an underlying mechanical base, as nodes with “remote brains” based on a cloud far away from them. The node would be connected to the cloud via a wireless network connection. In doing so, they have shifted the entire processing and storage load from the robot to the cloud. In effect no surplus

hardware is required for the robot (called a “Cellbot” in some cases) which only has minimal hardware requirements for its movement. Whenever the robot now has work to do, the cloud tells it exactly what to do after performing large calculations on its servers instead of on the robot hardware.

There are several robotics frameworks available for PC environments, but not many come for the mobile. The starting goal for the mobile phone will be the Android operating system. To reduce processing and storage requirements of the robot [i.e. images/video] will be stored on a cloud server. Windows Server 2008 with Hyper V is the target platform for the cloud services. It was chosen, because of its scalability and compatibility with windows based operating systems. It is not open source like Android but it is easier to configure and use than other open source cloud platforms.

Before the advent of Cloud robotics using Cellbots, it was not possible to make robots using a group of ordinary computers connected to some mechanical hardware. In most cases the hardware would be very sophisticated and then computers required would be server machines with large computing power. Even to achieve the level of functionality that is being tried to get with this Framework, a very large set up including a mainframe computer and 500 kilos of mechanical hardware would be required at the very least. A large team of technical people aside, a large test environment would also be needed with air conditioning and internal lighting. Hence the need for cloud robotics was felt and this invention has indeed made many lives easier.

3.4. Product Functions

The Framework consists of five parts, namely the Cellbot Controller, the Android App, the Cloud App, the Cloud Fabric and the User Agent Admin.

3.4.1. TheAndroid App

The Cellbot App provides the user with a very simple GUI from which to choose tasks from. The user selects tasks from the list upon which the app requests the cloud app for instructions. The downloaded instructions are passed on to the cellbot controller after data format conversion.

3.4.2. The Cellbot Controller

The cellbot controller provides no GUI and is simply working in the cellbot app's background. It takes one way instructions from the cellbot app and passes one way error messages to it. The instructions received from the cellbot app are converted to hardware specific language for the robot base. Its basic instructions are vector movement: forward, backward, left, right along with the distance in measure.

3.4.3. The Cloud App

The cloud app provides an interface to the cellbot app for video upstreaming and instructions downloading service. It compares the video being received with the already stored videos and creates algorithms to be sent to the cellbot app. It then simultaneously stores on the cloud and sends the algorithms to the cellbot app. The app also uses the object database on the cloud fabric to create on the fly algorithms in case there is a shift in a stored video and its algorithms. It continuously informs the cloud fabric of its resource requirements and its current resource usage levels.

3.4.4. The cloud fabric

The cloud fabric stores all the object images and algorithms and a database is used to handle their storage dynamically, meaning on-the-go. The fabric decides the amount of processing power required by the cloud app on a real time basis (scaling). It provides an interface to the user agent admin for monitoring and maintenance. The data stored on the fabric is indexed by a SQL database.

3.4.5. The User Agent Admin

The User Agent Admin gives a GUI for controlling the cloud fabric and cloud app. It provides all kinds of tools for observing resource usage and health, maintenance, configuration and functional overriding. It allows the user to completely shut down the Framework and restrict certain Android users from using cloud services.

3.5. User Classes and Characteristics

There are two kinds of users of the system, the Administrator and the Cellbot user. Their detail is given in the sections that follow.

3.5.1. Cellbot User:

The Cellbot user would be the owner of the Cellbot. His requirements include the movement of his Cellbot in an environment. Some tasks would be available at his disposal that he could make his Cellbot do. The presence of a cloud environment is unknown to this user and so are other technical details including the internal functioning of the Cellbot. The only technical details that are relevant to this user are the error messages received on his Cellbot application GUI regarding the Cellbot malfunctioning. In essence, he owns an

Android phone (along with the robot base) on which there is only one application installed – the Cellbot app.

This user is required to be proficient in English language and Android smartphone basics. He shall be able to install, uninstall and configure his Cellbot application. He is not required to have any hardware skills. However, he could attach his Android phone to the robot base and is quite efficient in connecting them via the Cellbot app GUI.

3.5.2. Administrator

The administrator is a technical person skilled in software and hardware installation, configuration and maintenance. He would be responsible for setting up the Framework from scratch and has knowledge of Android development, cloud development, operating systems and network configuration. He shall be handling the Framework via the User Agent Admin GUI. He shall be proficient almost all kinds of computer technology.

His responsibilities shall include maintaining the system, monitoring the cloud health and installing new features on the Framework when needed. He is also the creator of the test and deployment environments in which the Cellbot shall move around. The total infrastructure of the Framework is also set up by him.

3.6. Operating Environment (OE)

The Framework is to be designed around aMicrosoft based environment; therefore the environment of choice is the Windows Server Operating System with hypervisors for virtualization. The various kinds of OE are listed next.

3.6.1. The cloud and its OE:

For the cloud fabric, the Windows Server 2008 R2 Hyper-V hypervisor, released in 2008 as a long term support version of Windows Server shall be used. It contains support for open source cloud environments as many software packages come preinstalled on it to enable cloud based solutions to function smoothly. Many files and OS functions are built in specially for enabling cloud based applications. Also Linux is installable on ordinary Intel PCs and require little memory and storage for its existence.

For the purposes of virtualization and scaling, Hyper-V shall be installed on top of Windows Server, so that the cloud application could be given only the needed resources on demand. Hyper-V is a free world class cloud computing solution used by large companies for research and business purposes. It was selected as it is easy to use and has good documentation available online.

3.6.2. The Cellbot and its OE

The Cellbot shall be controlled by Android 2.3 OS that comes prebuilt inside most smartphones today from major vendors like Samsung and Motorola. The software development kits for Android are freely available and developers can make applications for the phones and readily deploy them.

The other part of the Cellbot shall be a robot base from iRobot ™, the Create ® 1.0. Create has a small body with wheels and can move at up to speeds of 5 km/h. it allows itself to be integrated with any Smartphone, and comes with a coding specification for developers. All applications developed for it must conform to the messaging specifications mentioned.

3.6.3. Languages used in OE

The project shall use C# and Java for all our development, as the Android OS is a java only platform. Also, Java is open source and it is a goal to make this Framework partially open source.

3.7. Design and Implementation Constraints

The cloud to be developed is a private cloud, not a public cloud. A private cloud is established on the local network of a company or institution and allows only specific people to access it via wired or wireless connections. Also our cloud would be based on three desktop computers, not server machines; hence all the equipment of a desktop computer (monitors, keyboards, mice, etc.) shall be part of the physical cloud, although useless to its overall function.

The Cellbot shall be kept in a limited environment and the reasons for that have been explained in section 3.2. The hardware of the cloud is outside the scope of the document and will not be discussed here.

3.8. User Documentation

The Framework shall be delivered with an illustrative user manual required for the Cellbot app that shall describe how to use the application. A troubleshooting guide shall also be given to help the user with technical problems like common errors and general maintenance.

A system admin guide shall be provided to explain cloud and Cellbot installation, configuration and maintenance issues. The entire protocols

governing the data flow along with relevant diagrams shall be included. A small user guide for the User Agent Admin shall be provided.

All the other documents of the development cycle, including the SRS, the analysis and design documents, the test cases and results shall also be provided. Source code shall be given in digital form.

3.9. Assumptions and Dependencies

There was a plan to test and deploy the Framework in a test environment. Usually robots like line followers are made to function in a custom-built environment that has minimal colors, objects and materials other than those required by the robot to carry out its task. Our deployment environment would be very simple as well. It would be a 16 foot squared area with one foot tall walls that would restrict the Cellbot vision to that environment only.

The reason for using such a deployment and test environment is directly inherent from our scope requirements. The team did not have the resources and time to make a commercial or first class level cloud robotics platform therefore it is not being developing it for external or indoor environments. The team shall train the Framework around our tailor made environment so that work could be done on a limited scope and targets achieved. Our target is not to build a large scale cloud robotics platform, but only to make a workable example of how cloud computing could be used to handle robots. Therefore our dependencies are as follows:

1. The Cellbot shall be operated in the environment that has beenbuilt for it.

2. The cloud shall not be used for any other application other than the cloud application that is deployed on it.
3. The Cellbot shall have only two tasks given to it for the purposes of the project
4. The software platforms shall remain the same for the cloud and the Cellbot, i.e. Windows Server 2008 with Hyper-V for the cloud and Android OS 2.3 for the Cellbot, respectively.
5. The hardware platforms shall remain the same for the project, i.e. the robot base would be the Create ® from iRobot ™.
6. The materials placed as obstacles in the path of the Cellbot at run time shall be from a very limited object set.

3.10. External Interface Requirements

The following sections list the external interface requirements.

3.10.1. User Interfaces

There are three user interfaces of the Framework: 1) the Cellbot app GUI for the Cellbot user 2) the User Agent Admin GUI for the system administrator and 3) the Cloud App that is accessible for the Administrator only for observation.

The purpose of the Cellbot GUI is to receive touch screen commands from the user for tasks to be done by the Cellbot and to display any error messages. The screen shot for the Cellbot app has been shown in figure 3.1 below, but the sample below is just a prototype GUI to be added to later.

Figure 3.1 Main screen for Android App at application launch, with example error

It does not show the subsequent screens, e.g. the video that shall start once a task is touched and the error message screen is not shown as well. The user

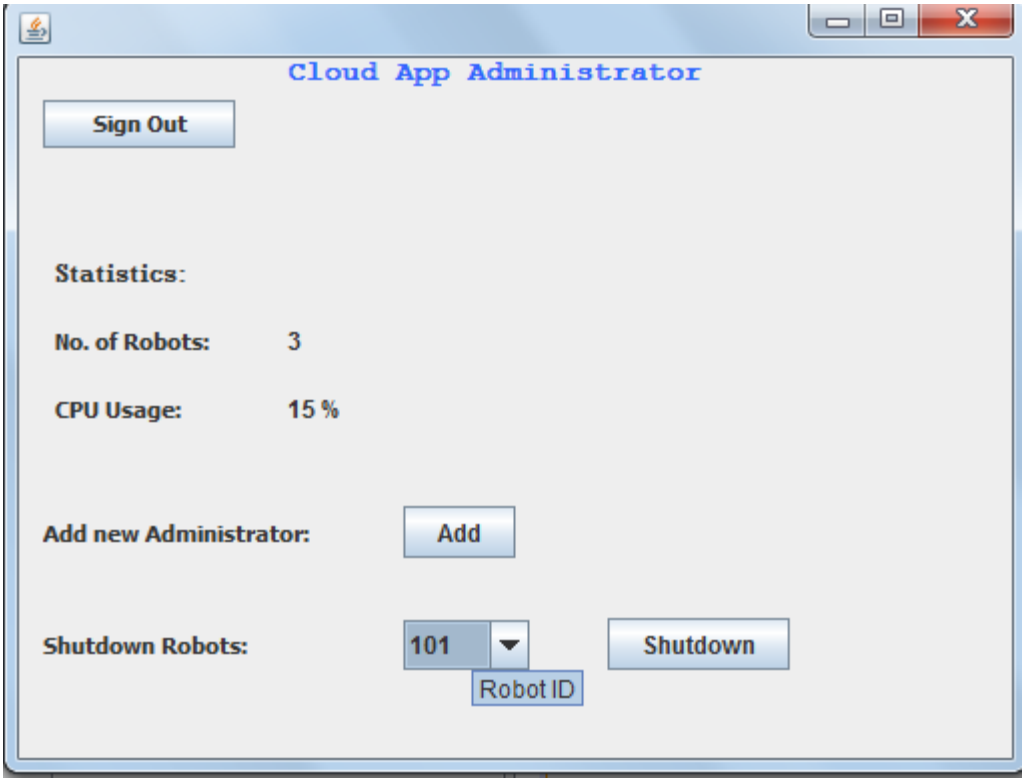

Figure 3.2 The Cloud Admin on its launch, gives status of core usage as well

interface for the User Agent Admin (shown in figure 3.3) application shall be like any desktop management program with tools for monitoring, controlling and configuration. The user could click on available buttons for the required action. The cloud app would be running on the cloud and cannot be handled

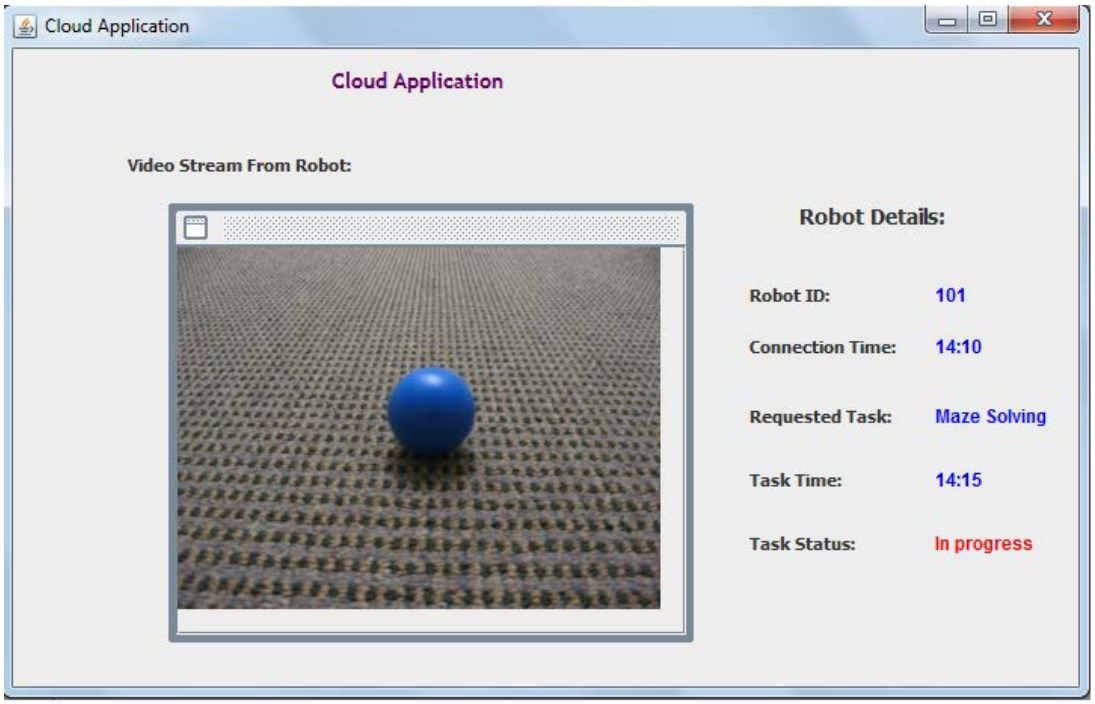

Figure 3.3 The Cloud App on its launch

by human effort; therefore the Administrator would only be allowed to view what it is doing at any given time.

3.10.2. Hardware Interfaces

The cloud shall be controlled by a combination of Windows Server 2008 R2 and Hyper-V while the Cellbot shall be under Android control. The User Agent Admin shall be housed on a laptop with Windows 7. All three components shall be connected via an 802.11g Wi-Fi router. As shown below in figure 3.4 the hardware communicates with each other in a multifaceted manner.

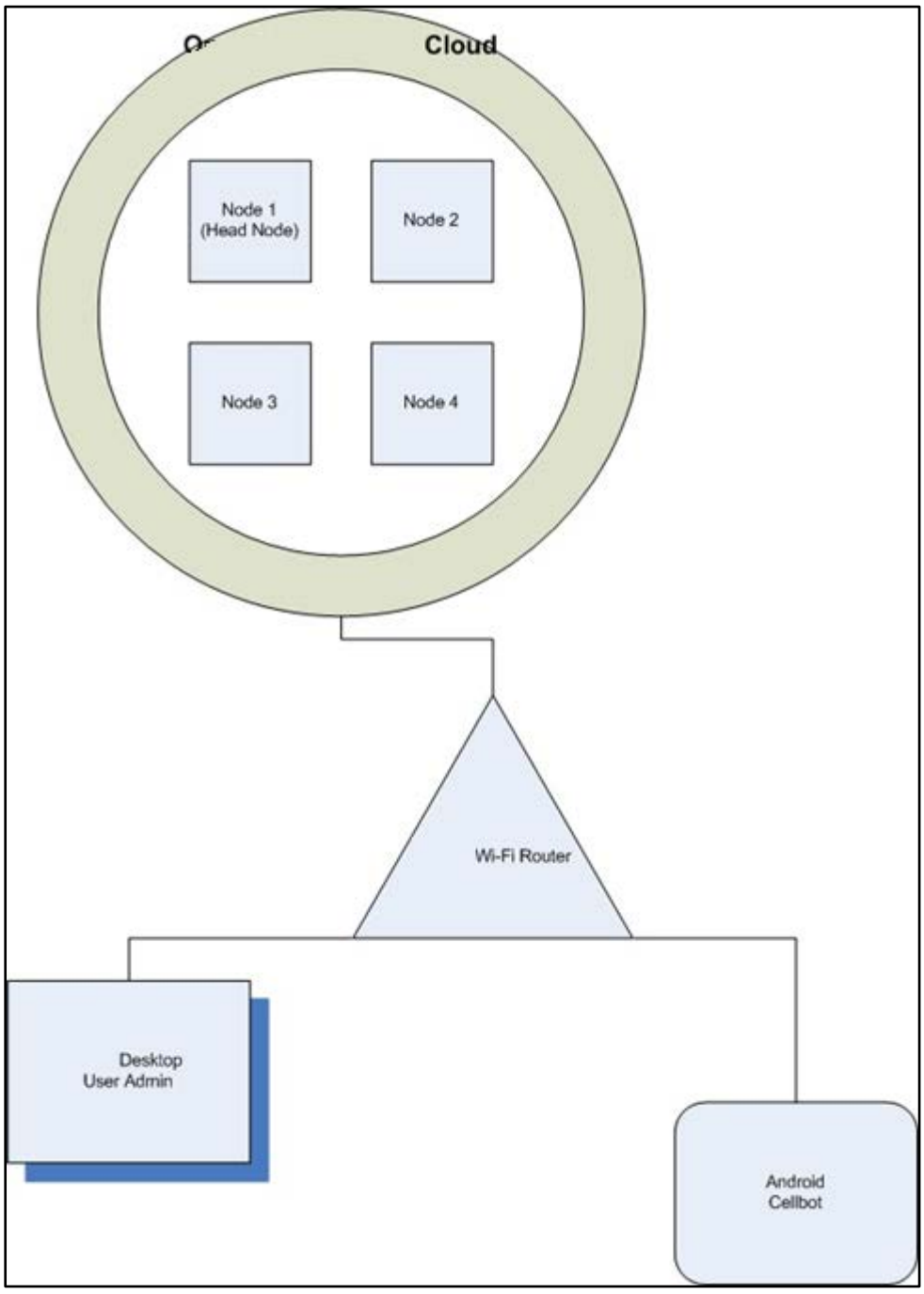

Figure 3.4 Hardware Interfaces

The cloud has four computers (nodes), with one of the computers for network connectivity (head node); The Cellbot has a capacitive touch screen enabled Android device.

3.10.3. Software Interfaces

The android application on start sends connection request to the application running on cloud; the app running on the cloud accepts the connection request and sends the list of available services/tasks in the cloud, the android app waits for the user to select a task, upon selection of the task the android app sends the user's requested task request to the cloud, the cloud app receives the user's requested task and on the basis of that requested task it starts the algorithm (it would be a computer vision algorithm) of the requested task. The software interfaces are shown in figure 3.5 below and it shows

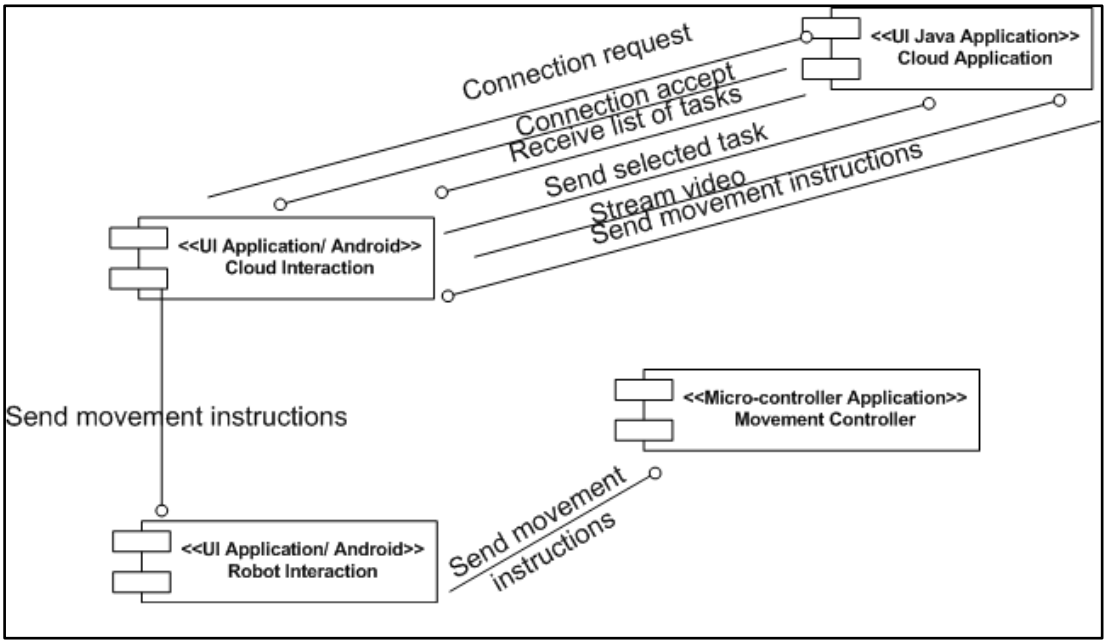

Figure 3.5 Software interfaces

certain components of the system connected with each other over software interfaces.

The android app starts streaming the video feed from Cellbot App to the cloud app, the cloud app receives that video feed and performs processing on the video feed, the output of processing is the movement instructions (forward, left, backward, forward), the cloud app sends these movement instructions to

the android app, the android app receives these movement instructions and pass them to the software burned on the robot's micro-controller, and on receiving these instructions robot starts moving in accordance with the instructions.

3.10.4. Communications Interfaces

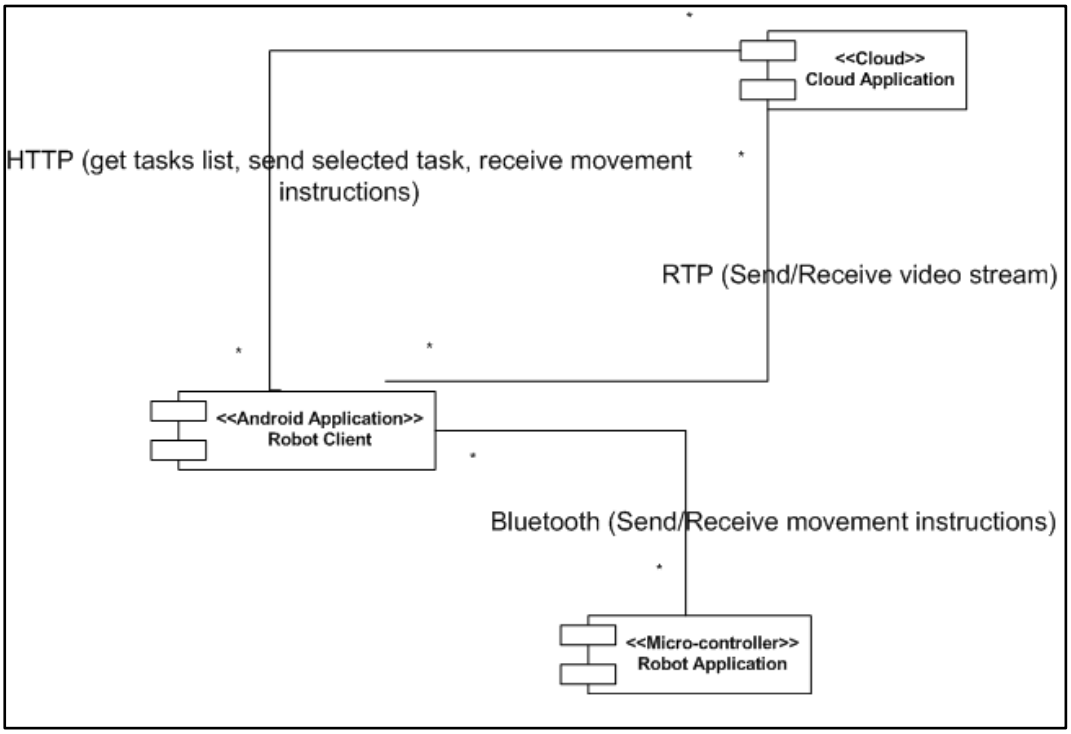

Figure 3.6 The software communications interfaces

A. Communication between the cloud application and Cellbot App:

Figure 3.6 above shows that to get the services from the cloud initially a connection will be established between the Cellbot App and the cloud using the HTTP protocol. After establishing the connection the user will be able to request a task from the cloud and the cloud will be able to send the movement instructions (move straight, move back, turn right and turn left) to the Cellbot app on the robot using the HTTP protocol.

B. Live Video Streaming:

For getting live video stream from the robot, the RTP/ RTSP (real time streaming protocol) will be used. The RTP connection will be initially established as soon as the user selects a task on the user interface, after establishing the RTP connection the Cellbot App will start sending the video stream to the cloud application and it continues to stream until the application is closed by the user.

C. Communication between robot's micro-controller and Cellbot controller:

The communication between the Cellbot Controller and robot's micro-controller takes place by the communication protocol called Bluetooth. By using this protocol robot's micro-controller can receive movement instructions from the controller.

3.11. System Features

This section describes the features of the system and their functionalities in detail. Each feature has a description, a stimulus/response sequence that tells how the users are expected to behave with the feature and what are the functional requirements from the feature.

3.11.1. Path planning service for Cellbots (Priority=*high*)

A. Description:

The cloud shall provide the service to Cellbots connected to it for navigating pre planned paths. These paths would be known to the cloud only and all

Cellbots connected to it would not have any knowledge of how to walk those paths. This path planning is a simple demonstration of how cloud computing could help robots in working without large hardware on their backs. The two types of paths that would be available to a Cellbot user are a simple straight line and a curved line with walls that arereferred to as a Maze. Selecting one of these tasks would make the robot first search for a line or a maze and then start treading it according to the instructions of the cloud.

B. Ideal path without real time obstacles:

The cloud shall have a Cellbot streaming live video of the path to it. Since the path would be already known to the cloud and not to the Cellbot, the cloud would help the Cellbot at each step of the way by sending specific instructions for every part of the path. The instructions sent by the cloud would include messages containing vectors and speed information, i.e. where to move, how much to move and at what speed. These instructions would be fed down to the Android app, which shall convert them to robot specific language and send it to the Cellbot controller, which shall make the robot base move.

C. Path with real time obstacle:

A problem could be encountered if a small object is placed in the path of the moving Cellbot while it is receiving instructions from the cloud. This object was not anticipated by the cloud. In this case the cloud would calculate on the fly an algorithm for the Cellbot after reviewing the live video stream and the object in front. It would send these new instructions as soon as it calculates

them, and the Cellbot would keep on moving after receiving them, going around the object or avoiding it in some other way.

D. Stimulus/Response Sequences

1. User selects task from the given task list on Cellbot App GUI.
2. The Cellbot App sends the request to the cloud and starts streaming a video upon response
3. The cloud tells the Cellbot to look for the path and takes information from the video being sent
4. Upon the cloud's detection of the path the Cellbot is given instructions on how to walk the path
5. In case an obstacle is not encountered on the path, the Cellbot is given instructions as they were once stored on the cloud
6. In case an obstacle is placed in real time on the path, the cloud calculates a new path and sends instructions for following to the Cellbot.
7. Upon path completion, the Cellbot is given instructions on how to walk back to the start of the path.

E. Functional Requirements

1. Cellbot app be in working state and show options for task selection
2. Task selection can be of two types: Line following or Maze solving
3. Video streaming to cloud is done at start of task till task completion
4. The Cellbot is under cloud control till task completion

5. The Cellbot App displays the video on the screen along with information on path completion percentage and cloud connection health
6. The Cellbot App displays when exactly it is downloading information from the cloud via an indicator e.g. a blinking light or status light on its GUI

3.11.2. Path and object information downloading from the cloud (Priority = *high*)

A. Description

The Cellbot does not have any information of the paths or objects placed therein on its hardware. It is the responsibility of the cloud to store details of the paths and objects on its database and provide their information to the Cellbot when required. The cloud has a large storage space and CPU power as well as memory that the Cellbot shall not have, therefore the cloud shall send information to the Cellbot in the form of movement instructions only after analyzing these paths and objects via the video being streamed to it and comparing them to those stored already on it.

This repository would be first trained to it by the Administrator. The cloud would be empty at the start and gradually be filled up by information of some objects and path algorithms by the Administrator. This information would be stored in a file system that would be indexed by a database. The database would provide the cloud app the information of these paths and algorithms when needed. This information request would not be invoked by any user or Cellbot, but by the cloud app itself.

Training of the objects would be done by entering information on size, shape and other features like photographs from various angles by the Administrator. Similarly path knowledge would be trained on the cloud by providing frame by frame directions to be stored in the database.

B. Stimulus/Response Sequences

1. The Cellbot app requests the cloud app for support on path planning
2. The cloud app analyzes the video being streamed to it and queries the database for information
3. The information is sent to the cloud app that uses images from the cell bot and compares them to the images from the database and selects an algorithm for movement based on certain metrics.
4. The information is downloaded to the Cellbot app by the cloud as movement instructions
5. Information is used for an object and sent down to the Cellbot in case an object is placed in the Cellbot's way in real time.

C. Functional Requirements

1. A database is installed on the cloud with relevant fields for path plans including dimensions, photographs, movement algorithms and others.
2. The cloud app communicates with the database whenever it needs to send instructions to the Cellbot app for movement.
3. The database would be accessible for change and training by the Administrator during all times.
4. Objects that are doubtful to the cloud app would be placed on a doubt queue for later resolution by the Administrator.

3.11.3. Simple system administration (priority = *high*)

A. Description

The cloud is supposed to be regulated throughout its functional life. This includes managing the cloud's resources, training the cloud, monitoring the cloud's usage, checking the cloud's health and debugging. All of these functions are available only to the system administrator via a GUI provided on the administrator's computer, which is not part of the cloud itself but instead connected to the cloud like the Cellbot.

The administrator shall have his own application to handle the framework, the User Agent Admin. This application would have GUIs for easy management that would allow the cloud to be dealt with at all times. The cloud would require handling at any given time.

This is also a To-be-added feature because Hyper-V gives its users many options for managing the cloud; however a detailed list of available management rights is out of this document's scope.

B. Stimulus/Response Sequences

1. The administrator opens up the User Agent Admin application from his computer.
2. The application establishes a connection to the cloud and downloads the current health and activity metrics of the cloud.
3. The administrator is shown graphs for resource utilization, scaling being done on the resources, number of Cellbots working with the cloud and task being performed.

4. The health of all four nodes is also available for individual checking and management, that tells if the node is currently connected to the others and if they have any software issues.
5. The administrator shuts down the cloud for maintenance if the need be.
6. The GUI gives option for connection closing, which disconnects the cloud from receiving videos from the Cellbot.
7. Training could be given to the cloud for new paths and objects.
8. The administrator opens up the database for entering information about new objects
9. Options for viewing the database and searching is provided
10. If new information is to be entered about a new path and/or object, the database allows the creation of a new entity.
11. For modifying information or adding information to an old path and/or object, the database allows the administrator to modify an old entity.
12. The administrator enters the new information

C. Functional Requirements

1. GUI for the User Agent Admin be very simple to understand and easy to use
2. The GUI should be window based and have keyboard shortcuts for easy access along with mouse control.
3. The User Agent Admin should have access rights and usernames and passwords for accessing it should be stored on the cloud
4. The GUI should have all available options for management on the left side of the main screen and once an option is clicked on, it should

open up on the right side of the main screen. The list of options should stay at its position at all times.

5. The options should be according to Windows Server's Server Manager or the console should be integrated into the User Agent Admin.
6. The data to be trained into the database should include photographs and their metadata.

3.12. Other Nonfunctional Requirements

This section lists down all the nonfunctional requirements that include the performance required from the cloud and the cellbot, as well as safety and quality standards that the system must adhere to.

3.12.1. Performance Requirements

The cloud shall be responsible for the performance of the entire framework; however the Cellbot and the user agent admin also have their specific performance requirements.

A. Cellbot

1. The Cellbot would be required to process tasks given to it within 100 milliseconds of the user pressing the buttons on screen.
2. Movement speed of the Cellbot shall be not less than 3 km/h under stress free and error-free operation.
3. Any task that requires more loading should have a progress indicator.
4. The user should be informed if the Cellbot is busy in receiving commands from the cloud

5. The Cellbot should not delay movement after the upper layers have given it the commands to move.

B. Cloud

1. The cloud should be scalable from 1 to 10 Cellbots with a 2 seconds response time.
2. Storage should be available for up to 10 Cellbots with an average of 10 GB per Cellbot.
3. There should be a 99% uptime for the cloud.
4. The processing power given at any time should be 90% of the maximum available CPU.
5. The cloud should quickly calculate on the fly algorithms and send them to the Cellbot

C. User agent admin

1. Given tasks should be responsive
2. Network errors should be handled gracefully
3. Tasks that take a lot of time to start or load should have progress indicators

3.12.2. Safety Requirements

The product does not have any important safety requirements. However users with radio waves related problems should not use the system. Some people have problems with touch screen phones e.g. electric shocks to sensitive skin. They should also not use the system.

3.12.3. Security Requirements

1. User Agent Admin should only allow restricted access and enforce certificates for login and user/data authentication.
2. The cloud nodes would not be accessible without user authentication. For both the User Agent Admin, and cloud nodes a singular entity would be responsible and no one else would be allowed access.
3. Cellbot does not require any user authentication nor any data authentication as all the data transferred from it to the cloud and back would be insensitive.

3.12.4. Software Quality Attributes

1. The software for the Framework should be reliable and appear error free. It should recover from failures within one minute and also display user friendly error messages.
2. The Framework should start up and be ready for function within five minutes and require no extra effort for initialization. All details of starting up and shutting down should be invisible to the users and administrator.
3. The GUI of the Cellbot and user agent admin should be fluid and smooth to use and must conform to all standard human computer interface principles.
4. Network connectivity should be made as smooth as possible and as fast as possible.

3.12.5. Other Requirements

This section describes the remaining few requirements of the system:

1. The product has components that are open source and therefore does not require any legal requirements.
2. Databases developed for the cloud shall be open source as well and based on MySQL. This database shall be controlled via the User Agent Admin component and deployed on the cloud to work with the cloud app and fabric.
3. The router that connects all three components should follow standard networking protocols. It should be in good condition and be fault free.

3.13. Summary:

The chapter gave a formal specification of the objectives that are to be met by the project syndicate members in order to bring the project to completion. The chapter gave an overview of the methods and techniques that shall be used to deliver the project and finished with certain use cases and other diagrams to outline the functional and nonfunctional requirements of the system.

Chapter 4

System Design Specifications

4. Introduction

This chapter describes the various models used as blueprints for the development of the project and its implementation during the next phases of the project lifecycle. It also describes in detail the various large and small components of the system that shall serve as a reference for future iterations of the design phase. Lastly, it is a supplement to the previous document, the Software Requirements Specification (SRS) submitted at the start of the project. Together, these two documents provide a clear roadmap for the development of the Cloud Robotics project.

4.1. Scope

The overall picture of the project is as follows: A small scale cloud is being used to drive a small dumb robot through a set of controlled indoor environments using only a camera and network connectivity. The video feed obtained from the camera mounted atop the dumb movable robot is used by the cloud for processing and instruction creation. The instructions are meant for the robot and it understands them as a set of commands for four way movement and speed. To achieve this goal, two applications, a server and a client shall work in tandem, over a wireless LAN, while the robot shall outsource 90% of its performance requirements and 99% of its storage requirements to the cloud.

The dominant design pattern used is the famous Model View Controller (MVC) paradigm. This was due to the nature of the project, which includes a cloud. Since a cloud is involved, it intrinsically relies on MVC, since for

clouds;a Presentation-Platform-Information model is the basis of development. In a cloud environment, Presentation is provided by end user clients like browsers, the Platform is always the hypervisor atop physical hardware, and the Information is also stored on the cloud and accessed via the hypervisor.

However, even though it is intended to build an application for cloud environments, one has to limit the scope due to time and hardware constraints and scale down to a very small version of a cloud, such that only demonstrative capabilities of the cloud are developed, in order to showcase an appreciation of the power of Cloud Computing. The purpose therefore, is to solely work on making a very basic cloud, not to build a large datacenter level cloud that handles millions of transcontinental clients.

Our cloud shall be based on two server machines and operate in a LAN environment with a shared storage. It shall only be developed for the robotics application being developed parallel to it as part of this project. The domain of the cloud shall not cover aspects such as Migration and public cloud bursting; neither shall it offer PaaS and IaaS versions, among other features available in commercial clouds today.

The Android App shall also follow the MVC architecture, as it would only allow its user to see services and shall not let the user access any data or “business logic”. The term Business Logic, or BL, shall be used somewhat throughout this document even though the processes described as part of the applications do not have any business-like nature.

The Android App shall be installed on an Android phone and shall only be able to upload videos and download text from the server deployed on the cloud. It shall have no other use except to transfer video to the cloud and get instructions from it for the robot underneath.

There shall be a total of three operation environments for the robot: two mazes and one wall-less path denoted by a colored line on the floor. All of these mazes shall be built by us and the application shall be configured to walk the robot through these environments only. The robot shall not be able to use services of the cloud to walk through other environments, external or internal.

Finally, since this robotic cloud is to be built on two 64 bit desktops, the team shall not be very keen on observing real-world cloud performance. In other words, there should be latencies involved, as well as downtime problems. However, since this application is very small in size compared to the two machines combined, it shall be supposed that it is deployed on a cloud environment.

The main driving force behind the creation of a cloud instead of a traditional server is that this cloud shall provide a single interface for any number of similar robots after one robot has successfully completed its predetermined path or maze. New robots with no knowledge of previous experience in these lab environments would simply have to download instructions and run them, instead of going through the time consuming path learning experiences.

4.2. Overview of the System Design:

The forthcoming sections are already mentioned in the Table of Contents and the figures in the document are indexed in the List of Figures at the beginning of this document. However, here is a rundown of all the sections in brief detail.

4.2.1. Deployment Diagram:

It describes the very high level architecture of the system, by listing in figures, all of the main parts of the system. The term “components” has not been used, specifically because there is a separate components diagram later on, which details all the technical components. The deployment diagram shows how the system looks from the hardware world in a non-technical context. This diagram will give the first impression of how the large parts are linked together and what the total parts comprise without going into unnecessary details.

4.2.2. Architectural Design

The architectural design, or the component diagram, shows somewhat more technical details than the deployment diagram. It shows an insider’s perspective of the system by describing the high level software components that perform the major functions to make the system operational. It overlooks the hardware aspects that were told in the Deployment Diagram.

4.2.3. Data Structure Design

UML class diagrams and database tables are described in this section. Class diagrams are integral for the description of low level components of the

software that include data storage and state details. Classes provide the means for the system to perform its functions. Tables stored in the database are also defined here. These tables shall be used in a relational DBMS for permanent storage of organized data.

4.2.4. State Chart Diagrams

These diagrams show the dynamic nature of the static UML classes. They describe how the system shall behave under various circumstances and data sets. They give the allowable states a dynamic class is to have, by showing it in various scenarios.

4.2.5. Use Case Realizations

Use cases tell how the user shall be able to interact with the system and how the system shall respond in return. They provide a means to show the functionalities of the system in a user-concentric manner, without paying attention to the dynamic behavior of the system or its underlying working. They directly relate to the functional requirements of the SRS.

4.2.6. Activity Diagrams

Activity diagrams follow a workflow based approach to describe the overall functioning of the system. They are a very good means to see how the steps involved in major tasks inside a system without going into technical details using a flow chart pattern.

4.2.7. Sequence Diagrams

Sequence diagrams show how different objects are involved in the completion of a functionality of the system. They have a unique format that allows the

reader to see how many objects are used and for how long for the completion of a system requirement.

4.2.8. UI Design

Some snapshots of the various graphical user interfaces are shown in this section that prototype the way a user shall be interacting with the system in a Windows Operating System on the cloud and an Android OS on the robot.

4.3. Design Models:

This section describes in much detail all of the design activities that have been carried out to model the system requirements into a concrete shape.

4.3.1. Deployment Diagram

The deployment diagram provides a physical look at the system with each processor and device indicated. Figure 4.1 below shows how each major physical part of the system is connected to each other in a deployment scenario.

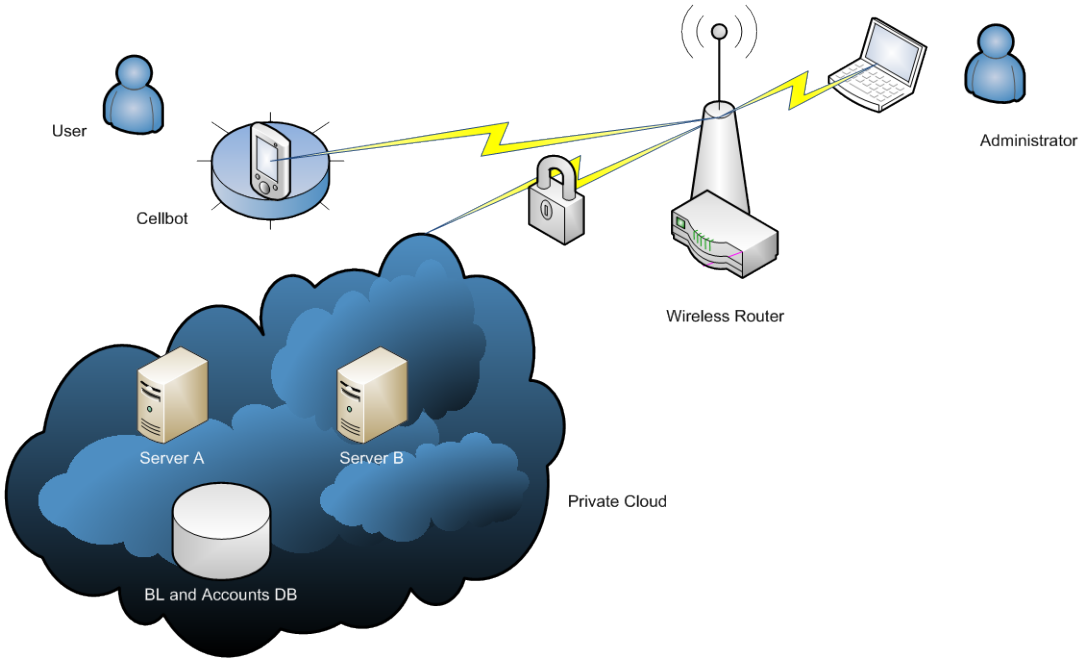

Figure 4.1 The deployment of the product, showing major physical components

Each physical location will have its own software unit and units in different physical locations will collaborate to provide the services that logically seem to be straddling the units.

In our system, there is a wireless router that creates a wireless LAN environment. All other hardware components connect to each other via this router through their network interfaces.

The cellbot connects to this router via an Android phone's Wi-Fi sensor. This cellbot can operate as long as it is in the Wi-Fi range of the router. The cellbot consists of an Android phone and an iRobot Create ® wheeled robot. The Android phone is connected to the robot via a Bluetooth device called the BAM Wireless Accessory via the phone's Bluetooth sensor.

The Private cloud (called *private* due to LAN-only availability) consists of two core i3 64-bit desktop machines that are configured with Windows Server

2008 R2. These two machines are connected via Ethernet and have a hypervisor running on them. This hypervisor gels the two machines into one single virtual machine with combined physical resources and power. Any application installed on either machine would be available on both machines in real time. These “cloud servers” also host the two databases to be used for the project. One of the machines shall be set as the interface machine to which the outside world shall connect. However it shall appear no different than the other desktop, and to the outside world both machines would appear as one.

Lastly, the Administration of the cloud shall be done remotely using a laptop computer that shall also connect to the rest of the system via the wireless LAN.

The two actors highlighted in the diagram, the User and the Administrator are the only two logical users of this system. The User (with a capital “U”), is supposedly a cellbot owner and has access to the services of the cloud through his Android application. The Administrator on the other hand, is a technical person and has full privileges to change all aspects of the system at will. He is responsible for maintenance and controls the system via his Administration portal on his laptop. He has put in place the protocols for the network and has defined the data in/outflow mechanisms. The word *Architect* was not used for this role, since it implies a non-maintenance nature. The Administrator also makes use of the cloud databases and performs troubleshooting tasks via his Administration portal.

Note that the cloud connects over a secure connection to the wireless router. It only allows certain people to access its services, using a username and password.

4.3.2. Architectural Design

Figure 4.2 below shows the previously mentioned components, their relationships with each other and theirconnection with the actors in the software design.

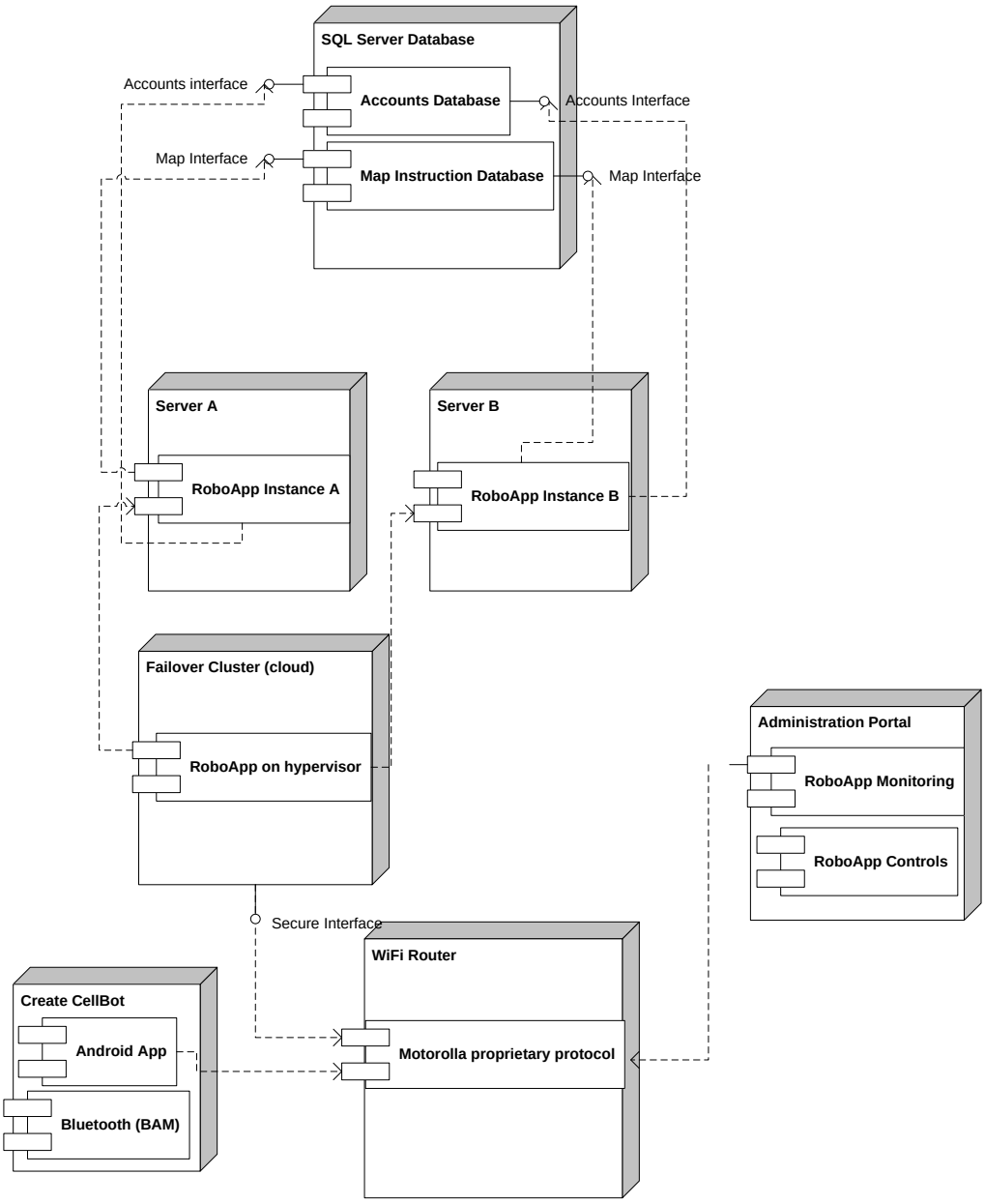

Figure 4.2 High level architecture shows the major software components

. At the highest level shown here each part has a name, an interface and an abstract detail. Here, abstract detail means a written down purpose, the given functions, the qualities (e.g. if it is dependent on other parts) and the constraints under which it must operate. It also tells about the needed resources, for example, any parts used by other parts that are outsiders to the

design, like physical devices (e.g., routers) and file share resources (e.g., software libraries).

All entities above are described below:

4.3.3. TP Link Proprietary protocols router:

A. Specification:

The TP Link IDV 350(w) router is a 3.5G indoor CPE and is based on a BCS5200 CPE chipset for indoor, small office/home office use. It is has Wave 3 modem capability and a VPN/WAN router platform with a surefire quiet-latency voice control. The Becham BCS5200 WiMAX CPE terminal thing consists of a fast working baseband Internal chipset and a 2 band straightconversion RFIC for operation in 2.3, 2.5 and 3.5 GHz bands. The BCS5200 gives a base platform for multiple CPE SKUs for WiMAX data alone as well as support for enhanced QoS for VoIP and other real time services. The project shall be primarily using it in the 3.5 GHz range.

B. Interface:

The router has five RJ-45 sockets, out which four are for router to PC connection, while one is for LAN or ADSL Modem to router connection.

4.3.4. Create ® Cellbot:

A. Specification:

The Android App is a stand-alone application that resides on an Android phone. The team has called it simply the “Android App”, since it is the only component in the entire system installed on the Android phone and also

performs minimal functions. It is based on Android version 2.3 release and follows a predefined Google API standard for smartphone application user interface. It shall be designed and built in the Android SDK available freely. The purpose of the application would be to use the camera on the phone to capture video streams and upload it to the cloud server. Its secondary objective would be to download instructions from the cloud and send them to the robot over the BAM accessory.

Bluetooth Adapter Module (BAM)gives the power of wireless control of the Create cellbot via a cloud server. The BAM attaches itself to Create's cargo bay, and you don't need any cables for it. BAM gives an imaginary serial port connection between the cellbot and a Bluetooth node (in our case, the Android). The Android phone could then talk with Create exactly in the same way it would if it were connected with a real live serial cable. BAM gives the human user the total wireless experience of the cellbot. It also shows Create's programmable IO, making it easy to connect more useful hardware.

B. Interface:

BAM has a Bluetooth Serial Port Profile available for custom configuration and scanning purposes. It lets a high power Class 1 Bluetooth radio be operated over large frequencies in the Wi-Fi range. The interface is controlled via an application installed on the Android phone. However the team shall be making this interface part of the Android App that shall be made as part of the project.

The Android App that shall be developed will have a standard touch interface with start GUI buttons and selectable commands. There shall be no command

line functionality to ease user friendliness. The user shall be provided with simple logical options to start and stop any service that is available.

4.3.5. Failover Cluster (cloud):

A. Specification:

thefailover cluster version of a server has become popular as an alternative to traditional Internet hosting. Special software called a hypervisor is used in conjunction with a server operating system to glue together multiple machines and to provide clients with a set of available virtual resource pools on which they may install their software. The benefit of such an approach is derived from highly available grid computing. This cluster shall be available to the Android App running on the cellbot to communicate with the database. It depends on the cloud's hypervisor, that from which exact server it provides the service to the cellbot. It could be either server A or server B, or even both. Both these servers would be accessed through a secure interface of the cloud. These two servers would jointly be hosting the Cloud App, hence forward called so, which is the heart of the system. The Cloud App would be responsible for handling the major functions of the system including image processing, database connectivity, sessions, and cloud scaling.

B. Interface:

The cloud is configured to let in only users with permitted usernames and passwords. These credentials would be stored on a database hosted on the cloud. The interface would only allow secure connections by storing the state

of the User and transferring data only after establishment of the secure channel.

4.3.6. Server A and B:

A. Specification:

These two servers shall host Cloud App in two identical instances. In case one instance fails, the other one shall handle the requests until both instances are able to share the load. One of the servers shall be responsible for outside connections. A limited demo of cloud bursting shall be shown on these two servers as part of the project. When the resources of either server A or B will be exhausted, the other remaining server shall be turned on. However this shall be purposely done, since due to the small size of the cloud actual cloud bursting shall not be prominently visible.

B. Interface:

These two servers shall be connected over NIC cards and shall be connected via RTSP and TCP protocols. To the Android, they shall appear invisible and appear as a collective cloud, since both of them would connect over a unique IP address.

4.3.7. SQL Server Database:

A. Specification:

This shall be a relational database handled using SQL Server 2008 RDBMS. It shall be installed on both the servers A and B. The state of the database shall be maintained across both servers and kept current using database policies. Two databases shall be part of the system, an Accounts database,

and a Map database. The accounts database shall hold information of the users and the administrators. The Map database shall hold text instructions of all the mazes and the path for the robots to download and follow. Both these databases shall be available to both servers and one server's portion of the database shall be accessible to the other server.

B. Interface:

the servers shall connect the databases using secure protocols that require explicit log on for administration purposes and implicit log on data fetching purposes (once during each sessions/service request).

4.3.8. Administration portal:

A. Specification:

The administrator role is assigned the responsibility of maintaining the system. He does it through this portal. In this portal he has the services to monitor the usage of the cloud, check its resource health and also to shut down servers if necessary. He could not administer the Android App from this portal, and in order to do so, he would have to manually go to the Android clients. This portal is meant only for the Cloud App. The Cloud App shall make use of the databases, which the administrator could change from this portal. Lastly, this portal could be used to handle users of the system and to change their rights and privileges.

B. Interface:

A simple GUI shall be provided to the administrator to view and check the health the cloud. In addition to this, a command line interface shall be

available for advanced functions like starting/shut down of the servers. Database administration would be done directly on a tabular interface that shall connect the database to the portal remotely. However, the administrator has to first explicitly login to the portal before making any changes.

4.4. Data Structure Design

The data structure design contains information about the software classes that make up the static portion of the software. There are three packages in total. The Android App (figure 4.3), the Cloud App (figure 4.4), and the user Agent Administrator (figure 4.5). All are described in the sections that follow next.

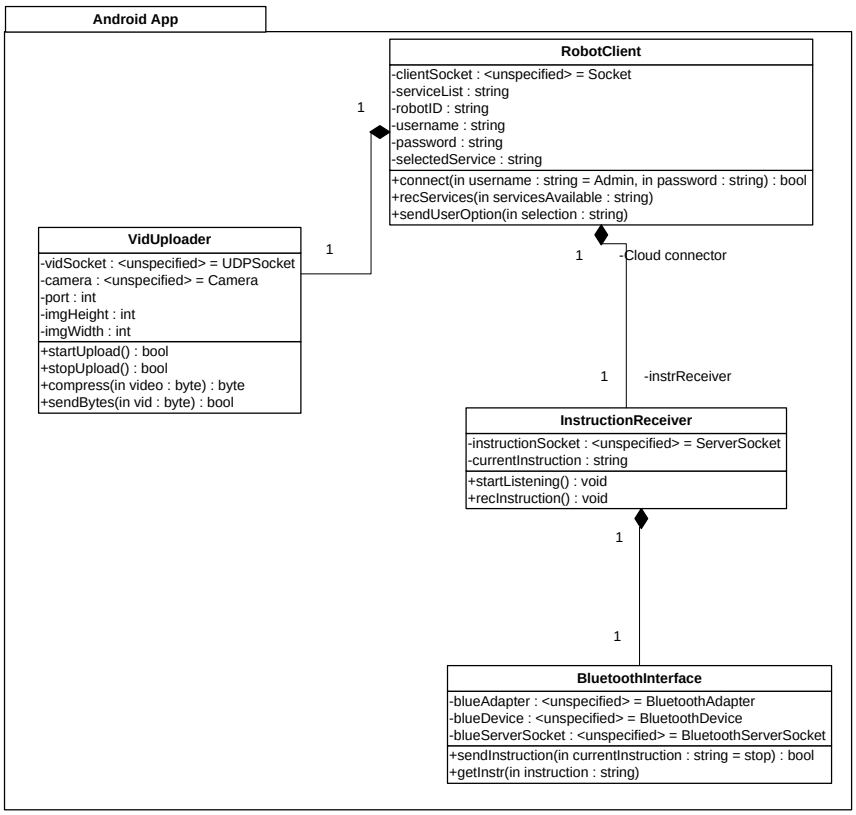

Figure 4.3Android App Package contains classes within classes according to the Android Spec.

4.4.1. Android App Package:

This package includes four classes: RobotClient, VidUploader, InstructionReceiver, and BluetoothInterface. Due to the programming nature and Google API specs, there has to be a main class, followed by smaller classes that are composed inside the main class. Therefore there are not any associations among classes in this package. The breakdown of these four is given in the tables below:

A. RobotClient Detail:

Class Name	RobotClient
Inherited Class	None
Contained Classes	VidUploader, InstructionReceiver
Associated Classes	None
Data Members	clientSocket, serviceList, robotID, username, password, selectedService
Member Functions	Connect(string, string), recServices(string), sendUserOption(string)
Data Structures	Socket, bool, string
Processing	This class is the main class on the Android App that handles connections to the Server and receives instructions from it.

Table 4.1RobotClient class detail

B. VidUploader detail:

Class Name	VidUploader
Inherited Class	None
Contained Classes	None

Associated Classes	RobotClient
Data Members	vidSocket, camera, port, imgheight, imgwidth
Member Functions	startUpload(), stopUpload() ,compress(), sendBytes(byte[])
Data Structures	Int, bool, byte, UDPSocket, Camera
Processing	This class has to manage the video upload stream to the server

Table 4.2 (cont'd) VidUploader class detail

C. InstructionReceiver detail:

Class Name	InstructionReceiver
Inherited Class	None
Contained Classes	BluetoothInterface
Associated Classes	RobotClient
Data Members	instructionSocket, currentInstruction
Member Functions	startListening(), recInstruction()
Data Structures	ServerSocket, string
Processing	This class maintains the instructions that are received from the server and converts them into language understandable by the Create ® robot

Table 4.3InstructionReceiver class detail

D. BluetoothInterface detail:

Class Name	BluetoothInterface
Inherited Class	None
Contained Classes	None

Associated Classes	InstructionReceiver
Data Members	blueAdapter, blueDevice, blueServerSocket
Member Functions	sendInstruction(string), getInstr(string)
Data Structures	BluetoothAdaptor, BluetoothDevice, BluetoothServerSocket, string
Processing	This class has to receive instruction strings, converted into low level robot language and send them via the BluetoothServerSocket connection down to the actual robot with which it maintains a connection. This class uses specific Android 2.3 SDK classes like BluetoothAdaptor, BluetoothDevice, and BluetoothServerSocket

Table 4.4 (cont'd)BluetoothInterface class detail

4.4.2. Cloud App Package:

This package contains a total of seven classes, out of which one is an abstract class, i.e. InstructionGenerator. This class is in turn implemented by two other classes, MazeSolver and PathFollowing. As shown in figure 4.4, this package is made for implementing the Cloud App, the main application of the system, to be deployed on the cloud. It acts as the server for the cellbot.

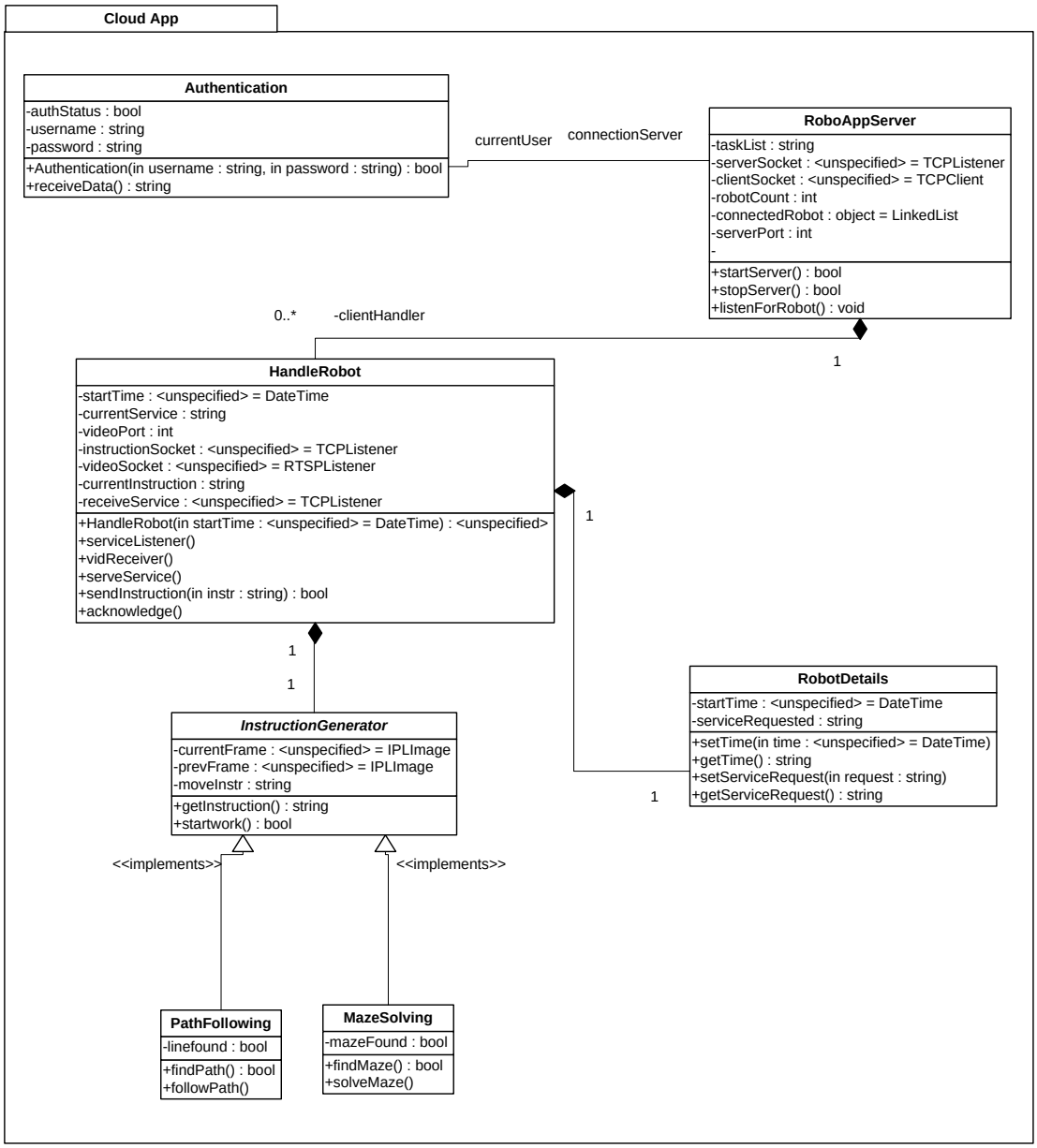

Figure 4.4The Cloud App Package contains classes that make up the business logic of the framework

A. Authentication details:

Class Name	Authentication
Inherited Class	None
Contained Classes	None
Associated	Cloud AppServer

Classes	
Data Members	authStatus, username, password
Member Functions	Authentication (string, string), receiveData()
Data Structures	Bool, string
Processing	This class is responsible for authenticating a robot client by connecting to the Accounts Database and verifying the User before allowing him to receive the services.

Table 4.5 Authentication class detail

B. Cloud AppServer details:

Class Name	Cloud AppServer
Inherited Class	None
Contained Classes	HandleRobot
Associated Classes	Authentication
Data Members	taskList, serverSocket, clientSocket, robot, connectedRobot, serverPort, Linked List
Member Functions	startServer(), stopServer, listenForRobot()
Data Structures	String, TCPListener, int
Processing	This class is the main server which maintains connections with the cellbot, however it does not serve instructions to the cellbot but only maintains session state

Table 4.6Cloud AppServer class detail

C. HandleRobot details:

Class Name	HandleRobot
Inherited Class	None
Contained	InstructionGenerator, RobotDetails

Classes	
Associated Classes	Cloud AppServer
Data Members	startTime, currentService, videoPort, instructionSocket, videoSocket, currentInstruction, receiveService
Member Functions	HandleRobot(DateTime), serviceListener(), vidReceiver(), serveService(), sendInstruction(string), acknowledge()
Data Structures	DateTime, RTSPListener, TCPListener, int, string
Processing	This class coordinates with RobotDetails and InstructionGenerator classes to fetch instructions from the map database and passes them down to the robot

Table 4.7HandleRobot class detail

D. RobotDetails details:

Class Name	RobotDetails
Inherited Class	None
Contained Classes	None
Associated Classes	HandleRobot
Data Members	startTime, serviceRequest
Member Functions	setTime(DateTime), getTime(), setServiceRequest(string), getServiceRequest()
Data Structures	DateTime, string
Processing	This class stores the details of the robot client including the time it connected to the robot and its unique ID.

Table 4.8RobotDetails class detail

E. InstructionGenerator details:

Class Name	InstructionDetails
Inherited Class	None

Contained Classes	None
Associated Classes	HandleRobot
Data Members	currentFrame, prevFrame, moveInstr
Member Functions	getInstruction(string), startwork()
Data Structures	IPLImage, string
Processing	This class is responsible for implementing the image processing capability of the system for either mazes or the path. It uses the IPLimage class of the EmguCV library

Table 4.9InstructionGenerator class detail

F. MazeSolver details:

Class Name	MazeSolver
Inherited Class	InstructionGenerator
Contained Classes	None
Associated Classes	None
Data Members	mazeFound
Member Functions	findMaze(), solveMaze()
Data Structures	Bool
Processing	This class implements image processing capabilities to help the robot find a maze in a laboratory environment

Table 4.10MazeSolver class detail

G. PathFollowing details:

Class Name	PathFollowing
Inherited Class	IntructionGenerator

Contained Classes	None
Associated Classes	None
Data Members	lineFound
Member Functions	findLine(), followLine()
Data Structures	Bool
Processing	This class is responsible for using the image processing capability to find the path in the lab environment and then to help the robot to follow the path till completion.

Table 4.11Pathfollowing class detail

4.4.3. Administration Portal Package:

This package contains three classes: User, Administrator, and Administration. These three classes shall be responsible for maintaining the information of the User and the administrator, plus the administration activities, as shown in figure 4.5 below.

They coordinate with the Accounts database to locate information for the clients as well as the administrators, since there could be more than one administrator and more than one simultaneous client. However, in this project a single client, single administrator system shall be focused on.

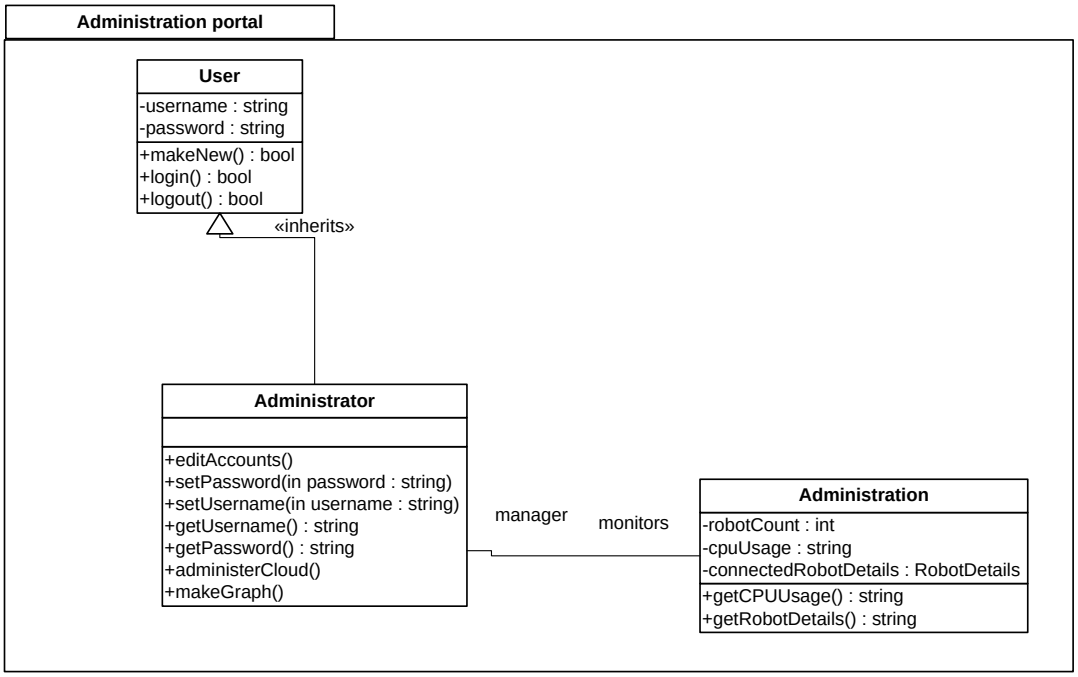

Figure 4.5 User Agent Administrator Package

A. User details:

Class Name	User
Inherited Class	None
Contained Classes	None
Associated Classes	None
Data Members	Username, password
Member Functions	makeNew(), login(), logout()
Data Structures	String
Processing	This class logs on/off the User when he tries to connect to the cloud via his Android App. It also facilitates the creation of a new account through the administration portal.

Table 4.12 User class detail\

B. Administrator details:

Class Name	Administrator
Inherited Class	User
Contained Classes	None
Associated Classes	Administration
Data Members	None
Member Functions	editAccount(), setPassword(string), setUsername(string), getUsername(), getPassword(), administerCloud(), makeGraph()
Data Structures	String
Processing	This class gives the administrator the rights to handle the administration policies for the cloud and to check the resource usage and health of the cloud.

Table 4.13 Administrator class detail

C. Administration details:

Class Name	Administrator
Inherited Class	None
Contained Classes	RobotDetails
Associated Classes	Administrator
Data Members	robotCount, CPUUsage, connectedRobotDetails
Member Functions	getCPUUsage(), getRobotDetails()
Data Structures	Int, string, RobotDetails
Processing	This class has the atomic functions for cloud administration and control and lets the Administrator class work in

	coordination with itself.

Table 4.14 (cont'd) Administration class detail

4.4.4. BL Databases:

The Business Logic Databases are stored in the SQL Server databases on the cloud itself. There are two such databases used in the system.

A. Map Database:

The Map database shall consist of two unrelated tables, titled "Maze" and "Path". They shall both consist of same entities: *Serial no, Version, TimeOfCreation, Algorithm, and Breakpoint.* The breakpoint entity shall be a number that shall describe the last number at which the last version ended and this version started in the algorithm steps.

B. Accounts Database:

This database shall consist of two tables, unrelated to each other, called "Users" and "Administrators". They shall consist of the same entities: *Username, password, DateOfAccountCreation.*

4.5. State chart Diagrams

The state chart diagrams are made for two classes that showed dynamic behavior: HandleRobot of the Cloud App package, and RobotClient of the Android App package. Their figures are included in appendix C and their descriptions are as under.

4.5.1. HandleRobot:

The HandleRobot object exhibits dynamic behavior from the Cloud App package as it is the only class with changing information over a certain period.

Other classes in its package are either single data carriers, whose data does not change or have a single state for the entire duration of a session and service request.

The HandleRobot has a total of eight states:

- **Connected:** the object establishes a connection with the database and the client.
- **Wait for command:**goes into waiting state for the user to select a valid task.
- **Response to client:**tells the client that it shall be processing its request and waits for an ack from the client.
- **Service started:**starts reading the video frame sent from the client and preprocesses it
- **Processing:** applies image processing algorithms on the frame to determine map location and suitable instruction fetching from the map database.
- **Instruction transfer:**It sends the fetched instructions to the client
- **Response completed:**the database sends a special character to indicate that the end of the map has been reached, during this state, it sends the client an acknowledgement and waits for the client to ack back.
- **Destruction:**threads are destroyed, and garbage collection is done.

4.5.2. RobotClient:

The Android App package has a top class that handles all the dynamic activities of the cellbot and all other classes are enclosed inside of it. Therefore the only behavioral object is that of RobotClient, which has nine states:

- **Disconnected:** when the cellbot is turned ON by the user, i.e. the Android App is selected from the cellphone interface, the resources are allocated inside the phone memory and all threads necessary are created.
- **Connected:**the Cloud App server is sent a request for connection and in this state the connection request packet is generated.
- **Waiting:**the server establishes a session, but is yet to send the cellbot the list of available tasks that have algorithms in its library. To the RobotClient object, this library is inaccessible and the server sends down the list of tasks in text version.
- **Service received:**services (tasks) are downloaded to the cellbot
- **Service selected:**the user selects a service from the list and this is sent to the server
- **Waiting for Instruction:**thecellbot waits for navigation instructions to be downloaded to it from the server
- **Service started:**instruction downloading begins to the cellbot
- **Instructions received**:the instructions got one at a time from the server are written to the Bluetooth Interface so that they could be sent to the robot underneath.

- **Service ended**:the special character is downloaded but not sent to the robot. After this, the threads are stopped and garbage collection is done.

4.6. Use Case Realizations

The use cases describe how the user interacts with the system and what pre-, post-conditions exist, as well as what kinds of user interaction sequences may occur. The figures are available in Appendix C.

4.7. Use Case List of Cellbot Android App:

Only the two most important use cases were documented here to avoid discrepancies.

1. Login
2. Select task

4.7.1. Use Case: Login

A. Use Case Requirement

- The App enables the user to login

B. Business Justification

- The app requires only authenticated Users to operate the cellbots.
- The User can be under check and security loopholes are minimized

C. Use Case Paths

- ***1. Normal****:*

  User authenticates himself

- ***2. Exceptional****:*

  User is unable to authenticate

1. Normal Path: User authenticates himself

Externals

- User

Preconditions

- The Android App is up and running, ready to take username and password

Interactions

1. The User inputs username and password in their respective fields and presses enter key, or an appropriate button.
2. The Android App shall authenticate the User after checking the username and password.

Post-conditions

- The Android App is displayed the phone home screen.

Frequency: High (Daily)

Criticality: High

Probability of Defects: Medium

Risk: High

2. Exceptional path: User is unable to authenticate himself

Externals:

- User

Preconditions:

- The Android App is up and running, ready to take username and password

Interactions:

1. The User inputs username and password in their respective fields and presses enter key, or an appropriate button.
2. The Android App shall not authenticate User due to incorrect username and/or password.

Post-conditions:

- The User is displayed the log in screen again with error message “wrong username/password, please enter correct password”

Frequency: High (Daily)

Criticality: High

Probability of Defects: Medium

Risk: High

4.7.2. Use Case: Select Task

A. Use Case Requirement

- The Android App enables User to log in to select a specific tasks from the given list only:

B. Business Justification

- The cloud requires users to operate the client only under its given services and none else.
- The application can be under check and security loopholes are minimized along with errors.

C. Use Case Paths

- ***1. Normal*:**

User selects service

- ***2. Exceptional*:**

  User doesn’t select service

1. Normal Path: User Selects Service

Externals

- User

Preconditions

- The Android is connected to the cloud and service list is displayed

Interactions

1. The User selects a given service by pressing finger on its area/button.
2. Android app starts the algorithm for the service selected after 10 seconds.

Post-conditions

- The cellbot starts to move around in the environment, looking for the service requested

Frequency: High (Daily)

Criticality: High

Probability of Defects: Medium

Risk: High

2. Exceptional path: User doesn’t select a service

Externals:

- User

Preconditions:

- The Android App is connected to the cloud and service list is displayed.

Interactions:

- The user turns off the device.

Post-conditions:

- Blank screen on the cellbot

Frequency: Low (Daily)

Criticality: Low

Probability of Defects: Low

Risk: Low

4.8. Use Case List for Administration Portal

All the most important use cases have been described below and the figure is in appendix C.

1. Change Username and password
2. Add new account
3. Edit account info
4. Administer BL Database

4.8.1. Use Case: Change Username/password

A. Use Case Requirement

- The Administrator should be allowed to change his username and password from time to time

B. Business Justification

- The app requires only authenticated administrators to use it
- The administrator can ensure that no one has stolen his credentials

C. Use Case Paths

- **1. Normal**:

Administrator changes his credentials

- **2. Exceptional**:

Administrator is unable to change his credentials

1. Normal Path: Administrator changes his credentials

Externals:

- Administrator

Preconditions:

- The Administration Portal is up and running, ready to take username and password

Interactions:

1. The Administrator selects an option to change his password and username
2. The Portal shall show forms for entry and keep password characters hidden when typed.
3. The administrator submits the new username and password after confirming his old password.
4. The Portal gives a success message

Post-conditions

- The Android App is displayed the phone home screen.

Frequency: Medium (Monthly)

Criticality: High

Probability of Defects: Low

Risk: High

2. Exceptional path: Administrator is unable to change his credentials

Externals:

- Administrator

Preconditions:

- The Administration Portal is up and running, ready to take username and password

Interactions

1. The administrator selects the option to change his credentials
2. Some forms and fields are given to him and he enters his credentials. The password characters are kept hidden as they are entered.
3. The administrator submits the credentials, however the old password given for confirmation does not match the one in the database.
4. Error message is show saying that password and username were not changed

Post-conditions:

- The User is displayed the log in screen again with error message “wrong username/password, please enter correct password”

Frequency: High (Daily)

Criticality: High

Probability of Defects: Medium

Risk: High

4.8.2. Use Case: Add New Account

A. Use Case Requirement

- The Cloud App and Android App shall have more than one User and Administrator

B. Business Justification

- The app requires only authenticated Users to operate the cellbots and authorized administrators to access the Administration Portal
- The User and Administrator can be under check and security loopholes are minimized

C. Use Case Paths

1. Normal:

Administrator adds a new account successfully

1. Normal Path: Administrator adds a new account successfully

Externals:

- Administrator

Preconditions:

- The Administration Portal is up and running, and the option to add a new user/administrator is available

Interactions:

1. The Administrator selects the button to add a new user/admin.
2. The form is displayed to administrator to enter the credentials of the user.
3. A checkbox is available to make the user an administrator.
4. The Administrator submits the information and the Portal displays the message for successful addition.

Post-conditions:

- The home screen of the Administration Portal is displayed.

Frequency: Low (Monthly)

Criticality: High

Probability of Defects: Low

Risk: Medium

4.8.3. Use Case: Edit Account info

A. Use Case Requirement

- Users could be converted into Administrators and their rights should be changeable.

B. Business Justification

- The app requires only authenticated Users to operate the cellbots and
- The User and Administrator can be under check and security loopholes are minimized
- There could be levels of Administrator, e.g. senior admin, junior admin, and each has their own levels of rights

C. Use Case Paths

1. Normal:

Administrator changes rights of account

1. Normal Path: Administrator changes rights of an account.

Externals:

- Administrator

Preconditions:

- The Administration Portal is up and running and option to edit accounts is available.

Interactions:

1. The Administrator selects the option to edit accounts.
2. He is given a list of available accounts
3. He selects one of the administrator accounts and edits its rights
4. Rights include rights to edit map database, Cloud App, and accounts database.

5. He selects “view only” for all three types for this certain account. Other options are “full access” and “hidden”.
6. He closes this account and opens another administrator account and assigns it the “junior” level from a drop down list. This makes the account only accessible to Cloud App monitoring activities and takes away all database rights.
7. He closes the account settings

Post-conditions

- The Administration Portal Home screen is displayed.

Frequency: High (Daily)

Criticality: High

Probability of Defects: High

Risk: High

4.8.4. Use Case: Administer BL Database

A. Use Case Requirement

- The administrator of the system should be able to directly access the database and be allowed to edit its data including images and text.

B. Business Justification

- The system shall continue learning and addition of more information is a must. Since the robot will be dumb, new information is required by it at all times.
- Sometimes, wrong information has to be corrected in the database
- Information is available only to the administrator for editing purposes, the client cannot change the information in the database in case of problems.

C. Use Case Paths

1. Normal:

Administrator adds path info

2. Normal:

Administrator adds object info

1. Normal Path: Administrator adds path info

Externals:

- Administrator

Preconditions:

- The Administration Portal is up and running and database administration option is available.
- Senior administrator, if available, is logged in

Interactions:

1. The administrator selects the database administration option and selects “edit map”.
2. He is given a list of available maps
3. He opens a map by selecting it, either from Maze table or from Path table.
4. He then selects a version
5. He adds to the text or corrects it.
6. He closes the table and saves the new information through a message that appears on closing the table.

Post-conditions:

- The Map database home screen is displayed.

Frequency: High (Daily)

Criticality: High

Probability of Defects: Medium

Risk: High

2. Normal Path: Administrator adds object info

Externals:

- Administrator

Preconditions:

- The Administration Portal is up and running, and database administration option is available.

Interactions:

1. The Administrator selects the database administration option and selects “edit objects”.
2. He has given a table consists of objects in database.
3. He selects an object by double clicking on it and opening a form
4. The forms has field that describe the dimensions of the object and has its snapshots from various angles, including its 3D graphical representation.
5. The administrator adds a new snapshot by clicking on “upload picture” button.
6. The new picture is displayed.
7. He changes the information about the object by clicking on relevant fields and typing the new information.
8. He closes the object form and is prompted to save the information.

Post-conditions

- The database administration home screen is available

Frequency: High (Daily)

Criticality: Medium

Probability of Defects: Medium

Risk: Low

4.9. Activity Diagrams

The main two activities of the system are selection of a task and its execution (on User’s end) and the system Administration (on Administrator’s end). These two tasks are defined in the activity diagrams. They are modeled in

general self-explanatory UML specification. Their figures are available in Appendix C. They describe how the two classes pass down data from the start to the end.

4.10. Sequence Diagram Design

The diagrams below show the sequence of procedures and functions called to complete three tasks: Administration, path following and Maze solving.

4.10.1. Administration

The administration and administrator classes have dynamic behavior. As shown in figure 4.6 below, the class enables the user to login to the system and then monitor the usage of the cluster, find out which robots are connected to the cloud at the moment and plot graphs for the human administrator's aid in analyses.

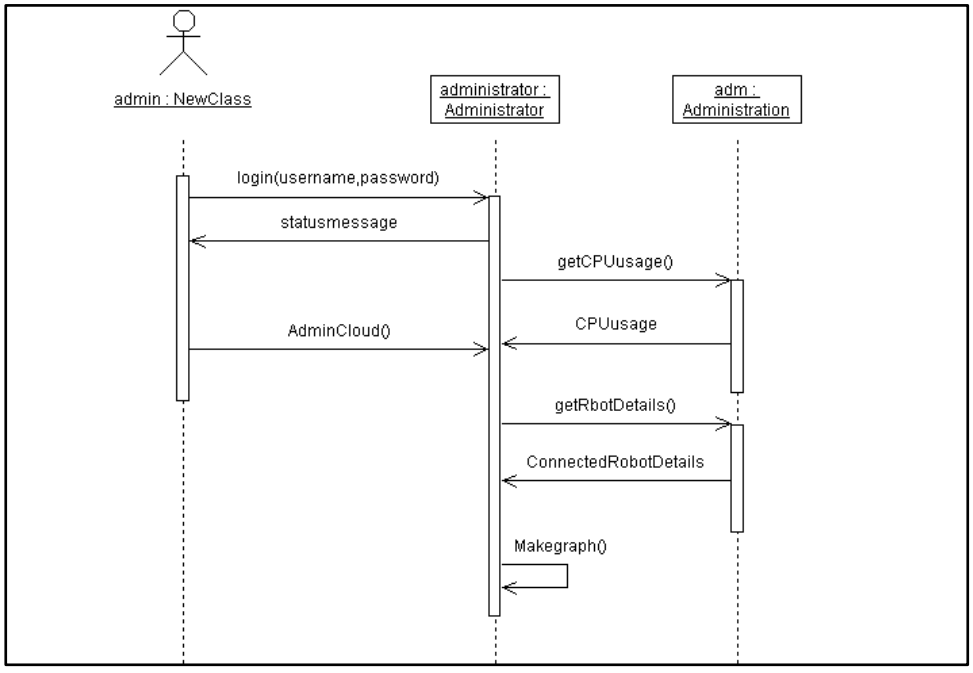

Figure 4.6Administration Portal Sequence Diagram

4.10.2. Path Following:

The path following service requires certain classes to function on its behalf, as show in figure 4.7 below. The classes server, authenticate and HandleRobot allow the system to define if an authenticated cellbot user is requesting services from the cloud. Then the robot details are shared with the cloud and the cloud tells it how to follow the path, through the functions getinstruction(), followpath(), and sendinstruction().

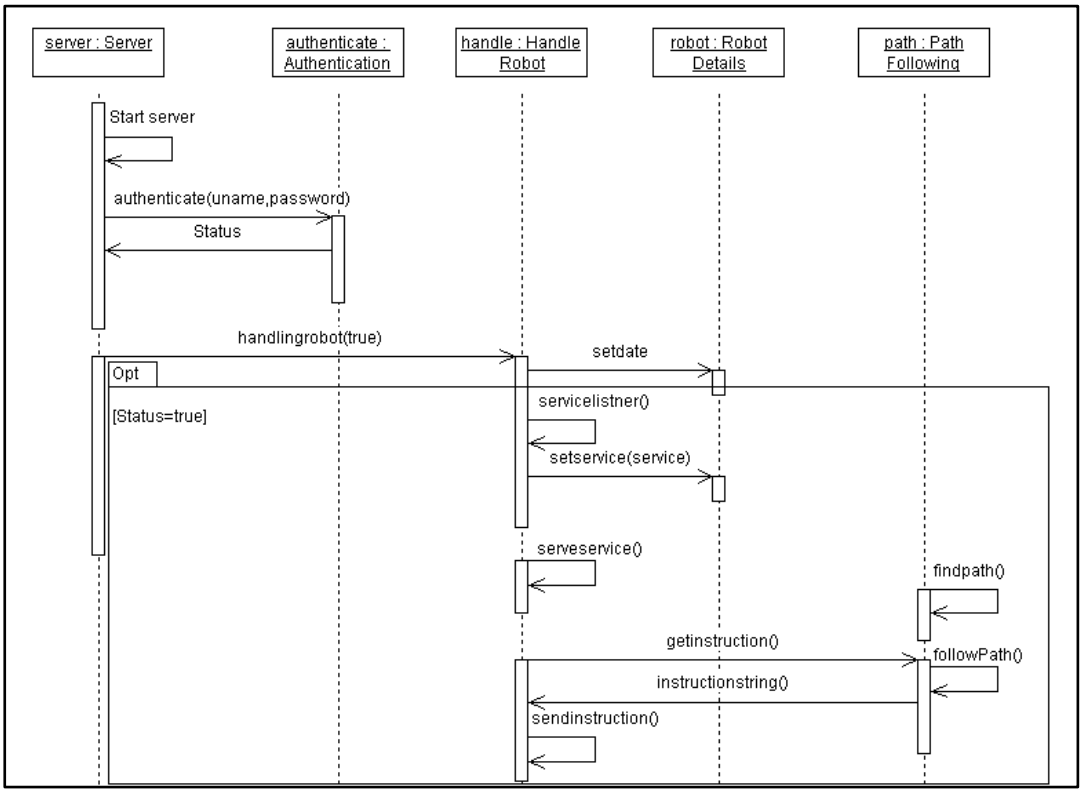

Figure 4.7 Path following service in action on the system

4.10.3. Maze Solving:

Maze Solving service requires the use of a database and subsequent querying by the cloud to the database. These requests are made by findMaze() function, as shown in figure 4.8 below. After the querying is finished, the rest of the activity is carried out by the solveMaze() function.

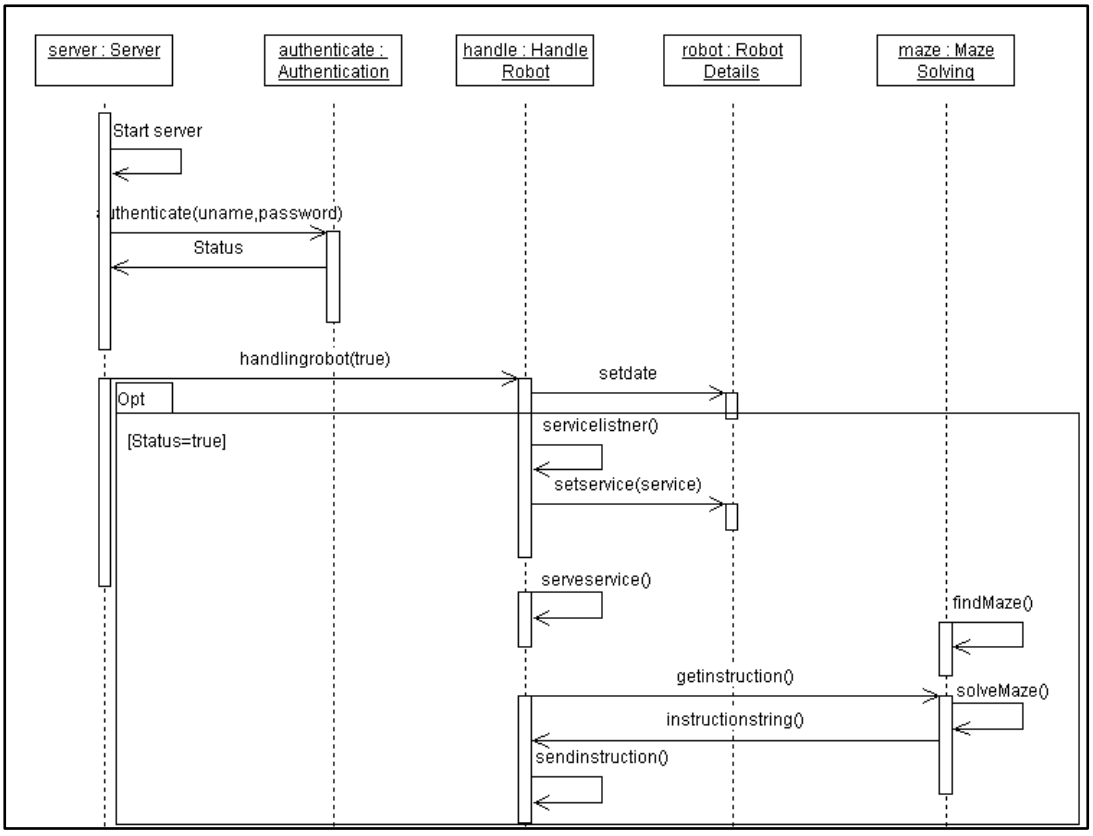

Figure 4.8 Maze solving service in action on the system

4.10.4. User Interface Design

Some of the user interfaces that have been developed so far as part of the prototype are displayed below. The user could login to the system through the welcome screen on the Admin portal as shown in figure 4.9.

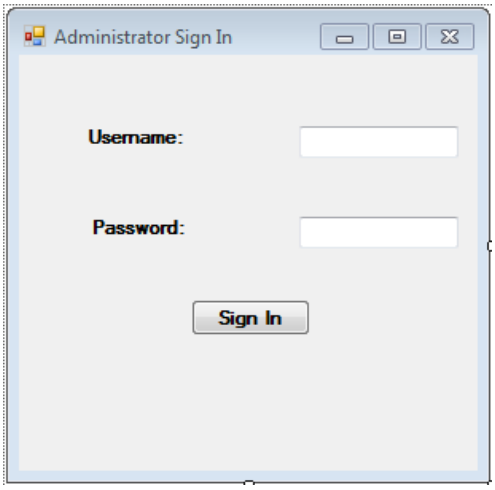

Figure 4.9Signing in to the Administrator Portal via the GUI

The user will next see a list of available management options (figure 4.10).

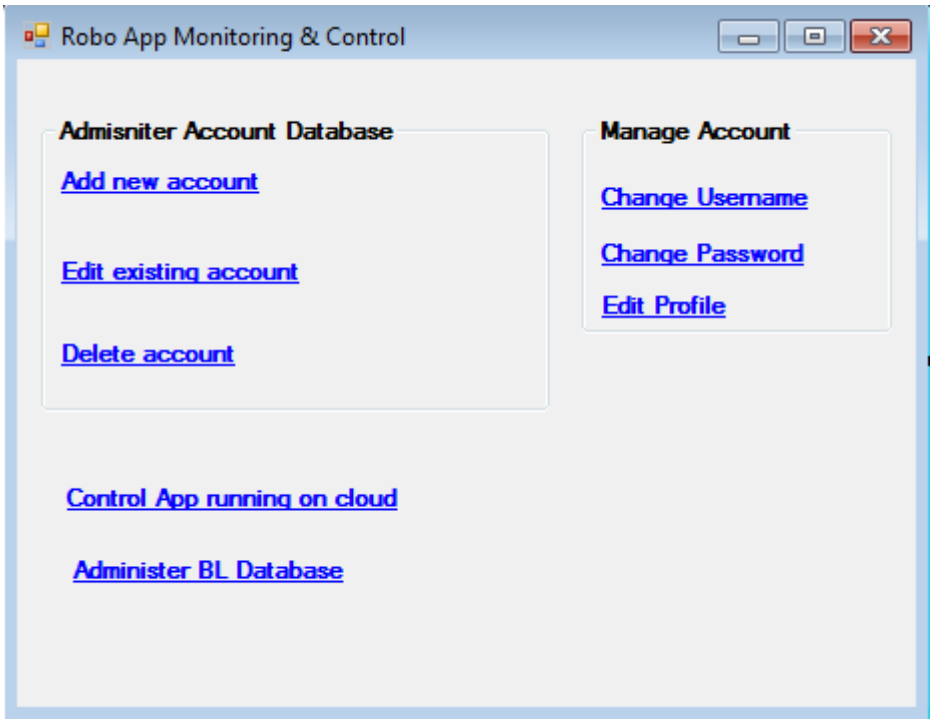

Figure 4.10 Managing the whole framework through the admin portal

The admin portal could also be used to see the statistics for the system, how many robots are using the system and what services are being requested, as shown in figure 4.11 below.

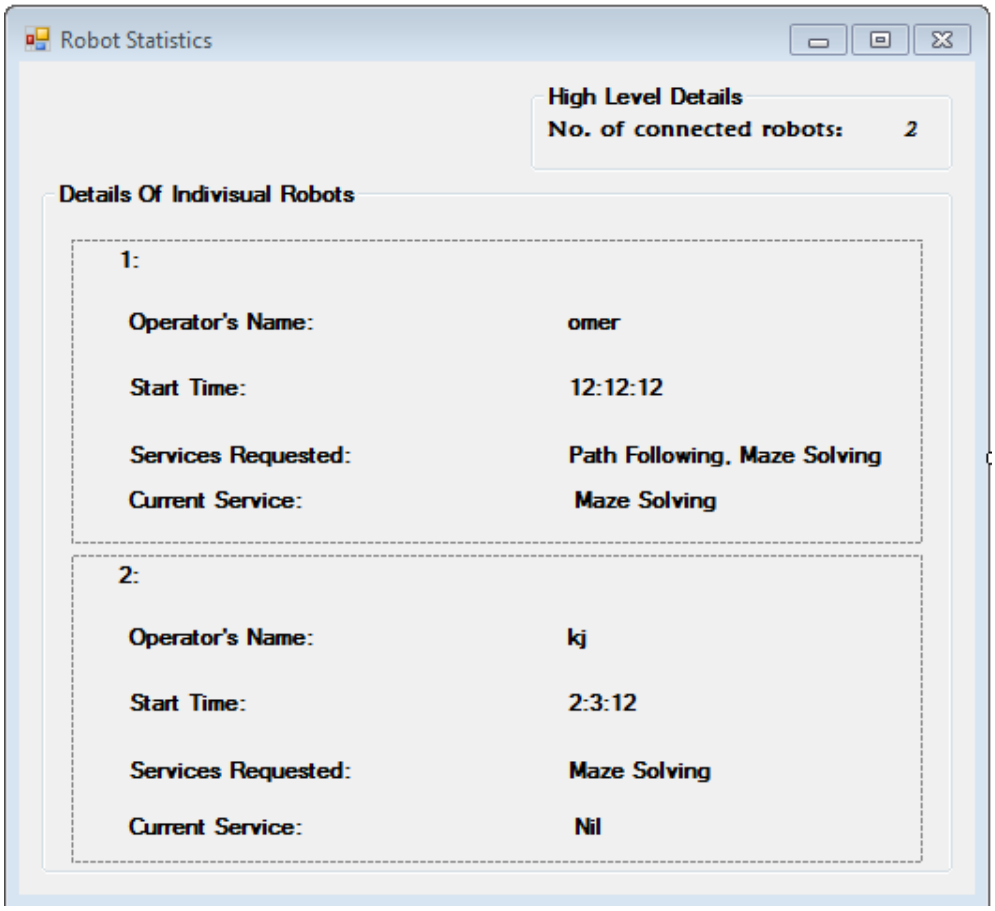

Figure 4.11Results of the client node health on the administration Portal GUI

4.11. Summary:

The chapter gave a detailed design description of the project, outlining the design for the product architecture, how it will be deployed, and its classes. The states those classes would be in are shown using state chart and sequence diagrams. Use case realizations are done to ensure a steady design of the use cases given during the requirements phase. Finally some UI designs were presented to show how the system would interact with the user.

Chapter 5

System Implementation

5. Introduction

This chapter discusses the details of the methods and techniques used to finally implement the system. There are two separate sections for both hardware and software implementation. The hardware section consists primarily of assembly and assembly stages that were undertaken to bring together different parts of the cloud subsystem in contact with the cellbot. The latter part of the hardware implementation talks about how the Bluetooth device was made to interact with the Android application.

The software implementation phase was divided according to topologicalcategories, namely, of software deployed over the cloud and that on the Android device. The sections below describe firstly the software parts that brought together the image processing procedures in tandem with the wireless communication processes to connect to the Android device for real-time streaming, image capture and upload/download routines using simple network streaming protocols.

5.1. Software Implementation

The software was developed separately for the cloud portion and the Android portion. The cloud portion includes a small database and the Cloud App deployed over a Windows Server 2008 R2 failover cluster. The Android portion contains the Android App deployed over the mobile phone device.

5.1.1. The CloudAppbackend

The popularity of image processing is continuously increasing as more and more digital cameras are available to the general public and the

computational power behind cameras is becoming larger. There are several computer vision and digital image processing libraries for lots of modern languages.OpenCV is just one of such libraries written for C by Intel and then supported and rewritten for C++ since version 2 by WillowGarage. EmguCV has been used, which is an open source OpenCV wrapper for C# environments. Its library has more than 500 complex functions including segmentation, tracking, image transformations, feature detection, and machine learning. It is freely available for Android development on Windows.

5.2.1.1 Development environments for developing Cloud App and its database:

This is minimal platform supported by OpenCV Java API. And it is set as default for OpenCV distribution. It is possible to use newer platform with OpenCV package, but it requires editingOpenCV project settings.

- Sun JDK 6
- Android SDK and its components (Android SDK tools revision 12,
- SDK Platform Android 2.2, API 8, rev 2
- Eclipse IDE 3.7 Indigo
- ADT Plugin for Eclipse
- MySQL 7

5.2.1.2 Implementation of Image Processing

The algorithm can be divided into two main steps: *feature selection* and *tracking*, as shown in the figure 5.1 below.

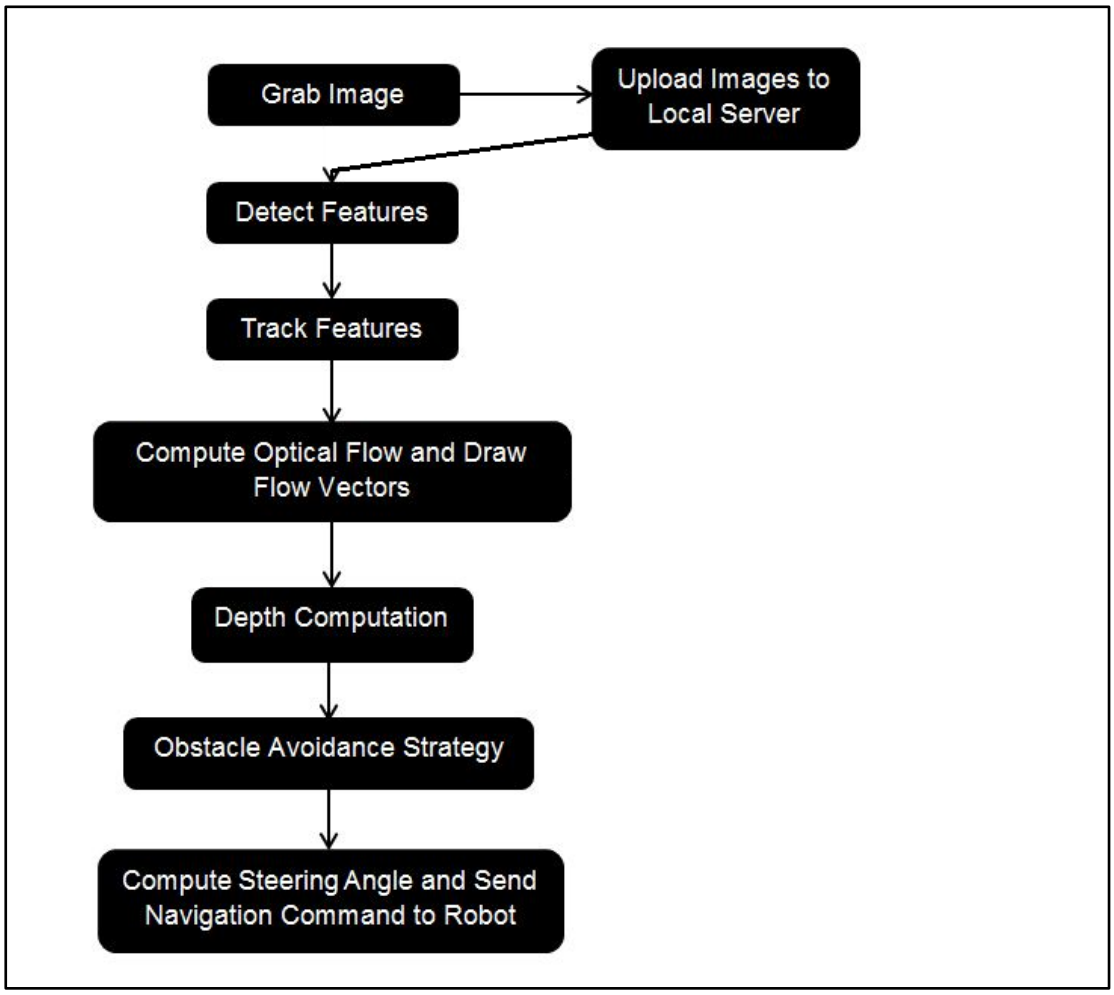

Figure 5.1 Basic Block Diagram of the image processing algorithm on Cloud App

A. Feature Selection:

In this phase of the algorithm, very distinct points called corners are found in a frame. These points have to be distinctive so that they can be found in subsequent frames. One common feature selection algorithm implemented in OpenCV and used in this project is the Shi-Tomasi algorithm[17]. This algorithm is provided in appendix D. The project team implemented this algorithm for feature selection.

B. Tracking:

To compute the optical flow vector between a point in one frame and that same point translated in a subsequent frame, the location of this point must be determined in the sequence of frames. Therefore this point must correspond to the same point in all the frames. This is known as tracking or correspondence. The tracking algorithm implemented and used in OpenCV is the Pyramidal Lucas-Kanade algorithm[18](given in appendix D). In this strategy, the image is decomposed into several cells. For each cell, the median magnitude and direction of the optical flow field are computed. Each cell is then compared to its neighboring cells for evidence of discontinuities. Here the depth of the features in the image sequences is used to identify obstacles. Obstacles in an image will have different depth values. In fact regions with significantly longer optical flow vectors are indicative of the presence of obstacles.

First of all the C# code will convert the YUV image to BGRA and then to HSV because this color space is more practical in case of color detection then other color spaces. Each pixel of the HSV image will be checked if it is inside a certain color range that matches the color of the torchlight. All pixels in the resulting image will be set to 1 if there is a light at that location and to 0 otherwise. The position of the light area inside the image is reported as well.

According to the function call above the C# code receives a byte array that contains the current camera image in YUV420 format, an int array for the resulting image in BGRA format, and finally a double array for light location

values. The first step is to use these parameters from the C++ appropriately. This is done by the following code:

```
jbyte* _yuv = env->GetByteArrayElements(yuv, 0); jint* _bgra =
env->GetIntArrayElements(bgra, 0); jdouble* _array = env-
>GetDoubleArrayElements(array, 0);
```

Algorithm 5.1 using parameters for image conversion

Now the image arrays can be converted to Mat matrices which is the most important data type in EmguCV. For this reason the width and the height parameters determine the dimensions of the images.

```
Mat myuv(height + height/2, width, CV_8UC1, (unsigned
char *)_yuv); Mat mbgra(height, width, CV_8UC4,
(unsigned char *)_bgra);
```

Algorithm 5.2 converting images into matrices

Using height + height/2 for image height in case of YUV is not the right technique, but this is how it has to be done with the YUV420 compression.

The YUV image is converted to HSV color space in two steps using cvtColorOpenCV function and then the mhsv matrix holds the new image.

```
cvtColor(myuv, mbgra, CV_YUV420sp2BGR, 4); Mat mhsv
= Mat(mbgra.rows,mbgra.cols,CV_8UC4);
cvtColor(mbgra, mhsv, CV_BGR2HSV, 4);
```

Algorithm 5.3 YUV to HSV conversion

The inRangeOpenCV function can determine if the pixels of an image are between two scalars. The result is stored in a one-channel matrix with the same size as the input image. The new one-channel matrix (mdetect) is

created to store the 1s and the 0s of torch light locations. lightLower and lightUpper are the limiting 3 dimensional scalars for the range checking. Since these scalars are used on mhsv (a HSV image) the 3 channels are interpreted as hue, saturation, and value. Each channel in mhsvare stored on 1 byte so constant values can be in the 0 and 255 range.

The predefined numbers in the following code mean that hue of the pixel is unimportant as all possible values between 0 and 255 are in range so white, red, and blue bright lights are all acceptable. However small saturation (between 0 and 10) and high value (between 220 and 255) are requested which means that the color intensity of the checked pixel is low while its brightness is high so pale, bright light pixels are searched for. Then inRange uses these scalars to store torchlight pixels in mdetect. Finally this 1-channel image is converted back to 4-channel BGRA image and the result is stored in the mbgra function parameter for further usage on the Java side.

```
vector planes; Mat mdetect =
Mat(mbgra.rows,mbgra.cols,CV_8UC1); Scalar lightLower =
Scalar(0, 0, 220); Scalar lightUpper = Scalar(255, 10, 255);
inRange(mhsv, lightLower, lightUpper, mdetect);
cvtColor(mdetect, mbgra, CV_GRAY2BGRA, 4);
```

Algorithm 5.4 Reconversion to 4-channel BGRA

It is not enough to know that there is a certain light patch on the scene but the patch location related to the robot is also important. This calculation can be done using image moments that is behind the momentsOpenCV function.

After each call rgba stores the calculated light image and first three elements of buffer contain light location information. It is not necessary to show the calculated light image but it is useful to know why the robot moves into a certain direction. So rgba is converted to a Bitmap in SampleView (see below) and then the bitmap is drawn on the canvas of the surfaceholder in the run method of SampleViewBase.

The navigation of the robot is performed in calculateMove of SampleViewBase. If there is not enough light (the 0th buffer value is below 100) then the robot stops. Otherwise the second coordinate of light blob is used to calculate horizontal direction based on the patch distance from the central line what is the current heading of the robot. Then two simple linear equations determine the left and the right motor speeds. Finally updateMotorControl is called with these intensity values.

5.1.2. Implementing the Private Cloud

The CloudAppis installed on top of a private cloud built using two desktop nodes. These two desktop nodes are part of a Failover Cluster – a term used in cloud computing to describe two fully independent and highly available co-working compute nodes connected to a single storage node. In case of failure at one compute node, the load of the application is shifted to the other compute node. In a real world scenario, there are hundreds of thousands of compute nodes connected to a SAN with a private cloud fabric such as Hyper-V inside of Windows Server 2008. The following section discusses how each of these technologies are used together to achieve a similar result on a smaller scale.

A. Configuring the failover cluster

In simple terms, a failover cluster is another word for cloud server architecture. Since cloud computing revolves around the idea of highly available servers, failover clustering brings this concept to reality. A failover cluster is a two word phrase that constitutes are group of computers (or nodes) designated as node A and B. When the cluster A stops functioning, it “fails over” to the failover B. Here, failover is a word for a cluster that is being used as a substitute. In essence, this increases the availability of the applications and services and users see a continuous, highly available cloud service being offered to them[19].

Cluster Shared Volumes (CSV) is a feature that simplifies the configuration and management of Hyper-V virtual machines in failover clusters. With CSV, on a failover cluster that runs Hyper-V, multiple virtual machines can use the same LUN (disk) yet fail over (or move from node to node) independently of one another. CSV provides increased flexibility for volumes in clustered storage—for example, it allows you to keep system files separate from data to optimize disk performance, even if the system files and the data are contained within virtual hard disk (VHD) files. CSV is available in versions of Windows Server® 2008 R2.

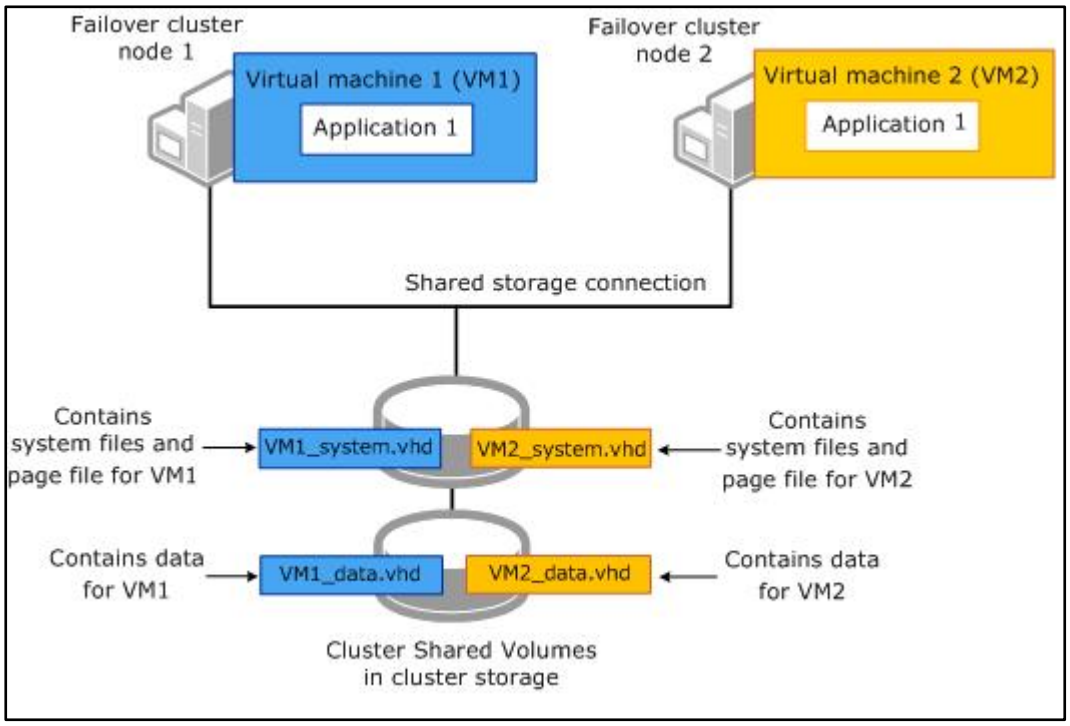

Figure 5.2 Implementing failover clustering with Clustered Shared Volumes in Windows Server 2008 R2

B. Physical node quorum in the failover cluster

Explaining in other words, a quorum for a cluster is actually the amount of nodes that must be working for that cluster to stay online. Resultantly, each node can cast a single “vote” to control whether the cluster stays online and serving. In themselves, the votes are either the nodes or a disk witness (explained later) or file share witness (also explained later). Each voting entity (the file share witness doesn’t count) has a shadow file of the cluster-config, and the Cluster service works to keep all shadow files exact forever. This is done such that the cluster stops running abnormally in case too many failures take place or if a sudden problem occurs with the internode communication channel. In reality, a node has to be able to support the full functionality of the applications that are failed over to it in case of a cluster failover, and so the quorum is not the only deciding factor here. Let’s take an example here, of a

cluster that has five nodes, and whose two nodes fail. Under normal conditions, the remaining three nodes should be failed over to and function normally. But in reality, those nodes would only serve the applications if they have the desired capacity to accommodate a failover.

The biggest problems with network failures are internode communication hassles. Some nodes might talk to each other in a small subnet of clusters but they might not be able to talk to another cluster in a faraway network. This causes the famous “split” problem, where one has to forcibly stop one of the subnets to stop acting as a cluster part. Otherwise major problems could occur.

In order to tackle the split problem, quorum voting comes into play once more. Here, the cluster software has a built in voting mechanism that decides within the sets of cluster if that specific subnet has quorum at that given moment or not. The voting then decides to either shut down the subnet prematurely or not. Since every cluster subnet has a quorum configuration file, it will know what constitutes a majority. In case that subnet has a majority, or if it doesn’t, the cluster will take the appropriate action. That subnet would continue to watch out for new nodes appearing on the subnet and whether they make quorum once again or not, however unless and until the cluster subnet reaches quorum, it will not be a part of the failover cluster.

For example, in a five node cluster that is using a node majority, consider what happens if nodes 1, 2, and 3 can communicate with each other but not with nodes 4 and 5. Nodes 1, 2, and 3 constitute a majority, and they continue running as a cluster. Nodes 4 and 5 are a minority and stop running as a

cluster, which prevents the problems of a “split” situation. If node 3 loses communication with other nodes, all nodes stop running as a cluster. However, all functioning nodes will continue to listen for communication, so that when the network begins working again, the cluster can form and begin to run.

C. Choosing the quorum configuration

There have been significant improvements to the quorum model in Windows Server 2008. In Windows Server 2003, almost all server clusters used a disk in cluster storage (the “quorum resource”) as the quorum. If a node could communicate with the specified disk, the node could function as a part of a cluster, and otherwise it could not. This made the quorum resource a potential single point of failure. In Windows Server 2008, a majority of ‘votes’ is what determines whether a cluster achieves quorum. Nodes can vote, and where appropriate, either a disk in cluster storage (called a “disk witness”) or a file share (called a “file share witness”) can vote. There is also a quorum mode called No Majority: Disk Only which functions like the disk-based quorum in Windows Server 2003. Aside from that mode, there is no single point of failure with the quorum modes, since what matters is the number of votes, not whether a particular element is available to vote.

There are four quorum modes:

1. **Node Majority**: Each node that is available and in communication can vote. The cluster functions only with a majority of the votes, that is, more than half.

2. **Node and Disk Majority**: Each node plus a designated disk in the cluster storage (the “disk witness”) can vote, whenever they are available and in communication. The cluster functions only with a majority of the votes, that is, more than half.
3. **Node and File Share Majority**: Each node plus a designated file share created by the administrator (the “file share witness”) can vote, whenever they are available and in communication. The cluster functions only with a majority of the votes, that is, more than half.
4. **No Majority: Disk Only**: The cluster has quorum if one node is available and in communication with a specific disk in the cluster storage.

Since there was an even number of nodes, but not a multi-tiered cluster in this project, the team chose the option “Node and Disk Majority”. All of these configurations are shown in the figure below:

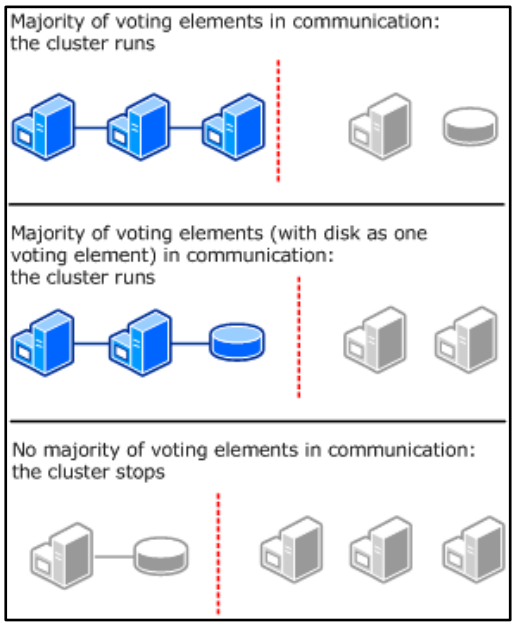

Figure 5.3 Node and Disk Majority quorum configuration

The three main reasons why quorum is important are: to ensure consistency, act as a tie-breaker to avoid partitioning, and to ensure cluster responsiveness.

Because the basic idea of a cluster is multiple physical servers acting as a single logical server, a primary requirement for a cluster is that each of the physical servers always has a view of the cluster that is consistent with the other servers. The cluster hive acts as the definitive repository for all configuration information relating to the cluster. In the event that the cluster hive cannot be loaded locally on a node, the Cluster service does not start, because it is not able to guarantee that the physical server meets the requirement of having a view of the cluster that is consistent with the other servers.

A witness resource is used as the tie-breaker to avoid “split” scenarios and to ensure that one, and only one, collection of the members in a distributed system is considered “official.” A split scenario happens when all of the network communication links between two or more cluster nodes fail. In these cases, the cluster may be split into two or more partitions that cannot communicate with each other. Having only one official membership prevents unsynchronized access to data by other partitions (unsynchronized access can cause data corruption). Likewise, having only one official membership prevents clustered services or applications being brought online by two different nodes: only a node in the collection of nodes that has achieved quorum can bring the clustered service or application online.

To ensure responsiveness, the quorum model ensures that whenever the cluster is running, enough members of the distributed system are operational and communicative, and at least one replica of current state can be guaranteed. This means that no additional time is required to bring members into communication or to determine whether a given replica is guaranteed.

D. Process of achieving quorum

Because a given cluster has a specific set of nodes and a specific quorum configuration, the cluster software on each node stores information about how many "votes" constitutes a quorum for that cluster. If the number drops below the majority, the cluster stops providing services. Nodes will continue listening for incoming connections from other nodes on port 3343, in case they appear again on the network, but the nodes will not begin to function as a cluster until quorum is achieved.

There are several phases a cluster must go through in order to achieve quorum. At a high level, they are:

As a given node comes up, it determines if there are other cluster members that can be communicated with (this process may be in progress on multiple nodes simultaneously).

Once communication is established with other members, the members compare their membership “views” of the cluster until they agree on one view (based on timestamps and other information).

A determination is made as to whether this collection of members “has quorum,” or in other words, has enough members that a “split” scenario

cannot exist. A “split” scenario would mean that another set of nodes that are in this cluster was running on a part of the network not accessible to these nodes.

If there are not enough votes to achieve quorum, then the voters wait for more members to appear. If there are enough votes present, the Cluster service begins to bring cluster resources and applications into service.

With quorum attained, the cluster becomes fully functional.

5.1.3. Administrating the CloudApp

The team has so far used the commonly used methodologies to implement our customized version of a private cloud. Next on, they proceeded to simple administration of the, which includes checking the resources, validating the operating conditions of the application and others. A list of common commands that are run using the Windows Server Powershell is provided below:

1. **Review status of clustered services and applications**: `Get-ClusterGroup`
2. **Review the resources in a cluster application**: `Get-ClusterGroup "Clustered Server 1" | Get-ClusterResource`Where Clustered Server 1 is a clustered server (such as a file server) that contains resources you want to review.
3. **Review detailed settings of resources in a clustered service or application:** `Get-ClusterGroup "Clustered Server 1" | Get-ClusterResource | Get-`

`ClusterParameter`WhereClustered Server 1 is a clustered server (such as a file server) for which you want to review resources and detailed resource settings.

4. **Bring a clustered service or application online:**`Start-ClusterGroup "Clustered Server 1"` Where Clustered Server 1 is a clustered server (such as a file server) that you want to bring online.
5. **Take a clustered service or application offline**: `Stop-ClusterGroup "Clustered Server 1"`
6. **Move a clustered service or application (This also tests failover):** `Move-ClusterGroup "Clustered Server 1"` Where Clustered Server 1 is a clustered server (such as a file server) that you want to test or move.
7. **Review the status of Cluster Shared Volumes**: `Get-ClusterSharedVolume`
8. **Move a Cluster Shared Volume to a different node:** `Move-ClusterSharedVolume "Cluster Disk 3"` Where Cluster Disk 3 is the Cluster Shared Volume you are moving to ownership by a different node.

5.1.4. The Android App Backend

Android applications vary in complexity from application to application, just like their Windows counterparts[20]. In our case, the application to be built were a simple 2D video capture and live streaming application on one hand, while a simple message receiver for the cellbot on the other. It does not have

any image processing function except for transmission of the captured live video to, which acts as its server. Once the server has processed the image and returned a set of text commands, it converts them into scripts for the iRobot Create cellbot and sends them over to it using its Bluetooth interface.

A. The UI thread

In the Android OS, upon application launch, the system creates a main thread, coincidentally called the ‘main’ thread. Later on it is also known as the UI thread, and is a very important part of the whole system because it has thedutyof dispatching the events to the appropriate components, including drawing and callout events. It is also the thread where your application interacts with running components of the Android UI toolkit.

For example, upon touching a button on the phone, the main thread messages the touch event to the component, which in turn sets its state to “pressed” and sends out an invalidate request to the event queue. The main thread dequeues the request and tells the component to redraw itself.

If you do not implement the algorithm properly, the uni-thread concept could crash the application due to bad performance. For example if things like database access and network queries take place on a single thread and other long operations as well, the main thread will block the user interface completely. Until these large operations complete, no other event would be undertaken. Hence, long screen block outs would be experienced. The ANR, or Application Not Responding error comes if the user interface remains hung due to the UI thread being blocked for more than 5 seconds.

Android 1.5 and later platforms offer a utility class called AsyncTask that simplifies the creation of long-running tasks that need to communicate with the user interface. The goal of AsyncTask is to take care of thread management. The following code snippet is an example of an image being uploaded onto the server from the live video stream:

```
private class DownloadImageTask extends AsyncTask<String, Void,
Bitmap>                                                          {
        protected   Bitmap   doInBackground(String...   urls)   {
                        return     loadImageFromNetwork(urls[0]);
                                                                 }

          protected    void    onPostExecute(Bitmap    result)    {
                                mImageView.setImageBitmap(result);
                                                                 }
 }
```

Algorithm 5.5 Image frame upload to server

As one can see, AsyncTaskmust be used by sub classing it. It is also very important to remember that an AsyncTask instance has to be created on the UI thread and can be executed only once, a quick overview of its working follows:

- The parameter types could be specified using generics as well as the final and progress values of the task
- doInBackground() function runs by itself on a worker thread
- UI thread invokes the onPostExecute(), onProgressUpdate(), and onPreExecute() functions on itself
- onPostExecute() gets the value given by doInBackground()
- One good thing is that the function publishProgress() could be called always in doInBackground() to run onProgressUpdate()

B. UDP datagram implementation for video streaming:

UDP is one of the main parts of the Internet Protocol Suite, which are network protocols to run the internet. Using UDP, applications can send/receive commands, also in UDP's case known as datagrams, to other applications on a network that understands IP without the need to set up special transmission channels or data paths.

At the simplest level, our Android App gets a frame(byte[]) array from the device camera and sends it to the server () using UDP. At the server side, first a `server_port` is opened on `port 12345` and a `DatagramSocket()` is opened on it with the local address of the server. Then a `byte[]` message is recorded with `message.getBytes()`. This message is sent in the new `DatagramPacket` to the server.

At the server side, the video is received in `DatagramPacket(message,message.length)` while it listens on a server port with `DatagramSocket`. The incoming messages are set to the maximum size using `byte[] message = new byte[1500];`

C. TCP packets for command transmission implementation

TCP is probably the most commonly used protocol, simply because it is used for so many applications such as HTTP, POP, SMTP, etc. TCP is a protocol which guarantees that the receiver will receive exactly what the sender sent - there will be no errors, it will be in the correct order, and everything will work just fine. That is why TCP was chosen for delivery of commands to the cellbot

via Android App working as the Cloud App's client. Figure 5.4 below shows how the server contacts the client in a sequence of steps.

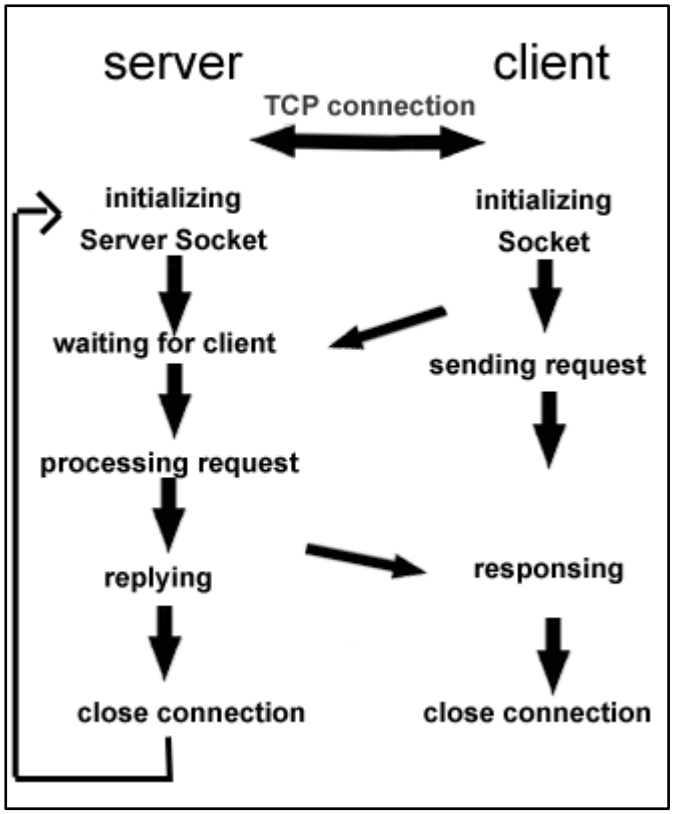

Figure 5.4 TCP sequence diagram

1. At the, a simple ServerSocket is established on port 12345 to listen to client connection requests. A new buffer is made to receive commands from the to be sent using `BufferedReader input =newBufferedReader(newInputStreamReader(s.getInputStream()));`
2. A new printer is made using
   a. `PrintWriter output = newPrintWriter(s.getOutputStream(),true);`
3. The commands are sent using `output.print()`function to the client.

4. A buffer on the Android App is created and it uses an `InputStreamReader` to send the incoming textual commands via the `BufferedReader` to the Bluetooth interface of the cellbot.

5.2. Hardware Implementation

The hardware portion of the system consists of an iRobot Create cellbot that uses a separate Bluetooth interface to communicate with the Android App. The Bluetooth Interface is called the BAM Wireless Accessory and comes with the iRobot Create for use on its serial port as an alternative to PC attached serial cables. The team selected the iRobot Create because of its ease of use for navigational demo purposes. Our aim was to use a simple wheel assembly robot that could execute basic movement commands via an Android phone placed on it. The iRobot Create fulfilled all the following criteria for robot selection:

- Twin wheel assembly for basic movement (up, down, left, right, stop)
- Lightweight and light hardware with customizable options
- Cargo bay
- Programmable interfaces
- Ease of use and little hardware development
- The hardware implementation details are given in the sections that follow:

5.2.1. The iRobot Create

iRobot® Create is a moving robo-development kit that allows users to flexibly program movement behaviors without the hassles of mechanical assembly

and low-level coding. Create's Open Interface (OI) gives a set of commands, such as drive and sensor commands, out which the drive commands were only used. With this development robot, one could easily develop new movements as well as add 3rd party components, and not care about integration issues at a mechanical level or low-level code. Further electronics could be added on top of the Create cargo bay, such as Android smartphones, robotics arms, and sensors. A top and bottom view of the robot is given in figures 5.5 and 5.6 below.

Create is used in the system as a test/demo node as a dumb terminal client that uses the capabilities of the cloud framework. In a real world scenario, this robot could be part of a larger army of iRobot Creates that all use the same cloud interface to download similar commands and execute them in the exact same way collectively. In our system, this single robot node acts a test bed for checking navigation applications in a simple and uncomplicated manner.

The robot has many sensors, few of which were of use to us, since the project was using an Android device's camera for our video streaming purposes. The only part of interest to us was the Cargo Bay Connector. It is a DB-25 connector used for programming. The software interface lets you manipulateCreate's behavior and read its sensors through a series ofcommands including mode commands, actuator commands,song commands, demo commands, and sensor commandsthat you send to Create's serial port by way of a PC ormicrocontroller that is connected to the Cargo Bay Connector.The team used this connector with the BAM module, described in a later section.

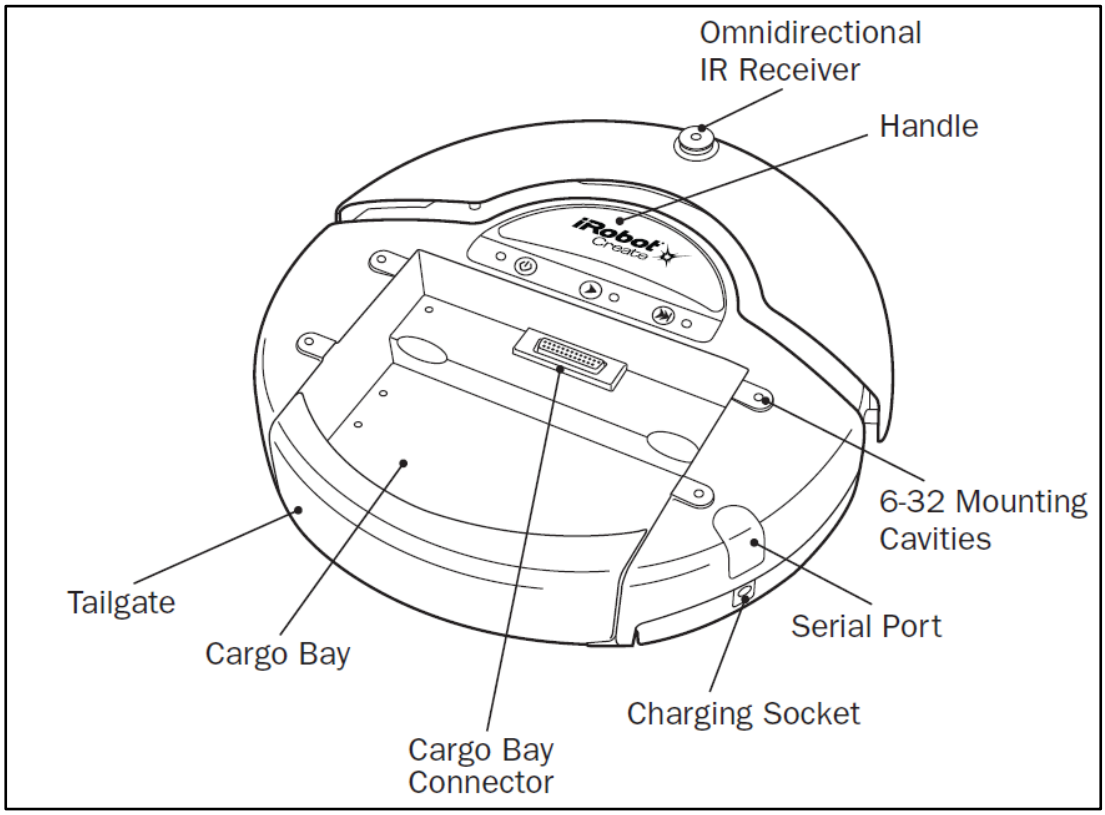

Figure 5.5 iRobot Create ® top view

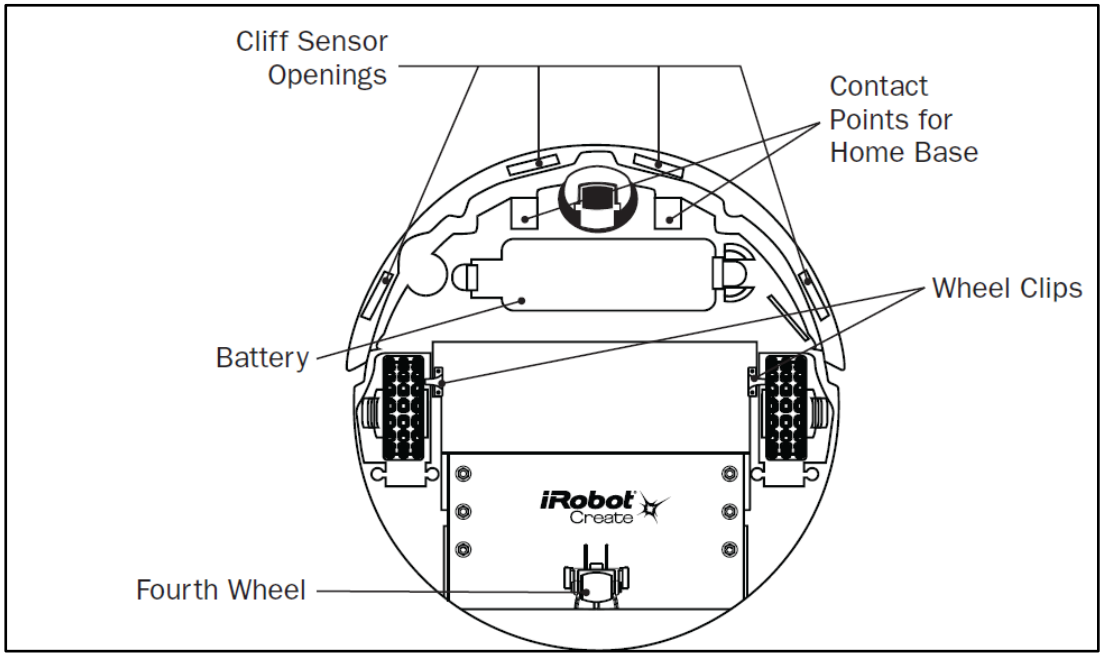

Figure 5.6 iRobot Create ® bottom view

A. The DB-25 Cargo Bay Connector

To use the OI, a processor capable of generating serial commands such as a PC or a microcontroller must be connected to the external Mini-DIN connector or the Cargo Bay Connector on Create. These connectors provide two-way, serial communication at TTL (0 – 5V) levels. The connectors also provide an unregulated direct connection to iRobot Create's battery, which you can use to power the OI applications. The Cargo Bay Connector also provides a regulated 5V power supply and several input and output pins.

The Cargo Bay Connector, located in the front middle of the cargo bay, contains 25 pins that you can use to attach electronics for peripheral devices such as additional sensors. The Cargo Bay Connector provides four digital inputs, an analog input, three digital outputs, three high-current low side driver outputs (useful for driving motors), a charging indicator, a power toggle input, 0-5V serial input and output, a 5V reference, battery ground and battery voltage.

5.2.2. iRobot Create Open Interface Commands

The Create comes with a set of programmable interfaces that allow developers to use it according to their own purposes. Create has an OI (Open Interface) with certain commands that follow the given pattern: Each command starts with a 1-byte opcode with optional additional data bytes. The most used commands are given below. Their opcodes are fed into the Android App module that connects with iRobot Create using the BAM interface.

A. Start OI command

Opcode:128

Data Bytes: 0

This command starts the OI. You must always send the Start command before sending any other commands to the OI.

B. Baud command

Opcode: 129

Data Bytes: 1

This command sets the baud rate in bits per second (bps)at which OI commands and data are sent according to the baud code sent in the data byte. The default baud rate at power up is 57600 bps, but the starting baud rate can be changed to 19200

C. Drive Command

Opcode: 137

Data Bytes: 4

This command controls Create's drive wheels. It takes four data bytes, interpreted as two 16-bit signed values using two's complement. The first two bytes specify the average velocity of the drive wheels in millimeters per second (mm/s), with the high byte being sent first. The next two bytes specify the radius in millimeters at which Create will turn. The longer radii make Create drive straighter, while the shorter radii make Create turn more. The

radius is measured from the center of the turning circle to the center of Create. A Drive command with a positive velocity and a positive radius makes Create drive forward while turning toward the left. A negative radius makes Create turn toward the right. Special cases for the radius make Create turn in place or drive straight, as specified below. A negative velocity makes Create drive backward.

- **Serial sequence**: [137] [Velocity high byte] [Velocity low byte][Radius high byte] [Radius low byte]
- **Drive data byte 1**: Velocity (-500 – 500 mm/s)
- **Drive data byte 2**: Radius (-2000 – 2000 mm)

Example:

To drive in reverse at a velocity of -200 mm/s while turning at a radius of 500mm, send the following serial byte sequence:

[137] [255] [56] [1] [244]

Velocity = -200 = hex FF38 = [hex FF] [hex 38] = [255] [56]

Radius = 500 = hex 01F4 = [hex 01] [hex F4] = [1] [244]

5.3. Summary:

The chapter described how the designs made during the system design phase were realized and implemented using techniques developed by the syndicate members. It shows how the two fold domain of cloud computing and robot path navigation was combined to achieve the objectives for cloud

robotics simulation over a Wi-Fi terminal. It details the techniques that were undertaken to deploy a failover cluster using Microsoft Hyper-V and how the robot application was split into partially the and the Android App. Lastly; it described the various commands that were constructed to move the cellbot correctly.

Chapter 6

Testing and Results Analysis

6. Introduction

Testing has been done at different level in this project to make sure the high quality of the end product. Different testing techniques have been adopted for removing the errors from the system. Different level of testing including unit testing, integration testing and system testing has been done so that the system could be checked in detail and the unwanted results could be removed from the end product. Different types of test cases have been made in every level of testing to make sure that the system provides its required result.

6.1. Unit testing

In unit testing, all the modules have been tested to make sure that they are all working efficiently without producing any kind of error. All the functionality of the software has been tested in this level of testing. Live video streaming module and the Cellbot's movement control module of the system has been tested in detail in at this level.

For the testing of live video streaming module of the android application, the connection between the user interface and the android application has been tested that they are working efficiently. Live video streaming module is also tested for the time delay issue to make sure that there will not be large time delay in the live video stream.

For testing the robot's movement control module of the android application, initially some dummy inputs were sent to the android application from the user end interface which sends the inputs to the BAM and Create to check the

input values and use its own program to run the specific movement command to verify that the control signals are successfully working.

Bluetooth connection to BAM is also tested in this level of testing to make sure that it will send the correct command to the mechanical part of the cellbot.

6.2. Integration testing

In integration testing, the testing during the integration of the different software modules and the testing during the integration of software and the hardware part has been done.

In first step of integration testing, the live video streaming module and the robot's movement control module are integrated and tested by running them in parallel. The quality of the live video stream has been tested that there must be no long time delay issue after integration of both modules. The robot's movement control module's working is also checked that it's working efficiently as it was before the integration of the both modules.

The testing during the integration of the software and the hardware is also done to make sure that there will be no problems after integrating the software (Android App) part with the hardware (BAM and Create both). Testing has been done after integrating the Android App (running on the phone) with the BAM and Create. During integration, connection between the Android App and the BAM has been tested, the connection between BAM and Create's DB-25 has been tested and the connection between the DB-25 and the mechanical parts of the robot has been tested.

6.3. System testing

System testing has been done after integrating all the software and the hardware parts. In this testing level, the system as a whole is tested to make sure that it's giving then required outputs without generating any kind of errors.

For system testing the inputs from the user have been given to the interface and then check that the connection has been made without generating any error and also test the live video streaming to make sure that there is not enough delay in the live video streaming.

Create's movement is also tested by giving some move command to the interface of the Android App so that it could be checked that the cellbot is moving in the desired direction without generating any kind of unexpected results.

6.4. Software Results

For checking the software results, the output of live video streaming and the command download from the cloud has been kept in mind. For analyzing the hardware results, the forward, left, right, backward, stop movementsof Create has been checked and it is working as expected.

6.4.1. Live video streaming module's results

The software part is divided into two major modules, one is the live video streaming module and the other one is the command downloading. For checking the results of the live video streaming module of the Android App, the live video stream has been analyzed at the user end interface and the

Cloud App. At the start of the project, it was expected that the live video streaming should have minimum time delay in streaming (less than 1 sec) and when the output of the live video streaming module has been analyzed the live video streaming module was giving accurate results with time delay of less than 1 sec in streaming the live stream from the Android App to the Cloud App end.

6.4.2. Command Downloading results

The commands were downloaded to the Android phone over the wireless network in less than 1 sec/command. The team divided the command download tests into categories of empty commands, light commands, and heavy commands. Heavy and light commands consisted of large and small amounts of test text commands, respectively. All three categories were able to download themselves to the Android App in less than 1 sec/command.

6.5. Hardware results

In the hardware testing part the Cellbot's movement and control responses were checked. This module of the Android App has been implemented to control the movement of the cellbot. For analyzing the movement control module's results, different movement inputs are given at the interface at the Cloud App end. Tester gave all the inputs including move up, move down, move right, move left, move forward and move backward. All the movements' commands have been tested and the cellbot was moving according to the given movement commands.

Hardware of the system includes all the structure of the iRobot Create, BAM, and the Android phone mounted on it. For analyzing the hardware results, the movement of the cellbot has been checked that it moves properly without facing jerking issues. Different movement commands were been given on the Cloud App interface for checking the results of the cellbot that it moves to the right direction as the input is given to it.

6.6. Analysis

The experiments were performed expansively so that large data could be sought on how the system performed for the Core i3 processors that were in use in our framework, as well as how much power they consumed at low and high clock rates. In this section, a sample of the data collected was given, as well interpreted. The setting for the entire experiments was as follows: 3 Intel Core i3 processors with a certain group of parallel threads executing on each.

For all three core i3 processors, these loads were too low: around 0.5-2.1% of the processor usage for less than 5 threads and 100% usage for threads reaching above 200. For higher load measuring scenarios, the team coded simulated threads in conjunction with special sleep breaks to model service idle modes. In the experiment setting,selected sleep times were 100ms, 500ms, and 1000ms, which represent the arrival rates of 4/second, 1/second, and 0.5/second, respectively. Unfortunately, the team was at no position to verify the real world delays in service, and had to rely on the assumption that wireless service is marginally fair. Video and image frame processing for object recognition and their related delays were not included in this part of the simulation as they were negligible due to their minute compute requirements

on the cloud. The advantage in this whole test bed was that thousands of simulation threads could be coded for stress testing on the cloud hardware.

For our experiments, fifty percent of the threads were made to run in a continuous loop and were used as clients that could give requests to the cloud server. Rest fifty percent of threads ran in a continuous loop and was used as application services to serve resources to clients. Then, processor usage data from Windows Task Manager was recorded for two minutes each, with an average calculation at the end for result. Repeated experiments were done for various types of sleep times per iteration and different clock rates for Intel Core i3. The data collected are shown in Figures below.

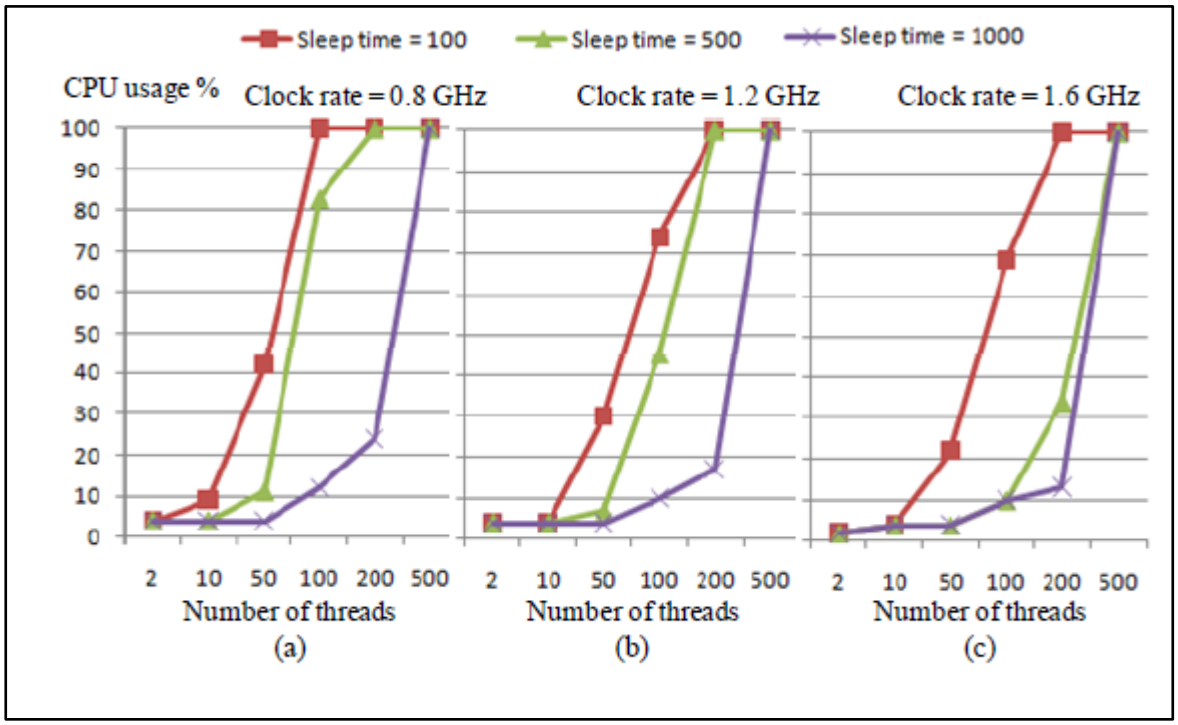

Figure 6.1 Testing the output of the system at very low clock rates for the Intel Core i3 processor

The usage data found from our experiments is a visible proof of the vast amount of the services a cloud server could host and provide. The threads in

the figure (X Axis) are used to show the services, while the different sleep modes are shown by the colored lines to model requests.

As is shown in the figure, these processors were ready to take a very large number of service requests, even at low clock rates. For purposes of our demo environment, these service requests might be low and light weight, however, given that a linear increase in the request/service provision ratio is seen, one could predict that real world applications demanding millions of services would run very smoothly on the same hardware. The good thing about this setting is that by using a hypervisor such as Hyper-V, a cloud server could scale down to low clock rates in a low demand scenario, and scale up in a high demand scenario without greatly affecting its performance. By scaling down, it could reduce power consumption and processor heating, as they both rise exponentially to clock rate doubling.

6.7. Summary:

This chapter focused on the testing of the product to check for verification and validity of the initial objectives set down in the software requirements gathering phase. All relevant parts of a software testing were carried out including unit, integration and system testing to ensure that the project met its objectives correctly. The results obtained from tests were then analyzed and conclusions were made about how the product was finally performing and whether any improvements need to be made.

Chapter 7

Conclusion and Future Work

7. Introduction

With this chapter, the documentation of the project comes to a formal closure and ways are discussed for further extension as far as the system functionality enhancement is concerned. To make this framework worthy of use for robot training, testing and demo, as well as large scale deployment purposes, more functionality needs to be added.

7.1. Future Work

To integrate Language and Action in the field of robotics, some challenges were given toward this end[21]. These have been vastly described in the literature for Cloud Robotics since its inception by major software firms like Google Inc. These challenges were filed under the heading of developmental robotics for the next ten years. They impose certain requirements on the underlying machines that make the entire Cloud Robotics architecture very computationally and storage demanding. Therefore the work to be seen in the near future will mainly focus on the integration of the human language and computer action domains for social and military robotics with further aiding in social/behavior analysis.

The concepts given by us so far and those to be given during the coming years will primarily drive their energies from socialization techniques of advanced mobile robots and their architectural capabilities. An artificial system would be made able to understand some sort of developmental learning technique that allows it to grow due to inspirational needs. Such an implemented system, either on mobile platforms, or on Cloud Computing

platforms, have rich application domains in fields of military, rehabilitation and surgery, space and volcanic research, deep sea travel, and general security.

7.2. Conclusion

The project showed a small test framework that used a distributed failover cluster for server and mobile phone robot, or cellbot, for client. The framework employed the failover cluster as a Cloud server using Hyper-V role of Windows Server 2008 ®. In order to provide a proof-of-concept for the comparatively new cloud robotics field, this framework shows how exactly, using minimal hardware and software, Cloud Robotics could be implemented on much larger scales. Since this framework was developed for a test and demo environment, only a minor domain of Robotics, i.e. autonomous navigation was used to demonstrate the combined power of the cloud computing and robotics. Our robot, under the guidance of a cloud server, was able to navigate a maze and traverse a line without any onboard storage or computational facilities. All the storage/compute nodes were in the cloud, whereas the client cellbot was only supposed to interpret the commands sent to it for movement. Thus the cellbot acted as a dummy client, and ran on very inexpensive hardware, yet it performed functions similar to those of commercial and expensive robots with onboard computers. Keeping in view, these low hardware implementation costs, the cloud framework can now be used for more complex, and human collaborative/adaptable tasks.

Architecturally, the framework justified the need for cloud computing as the server, instead of a simple web server. The software requirements and objectives to be met were shown. Next the implementation and design

backgrounds were explained in detail. At the end, testing was done on the finished product and analysis with results was elaborated for the reader to decide if the framework is feasible for their uses or not.

Our approach was aimed at making a very simple framework that demonstrated how computers could utilize cloud robotics to gain a physical advantage over their predecessor systems that could only compute and show results, yet not do any physical work. Even though physical work has been carried out in factories and assembly lines, however programming those modules is extremely difficult and inflexible. Our framework shows how easy it is to use robotics as the augmented limbs of a computing system, which extends its reach into the physical, non-binary domain. By distributing the workload among several nodes, the teamwas able to achieve efficiency, speed and timeliness. Finally the future work possible for the project was presented.

7.3. Summary:

The chapter gives a brief conclusion of the entire project, how it began and what phases it went through to reach completion. It describes any new techniques that could be added in the future to aid the development of the product expansion. It tells how the product could be given advanced supplications in order to adapt it to real world environments. Lastly it gives an overall conclusion about the project and tells whether the expectations of the syndicate members were met or not with regards to the research.

BIBLIOGRAPHY

[1] S. Kumar, "Binocular Stereo Vision Based Obstacle Avoidance Algorithm for Autonomous Mobile Robots," in *Advance Computing Conference, 2009. IACC 2009. IEEE International*, Patiala, 2009.

[2] K. Nishiwaki, J. Kuffner, S. Kagami, M. Inaba and H. Inoue, "The experimental humanoid robot H7: a research platform for autonomous behaviour," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences,* vol. 365, no. 1850, pp. 79-107, 2007.

[3] R. Arumugam, "DAvinCi: A cloud computing framework for service robots," in *2010 IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, 2010.

[4] B. K. Kim, "Web Services Based Robot Control Platform for Ubiquitous Functions," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005*, Barcelona, 2005.

[5] L. Vasiliu, B. Sakpota and H.-G. Kim, "A Semantic Web Services Driven Application on Humanoid Robots," in *Proceedings of the The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06)*, Gyeongju, 2006.

[6] G. Metta, G. Sagerer, S. Nolfi, C. Nehaniv, K. Fischer, J. Tani, T. Belpaeme, G. Sandini, F. Nori, L. Fadiga, B. Wrede, K. Rohlfing, E. Tuci, K. Dautenhahn, J. Saunders and A. Zeschel, "Integration of Action and Language Knowledge: A Roadmap for Developmental Robotics," *IEEE Transactions on Autonomous Mental Development,* vol. 2, no. 3, pp. 167-195, 2010.

[7] R. Pereira, "An Architecture for Distributed High Performance Video Processing in the Cloud," in *IEEE 3rd International Conference on Cloud Computing (CLOUD)*, Miami, FL, 2010.

[8] R. Buyya, "Market-Oriented Cloud Computing: Vision, Hype, and Reality of Delivering Computing as the 5th Utility," in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09)*, Shanghai, 2009.

[9] Y. Chen, "Robot as a Service in Cloud Computing," in *Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE)*, Nanjing, 2010.

[10] J. Rhoton, Cloud Computing Explained: Handbook for Enterprise Implementation, Vienna: Recursive Limited, 2010.

[11] Google and Willow Garage, "Google I/O 2011," Youtube, San Francisco, 2011.

[12] Google, "Android," Cellbots, 23 May 2011. [Online]. Available: http://www.cellbots.com/category/android/. [Accessed 20 November 2011].

[13] M. B. M. Waibel, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle and R. van de Molengraft, "RoboEarth," *Robotics & Automation Magazine, IEEE,* vol. 18, no. 2, pp. 69-82, 2011.

[14] C. Davies, "Hasbro Android Robot Toys get I/O video playtime," SlashGear, 11 May 2011. [Online]. Available: http://www.slashgear.com/hasbro-android-robot-toys-get-io-video-playtime-11151542/. [Accessed 20 November 2011].

[15] K. Drummond and N. Shachtman, "Darpa's Next Grand Challenge: Build Us Lifelike, Humanoid Robots," Wired, 5 April 2012. [Online]. Available: http://www.wired.com/dangerroom/2012/04/darpa-humanoid-robots/. [Accessed 5 April 2012].

[16] G. Clarke, "Father of Java abandons Google for floating robot cloud," The Register, 31 August 2011. [Online]. Available: http://www.theregister.co.uk/2011/08/31/gosling_liquid_robotics/. [Accessed 20 November 2011].

[17] J. Shi, "Good Features to Track," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '94)* , Seattle, WA, 1994.

[18] J. Y. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker," *Intel Corp, Microprocessor Research Labs, OpenCV Documents,* vol. 3, no. 1, pp. 1-10, 1999.

[19] S. Cummins, Pro SharePoint 2010 Disaster Recovery and High Availability, New York: Apress , 2011.

[20] J. Friesen and D. Smith, Android Recipes, New York: Apress, 2011.

[21] N. Kwak, "3D grid and particle based SLAM for a humanoid robot," in *9th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2009)* , Paris, 2009.

Appendix A

Glossary:

A

Android:An open source mobile phone operating system made by Google. It comes with software tools to develop applications for its operating system based phones.

C

Cellbot:Cellbots are a new technology that combine smartphones with any mechanical component for movement or physical arbitration and make up a functional robot. Examples include Cellbots from the toy company Hasbro ©.

Cloud bursting:Cloud bursting is an application deployment model in which an application runs in a private cloud or data center and bursts into a public cloud when the demand for computing capacity spikes. The advantage of such a hybrid cloud deployment is that an organization only pays for extra compute resources when they are needed.

Cloud Computing: Computation, software, data access, and storage services not requiring end user knowledge of neither location nor configuration for mass distribution and usage.Typically described as an IT service model for enabling convenient, on-demand network access to a shared pool of computing resources.Usually mentioned simply as the ‘Cloud’.

Cloud scaling:The option to increase the available virtual servers and processing power/storage capacity during peak performance timings without any noticeable change.

F

Failover Cluster: Failover clusters (also known as High Availability clusters or HA Clusters) are groups of computers that support server applications that can be reliably utilized with a minimum of down-time. They operate by

harnessing redundant computers in groups or clusters that provide continued service when system components fail. Without clustering, if a server running a particular application crashes, the application will be unavailable until the crashed server is fixed. HA clustering remedies this situation by detecting hardware/software faults, and immediately restarting the application on another system without requiring administrative intervention, a process known as failover. As part of this process, clustering software may configure the node before starting the application on it. For example, appropriate filesystems may need to be imported and mounted, network hardware may have to be configured, and some supporting applications may need to be running as well.

HA clusters usually use a heartbeat private network connection which is used to monitor the health and status of each node in the cluster. One subtle but serious condition all clustering software must be able to handle is split-brain, which occurs when all of the private links go down simultaneously, but the cluster nodes are still running. If that happens, each node in the cluster may mistakenly decide that every other node has gone down and attempt to start services that other nodes are still running. Having duplicate instances of services may cause data corruption on the shared storage.

H

Hyper-V:A private cloud computing tool for desktop and network virtualization which provides options like failover clustering and that is available for software development for private and public clouds via free downloads through MSDN subscriptions.

Hypervisor: In computing, a hypervisor, also called virtual machine manager (VMM), is one of many hardware virtualization techniques allowing multiple operating systems, termed guests, to run concurrently on a host computer. It is so named because it is conceptually one level higher than a supervisory program. The hypervisor presents to the guest operating systems a virtual operating platform and manages the execution of the guest operating systems. Multiple instances of a variety of operating systems may share the virtualized hardware resources. Hypervisors are installed on server hardware whose only task is to run guest operating systems. The term can be used to describe the interface provided by the specific cloud computing functionality infrastructure as a service (IaaS).

M

Migration:Live migration is the movement of a virtual machine from one physical host to another while continuously powered-up. When properly carried out, this process takes place without any noticeable effect from the point of view of the end user. Live migration allows an administrator to take a virtual machine offline for maintenance or upgrading without subjecting the system's users to downtime.

S

SaaS, PaaS, and IaaS: Software as a service, sometimes referred to as "on-demand software," is a software delivery model in which software and its associated data are hosted centrally (typically in the (Internet) cloud) and are typically accessed by users using a thin client, normally using a web browser over the Internet.

Platform as a service (PaaS) is a category of cloud computing services that provide a computing platform and a solution stack as a service. In the classic layered model of cloud computing, the PaaS layer lies between the SaaS and the IaaS layers. PaaS offerings facilitate the deployment of applications without the cost and complexity of buying and managing the underlying hardware and software and provisioning hosting capabilities, providing all of

the facilities required to support the complete life cycle of building and delivering web applications and services entirely available from the Internet.

Infrastructure as a Service is a provision model in which an organization outsources the equipment used to support operations, including storage, hardware, servers and networking components. The service provider owns the equipment and is responsible for housing, running and maintaining it. The client typically pays on a per-use basis.

Characteristics and components of IaaS include:

- Utility computing service and billing model.
- Automation of administrative tasks.
- Dynamic scaling.
- Desktop virtualization.
- Policy-based services.
- Internet connectivity.

SDK:Software Development Kit - A set of software creation tools that enable a programmer to create applications from a pre-designed software framework.

Appendix B

1. UML Use Case Diagram for Android App (section 4.7)

Figure B.1 Using the Android App

2. UML Use Case Diagram for Cloud App Admin Portal (section 4.8)

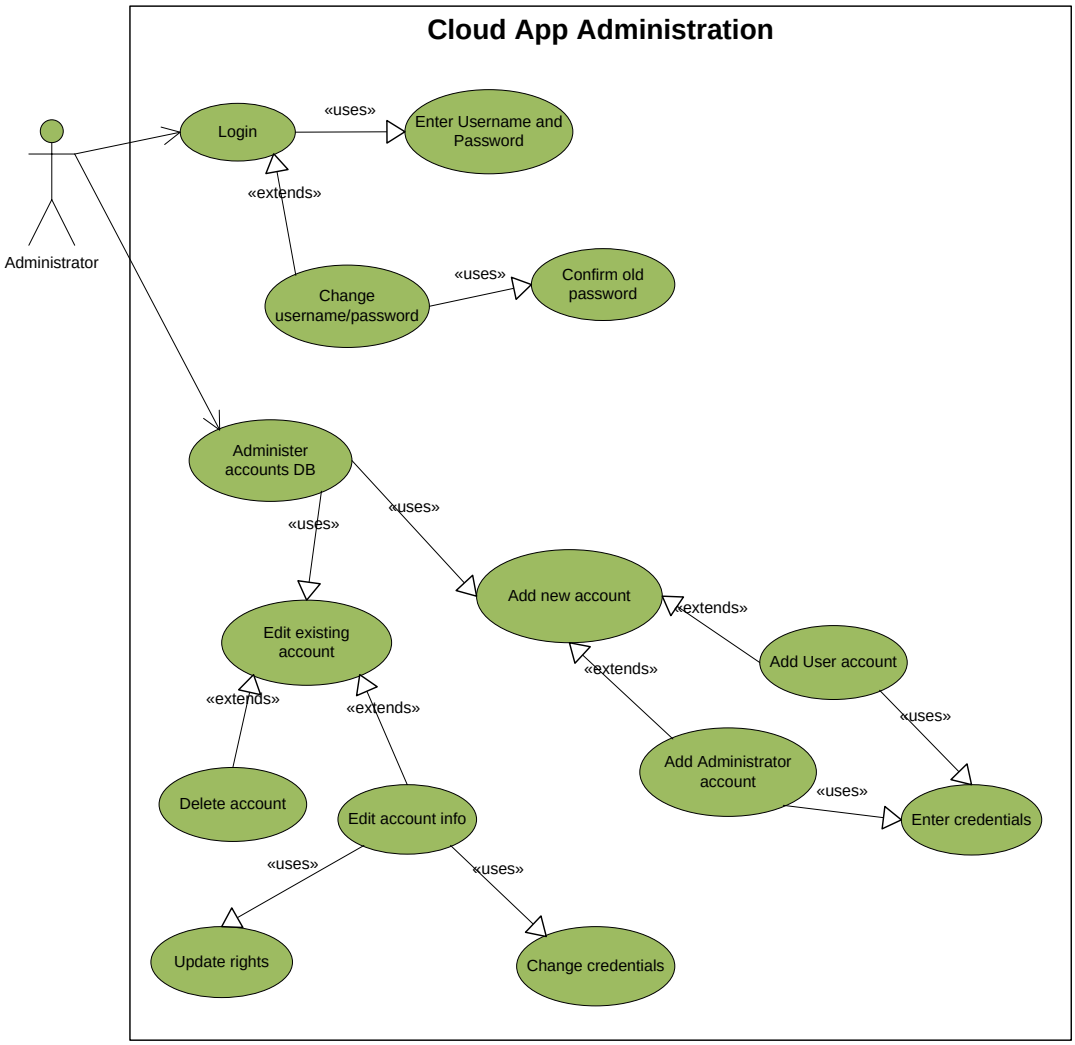

Figure B.2 Administering the Cloud App

3. UML Activity Diagrams(Section 4.9).

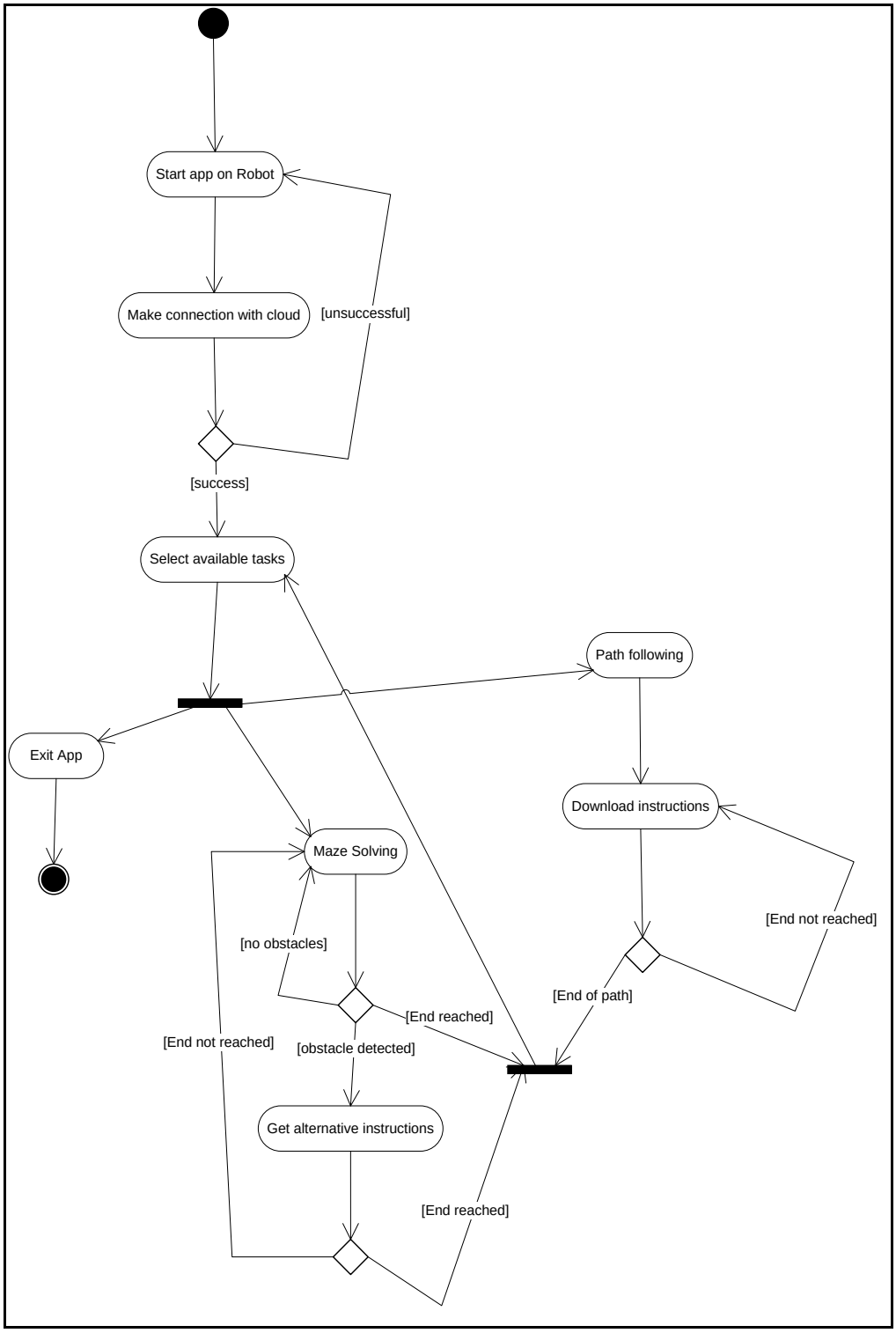

Figure B.3 Service Selection Activity

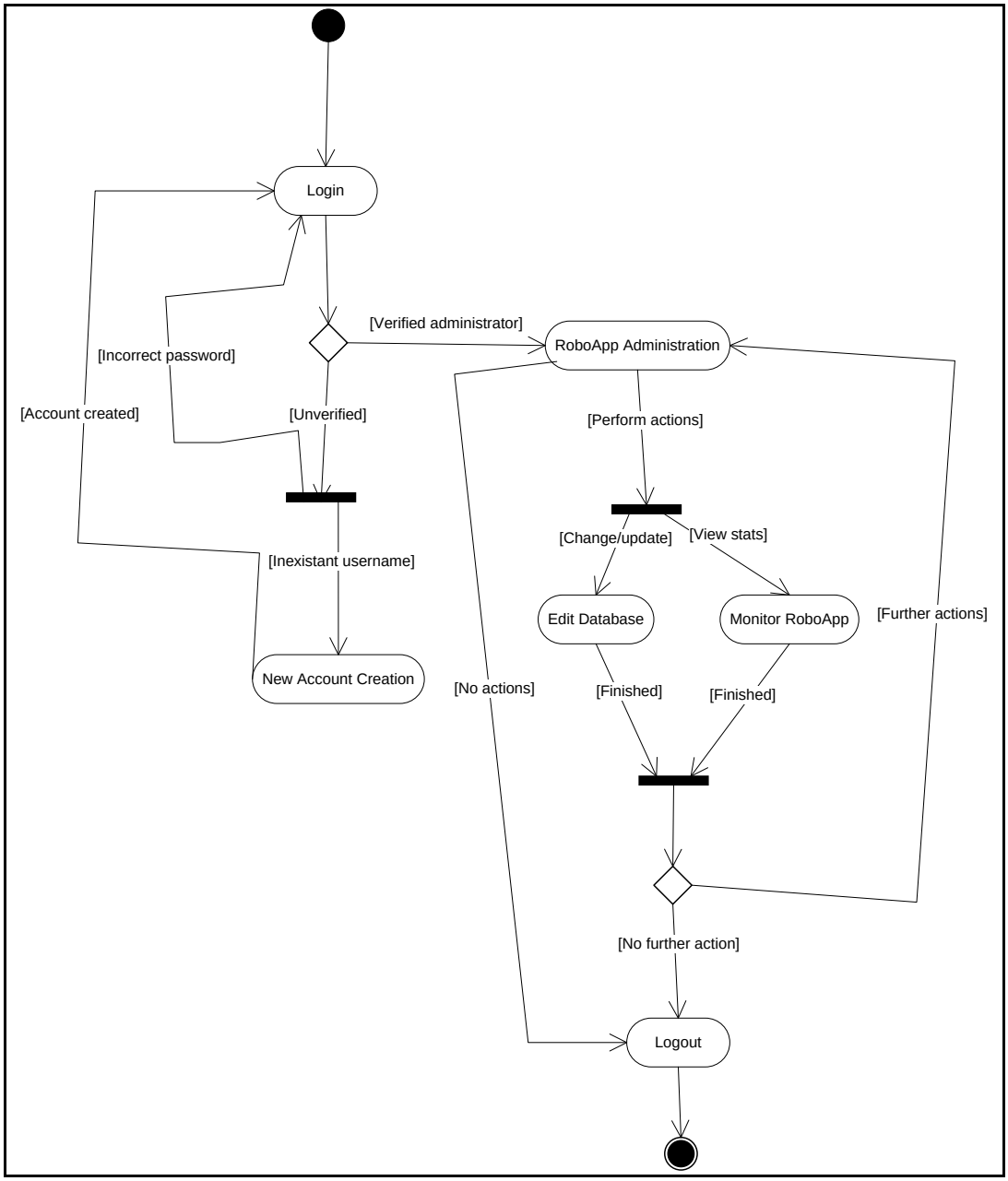

Figure B.4 Administration Activity

4. State Chart Diagrams (section 4.5)

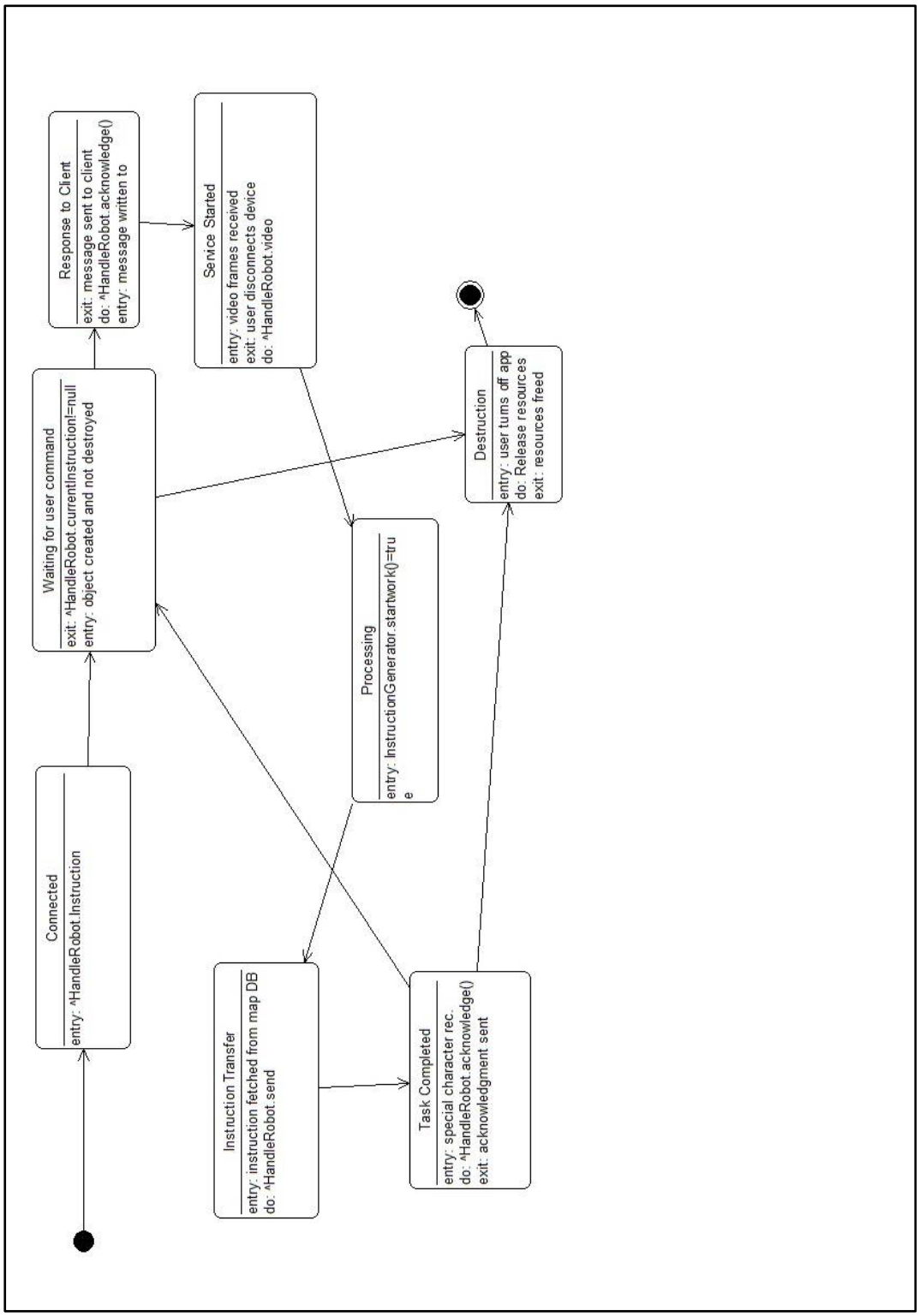

Figure C.5 State Chart for class HandleRobot

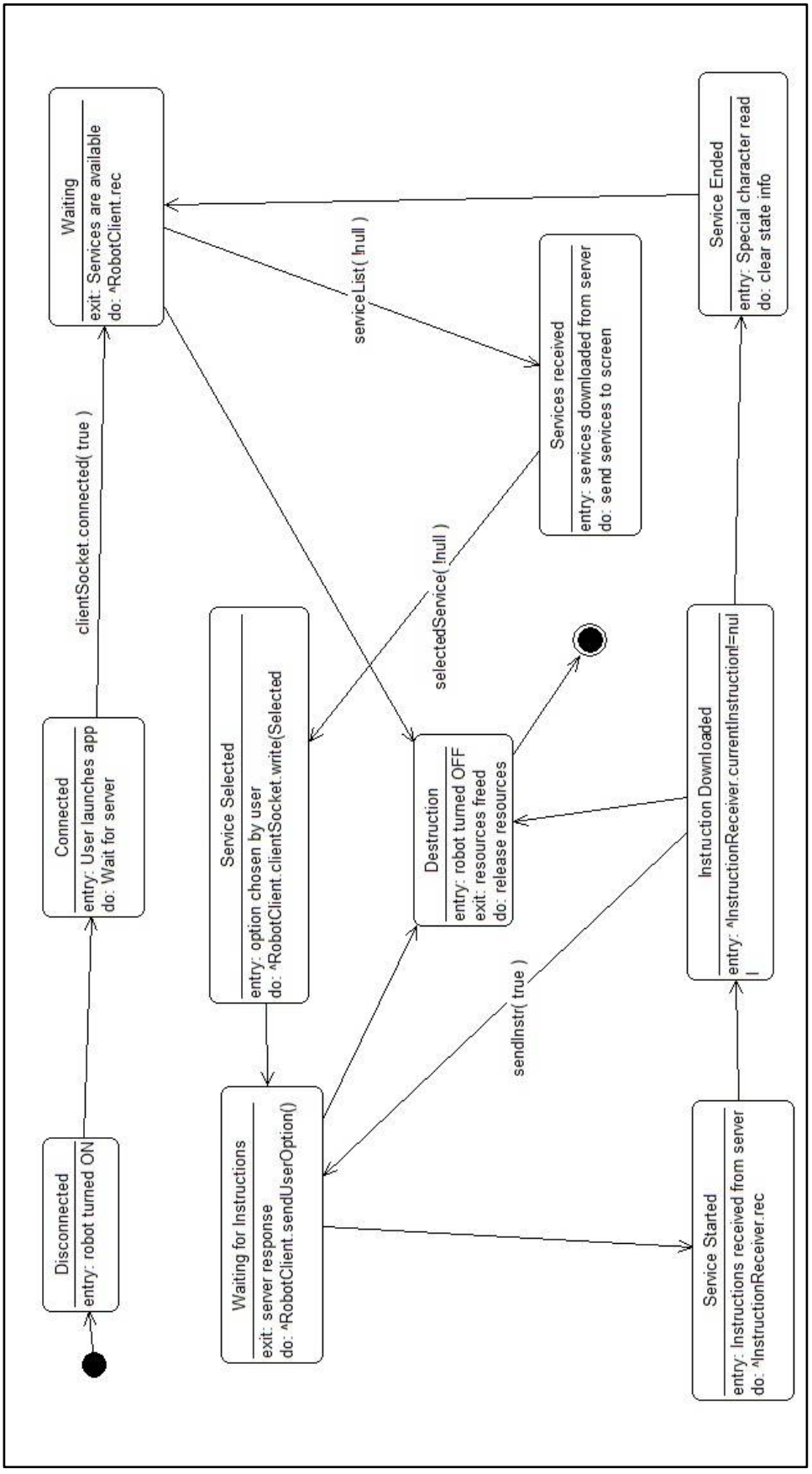

Figure B.6 State Chart for class RobotClient

Appendix C

1. Shi-Tomasi algorithm

Point motion for a deformation area in a picture's pixels matrix D is described by:

$$J(Ax + d) = I(x)$$

where

- J is the current image
- I is the 1st image
- A = 1+D (1 is 2x2 identity matrix and D is the deformation matrix)
- d is the translation

D= $\begin{bmatrix} dxx & dxy \\ dyx & dyy \end{bmatrix}$

D is usually set to zero. Now we measure dissimilarity c.

$$c = \iint_w [J(Ax + d) - I(x)]^2 w(x) dx$$

- w(x) can be 1 or Gaussian function to emphasize center of window
- Differences are squared and summed
- Solved using Newton Rap son method
- Yields 6x6 system of linear equations
- Smaller system available for tracking only

Now the system is to be linearized:

$$J(Ax + d) = J(x) + g^T(u)$$

Where:

g is the unknown guesses of the deformation matrix D.

T = $\iint_w \begin{bmatrix} U & V \\ V^T & Z \end{bmatrix} wdx$

Where:

$$U = \begin{bmatrix} x^2 g_x^2 & x^2 g_x g_y & x y g_x^2 & x y g_x g_y \\ x^2 g_x g_y & x^2 g_y^2 & x y g_x g_y & x y g_y^2 \\ x y g_x^2 & x y g_x g_y & y^2 g_x^2 & y^2 g_x g_y \\ x y g_x g_y & x y g_y^2 & y^2 g_x g_y & y^2 g_y^2 \end{bmatrix}$$

$$V^T = \begin{bmatrix} x g_x^2 & x g_x g_y & y g_x^2 & y g_x g_y \\ x g_x g_y & x g_y^2 & y g_x g_y & y g_y^2 \end{bmatrix}$$

$$Z = \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} .$$

If the two eigenvalues of Z are λ_1, λ_2 then our window of the corner is $\min \ (\lambda_1, \lambda_2) > \lambda$

Where:λ is a predefined threshold.

2. Pyramidal Lucas-Kanade Algorithm

This is a popular differential method for optical flow estimation and feature tracking. The algorithm below has been taken from the paper *Pyramidal Implementation of the Lucas Kanade Feature Tracker Description* (by Jean-Yves Bouquet, Intel Corp.)

Goal: Let $\mathbf{u}$ be a point on image I. Find its corresponding location $\mathbf{v}$ on image J

Build pyramid representations of I and J: $\{I^L\}_{L=0,\dots,L_m}$ and $\{J^L\}_{L=0,\dots,L_m}$

Initialization of pyramidal guess: $\mathbf{g}^{L_m} = [g_x^{L_m} \;\; g_x^{L_m}]^T = [0 \;\; 0]^T$

for $L = L_m$ **down to** 0 **with step of** -1

Location of point $\mathbf{u}$ *on image* I^L: $\mathbf{u}^L = [p_x \;\; p_y]^T = \mathbf{u}/2^L$

Derivative of I^L *with respect to* x: $I_x(x,y) = \dfrac{I^L(x+1,y) - I^L(x-1,y)}{2}$

Derivative of I^L *with respect to* y: $I_y(x,y) = \dfrac{I^L(x,y+1) - I^L(x,y-1)}{2}$

Spatial gradient matrix: $G = \displaystyle\sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} I_x^2(x,y) & I_x(x,y)\,I_y(x,y) \\ I_x(x,y)\,I_y(x,y) & I_y^2(x,y) \end{bmatrix}$

Initialization of iterative L-K: $\overline{\nu}^0 = [0 \;\; 0]^T$

for $k = 1$ **to** K **with step of** 1 (or until $\|\overline{\eta}^k\| <$ accuracy threshold)

Image difference: $\delta I_k(x,y) = I^L(x,y) - J^L(x + g_x^L + \nu_x^{k-1}, y + g_y^L + \nu_y^{k-1})$

Image mismatch vector: $\overline{b}_k = \displaystyle\sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} \delta I_k(x,y)\,I_x(x,y) \\ \delta I_k(x,y)\,I_y(x,y) \end{bmatrix}$

Optical flow (Lucas-Kanade): $\overline{\eta}^k = G^{-1}\,\overline{b}_k$

Guess for next iteration: $\overline{\nu}^k = \overline{\nu}^{k-1} + \overline{\eta}^k$

end of for-loop on k

Final optical flow at level L: $\mathbf{d}^L = \overline{\nu}^K$

Guess for next level $L-1$: $\mathbf{g}^{L-1} = [g_x^{L-1} \;\; g_y^{L-1}]^T = 2\,(\mathbf{g^L} + \mathbf{d}^L)$

end of for-loop on L

Final optical flow vector: $\mathbf{d} = \mathbf{g}^0 + \mathbf{d}^0$

Location of point on J: $\mathbf{v} = \mathbf{u} + \mathbf{d}$

Solution: The corresponding point is at location $\mathbf{v}$ on image J

APPENDIX D

BAM specifications:

The Element Direct BAM (short for Bluetooth Adapter Module) enables wireless control of the iRobot® Create™ robot from a Windows, Macintosh, or Linux PC. The BAM connects to the Create's cargo bay port – without any extra wires or cables. The BAM provides a virtual serial port connection between a Bluetooth host and Create. A PC can communicate with Create in the same way it would as if it were attached with a serial cable. The BAM gives a user complete wireless control of Create. It also exposes Create's programmable IO, making it easy to connect additional hardware. Bluetooth is a communications system intended to replace the cables connecting electronic devices. Unlike Wi-Fi and other wireless systems, Bluetooth is both a low-power and a low-cost option, making it ideal for use with iRobot Create. Bluetooth operates in the unlicensed ISM band at 2.4 GHz and employs a frequency hopping radio to combat interference from other RF sources.

About BAM:

The BAM is a Class 1 (high power) Bluetooth device. It is capable of communicating with a Bluetooth-enabled host, such as a desktop PC, laptop, or PDA. There are two basic types of Bluetooth hardware: Bluetooth hosts and Bluetooth devices. A laptop PC is an example of a Bluetooth host. A wireless mouse, wireless printer, and the BAM are all examples of Bluetooth devices. In order to connect to the BAM, you must have a computer which can act as a Bluetooth host. Please consult your computer's documentation to determine whether it is a Bluetooth-enabled host.

Windows desktop PCs generally require an external USB Bluetooth radio.

Laptops may have an internal radio or require an add-on USB Bluetooth radio.

Androids may be a Bluetooth host already, or they may require an add-on card.

BAM Features:

The BAM appears to the host computer or PDA as a Bluetooth Serial Port. It provides a virtual serial cable connection between the host and Create. From the host's prospective, Create is connected to a wired serial port, and the host can communicate and control Create through the iRobot Open Interface as if Create were attached to the host with its included RS-232 serial cable. The BAM has an IO Connector which enables users to add their own hardware to Create. The DB-25 connector on the bottom edge of the BAM plugs directly into the Create's Cargo Bay Connector.

Indicator LED:

A red indicator LED displays the BAM's current state. It provides a simple method to determine whether the BAM is connected or disconnected to the host. When Create and BAM are first powered on, the LED will flash 26 times quickly. When the BAM is connected to a Bluetooth host, the LED will flash once per second. When the BAM is disconnected from a host, the LED flashes once every 3 seconds.