Survivable Cloud Multi-Robotics Framework for Heterogeneous Environments

by

Vikash Ramharuk

submitted in accordance with the academic requirements for
the degree of

Master of Science in Computing (Computer Science)

at the

School of Computing, College of Science, Engineering and Technology (CSET),
University of South Africa

Supervisor: Professor Isaac Olusegun Osunmakinde

February 2015

DECLARATION

Student number: 50089625

I, Vikash Ramharuk declare that the dissertation titled **Survivable Cloud Multi-Robotics Framework for Heterogeneous Environments** is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references.

_____ _____

SIGNATURE DATE

DEDICATION

This study is dedicated to my late parents

Romesh Ramharuk and *Charmaine Bisnath*,

for their courage and support in ensuring I persevered with my studies.

Thank you for always telling me to *reach for the stars*.

Wish you were here!

ACKNOWLEDGEMENTS

Only one who devotes himself to a cause with his whole strength and soul can be a true master. For this reason mastery demands all of a person ~ Albert Einstein

I would sincerely like to thank my supervisor, Professor Isaac Osunmakinde, from an academic perspective. You were great and assisted me greatly in guiding me through this study. You will forever be part of one of the most significant experiences in my life. Thank you for your guidance and hard work put into this research study. Thank you for all the critical reading, unfailing support, supervision and dedication to quality. Most of all thank you for never allowing me to give up. A famous quotation by Albert Einstein best sums up the relationship we shared during this research study: *"I never teach my pupils, I only provide the conditions in which they can learn" – Albert Einstein.*

Estelle De Kock: Thank you for all your support. It might not be often that someone outside the supervisor, family and friends are acknowledged for their support but without you I do not think I would have been able to complete my research study. Thank you for actually finding Prof. Osunmakinde to be my supervisor after I was registered incorrectly for IS instead of Computer Science. Thank you for ensuring I get registered every time. Most of all, thank you for attending to my numerous queries and questions, no matter what time of the hour it may be.

Last but not least, I will forever be grateful to my friends, Clinton, Runga, Shaun, Rama and Jason, my family and colleagues who supported me through this journey and provided me with support and motivation when it was sorely needed. Sometime we take the journey only to realize the solution is just the realization of the problem really is …

ABSTRACT

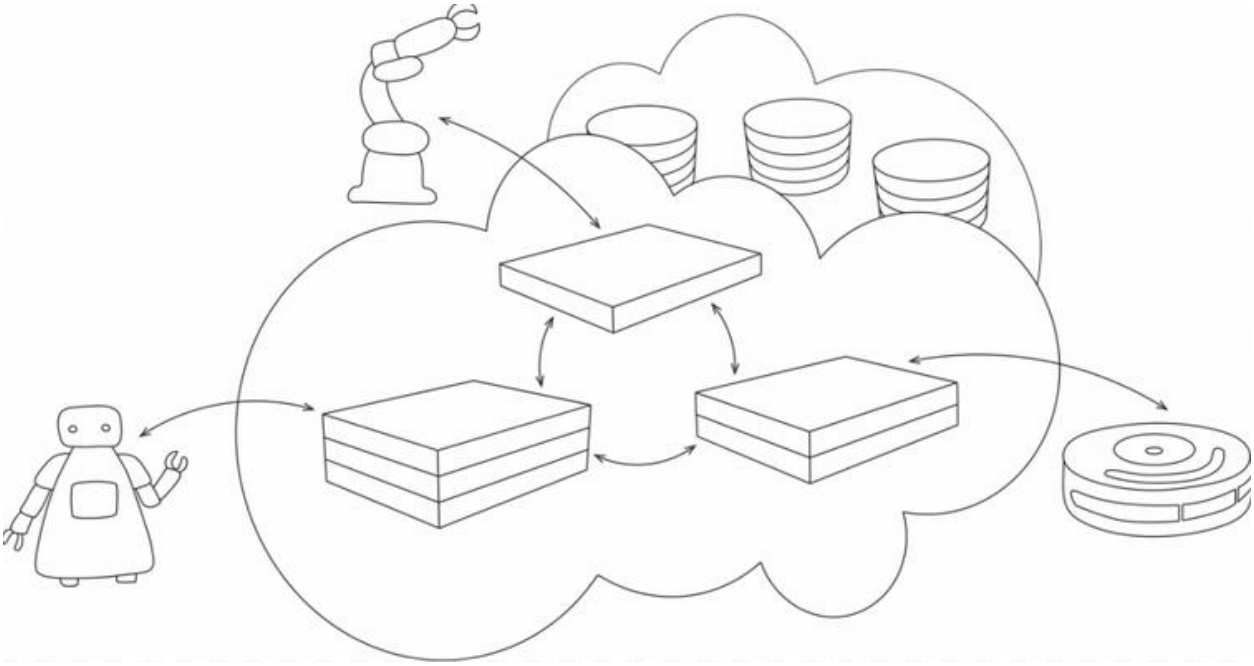
The emergence of cloud computing has transformed the potential of robotics by enabling multi-robotic teams to fulfil complex tasks in the cloud. This paradigm is known as "cloud robotics" and relieves robots from hardware and software limitations, as large amounts of available resources and parallel computing capabilities are available in the cloud. The introduction of cloud-enabled robots alleviates the need for computationally intensive robots to be built, as many, if not all, of the CPU-intensive tasks can be offloaded into the cloud, resulting in multi-robots that require much less power, energy consumption and on-board processing units.

While the benefits of cloud robotics are clearly evident and have resulted in an increase in interest among the scientific community, one of the biggest challenges of cloud robotics is the inherent communication challenges brought about by disconnections between the multi-robotic system and the cloud. The communication delays brought about by the cloud disconnection results in robots not being able to receive and transmit data to the physical cloud. The unavailability of these robotic services in certain instances could prove fatal in a heterogeneous environment that requires multi-robotic teams to assist with the saving of human lives. This niche area is relatively unexplored in the literature.

This work serves to assist with the challenge of disconnection in cloud robotics by proposing a survivable cloud multi-robotics (SCMR) framework for heterogeneous environments. The SCMR framework leverages the combination of a virtual ad hoc network formed by the robot-to-robot communication and a physical cloud infrastructure formed by the robot-to-cloud communications. The Quality of Service (QoS) on the SCMR framework is tested and validated by determining the optimal energy utilization and Time of Response (ToR) on drivability analysis with and without cloud connection. The experimental results demonstrate that the proposed framework is feasible for current multi-robotic applications and shows the survivability aspect of the framework in instances of cloud disconnection.

Keywords:

Cloud, Disconnection, Drivability, Multi-robotics, Robot to Cloud, Robot to Robot, Statistical Region Merging, Survivability.



What if robots and automation systems were not limited by onboard computation, memory, or programming?

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
AI	Artificial Intelligence
API	Application Programming Interface
CPU	Central Processing Unit
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JAX-RS	Java API for RESTful Web Services
JAX-WS	Java API for XML Web Services
JEE	Java Enterprise Edition
MRS	Multi Robotic System
QoS	Quality of Service
R2C	Robot-to-cloud
R2R	Robot-to-robot
ROS	Robot Operating System
RPI	Remote Procedure Call
SDK	Software Development Kit
SDL	Software Development Lifecycle
SMTP	Simple Mail Transfer Protocol
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
URI	Uniform Resource Identifier
WS	Web Services
WWW	World Wide Web
XML	Extensible Mark-up Language
XSD	XML Schema Definition

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

Multi-robots are slowly becoming an integral part of the society. In recent years, multi-robots have been performing primary tasks such as obstacle avoidance, vision processing, localization, path planning and environment mapping [1]. The past decade has also seen the introduction of multi-robots into the industrial sector to perform repetitive and dangerous tasks such as assembly and packaging [2]. The introduction of multi-robots into these different environments has also had an important economic and social impact on human lives over the past decade [3]. These multi-robots have been very successful in these areas owing to the precision, accuracy and speed that can be achieved in these structured environments, with every robot working independently of others [3].

The term "cloud robotics" was introduced in 2010 by James Kuffner at Google [4]; it describes a new paradigm in robotics in which robotic services are leveraged off the cloud in order to reduce the computational processing power and data storage required locally by robots. This can be seen as the next generation of networked robotics, which refers to a group of robotic devices connected via a wired/wireless communication network [5]. Conventional robots such as PR2 robots, Robo-dogs and aerial unmanned vehicles have all been built with large computational processing units (CPU) and large batteries to provide power to the robots, because all the computational tasks such as map building, navigation and object recognition had to be performed locally and required a substantial amount of pre-programmed information [6]. The introduction of cloud-enabled robots alleviates the need for such robots to be built, as many, if not all, of the CPU-intensive tasks can be offloaded into the cloud, resulting in multi-robots that require much less power, energy consumption and onboard processing units. The paradigm of cloud robotics also paves the way forward for multi-robots to expand their services in the future and offers the possibility of multi-robots sharing services and information with one another by creating a central knowledge base in the cloud [6].

During the past few years, scientists and researchers all over the world have started attempting to build cloud computing frameworks to support both individual robots and multi-robots. Most

notably, researchers at the Institute of Dynamic Systems and Control in Switzerland have created an open-source project called RoboEarth in an attempt to build a web community for robots to share information and object models autonomously [7]. The RoboEarth cloud engine is a platform as a service (PaaS) that allows other users to run their robotic services on the RoboEarth cloud [7]. A team of researchers in Singapore has created the DAvinCi project [8], which was based on the robot operating system (ROS) [9] messaging framework and used the ROS master node to gather data into the Apache Hadoop cluster [10], and illustrated the benefits of cloud computing by parallelizing the FastSLAM algorithm [11].

The Ubiquitous Network Robot Platform (UNR-PF) [12] is an ongoing project that primarily focuses on using the cloud as a mechanism to create a network between robots, sensors and mobile devices. Rosbridge is another open-source project, which focuses on the external communication between a robot or a multi-robot team and a single ROS environment in the cloud [13]. The emergence of smartphones has enabled Google developers to create Android-powered robots that allow users to control them using their smartphones on platforms such as Lego Mindstorm and iRobot [14]. The use of cloud robotic models has also been extended to the medical industry, which has robots accessing the cloud infrastructure to gather data to assist children in a hospital in Italy with face detection and speech recognition [15].

1.2 PROBLEM STATEMENT

The start of the new century has seen a shift in the requirements of robots. The robots of the last century were geared more towards mass production of products where they carried out repetitive tasks in a controlled environment [1]. However, the current trend is to have robots that are capable of solving complex problems, such as path planning, object recognition and navigation, in less controlled environments [1]. This has subsequently resulted in a need for robots that are more intelligent and autonomous, with the ability to detect drivable road regions in heterogeneous environments.

In any cloud robotic environment, as the data and the number of robots and robotic services increases, the efficiency, reliability and fairness of the data retrieval become increasingly challenging [8]. Another challenge brought about by the increasing number of robots in a cloud

robotic environment is the multiple parallel client robot requests sent to the cloud server. If these requests are not handled satisfactorily and complied with within the agreed time period, this would result in a breach of the quality of service (QoS) agreement [8]. Resource scheduling therefore, becomes critical in a cloud robotic model to ensure that the requests are handled fairly, efficiently and reliably and are fulfilled within the constraints of the QoS.

While the benefits of cloud robotics are clearly evident and have resulted in an increase in interest among the scientific community, one of the biggest challenges of cloud robotics is the inherent communication challenges brought about by disconnections between the multi-robotic systems (MRS) and the cloud. The worst-case communication delays in the existing cloud robotic models can be generally represented by $O(N \log N)$, where N is a node in the network, and two nodes are not within communication range with each other and therefore not neighbors [11]. The communication delays brought about by the cloud disconnection results in robots not being able to receive and transmit data to the physical cloud, thus resulting in the robots being unable to complete certain tasks. The unavailability of these robotic services in certain instances could prove fatal in heterogeneous environments that require multi-robotic teams to assist with the saving of human lives.

In many instances multi-robots have to work together as a group of multi-robot teams to achieve a common objective. The MRS are required to be cooperative while fulfilling tasks in dynamic, complex and uncontrolled environments, where the primary challenge for the robots would be to understand and interpret the immediate environment using robotic vision to detect drivable road regions [2]. This has resulted in robots being built with complex system architectures and large computing needs. In addition this has contributed to robots requiring large onboard computational devices, dedicated power supplies and large local data storage, all of which become responsible for a substantial amount of the robot's energy consumption. It is neither feasible nor cost-efficient for these robots to move in heterogeneous environments with large onboard computational devices and local data storage. Furthermore, this results in duplication of work for the robots, increased operational overhead, longer roll-out periods and limited processing capability [3].

The advancement of cloud robotics for MRSs is limited by the resource, information and communication constraints in the current framework. During the past few years, first efforts have emerged to build a cloud-computing framework for multi-robotic teams [7, 8]; however, the development of a survivable cloud multi-robotics (SCMR) framework for heterogeneous environments is relatively unexplored.

1.3 RESEARCH OBJECTIVES

The research objectives for this study are epitomized by the following question:

What if robots and automation systems were not limited by on-board computation, memory, or programming?

The emergence of cloud computing in the 21st century has enabled the realization of large parallel computation and real time sharing of vast data resources. The ability of robots and automated systems to leverage the possibilities brought about by cloud computing characterizes the scientific rationale for the study. Therefore the primary objective of the study is to develop a framework for cloud multi-robotic teams that can overcome some challenges, such as cloud disconnection in heterogeneous environments. The following list describes the additional sub-objectives of the study:

- Implement, characterize, formulate and perform visual observations on the proposed SCMR framework to detect drivable road regions with and without cloud connection in heterogeneous environments with the objective of determining the robustness, interoperability, mobility and reliability of the proposed framework.
- Implement the proposed SCMR framework and conduct detailed experiments by validating the optimal QoS in terms of fast response, reliable response and resource scheduling with and without cloud disconnection.
- Evaluate the proposed and implemented SCMR framework against the design requirements and related solutions and formulate an elaborate optical task energy execution model that will identify whether a task should be executed locally by the robot or offloaded into the cloud server with the intention to conserve energy.

All the work done to date in the field of cloud robotics has contributed significantly to the body of knowledge and illustrated the potential benefits that can be achieved in cloud robotic models by offloading data-intensive and computationally intensive tasks into the cloud, thereby

enabling researchers to realize the elevated goal of cloud robotics. However, the complexities concerning cloud disconnection, cloud-robotic optimal energy execution and cloud-robotic QoS for multi-robotic teams working in heterogeneous environments remain relatively unexplored. Therefore, in summation, the motivation for this study is the development of a comprehensive framework that will cater for these complexities that have been left unexplored and contribute to the ever growing body of scientific knowledge.

1.4 RESEARCH QUESTIONS

The research questions defined in this section are set against the purpose of the study, which is to develop a cloud multi-robotics framework. In order to construct a functional and comprehensive cloud multi-robotics framework, it is necessary to understand the current computing models for cloud robotics, current capabilities, challenges and technologies in cloud robotics, technical architecture and architectural design principles for multi-robotics using cloud services. The overall dissertation is therefore based on the following main question and supporting research questions:

Main research question*: How can a cloud multi-robotic framework be developed for surviving cloud disconnection over independent on-board processing by considering a team of robots' perception/vision, path planning and navigation?*

The following is a list of the supporting questions that are addressed in this study to achieve the stated objective comprehensively and answer the main research question stated above adequately:

Supporting research question 1: *How can quality and reliable multiple drivable regions be simultaneously obtained with and without cloud disconnections in heterogeneous environments?*

Supporting research question 2: *How is the QoS optimized in terms of fast response, optimal task execution policy and time of response in the cloud multi-robotics framework?*

Supporting research question 3: *To what extent does the level of complexity analysis of the developed framework efficiently reduce the computational energy consumed overheads based on on-board processing and cloud mechanisms?*

1.5 RESEARCH DESIGN

The purpose of this study is to propose a cloud multi-robotic framework that could provide a basis for robots working in heterogeneous environments to execute specific tasks. In order to achieve this purpose, a combination of research techniques was used in this study. These include literature reviews, argument constructions, propositions, prototyping, contextual evaluations, experimentation and testing.

A technical survey was conducted on the existing literature on cloud robotics and cloud computing, with specific reference to the issue of cloud disconnections for systems connecting to a cloud access point. Through this survey, key information was extracted to identify the most salient characteristics in the existing cloud robotic frameworks and the current challenges faced in the field of cloud robotics, which at this point still leave MRS vulnerable to cloud disconnections.

Using the information studied, classifications were proposed of different types of cloud robotic models, together with the pros and cons of each model. Thereafter a hybrid cloud robotic model to support the proposed SCMR framework was constructed through further arguments and propositions. A prototype of the SCMR framework, as a proof of concept of the framework, was developed using a client/server application built in Java. The prototyping of the framework included evaluating suitable infrastructure, prototype design, test case construction and evaluation of test results. The prototype was scientifically tested using the statistical region merging SRM technique to illustrate the survivability aspect of the framework, which indicated that even in the event of cloud disconnection mobile robotics systems were able to perform drivable road detection.

1.6 RESEARCH CONTRIBUTIONS

This section describes the overall research contributions of the study to the scientific body of knowledge, resulting in accredited and peer-reviewed publications.

1.6.1 Contributions to the scientific body of knowledge

The research study focuses on addressing some of the technical challenges faced in cloud robotics by proposing an SCMR framework, which would pave the way for more intelligent, robust, efficient and interoperable cloud multi-robotic solutions. Therefore, the major contributions in the study are the following:

- Development of a new SCMR framework that could assist with offloading the computational burden from local multi-robots to the cloud and handle disconnections between the cloud and the multi-robots in heterogeneous environments.
- The modelling of the theory in wireless communication, Big-O notations, concepts of cloud multi-robotics, derivations of energy utilization equations and drivability analysis based on an image-processing technique known as SRM.
- Detailed experimental evaluations conducted on publicly available heterogeneous terrains validated by measuring QoS and determining the optimal energy utilization on MRS drivability analysis with and without cloud connection for engineering practices.

1.6.2 Publications resulting from this study

The following peer-reviewed publications were generated and published as a result of the scientific research conducted for this dissertation:

- Ramharuk V. and Osunmakinde I.O. 2014. Development of a Survivable Cloud Multi-Robotics Framework for Heterogeneous Environments. International Journal of Advanced Robotics Systems, InTech Publishers, Vol. 11(164), ISSN: 1729-8806, (ISI journal).
- Ramharuk V. and Osunmakinde I.O. 2014. Cloud Robotics: A Framework Towards Cloud-enabled Multi-robotics Survivability. In proceedings of the South African Institute for Computer Scientists and Information Technologists (SAICSIT 2014) conference Gauteng, South Africa, ACM ISBN: 978-1-4503-3246-0, pp 82 - 92, (Best paper award - Computer Science), *DoE accredited*

1.7 RESEARCH ETHICAL CONSIDERATION

In accordance with the procedures and processes set out by the UNISA Computing College Research Ethics Committee the scientific integrity of the study was protected by applying the following measures:

- No data was manipulated or forged during the experimental evaluations of the SCMR framework
- The findings of the study were reported on the basis of the actual test results and these findings were not distorted in order to support preconceived views.
- Plagiarism was avoided to ensure that the work presented is original and authentic.
- The dissertation has been run through plagiarism software successfully.
- All sources that were used and that contributed to the study were referenced accordingly.

1.8 RESEARCH SCOPE AND LIMITATIONS

This section outlines the research scope and limitations of the study.

1.8.1 Research scope

The focus of this study is mainly on enabling multi-robotic teams to offload computationally intensive tasks to the cloud server for processing and ensuring that in the event of cloud disconnection the multi-robotic teams are able to complete these tasks without too much delay. In addition, the critical discussion in this dissertation deals with the concepts of cloud robotics, cloud computing, cloud disconnection and robotic vision and image processing.

1.8.2 Research limitations

The SCMR framework prototype developed for this dissertation is intended to demonstrate the proof-of-concept implementation, rather than the actual realization of a fully-fledged cloud multi-robotic team environment executing specific robotic tasks. Furthermore, it should be noted that the prototype was implemented and tested in the Java environment, but could also be tested in other environments, such as Microsoft .NET.

Other concepts that form part of cloud robotic concepts, such as resource sharing, machine sharing and learning and robotic hardware, could be beyond the scope of this study. In additional, security measures for protecting robotics systems against malicious intent is an area that has been well researched in the scientific community and has therefore not been considered as part of this dissertation. Nevertheless, references to these concepts are made in subsequent chapters of this study, for clarifying and providing more insight into the proposed solution.

1.9 RESEARCH STRUCTURE

In order to develop an SCMR framework for heterogeneous environments and answer the research questions, this study comprises five chapters, as outlined in Figure 1. In chapter 1, this chapter of the study, an introduction to the study, motivation for the study and research questions is presented. In addition, a brief summary of the other aspects of the study is presented. In chapter 2, the paradigm of cloud robotics, its history, background and the current technologies used to support cloud robotics are discussed.

In Chapter 3 detailed insights into the design, development, services and evaluation mechanisms of the SCMR framework are presented. In addition the use of the SCMR framework to perform drivability analysis using the SRM technique is discussed. The proposed SCMR framework is formulated and tested in chapter 4. The experimental results are then presented, illustrating the survivability aspect of the framework, together with the QoS. The study is concluded in Chapter 5 by providing the contributions made by the study and highlighting some of the outstanding challenges that remain to be addressed by the proposed SCMR framework. The intention is that some of these challenges could potentially form the basis for future research.

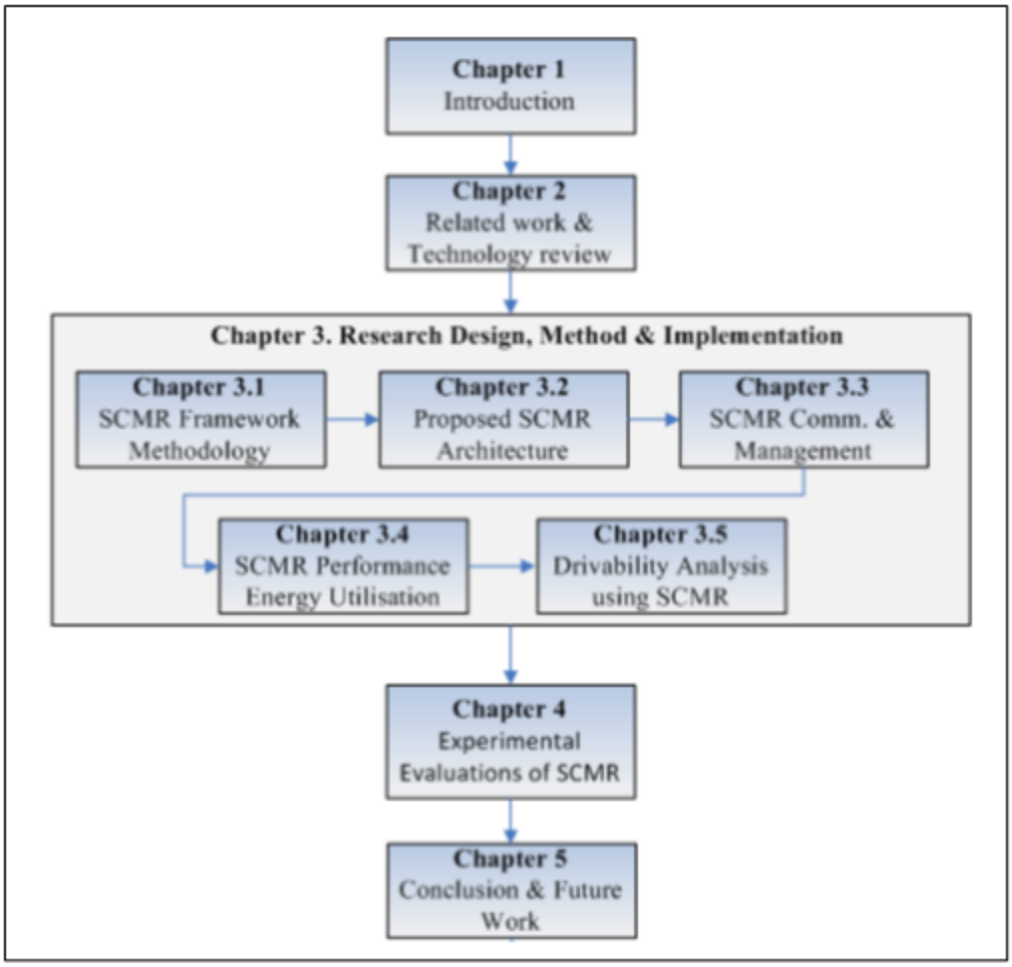


Figure 1.1: Dissertation outline

CHAPTER 2: LITERATURE REVIEW AND THEORETICAL BACKGROUND

2.1 INTRODUCTION

The purpose of this chapter is to review literature and technologies related to cloud robotics. The review is required to gain insight into the dissertation's subject matter and to avoid repeating similar experiments and implementations that have previously been published by accredited scholars and researchers. The review is also important as it provides direction for the research study and lays down the foundation for the analysis of problems and solutions that have previously been researched. These would include identifying trends, challenges, what works and what does not work, as well as proposing solutions to address the current problems in the area of cloud robotics. Therefore this review is also focused on acquiring knowledge about the problems that exist and exploring ways to solve them.

As depicted by Figure 2.1, this chapter consists of seven main sections. Section 2.1 provides the introduction to the chapter, which is then followed by section 2.2, which provides an extensive discussion on cloud computing. Thereafter section 2.3 describes in detail the paradigm of cloud robotics, while section 2.4 describes the different cloud robotic models that are currently available, including the proposed model used in this dissertation. Section 2.5 introduces the constant challenge of cloud disconnection and the impact it has on cloud robotics. The focus of the study then shifts to the key scientific concept of robotic vision in section 2.6 and describes in detail the methods for autonomous navigation, including key topics such as drivable road detection and image processing. Finally, section 2.7 provides a summary of the chapter.

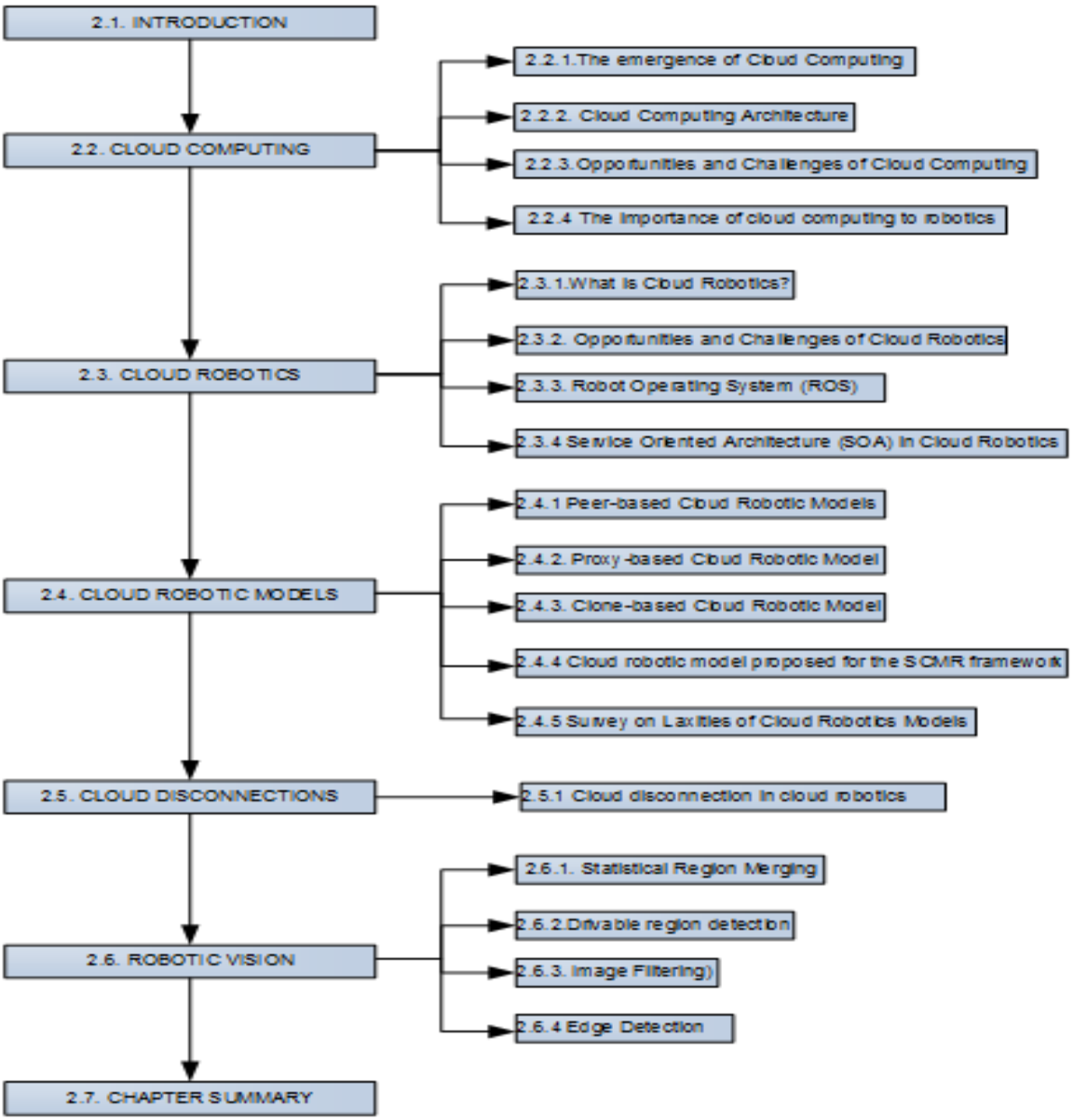


Figure 2.1: Graphical representation of Chapter 2

2.2 CLOUD COMPUTING

In sections 2.2.1 to 2.2.3 a description of what cloud computing is, its history, architecture, challenges, opportunities and the economic impact of cloud computing are described.

2.2.1 The emergence of cloud computing

Cloud computing is an internet-based paradigm where all the computing resources such as processing, memory, and storage are hosted in a cloud, which consists of multiple computers

interlinked together in a complex manner [16]. The initial concept of cloud computing dates as far back as the 1960s, when John McCarthy indicated that "computation may someday be organized as a public utility" and this was further characterized in the book by Douglas Parkhill titled "*The Challenge of the Computer Utility*", which was first published in 1966 [17]. The history of the term *cloud* is derived from the field of telecommunications; telecommunication service providers (SP) were the first companies to offer virtual private network (VPN) services with negotiated QoS between the consumer and the telecommunication SP at reduced costs [18]. VPN enabled the telecommunication SP to switch traffic in order to balance the utilization of the overall network as opposed providing dedicated point-to-point (P2P) data circuits to its customers, which resulted in bandwidth wastage [17].

More recently, however, researchers and scholars in the science and computing communities have attempted to define the actual concept of cloud computing. One of the most elaborate and complete definitions of cloud computing is given by [18] as part of their research work; they indicate that "Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiations between service providers and its consumers."

The major global technology companies, such as Google, IBM, Microsoft and Amazon, have also realized the potential benefits of cloud computing brought about by incorporating the concepts of parallel and distributed computing to provide shared resources; hardware, software and information, to computers or other devices on demand [17]. This has resulted in these technology companies providing distributed systems that follow the "pay as you use" model as opposed to the traditional licensing model, which resulted in customers paying for facilities they do not use in a software package. Amazon, for example offers its customers Elastic Compute Cloud (EC2) and Simple Storage Service (S3). Amazon customers on EC2 can perform computation on the stored data in the cloud, while customers on S3 can store and access their personal data in the cloud. This approach of making the hardware infrastructure available as a service is called infrastructure as a service (IaaS). Another approach is to provide the platform, which can also include the required operating system and software, over the

hardware infrastructure. This is called a PaaS; the Google application engine is a perfect example of PaaS [10]. A more common approach to cloud computing in recent years has been to provide both the application and hardware as a service. This is called software as a service (SaaS) and examples include Google Docs, Dropbox, ZOHO and Salesforce.com [20]. The next section elaborates further on the stack of services in cloud computing, together with the architecture that is required to enable these services.

2.2.2 Architecture of cloud computing

In general, cloud computing can be visualized as X as a service (XaaS), where X can be virtually anything [21]. Some examples of X include:

- Hardware as a service (Haas),
- Communication as a service (Caas),
- Database as a service (Daas), and
- Storage as a service (Saas).

Figure 2.2 represents a conceptual view of the cloud computing architecture, indicating the services that cloud computing can perform. The basic cloud services are represented at the bottom of Figure 2.2, such as cloud infrastructure and cloud storage. Moving higher up in the stack brings about the more complex cloud services, such as cloud applications and cloud clients [22].

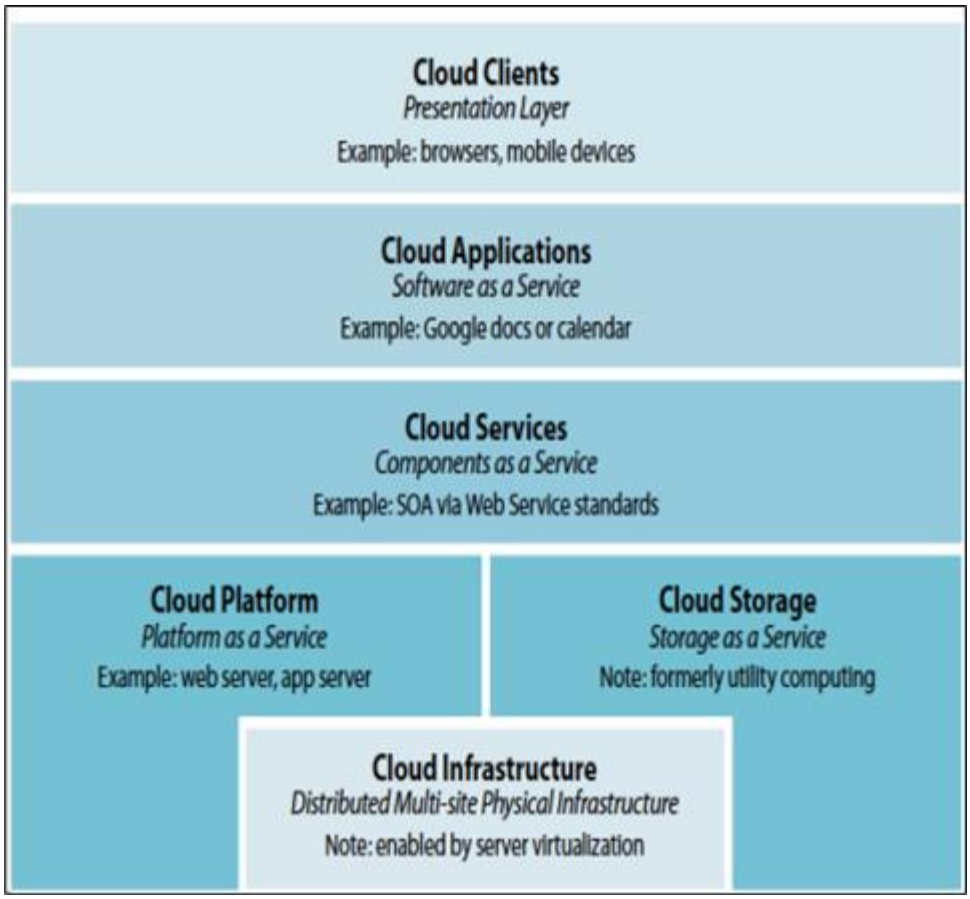


Figure 2.2: Cloud computing services presented as a stack of services [22]

There are many different ways in which the stack of cloud services could be represented, but at a high level there are three prominent approaches to cloud services that depend on the manner in which the resources are made available [22], namely IaaS, PaaS and SaaS. Figure 2.3 shows the cloud computing approaches together with examples of each.

	SaaS	PaaS	IaaS
	Software as a Service	*Platform as a Service*	*Infrastucture as a Service*
Applications	-Government Apps -Communication (email) -Collaboration -Productivity tools (Office) -ERP	-Application Development -Security Services -Database Management	-Servers -Network -Storage -Manageemnt -Reporting
Examples	-Salesforce.com -Netsuit -Oracle -IBM -Google Apps	-GAE -Microsoft Azure -Amazon EC2	-GoGrid -Flexiscale -Joyent

Figure 2.3: Cloud computing services and examples [23]

SaaS is delivered as a cloud application over the internet, which eliminates the need for consumers to install and run applications on the end-users' systems [23]. Therefore consumers of the hosted application in the cloud do not need to worry about the infrastructure or the platform and access to the actual platform is made available through many devices by a web browser.

PaaS is delivered as a computing platform using the cloud infrastructure and typically has all the applications required by the consumer deployed on the infrastructure [23]. Therefore consumers of the hosted platform in the cloud do not need to worry about purchasing the required hardware and software to run the application. Furthermore, PaaS offers its consumers the flexibility to deploy applications into the cloud infrastructure as well as being able to provide its consumers with the required environments for the lifecycle of system software such as testing, development, maintenance and hosting of web applications [24].

IaaS is delivered as the required infrastructure by the consumer that is hosted in the cloud, thus reducing the need for the consumer to purchase required servers, data centres and network resources [23]. With the IaaS approach consumers lose the flexibility of controlling the underlying infrastructure, but have an advantage in being able to control the operating system, processing, memory, storage and deployment of the software application. Furthermore,

consumers of IaaS only need to pay for the time duration they use the service, enabling them to deliver services faster; these services are quickly scalable and more cost-efficient [24].

A typical cloud computing system can be divided into two sections: the front end, which is the consumer, and the back end, which is the provider [22]. The front end is what the client would generally see, such as the applications screens, whereas the back end is the actual cloud system containing the various computers, servers and databases. Therefore the consumer/provider relationship plays a vital role in the architecture of cloud services, as it is the connecting thread at all levels of the stack of cloud services and also is hugely dependent on the network to be able to provide connectivity between the consumer and provider [10].

A key feature of cloud computing is its ability to abstract the complexity of providing the infrastructure for its consumers. In doing so, cloud computing has three business models that consumers can choose from for the deployment of their cloud computing services. The three basic business models are the *public cloud,* which is accessible to anyone via the internet, the *private cloud,* which is only accessible within the consumer's organization, and the *hybrid cloud,* which is a combination of both the private and public cloud models [25]. Figure 2.4 shows an overview of the three basic cloud computing deployment models.

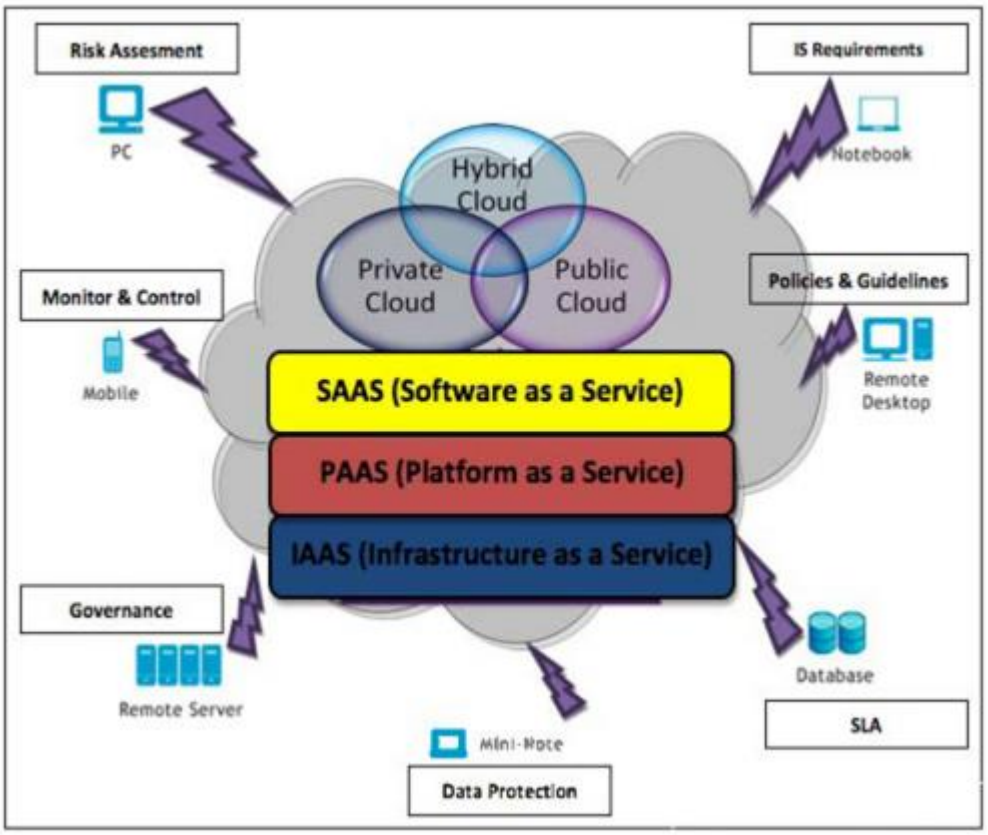


Figure 2.4: Cloud computing deployment models[25]

A *public cloud*, as its name indicates, is a cloud infrastructure made available to the public, meaning it makes resources of the cloud infrastructure available to anybody who needs to consume them. It is generally a pay-as-you-use service, with the users being required to pay only for the time duration they use the service [23]. This model reduces the operational costs on information technology (IT) expenditure, but is less secure than the other models, as all the applications and data on the public cloud are more prone to malicious attacks [24]. To access the publicly available resources consumers are required to register with a cloud provider such as Amazon Web Services (AWS) or IBM Blue Cloud.

A *private cloud* is a more secure model, as it is deployed behind the firewall of the company and the infrastructure of the cloud is not managed by a third-party provider [24]. This model enables the company to manage its own security maintenance and upgrades and also provides additional control over the deployment of applications and use of the cloud infrastructure [23]. A private cloud could be compared to an intranet, as all the applications and resources are

pooled together and made available to users within a particular organization. This model is generally adopted by companies that want to manage their own data and control their cloud infrastructure.

A *hybrid cloud* is a combination of the public and private cloud model. In this model a private cloud is used and linked to one or more external cloud services that can either be a public cloud, private cloud or community cloud [24]. Even though the different clouds are interlinked to form a hybrid cloud, they still remain unique individual entities [24]. This model has the advantage of portability of applications and data as compared to the other models. Furthermore, the hybrid model provides additional flexibility in enabling the organization to serve its needs in the private cloud and in the event the organization requires additional resources it can leverage off the public cloud to acquire the additional demand [23].

Finally, a *community cloud* is a jointly constructed cloud between organizations that share their cloud infrastructure, policies and requirements [23]. The cloud infrastructure is generally hosted by a third-party organization or by one of the organizations in the community [24].

2.2.3 Opportunities and challenges of cloud computing

Cloud computing brings about many advantages to businesses, such as low initial capital investment, reduced time for new services, cheaper maintenance and operational cost, increased utilization through virtualization and easier disaster recovery [19]. However, in addition to the widespread enthusiasm for cloud computing, several challenges, such as the reliability and privacy of data, availability of service, computation scalability and concerns about the security of confidential customer data, have become obstacles to the adoption of the cloud [20]. The current challenges and opportunities of cloud computing are well documented and have been researched in recent years [25]. Therefore, in this research study, a summary of the top ten challenges and opportunities is provided in Table 2.1, the primary focus of this study being the survivability aspect of robots in instances of cloud disconnection.

Table 2.1: Top 10 Challanges to and Opportunities for Growth of Cloud Computing [25]

	Challenge	Opportunity
1	Availability of Service (Disconnections)	Use Multiple Cloud Providers; Use Elasticity to Prevent Denial of Service (DoS)
2	Data Lock-In	Standardize APIs; Compatible SW to Enable Surge Computing
3	Data Confidentiality and Auditability	Deploy Encryption, Virtual Local Area Networks, Firewalls; Geographical Data Storage
4	Data Transfer Bottlenecks	FedExing Disks; Data Backup/Archival; Higher Switches
5	Performance Unpredictability	Improved Virtual Machine (VM) Support; Flash Memory; Gang Schedule VMs
6	Scalable Storage	Invent Scalable Store
7	Bugs in Large Distributed Systems	Invent Debugger that Relies on Distributed VMs
8	Scaling Quickly	Invent Auto-Scalar; Snapshots for Conservation
9	Reputation Fate Sharing	Offer Reputation-guarding Services such as those for email
10	Software Licensing	Pay-for-use Licenses; Bulk Use Sales

2.2.4 The importance of cloud computing to robotics

In recent years many researchers [5], [6], [8], [10] have described the use of algorithms, techniques, frameworks and approaches for networked robots to coordinate exploration, path finding, map building and autonomous navigation. Many of these approaches can be parallelized and distributed by moving some of the functionality of the robots to a backbend multiprocessor system, which can be accessed by other robots as well [10]. In doing so, robots become less dependent on their own local resources and leverage more on the potential of the cloud architecture.

Robotic tasks such as map building and vision processing are indeed computationally intensive. Many robots are built with on-board batteries that have limited power consumption and therefore processing computational intensive tasks are extremely difficult for robots, as they would have minimal energy to spare [4]. An option to overcome this barrier would be to increase the size of the on-board computational devices on the robots, but this is neither cost-efficient nor feasible [4].

Current robots are also limited by the embedded systems that they carry and are unable to cater for all possible situations [5]. By leveraging the potential of the cloud architecture robots could potentially have system updates done without requiring on-site repairs or actually stopping the mobile robotic system [4]. Replication of maps by the individual robots, which results in wastage of time, increased inefficiency and duplication of effort, can be solved by creating a single global map in the cloud infrastructure, which can be accessed by all the other robots [4]. Furthermore, the time in which robots are introduced into a new environment is substantially reduced and the potential of machine learning among the robots is also facilitated by leveraging the cloud architecture [7].

It is clearly evident that the robots of today can be even more promising by leveraging the potential of the cloud architecture and would also be able to extend their current range of capabilities. The next section therefore further elaborates on the paradigm of cloud robotics, explaining what cloud robotics is, the current challenges and opportunities that exist in cloud robotics, as well as the technological innovations taking place in this scientific field.

2.3 CLOUD ROBOTICS

Cloud robotics in a nutshell is the scientific field that applies cloud computing technology to robots. In order to gain an appreciation for this rapidly growing field of cloud robotics, a brief overview is given in section 2.3.1 of the main developments in cloud robotics to date. Section 2.3.2 addresses some of the current challenges and opportunities in cloud robotics. The prominent operating system used in cloud robotics is discussed in section 2.3.3 and section 2.3.4 introduces the cloud computing concept of service-oriented architecture required in cloud robotics.

2.3.1 What is cloud robotics?

The field of robotics research has come a long way since scientists demonstrated the world's first human-looking robot, called "Elektro", back in 1940, which stood two meters in height, weighed 120 kilograms and had the capability of talking and walking by voice commands [26]. Today a wide range of robots exists, which are much smaller and lighter and much more intelligent. These robots are required to perform labour-intensive, repetitive, tedious and sometimes dangerous tasks that humans cannot carry out [27]. Over the past decade robots have been successful in performing these tasks in the comfort of structured environments [28]. However, more recently robots have been required to perform these tasks in highly unstructured and non-deterministic environments, which would require them to have additional computing power [10].

To overcome this problem and increase the current capabilities of robots, the term "cloud robotics" was coined in 2010 by James Kuffner at Google [4]. It describes a new paradigm of robotics in which robotic services are leveraged off the cloud in order to reduce the computational processing power and data storage required locally by robots. In terms of the dissertation, this would be the definition of cloud robotics to be used throughout. Figure 2.5 represents a schematic view of the concept of cloud robotics, illustrating that cloud-enabled robots could offload computationally intensive tasks to remote servers hosted in the cloud, thereby reducing the size of the on-board computational devices on the robots [26]. Furthermore, the services offered by the robots could be extended to include capabilities such as object recognition, people recognition, navigation and knowledge sharing [26], as illustrated in Figure 2.5.

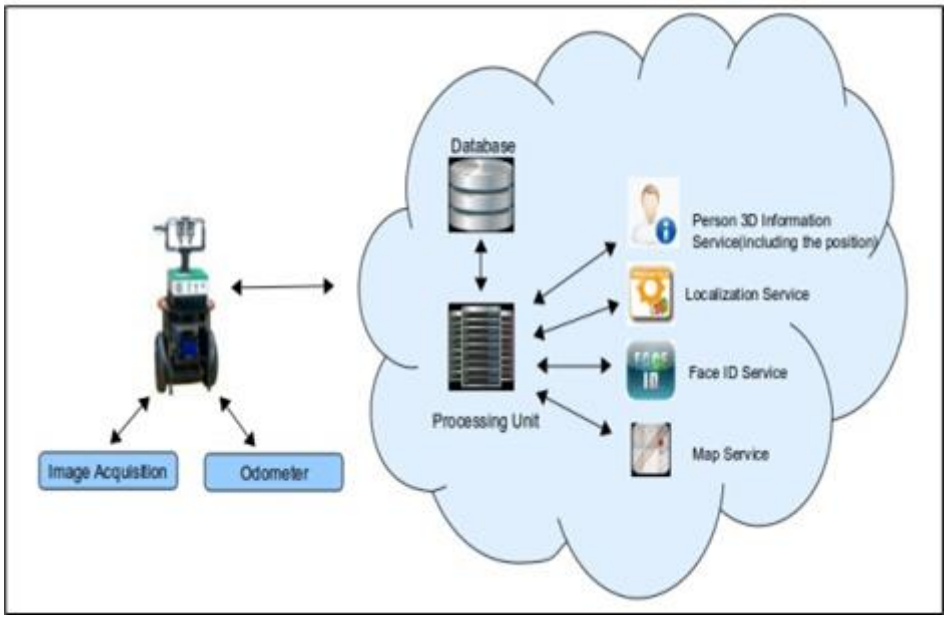


Figure 2.5: Schematic diagram of cloud robotics concept [26]

Cloud robotics removes the need for robots to carry around with them large on-board computational devices by offloading computationally intensive tasks to the cloud server. Some of these tasks include path planning [6], mapping [7], image recognition [8] and navigation. Continuous improvement in the data transfer rate [9] and growth in the wireless communications industry have also made it much more feasible to execute these robotic tasks in the cloud. The potential benefits brought about by cloud robotics have resulted in major global technology companies such as Google and IBM playing an active role in the development of cloud robotic services. Some examples of these cloud robotic projects are presented in section 2.4.4, where the laxities of cloud robotics are also explained.

The research topics relating to cloud robotics and its related technologies are indeed increasing the area of interest among the scientific community. On one side of the coin the basic concepts of cloud robotics appear to be a very attractive area of research for the future development of robotics [27]. This would result in robots of the future that require more computational power in order to execute more demanding and complex tasks. On the other side of the coin, some of these concepts are not particularly new, with some concepts, such as using web services for robotic tasks, dating back to the 1990s. However, given the advancements made in some of these concepts over the years, it is now worth giving these concepts a renewed try [27]. The

next section therefore presents some of the opportunities brought about by the recent interest in cloud robotics, together with some of the challenges that are inhibiting the adoption of cloud robotics.

2.3.2 Opportunities and challenges of cloud robotics

The cloud, together with its associated users, is a massive source of computation and data about the semantics and manipulation of objects [39]. An example of this is the several image-labelling projects that exist using machine learning in the cloud and the millions of digital photos that are uploaded by users [40]. Cloud robotics leverages the paradigm of cloud computing by enabling robotic systems to share vast of amounts of real time data and providing them with the agility and flexibility of parallel computational processing. This section is therefore aimed at further detailing the opportunities and challenges that exist in this scientific field of research.

Many current autonomous robots are limited by their existing programming, memory and onboard computation [4]. Cloud robotics now makes it possible for autonomous robots to offload this computational burden to the cloud and make use of the vast amount of data and services available in the cloud. The Google autonomous driving project is a typical example of this opportunity brought about by cloud robotics, which indexes maps and uploads images to the cloud server using the satellite and crowdsourcing [39]. The concept of robots using the cloud to share data, incorporating the elements of open source, open access and crowd sourcing, is encapsulated by a quotation from Steve Cousin of the Willow Garage project who says that, "No Robot is an island" [41]. The report in [4] highlights the following five areas of opportunity in the field of cloud robotics and automation:

- Access to a global catalogue of images, object data, maps for mobile robotic systems
- Massive parallel computation for on-demand tasks such as motion planning and statistical modelling
- Sharing of outcomes, policies and trajectories from robots
- Sharing of experiments, open code, designs, data and hardware construction from humans
- Guidance from humans, such as call centres, on error recovery and exception handling

Big Data is another paradigm that has gained substantial attention in the scientific community in recent years and refers to data sets that are beyond the capabilities of existing relational databases [4]. These large data sets provide an opportunity to robotic systems to leverage the ever-growing library of images, maps and other forms of data relevant to robots. For example, the Columbia Grasp dataset and MIT kit object dataset have become readily available and are used in evaluating grasping algorithms [41].

Cloud robotics allows robots to share data and information from different tasks in a variety of environments. The type of information that could potentially be shared by the robot includes environment conditions, control policies, sensor information, data on performance and outcomes from a particular task [42]. Robots can share this newly gained information, which would mean that they do not have to go through the difficult learning process again, thereby enabling machine learning [42]. One such project is the RoboEarth cloud engine, which acts as acknowledge repository for robots for easy access and integration [12].

The cloud also enables humans to share code, data and hardware and software designs. Cloud robotics has the opportunity to leverage the success of open source software [4]. This opportunity is epitomized by ROS, which has become the open-source meta-operating system for robotics and provides developers with libraries and tools to create robot applications [29]. In addition, much other software and many libraries for developing robotic applications have become open-source, enabling students and researchers to set up robot applications quickly and share the results of the robot experiments.

The navigation of robots involves the execution of several robotic tasks, such as path planning, building a map and localization, all of which are extremely computational and data-intensive. Current robot implementations are limited by onboard computational resources, making the robot navigation unreliable [23]. Cloud robotics is a very promising solution for future cloud-enabled robot navigation systems, as it would avoid these challenges. Through the cloud, commercially available maps such as Google maps can be accessed to provide reliable, on-demand and agile navigation solutions for robotic applications [23].

Though many opportunities exist in the field of cloud robotics, there are still challenges that need to be overcome before realising the promised potential of cloud robotics. Figure 2.6 highlights the taxonomy of challenges currently faced in the field of cloud robotics and breaks it down to six functional areas, namely operational, end user, service level, privacy and security, context awareness and data management [38]. The focus for this dissertation is on the functional area of operational challenges faced in cloud robotics. In addition to this topic, other challenges faced in cloud robotics that form part of the taxonomy of challenges illustrated in Figure 2.6 are also briefly discussed.

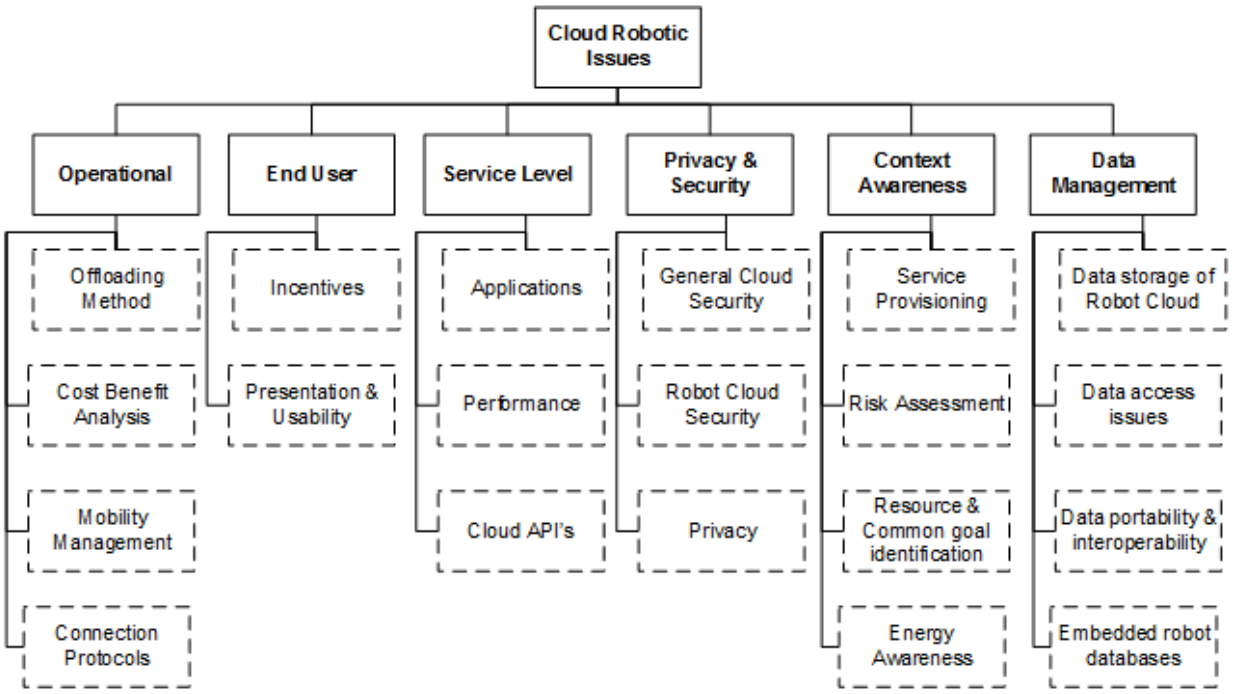


Figure 2.6: A taxanomy of challenges in cloud robotics [38]

The main advantage that cloud robotics brings is the ability to offload computationally intensive tasks to the cloud server for processing. However, the decision on which robotic tasks to offload to the cloud server requires an optimization framework that will take into consideration many factors, such as bandwidth, size of data, latency, energy consumption and QoS [23]. In general the task optimization framework should consider at least three offloading strategies, namely standalone robot execution, standalone cloud execution and a collaborative strategy, to execute the tasks among the networked robots. Another challenge with computational offloading is the ability to allocate virtual machines spread across multiple data

centres optimally to execute robotic tasks offloaded into the cloud and manage virtual machine migrations [10].

The decision on whether to execute the task in the cloud or on the robot is dependent on the delay sensitivity of a particular task [19]. A challenge often faced in cloud robotics is the inherent packet delivery failures and network failures often found with wireless communication systems [10]. Cloud-enabled multi-robotic systems executing tasks in heterogeneous environments often have to rely on wireless communication systems and failure to deliver the data packet to the cloud server to execute a particular robotic could potentially have catastrophic ramifications.

End user challenges relate to issues that are directly involved with the users of the cloud robotic systems and include topics such as cost, participation and interoperability [38]. Although there has been a lack of focus on some of the end user challenges in cloud robotics [23], the presentation of user interfaces does appear to be a valid challenge. Cloud robotic systems could span a vast number of different platforms and having to design and develop separate user interfaces for each and every type of robotic system could be unrealistic and inconvenient [38].

A major challenge faced in both cloud computing and cloud robotics today is the issue of security and trust [10]. At a very minimum in cloud robotics the virtual machine that is executing the robotic task should be trustworthy. If this is not the case, then there could be instances of sabotage through which a malicious virtual machine prevents a task from being executed without the robot being aware of the damage [10]. The ability to authenticate, account and audit (AAA) for a robot requesting a particular task to be executed in the cloud server is imperative. This is even more important when executing a task or retrieving confidential data in a public cloud to ensure that the task does not contain any malicious or hidden code [20]. These challenges are inherent to cloud computing, but specific cloud robotic security and threat issues, such as robot botnets, battery exhaustion attacks and robot targeted attacks, need be considered as well [38].

Based on the related literature, the challenges of determining an optimal task execution policy, efficient cloud robotic energy execution models and QoS in instances of communication delays have not been sufficiently covered. Therefore these are the gaps that the SCMR framework addresses and that are extensively detailed in the experimental results section of the dissertation.

2.3.3 Robot Operating System

ROS is an open-source, meta-operating system for robots that abstracts most of the implementation-dependent logic inherent in realizing a robot's system, which incorporates sensing, planning, navigation and control [29]. ROS was initially designed as part of the STAIR project at Stanford University to meet a specific set of challenges encountered by service robots and is based on the following five philosophical goals [30]:

- Peer-to-peer
- Tools-based
- Multi-lingual
- Thin
- Free and open-Source.

ROS itself fundamentally comprises nodes, topics, messages and services. Nodes are the processes that perform the computation and a robot system is typically comprised of many nodes [30]. The communication between the different nodes in a robot system is done through messages [30], which are defined by a strictly typed data structure (integer, floating point, Boolean, etc.). Messages in ROS can also be composed of other messages, as well as arrays of other messages. A topic is generally a string such as "map" or "video" and a node sends a message by publishing it to a particular topic. A node that is interested in a certain type of data will subscribe to a particular topic, which is known as the publish/subscribe model [29]. A single node may publish/subscribe to multiple topics and there may be multiple concurrent subscribers/publishers of a single topic [30]. Finally, for synchronous transactions, ROS makes use of a service that is similar in nature to web services, is defined by a sting name and strictly typed response and request messages that are passed over the UCP/IP or UDP protocol [30]. The topic-based publish/subscribe model in ROS is a flexible communications paradigm

for asynchronous transactions, but not ideal for synchronous messages. Instead, the ROS services provide more flexibility for synchronous services by ensuring that only one node can advertise a service of a particular name [29]. An example of such a service is "Classify_Image". The ROS master node keeps track of all the topics and services in addition to the registration of new nodes being introduced into the ROS environment, which are all configurable in an XML file. Figure 2.7 is an illustrative output of ROS nodes, topics, messages and services and shows the topic-based publish/subscribe model.

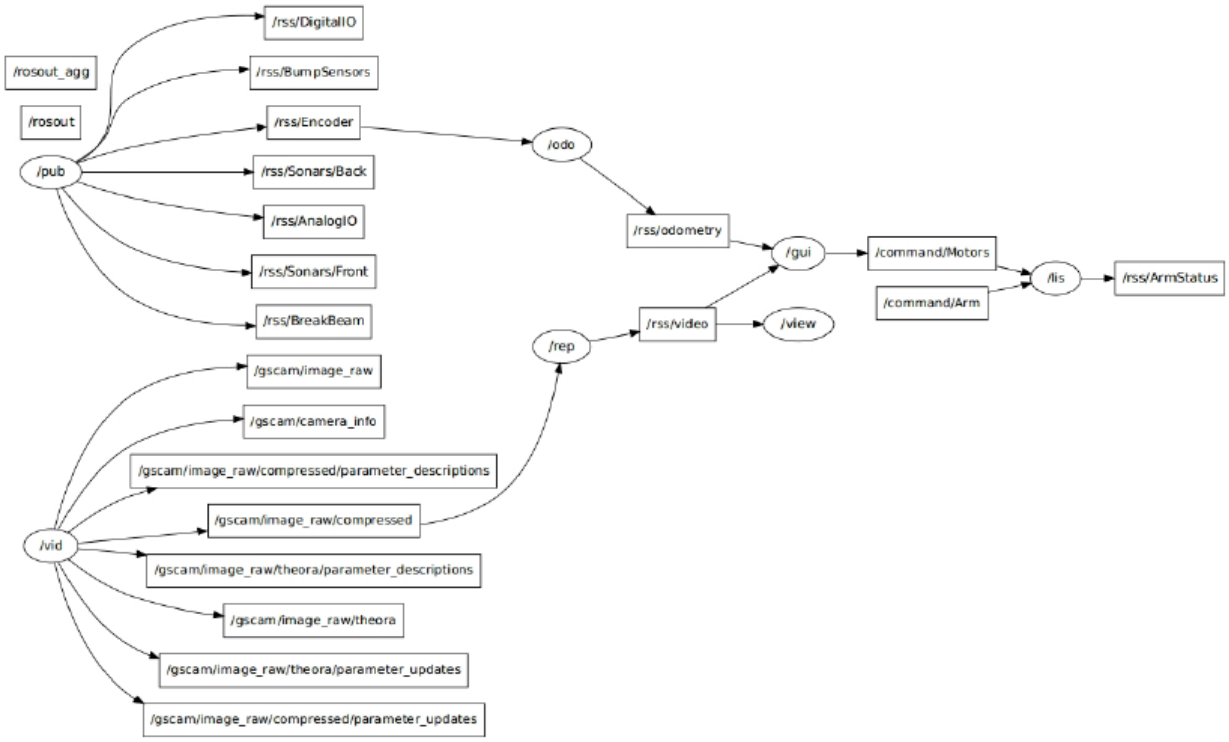


Figure 2.7: Example output of ROS nodes and topics [29]

Several development environments exist today for the development of robotic functions, such as Mobile Robotics Development Studio (MRDS), and ROS, RT-Middleware (RTM) [31]. It is difficult to indicate which the best framework for all robotic software is because the field of robotics is too wide-ranging for a single solution. However, one of the attractive features of ROS compared to the other robotic environments is that it provides a loosely distributed platform, which enables a modular communication mechanism for message exchange [32]. Furthermore, ROS modules developed for specific robotic functions are often published on the web and made available for reuse [31]. Table 2.2 provides a summary of the robotic

environments together with their main features, which also indicates the reasons for ROS being one of the most widely used operating system for robots.

Table 2.2: Characteristics of Robotic Development Environments [32]

	Player/Stage	Gazebo	ROS	Simbad	CARMEN	USARSim	MRDS	MissionLab
OS	Linux, Mac, Win.	Linux	Linux, Mac, Win.	Linux, Mac, Win.	Linux	Linux Win.	Win.	Linux
Simulator Type	2-D	3-D	2-D, 3-D	3-D	2-D	3-D	3-D	3-D
Programming Language	Player(any) Stage(C, C++, Python, Java)	C, C++, Python, Java	C++, Python, Octave, LISP, Java, Lua	Java	C, Java	C, C++, Java	VPL, C#, Visual Basic, JScript, Iron-Python	VPL
Documentation	Low-Level	Low-Level	High-level	High-Level	Low-Level	High-Level	High-Level	High-Level
Tutorial	Yes	Yes	Yes	Limited	Limited	Limited	Yes	Limited
Portability	Yes	Yes	Yes	Limited	Yes	Yes (using Player)	Yes	Yes
Sensors	odometry, range	odometry, range, camera	odometry, camera, range	vision, range, contact	odometry, range, GPS	odometry, range, camera, touch	odometry, range, camera	odometry, range
Debugging/ Logging	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
Graphical User Interface	No	No	No	No	No	Yes	Yes	Yes

2.3.4 Service Oriented Architecture in Cloud Robotics

Service-Oriented Architecture (SOA) is a software system that consists of loosely coupled services that communicate with one another through standard interfaces using a standard message exchanging protocol [33]. It is an architectural model that is used to increase efficiency, agility, and productivity by building the solution logic on services that become the primary means to engage with an enterprise, which supports the underlying goals associated with SOA [26]. SOA consists of a combination of technologies, application programming interfaces (APIs), infrastructure and different products, which are in most instances implemented using web services.

The three main components of SOA are a service provider, a service consumer and a service registry, as depicted in Figure 2.8. The service providers publish their services on the service registry and the service consumer searches for the service on the service registry and then invokes the service. These components facilitate a high level of service abstraction while still enabling the service provider to decouple the service being provided from any vendor-specific proprietary implementation details [34]. Furthermore, these components also facilitate the key principals of SOA, such as service reusability, service loosely coupling, service contracts and service composability, which are desirable characteristics of a SOA implementation [33].

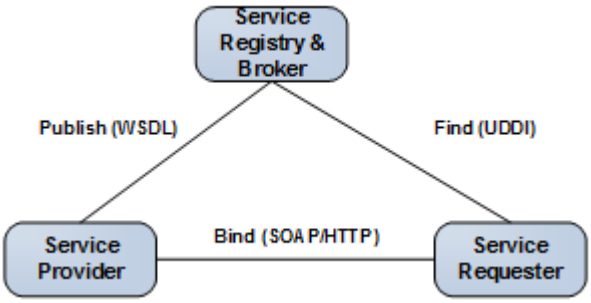


Figure 2.8: Overview of SOA architecture [34]

A number of software development enterprises, such as IBM, SAP, Oracle and Microsoft, have adopted and applied SOA to produce and deliver service-based applications [33]. These global technology companies have implemented SOA in order to promote reusability, scalability, interoperability and lowering of software production costs [34]. However, SOA and cloud computing have thus far primarily been limited to e-commerce and enterprise systems [33]. The technologies of SOA that form the foundation of cloud computing are now also being introduced into the field of robotics and embedded systems [34]. MRDS was one of the first robotic development software applications to incorporate the concepts of SOA by defining a mapping layer that decoupled the virtual services layer from the physical robot layer and used REST services instead of SOAP services to improve real time communication with the robot [32].

In Japan a team of researchers leveraged SOA to control a swarm of robots [35] dynamically, while another team of researchers was able to implement SOA successfully in a virtual remote laboratory environment using Lego NXT to perform physical experiment installations [36].

Encapsulating the paradigm of both SOA and cloud computing, researchers from the Arizona State University were able to implement the concept of robot-as-a-service (RaaS), which enforces the design and implementation of a robot as a SOA unit [33]. In order to achieve this the RaaS team had to implement services for performing robotic functionality, broker services for publishing and discovery robotic services, as well as services for direct client access [33]. Finally in [37], the authors describe the use of semantic web services in robotics to facilitate corporation between robots for executing joint tasks. This SOA implementation enabled the isolated robots to communicate among themselves semantically, allowing atomic robotic operations to be executed and readjustments done in real time [37].

The work done in these projects clearly indicates that much effort has been made in applying the SOA principles to the field of robotics and automation, which highlights the growing importance of SOA in robotics. It is important to note that this section is used to highlight the impact that SOA has on the field of robotics and automation and that the detailed implementation of the SOA principles used for the SCMR framework is extensively discussed in section 3 of the dissertation. The advancements made in both SOA and cloud robotics has also paved the way for the realization of different cloud robotic models. The next section further details the different cloud robotic models that are currently available.

2.4 CLOUD ROBOTIC MODELS

In this section the different cloud robotic models available are discussed, together with their advantages and disadvantages. The existing cloud robotic models are the peer-based, proxy-based and clone-based robotic models.

2.4.1 Peer-based cloud robotic models

The peer-based model is illustrated in Figure 2.9, in which each robot and each virtual machine (VM) in the cloud is regarded as a computing unit. The main advantage of this cloud robotic model is that the robots and VMs form a fully distributed network that allows for certain robotic tasks to be divided into smaller tasks that can be executed in a subset on the distributed network [7]. The main disadvantage of the peer-based cloud model is the decentralizing of the network created by the one-to-one mapping of robot and VM in the cloud, which makes it difficult to

administer. Another disadvantage is the low number of network connections between the individual robots and the physical cloud infrastructure. The DaVinCi project [8] and the Robot-Cloud project [4], which promotes the creation of low-costs robots using cloud robotics, are examples of peer-based cloud robotic models that have the individual robots and the VM in the cloud computing as a single unit.

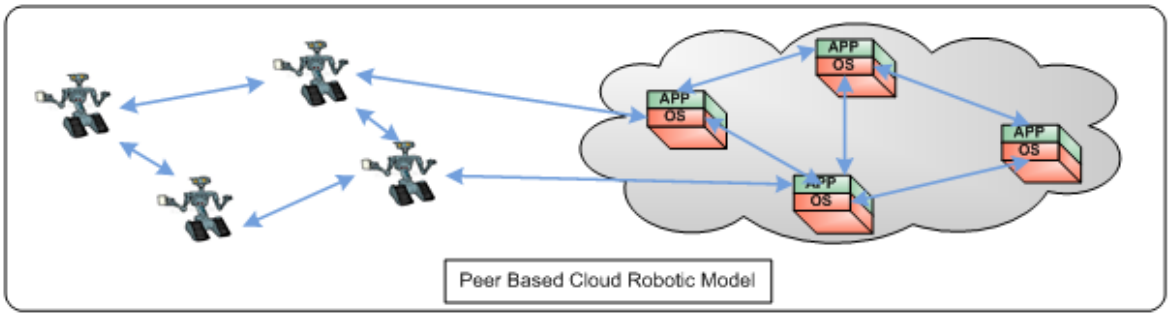


Figure 2.9: Peer-based cloud robotic model

2.4.2 Proxy-based cloud robotic models

The proxy-based model consists of a multi-robotic team that has one robot functioning as the leader of the team. All communication between the individual robots in the MRS and the robot leader acts as a proxy for the physical cloud infrastructure, thus bridging the communication between the robotic network and the cloud. The main advantage of the proxy-based robotic model is its increased interoperability as opposed to the other cloud robotic models. Interoperability in this sense refers to the additional complexities brought about by maintaining a cloud robotics infrastructure [7]. The main disadvantage of the proxy-based cloud robotic model is the number of connections between the robots and the cloud. In this model only one network connection can be established at a time and in instances where the group leader is unavailable, communication to the cloud will not be possible. Figure 2.10 is an illustration of the proxy-based cloud robotic model showing the robot leader as the proxy to the cloud server.

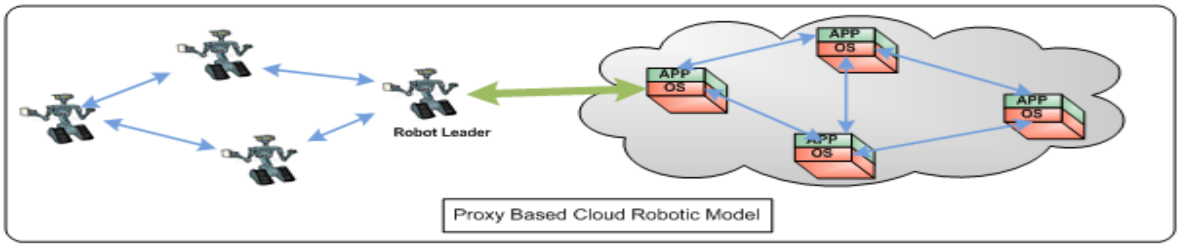


Figure 2.10: Proxy-based cloud robotic model

2.4.3 Clone-based cloud robotic models

In the clone-based model each individual robot in the MRS is required to map to a system-level clone in the physical cloud infrastructure [7], as illustrated in Figure 2.11. The main advantage of this model is that a particular robotic task can be executed either in the cloud clone or locally on the individual robots. The main disadvantage of the clone-based cloud robotic model is the increased complexity brought about by maintaining and operating the cloud robotic infrastructure, which reduces its interoperability. Another disadvantage of the model is the increased VM migrations required in order to support robot mobility in this particular model [7]. An example of the clone-based robotic model is the Rapyuta project [5], which has each individual robot connecting to a system level clone in the cloud, enabling them to offload heavy computation to the cloud server.

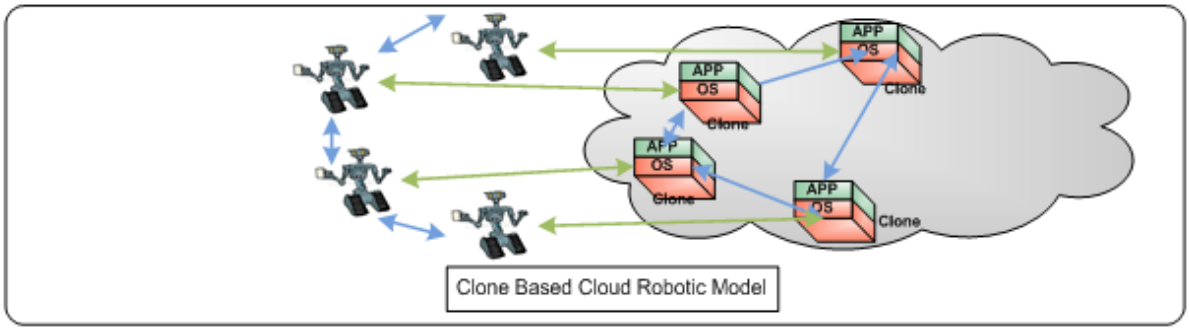


Figure 2.11: Clone-based cloud robotic model

2.4.4 Cloud robotic model proposed for the SCMR framework

In determining an elastic computing model to use for a particular cloud robotic framework, the three main factors that need to be considered are network conditions, application requirements and resource availability [7]. This study makes use of a hybrid model, leveraging the benefits of both the proxy-based and clone-based cloud robotic models. It should be noted that the experimental section of the dissertation provides a detailed comparative evaluation of the different cloud robotic models.

The proposed SCMR framework consists of two tiers: a robot-to-cloud (R2C) tier and a robot-to-robot (R2R) tier, in an attempt to overcome some of the challenges faced by the current cloud robotic models. This model differentiates itself from existing models in that it leverages off two cloud infrastructures. The first is the physical R2C infrastructure and the second the

virtual ad hoc cloud infrastructure created by R2R communication, which forms a computing unit.

The proposed cloud robotic model for R2C communication is illustrated in Figure 2.12, which has the robot leader corresponding to a system-level clone in the cloud. A task can therefore be executed using the robot leader in situations where individual robots have no access to the network or in its clone in the cloud. In the physical cloud infrastructure VMs may be created to form a distributed computing environment, which would enable real time responses and reliable data to be provided to the individual robots. The other individual robots can also access the VMs created, as well as the system level clone.

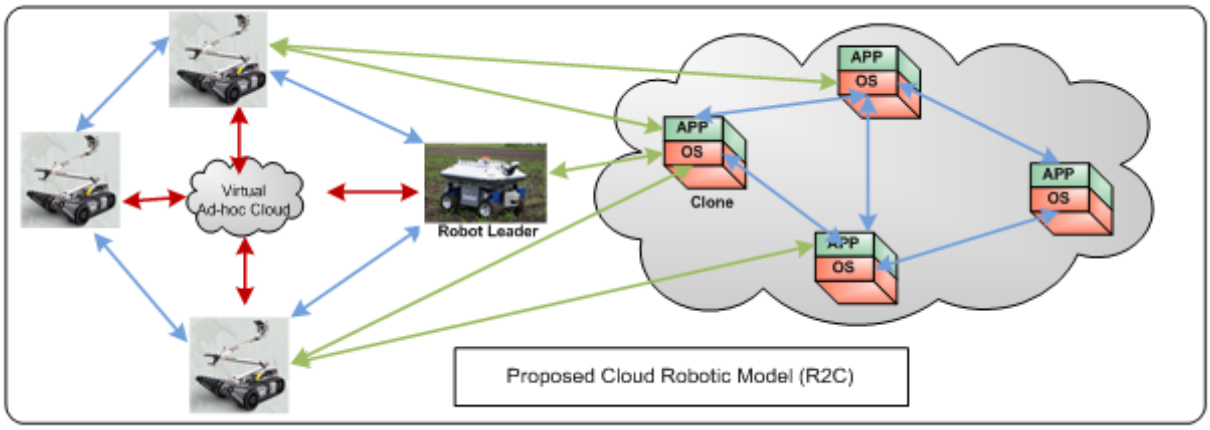


Figure 2.12: Robot-to-cloud cloud robotic model

The R2R cloud robotic model is illustrated in Figure 2.13, showing the creation of the virtual ad hoc cloud in instances of cloud disconnection or unavailability of the cloud server. In this model a robotic task is executed by offloading the task to the robot leader. A virtual ad-hoc cloud (computing unit) is created by the R2R communication and resources are shared among the robots in this virtual cloud environment.

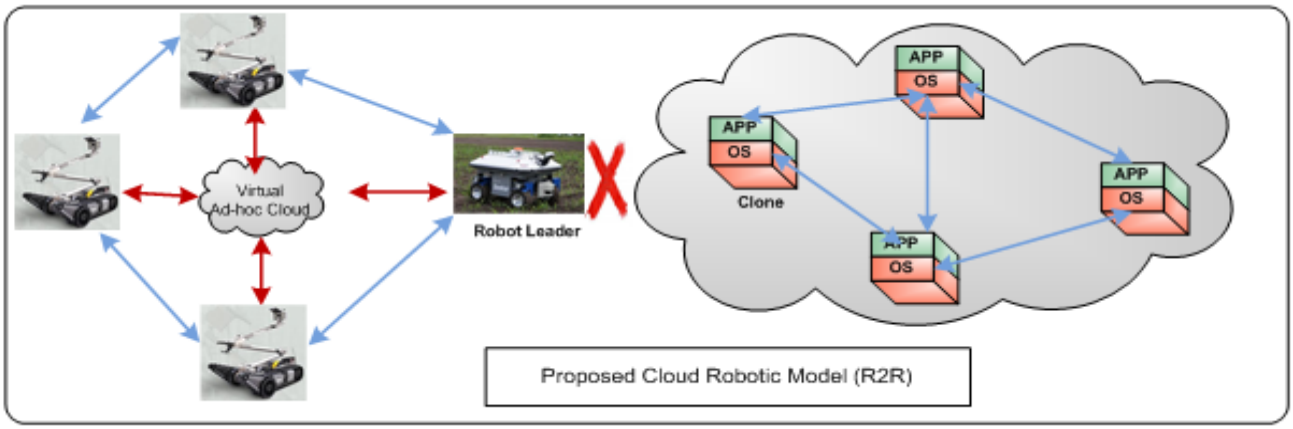


Figure 2.13: Robot-to-robot cloud robotic model

2.4.5 Survey on laxities of cloud robotics models

During the past few years, scientists and researchers all over the world have started attempting to build cloud computing frameworks to support both individual robots and multi-robots. Most notably, researchers at the Institute of Dynamic Systems and Control in Switzerland have created an open-source project called RoboEarth in an attempt to build a web community for robots to autonomously share information and object models [12]. The RoboEarth cloud engine is a PaaS that allows other users to run their robotic services on the RoboEarth cloud [7]. A team of researchers in Singapore have created the DAvinCi project [10], which was based on the ROS [10] messaging framework and used the ROS master node to gather data into the Apache hadoop cluster [13] and illustrated the benefits of cloud computing by parallelizing the FastSLAM algorithm [14].

The Ubiquitous Network Robot Platform (UNR-PF) [8] is an ongoing project that primarily focuses on using the cloud as a mechanism to creating a network between robots, sensors and mobile devices. Rosbridge [10] is another open-source project, which focuses on the external communication between a robot or a multi-robot team and a single ROS environment in the cloud. The emergence of smartphones has enabled Google developers to create Android-powered robots that allow users to control them using their smartphones on platforms such as Lego Mindstorm and iRobot [39]. The use of cloud robotic models has also been extended to the medical industry, which has robots accessing the cloud infrastructure to gather data to assist children in a hospital in Italy with face detection and speech recognition [15].

All projects mentioned above contribute significantly to cloud robotics. These projects illustrate the potential benefits that can be achieved in cloud robotic models by offloading data-intensive and computationally intensive tasks into the cloud, enabling researchers to realize the elevated goal of cloud robotics. However, there are still some inhibitors, barriers and challenges that require attention. According to a report by [16], the following are some of the current challenges faced with cloud robot models:

- Many cloud robotic models make the assumption that resources in the cloud are unlimited, which is not the case.
- Cloud robotic services are actually limited. These services could be dependent on the network bandwidth for offloading data or the number of parallel connections allowed to the cloud.
- The number of individual cloud robotic hosts in an MRS is limited.
- Maximizing the effectiveness of available resources in the cloud with on-demand robotic requests remains a further challenge.
- Compatibility of data retrieval is an ongoing challenge brought about by different robots accessing the cloud to execute tasks.

In order to understand the rationale for this work better. an empirical survey was conducted by sampling articles on cloud robotic models from the Science Direct and Institute of Electrical and Electronic Engineers (IEEE) databases. The following keywords were used in the selection sampling that took place during May 2013: Cloud Robotic Models, Cloud Robotics, Cloud Disconnections, Multi Robotics, cloud computing, quality of service, drivability, framework, networked robots, ROS, energy models and laxities. Figure 2.14(a) shows the findings from the empirical survey. Of the 40 accredited journals sampled, 75% indicated that the current cloud robotic models have room for improvement by illustrating the shortcomings of current cloud robotic models and indicating future research work that can be done in the field of cloud robotics; 25% were undecided and did not stated whether or not there is room for improvement in the current cloud robotic models.

A subset of the accredited journals, which indicated that cloud robotic models do require improvement, was then taken to determine if they highlighted the issue of cloud disconnection. This is depicted in Figure 2.14(b). Interestingly, 83% of the journals did not highlight the issue

of cloud robotic disconnections and assumed and that a connection between the cloud and the robot was always available. In heterogeneous environments multi-team robots can often find themselves in locations that do not have access to the cloud infrastructure or the connection between robot and cloud is limited by network bandwidth.

Furthermore, communication protocols between R2C and R2R in an MRS could lead to high latency and performance degradation on the network. Only 17% of the journals mention the issue of cloud disconnections, but do not completely address the problem for multi-robots executing tasks in heterogeneous environments. Results from this survey indicate that there are some laxities in the current cloud robotic models and highlight a need for more work to be conducted on cloud multi-robots in heterogeneous environments and the issue of cloud robotic disconnections.

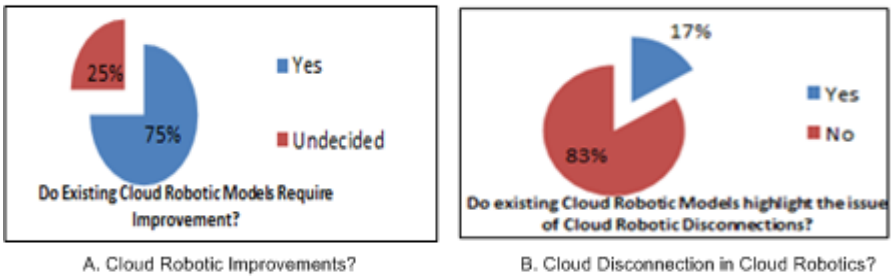


Figure 2.14: Do existing cloud robotic models require improvement and do they highlight the issue of cloud disconnection?

2.5 CLOUD DISCONNECTIONS

The *cloud* has been used as a metaphor for the internet since the inception of the World Wide Web (WWW) in the early 1990s, where software, hardware, resources and information are shared [25]. One of the major challenges in the paradigm of cloud robotics which has been inherited from cloud computing is the connection to the cloud server [23]. The benefit of leveraging the cloud for computational processing is only a benefit if access to the cloud is open and accessible. Therefore this section aims to provide an overview of the current cloud disconnection in cloud robotic models, as well as the technologies used in cloud robotic models to connect to the cloud server. The consequences of cloud disconnections are also discussed, emphasizing the importance of ensuring that a robot executes its tasks even in the event of cloud disconnection, which turns out to be the fundamental concept of this dissertation.

2.5.1 Cloud disconnection in cloud robotics

In a cloud robotic model the individual robots in an MRS can only communicate with one another if they are within communication range of one another and can only communicate with the physical cloud infrastructure if the individual robots are within range of the cloud access points. Before proceeding to describe disconnections in cloud robotic models, it is imperative to first understand the main connection protocols for communication in a cloud robotic model. Current research indicates that the main connection protocols for communication in cloud robotics are Wi-Fi, Bluetooth and 3G (third generation network for telecommunications), the most popular being the use of Wi-Fi for several reasons, including lower energy consumption and longer radius ranges [38]. An overview of the connection protocols used, together with their pros and cons, is presented in Table 2.3.

Table 2.3: Comparison of Connection Protocols Used in Cloud Robotics [38]

Protocol	Frameworks	Pros	Cons
Wi-Fi	Spectra, Chroma, Cuckoo, Cloudlets, MobiCloud, CloneCloud, Virtual cloud	Superior communication range and lower energy consumption	Security concerns and interoperability issues between the different Wi-Fi brands
3G	Cuckoo, MAUI, CloneCloud	Near-universal coverage	High energy consumption and increased latency resulting in poor performance
Bluetooth	Cuckoo, MMPI	Lower power use and increased availability	Limited communication range

In most Cloud Robotic models Wi-Fi is used to create a gateway in which the cloud robotic services are linked with the robots [10]. This enables the dynamic creation of a network of robots in an unstructured manner, enabling individual robots to enter and leave the network as and when required. The unavailability of cloud robotic services to the individual robots in an

MRS is brought about by the disconnection to the connection protocols for communication in a cloud robotic model. This means that the individual robots in the MRS are unable to receive and transmit data to the physical cloud, resulting in the robots being unable to complete certain tasks. Cloud disconnection can also occur as robots enter or leave a network, run out of battery power and loss of network signal, as well as in the case of hardware failure. This could prove fatal in heterogeneous environments that require multi-robotic teams to assist with the saving of human lives. Disconnections in a cloud robotic model can be brought about by robot mobility as robots enter and leave a particular network, unpredictable failures, unforeseen environment obstructions, loss of battery power in the robots, network failure or hardware failure [38].

2.6 ROBOTIC VISION

The visual capabilities of a multi-robotic team are extremely important in order to enable navigation to a heterogeneous environment successfully. The autonomous navigation of robots is typically divided into three components, which are the perception module, the mechanical module and the control module, as illustrated in Figure 2.15. The perception module comprises sensors with high-quality visual capabilities that are important for autonomous navigation, as these are used to capture the robots' current position (*x*, *y*, *z*) and determine which region is safe for autonomous navigation [43]. The mechanical module and control module are out of the scope of this research study, as illustrated in Figure 22.

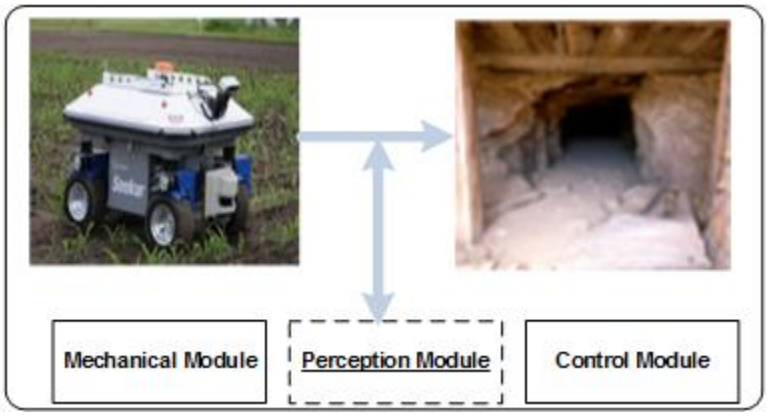


Figure 2.15: Autonomous robot navigation modules

Recent advancements in robotics and automation have prompted the research community around the world to focus on developing new methods for autonomous navigation, while also improving on the existing methods [44], [45]. Most of the research on autonomous navigation has focused on improving the robots' vision in different environments; however, very little attention has been paid to autonomous navigation in underground terrains such as mines [46]. A possible reason for this lack of research could be attributed to the roughness, technological and environmental constraints of underground terrains compared to structured and unstructured surface terrains [44]. The past decade has seen an increase in the study of autonomous navigation in heterogeneous environments [44], [46] and the challenge of finding a robust algorithm applicable in different terrains remains a challenge. Several segmentation algorithms, such as the k-means, entropy and edge detection, currently exist for autonomous navigation and much research has been conducted on the benefits of these algorithms to assist with autonomous navigation [46]. The dissertation adopts a statistical approach based on the SRM model in view of the interesting statistical properties, such as separability and homogeneity, of the SRM model [47]. The next section elaborates on the topic of SRM.

2.6.1 Statistical Region Merging

SRM is a robust algorithm introduced by Nock and Nielson in 2004 for segmenting an image into similar regions of colour or intensity. It combined the relevant ideas of computational learning theory with machine learning theory [13]. It is based on a model of image generation that groups together an interference problem and is attractive to use, as it removes the assumptions of data distribution. Furthermore, the SRM model is increasingly being adopted as a growing/merging technique because of its ability to handle significant noise corrosion and occlusion with the sort function, and its agility in performing multi-scale segmentation [48]. SRM does, however, have the disadvantages of over-merging and sensitivity to the sort function [13]. Over the years SRM has been successfully used in the fields of medical science and computer science for image processing [48] and is also used in some of the experiments for this dissertation.

In SRM a region is a group of connected pixels with similar homogeneity and feature properties, while image segmentation refers to the process of partitioning an image into

multiple regions [13]. Segmentation provides the ability to interpret parts of an image by transforming the pixels of an image into interpretable regions and objects [13]. The part of interest in an image becomes the object, while the rest of the image becomes the background. For an image I and homogeneity predicate H_p the segmentation of an observed image I is a partition K of I into a set of G regions, $R_1, R_2, \ldots . R_G$ such that the following conditions are true [48]:

- $H_p\left(R_g\right) = true\ \forall g$
- $H_p\left(R_g\ \cup\ R_h\right) = false\ \forall\ adjacent\left(R_g, R_h\right)$
- $\bigcup_{g=1}^{G} R_g = I\ with\ g\ \neq h\ and\ R_g\ \cap\ R_h = \emptyset.$

Segmentation is modelled as an inference problem in SRM by conducting a statistical test bed on a merging predict and has been adopted in remote image sensing and medical imaging [48]. In [49] SRM is used in an attempt to analyse skin cancer by detecting the borders of an infected image using skin imaging technology. The technique of region merging produces regions of homogeneity by iteratively combining smaller regions. In SRM the regions are merged using a merge-find set defined in [13] as follows:

- Find: Determines if two elements (pixels) are in the same subset.
- Union: Merges two subsets (sub-region) into a single subset (region) based on some criteria.

A downfall of SRM, as indicated, is over-merging of an image, which results in the observed image containing multiple true regions. However, research in [49] indicates that the error brought about by over-merging is minimal, as the SRM algorithm manages an accuracy close to optimum. The SRM algorithm is dependent on the interaction of the estimated cluster, Q, and the merging predict, where the parameter Q is used to modify the statistical complexity of a scene [13]. The merging predict in Equation 2.1 was put forward by [13] based on the independent merged difference inequality:

$$P(R, R') = \begin{cases} true & if\ \forall c\ \epsilon\ (R, G, B), |\ \bar{R}'_c - \ \bar{R}_c| \ \leq T \\ false\ otherwise \end{cases} \tag{2.1}$$

where R and R' are two regions of I, $\bar{R}_c$ represents the observed average for channel c in region R,

$$T = \left| \sqrt{k^2(R) + k^2(R')} \right| \tag{2.2}$$

and $R_{|R|}$ represents the set of pixels with $|R|$ pixels.

$$k(R) = \sqrt[g]{\frac{1}{2Q\ |R|}\ \ln(6|I|^2 R_{|R|})} \tag{2.3}$$

Let I be an observed image with pixels $|I|$ that each contains three (R, G, B) values belonging to the set {0, 1, • • •, g - 1 pixels} where g = 256. The observed image I' is created by sampling each pixel of the RGB channels. Each colour level of each pixel in the observed image I' takes a value in the set Q with values of [0, $\frac{g}{Q}$]. The parameter Q quantifies the statistical complexity of I', model generality, and the statistical solidity of the task [13]. Therefore, the optimal statistical regions in I' satisfy both the property of homogeneity and separability [48]. Finally the sort function, where P'_a, P_a represent pixel values of a pair of adjacent pixels of the colour channel, is presented in Equation 2.4

$$f(P, P') = \max_{a \epsilon R,G,B} |P_a - P'_a| \tag{2.4}$$

2.6.2 Drivable region detection

The successful completion of a robotic task is dependent on the ability of a robot to navigate successfully to the desired location where a particular task is being performed. Furthermore, it is important for the robot to be able to navigate to the desired location autonomously with little or no human intervention. Drivable road detection is still attracting much interest from the scientific community, although researchers have been studying it for the past decade [48]. This is evident from high-profile competitions, such as RoboCup and the Defence Advanced Research Projects Agency (DARPA) challenges [49] that are used to develop robots capable of assisting humans in responding to natural and man-made disasters.

An autonomous robot should be able to identify drivable road regions and drive along them in drivable weather conditions with the ability to minimize dangers brought about by the environment and maximize speed [50]. However, autonomous robotic navigation has long been stressed as a major challenge in the field of robotics. This has resulted in many researchers looking at several diverse mechanisms to overcome this challenge [51]. Researchers from the Applications Research Lab at Intec Corporation [52] used a hierarchical Bayesian network for image segmentation and detecting off-road drivable region detection for autonomous

navigation. Using the Bayesian network, the researchers were able to account successfully for dependencies between neighbouring pixels in the dimensions of the images. In [53] researchers developed a machine vision algorithm that generates the region of interest (navigable area for robots) from a dynamic threshold method. In an alternative approach in [54], researchers investigated the possibility of using statistical feature analysis (SFA) combined with a breadth-first search algorithm to segment different intensity regions of similarity in a road image. They used a metric derived from the Bhattacharyya distance and voting scores to derive the drivable road regions.

The research done in the area of drivable road detection illustrates that the ability of a robot to recognize objects in a heterogeneous environment is paramount. In general, one of the first steps in road region detection is identifying colour and image filtering, followed by edge detection [50]. Once the edges of an image have been detected subsequent techniques, such as region segmentation and removal of salient pixels, can be used to produce the drivable road image. The next few sections therefore further discuss the topics of colour spaces, edge detection and image filtering.

2.6.3 Colour spaces

In terms of robotic vision, a colour model can be described as an abstract mathematical model representing the different colours as ordered lists of numbers [50]. These lists typically consist of three or four colour models such as RBG (red, green, and blue) or CMYK (cyan, magenta, yellow, and key). Colour spaces are a result of associating one of these models to interpret an image and the values associated with each colour generally range between 0-255 (8-bits), corresponding with different colour intensity [50].

Colour spaces are important in helping a robotic system understand the colour of an image as they produce information about object location and this is required for processing purposes. Generally, an image captured by a robotic system is based on three primary colours, red, green and blue, which is known as the RGB model [51]. However, several other colour models exist today that represent colour spaces such HSV (hue, saturation, value), also known as HSB (hue,

saturation, brightness), HSL (hue, saturation, lightness/luminance) and CMY (cyan, magenta, yellow).

2.6.3.1 Generic colour models

The primary colour model as indicated is the RBG colour model, which uses additive colour mixing to describe the kind of light required to produce a given colour. In the RGB model individual colours are stored as a RBG triplet *(R, G, B)* ranging between 0 and 255 [50]. For example, the, *(255, 255, 255)* triplet will result in a white image, while the triplet *(0, 0, 0)* will result in an image that is completely black. Therefore the combination of these colours typically forms the basis for the secondary colours. The RGB model in certain instances has also been extended to include the additional alpha channel, thereby producing the RGBA colour model, which is used to indicate transparency [50].

CMYK is a colour model based on subtractive colour mixing predominantly used in the printing process of images [50]. The desired colour in the CMYK model is produced by determining the inks required to reflect the light from the substrate. As a base the white substrate is used and ink is subtracted from the white base to produce the required image. The different inks stored in the CMYK model are often cyan, magenta, yellow and black.

HSV is a model commonly used by artists and is a transformation of the RGB colour space, as its components are relative to the RGB colour space [52]. This model represents colours in their more natural form of hue and saturation rather than in terms of additive or subtractive colour components. HSL is an extension of the HSV model, which has the lightness component of the model replacing the brightness component of the HSV model.

In this dissertation the RGB colour model is used because the triplet values of the RGB colour channels are greatly correlated with one another. Often researchers use the RGB model to transform to one of the other colour models because of this high correlation that the RGB model provides [50]. Furthermore, RGB is used as the feature extraction in this study owing to its simplicity of implementation and its non-linearity with visual inspection. Other colour models, such as HIS (hue, saturation, and intensity) and CIELUV are computationally

intensive and it will not make sense to use them for this study, as this will increase the computational burden on the cloud robotic framework. Many image-filtering and edge-detection techniques have been successfully applied to the RGB colour model to produce the required image. The next section therefore introduces some of the commonly used image-filtering techniques.

2.6.4 Image filtering

Noise elimination is one of the major aspects of computer and robotic vision in the field of image processing, because noise leads to degraded images. Noise is generally referred to as the manifested undesirable information contained in an image, which does not in any way relate to the image but disturbs the presentation of an image [54]. The unwanted noise information is generally introduced into an image through image acquisition, using a camera of poor quality, accusation conditions or the scene of the environment [54]. Image filtering or digital filtering is used as the prominent technique to remove noise from digital images [46]. Several different filters are used for image enhancement and noise removal, which can be either linear or non-linear.

Linear filtering is a process in which the output pixel is a linear combination of the neighbourhood pixels, but has the disadvantage of sometimes producing a blurred image [54]. This has therefore resulted in a variety of alternative smoothing techniques that are non-linear. The median filter (MF) technique is one of the most widely used non-linear filtering techniques and is very efficient when considering a small neighbourhood of pixels. However, in cases of a large neighbourhood of pixels or with an image containing a large amount of noise the MF technique is not as successful [50]. To overcome these challenges, several variations of median filters have been developed, such as the multistage filter, recursive median filter and detection-estimation median filter, which incorporated a noise detection algorithm [54].

The weighted median (WM) filter is an extension of the median filter, which operates by giving more weight to certain values in the image window. This technique is seen more as an image-smoothing technique, as the smoothing of a particular image can be controlled through the

weighted variables. An extension of the weighted median is the centred weighted median (CMW) filter, which performs better than the median filter [54].

2.6.4.1 Existing filtering algorithms

The different number of image-filtering techniques over the years has resulted in the measure of similarity $K(x_i, x_j, y_i, y_j)$ being defined is several different ways. Figure 2.16 illustrates the different number of similarity kernels that lead to multiple different filters. These filters include the classical regression filters, the bilateral filters and non-local means filter, to name but a few. The next few sections details some of the existing filtering algorithms.

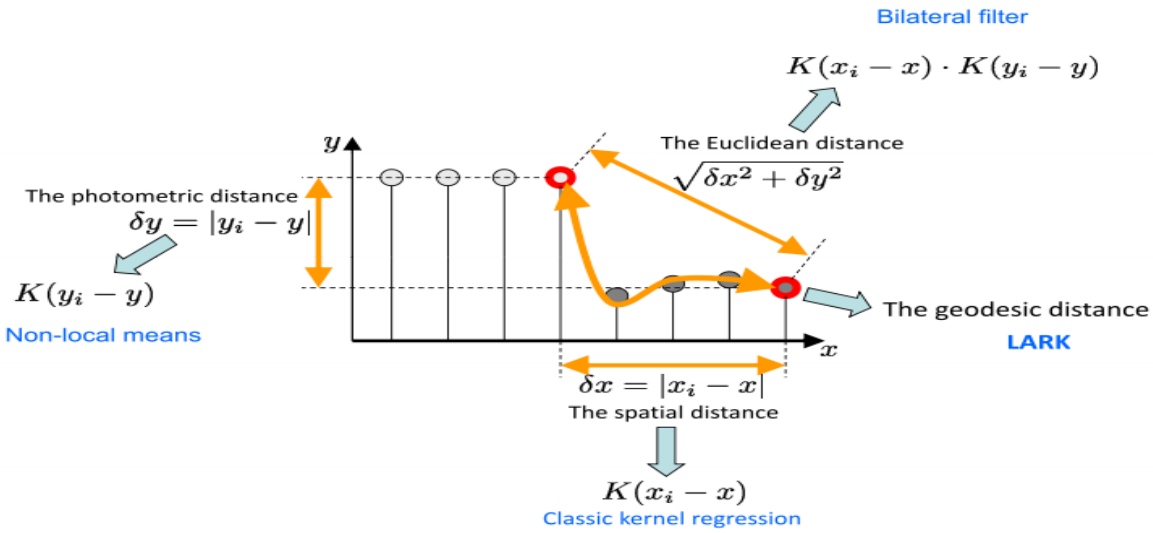


Figure 2.16: Image-filtering similarity metrics and resulting filters [55]

2.6.4.2 Classical regression filters

The most commonly used regression filter is the Gaussian kernel, which measures the distance between two pixels in the subjected image by considering their Euclidean distance. It is worth noting at this point that the Gaussian filter is used in this study as part of the pre-processing process of the drivable road detection because of its simplicity and efficiency. Further details on the Gaussian filter are provided in chapter 3 of the study when describing the methodology of the SCMR framework. The Gaussian kernel itself can we expressed by the following polynomial:

$$K(x_i, x_j, y_i, y_j) = \exp\left(\frac{-\| x_i - x_j \|^2}{h_x^2}\right) \tag{2.5}$$

where h_x is the variance parameter used in the local image statistics to improve the algorithm's performance [55]. One major disadvantage of the classical regression filters, however, is the low adaptivity to the underlying structure. This is often overcome by using additional filters to remove noise from picture frames.

2.6.4.3 Bilateral filter

The bilateral filter was formed as an extension to the classical regression filter to overcome some of its challenges in removing noise and salient pixels from picture frames [51]. In the bilateral filter the spatial and photometric distances between two pixels are presented as separable entities and can be represented as follows:

$$K(x_i, x_j, y_i, y_j) = \exp\left(\frac{-\| x_i - x_j \|^2}{h_x^2}\right) \exp\left(\frac{-\| y_i - y_j \|^2}{h_y^2}\right)$$
$$= \exp\left\{\frac{-\| x_i - x_j \|^2}{h_x^2} + \frac{-\| y_i - y_j \|^2}{h_y^2}\right\} \quad (2.6)$$

where the weighted Euclidean distance between vectors (x_i, y_i) and (x_j, y_j) produces the similarity metric [55]. The main advantages of this filter is its simplicity in that it has only two control variables (h_x, h_y) and is computationally simple to calculate. The major drawback of this filter, however, is that in low signal-to-noise scenarios the performance is not effective [55].

2.6.4.4 Non-local means filter

The Non-local means (NLM) filter, originally proposed in [56], is a simple generalization of the bilateral filter. The main difference between the filters is that in the bilateral filter e similarity kernel (photometric term) is measured point-wise while in the NLM filter it is measured patch-wise. The second difference is that in the NLM filter the distance between the patches is ignored, while in the bilateral filter this geometric distance is taken into consideration [55]. In the case of the NLM filter this tends to lead to a strong contribution from the image patches that may not be near the region of interest in a particular frame. The NLM kernel is therefore summarized mathematically as follows:

$$K(x_i, x_j, y_i, y_j) = \exp\left(\frac{-\| x_i - x_j \|^2}{h_x^2}\right) \exp\left(\frac{-\| y_i - y_j \|^2}{h_y^2}\right) \text{ with } h_x \to \infty \quad (2.7)$$

where y_i and y_j refer to the patches of pixels centered around x_i and x_j respectively. It is noted in [56] that it could be computationally impractical to compare all patches y_i to y_j and that rather than h_x being infinite it is reasonably limited to the neighborhood of pixels of y_j instead. In practice h_x is never infinite and despite its popularity among researchers the NLM filter has many flaws that are to be addressed before it can become a powerful and efficient filtering algorithm [55].

2.6.5 Edge detection

Edge detection is a termed in image processing and computer vision for identifying areas in an image of sharp contrast [57]. In terms of robotic vision it is a task of fundamental importance, as edges help characterize object boundaries and can therefore assist the robot in identifying the objects of an environment. It is also useful for segmentation and registration [55]. Edge detection primarily consists of three steps: filtering, enhancement and detection. A common technique used to identify edges in an image is to consider the brightness values of neighbouring pixels and determine if they are significantly different [57].

Over the years various edge detection techniques have been proposed; however, a majority of the different techniques can be grouped into either the Gradient method or the Laplacian method [57]. The Gradient method is a search-based method that detects the edges of an image by formulating the measure of the edge strength using a gradient magnitude [55]. The Laplacian method, on the other hand, determines the edges of an image by searching for zero crossings in the second derivative of the subjected image [55].

Edge detection takes into consideration two derivatives $\partial_x f(x,y)$ and $\partial_y f(x,y)$, where the variables x and y represent the row and column coordinates respectively in a continuous image [55]. Gradient magnitude and gradient orientation are two important functions in edge detection, which are represented by mathematical directional derivatives. The gradient magnitude is represented as:

$$|\nabla f(x,y)| = \sqrt{(\partial_x f(x,y))^2 + (\partial_y f(x,y))^2} \tag{2.8}$$

and the gradient orientation is defined as:

$$< \nabla f(x,y) = \arctan\left(\frac{\partial_y f(x,y)}{\partial_x f(x,y)}\right). \quad (2.9)$$

Using these derivatives, many edge detection methods have been derived over the years. The next section details the use of these derivatives in some of the leading edge detection methods.

2.6.5.1 Canny edge detection algorithm

The Canny edge detection algorithm was developed by John F. Canny in 1986 and uses a multi-stage algorithm to detect edges in images [58]. Canny set out to improve on all the existing edge detection techniques that were available at the time and this included decreasing the error rate, localizing edge points and having only one response to a single edge [58]. Using these criteria, the Canny edge detection algorithm first smooths an image to reduce the level of noise and then detects the image gradients that contain high spatial derivatives. After determining the pixels containing the high spatial derivatives the algorithm subdues other pixels that are not at that maximum. Thereafter the algorithm uses a hysteresis to track the remaining pixels that have not been subdued. The hysteresis makes use of two thresholds: If the magnitude is above the threshold and edge is detected or if the magnitude is below the first threshold, when it is [58]. The Canny edge detection algorithm consists of multiple tests. The next few paragraphs describe the steps required in the algorithm.

Step 1: The first step in the Canny edge detection algorithm, before attempting to detect any edges, is to remove the noise from an image. The filter used in the algorithm to remove this noise is the Gaussian filter because of the simple mask it uses for computing noise in a particular image [59]. Thereafter Gaussian smoothing is performed, using convolution methods, which results in a convolution mask being overlaid on the image to manipulate a matrix of pixels at a time [59].

Step 2: Once the noise elimination and smoothing of an image has been done, the next step is to detect the strength of a particular edge by using the image gradient. The algorithm then uses the Sobel operator [60] to create a pair of 3x3 convolution masks and performs a 2D spatial gradient measurement of the image [59]. The first 3x3 convolution mask is used to estimate the gradient in the columns, while the second 3x3 convolution mask is used to estimate the

gradient in the rows. Figure 2.17 shows the pair of 3x3 convolution kernels used in the algorithm.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 2.17: 3x3 convolution kernels used in Canny edge Ddtection

Using the matrices G_x and G_y the edge strength is the formulated as follows:

$$|\mathrm{G}| = |G_x| + |G_y| \tag{2.10}$$

Step 3: The third step in the algorithm is to determine the direction of the edge, using the gradients (x and y). In general if G_y has a value of zero then the edge direction will be zero degrees, otherwise the edge direction is 90 degrees [59]. The following polynomial is used to detect the edge direction:

$$\mathrm{Theta} = \mathrm{invtan}\left(\frac{G_x}{G_y}\right) \tag{2.11}$$

Step 4: After determining the direction of the edge, the next step is to relate the direction of the edge to the image under study [58].

Step 5: Thereafter the algorithm uses non-maximum suppression to follow through on the edge direction calculated in step 3 and 4 and suppress any pixel value that is not considered an edge by defaulting the pixel value to zero [59].

Step 6: Finally, hysteresis [60] is used to remove any streaking that may have occurred. Two thresholds (T1 and T2) are used to accomplish this task. The algorithm then defines an edge as any pixel value above T1 and any pixels connected to this edge pixel that have a pixel value greater than T2 [59].

2.6.5.2 Robert's cross-operator

A commonly used edge detection algorithm is Robert's cross-operator, which performs a 2D spatial gradient measurement on the image and is computationally very efficient. The absolute magnitude of the spatial gradient of the image is represented by the output of each pixel value.

Robert's cross-operator makes use of a pair of 2x2 convolution kernels, namely G_x and G_y, as illustrated in Figure 2.18. It can be seen that the kernel G_x is simply the inverse (rotated 90°) of the kernel G_y. The kernels are designed to respond optimally to the identified edges in an image that runs at 45° [55].

Figure 2.18: 2x2 convolution kernels used in Robert's cross-operator

Separate measurements of the gradient component can be produced by applying each kernel separately to the image [61]. The output of applying each kernel to the image can then be combined to determine the absolute magnitude of the orientation of that gradient [61], which is given by the following formula:

$$|G| = \sqrt{{G_x}^2 + {G_y}^2} \tag{2.12}$$

where the absolute magnitude is computed faster and more efficiently. Robert's cross-operator also makes provision for the angle of orientation of the edge relative to the pixel grid orientation [61] and this is given by:

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) - \frac{3\pi}{4} \tag{2.13}$$

2.6.5.3 Sobel operator

The Sobel operator is a computationally efficient algorithm based on convolving an image with small integer valued filters in horizontal and vertical directions [60]. The horizontal and vertical filters are represented by a pair of 3x3 convolution kernels, as shown in Figure 2.19, where the kernel G_y is simply the kernel G_x rotated 90°.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 2.19: 3x3 convolution kernels used in Sobel operator

The edge of the image is detected by overlaying (convolving) the kernels on the image. Thereafter the pair of kernels perform a 2D spatial gradient measurement on the image [60], to determine the absolute gradient magnitude at each point of the grayscale input image. The kernels are also designed to run vertically and horizontally relative to the pixel grid. The gradient magnitude of the image in the Sobel operator is given by Formula 2.14 and the angle of orientation of the edge is given as:

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{2.14}$$

It should also be noted at this point that the Sobel algorithm is chosen as the edge detection technique used in the proposed SCMR framework because of its efficiency in detecting edges. Further detail on the design of the Sobel filter in the proposed SCMR framework is presented in chapter 3 of the dissertation.

2.6.5.4 Laplacian of Gaussian

The Laplacian of Gaussian is one of the most commonly used edge detection algorithms in visual processing systems, including robots, because of its ability to identify regions in an image of rapid intense change effectively and efficiently [59]. In most visual processing systems an image is typically smoothed by using the Gaussian filter to reduce the level of noise in a particular image.

In applying the Laplacian of Gaussian algorithm, the input is generally a greyscale image and the output is another greyscale image. The Laplacian $L(x, y)$ of an image with pixel intensity $I(x, y)$ can be represented by the following polynomial [59]:

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \quad . \tag{2.15}$$

Three commonly used kernels are used in the Laplacian of Gaussian algorithm, which is used to approximate the second derivatives of the Laplacian [59]. These commonly used kernels are

shown in Figure 2.20. The one downfall of the kernels is that they are very sensitive to noise, but this is overcome using the Gaussian filter.

1	1	1
1	-8	1
1	1	1

-1	2	-1
2	-4	2
-1	2	-1

0	1	0
1	-4	1
0	1	0

Figure 2.20: Three commonly used kernels in the Laplacian of Guassian algorithm

2.7 CHAPTER SUMMARY

In summary, the related literature and technology overview illustrate that the concept of cloud computing has shifted the manner in which computing resources and applications are developed and delivered. The paradigm of cloud robotics makes use of the concept of cloud computing and extends the current capabilities of robots by enabling the availability of robotic services and resources over the internet [5].

This chapter also presented a summary of related work by providing an overview on the laxities of the current existing cloud robotic models, especially with regard to frameworks that are closely linked to the proposed work in this dissertation. Although cloud robotics is still in its infancy, researchers across the world have made some strides in making sure that cloud robotics becomes a reality. Some of the related literature used in this chapter covers the RoboEarth project [7], DAvinCi project [10] and the Rosbridge project [4].

These projects, as well as other cloud robotic projects sponsored by major technology companies such as Google [9], IBM and Microsoft, and their related technologies were also evaluated, in order to determine the relevance of the proposed work and the limitations of the existing frameworks in relation to the objectives of this study. Finally, the concept of robotic vision was introduced, together with relevant literature on existing image processing and edge detection techniques. The amount of energy required to process these computationally intensive algorithms was also highlighted. The next chapter introduces the proposed SCMR framework

CHAPTER 3: RESEARCH METHODOLOGY AND DESIGN

3.1 INTRODUCTION

The purpose of this chapter is to describe the research methodology and design used in the study. The research methodology and design have been developed to meet the requirements of the study and also detail the manner in which the research study was conducted. A comprehensive research design is required to ensure that the study is able to provide trustworthy answers to the research questions. Therefore the chapter begins with a discussion on the methodology used to evaluate the proposed SCMR framework in order to develop a comprehensive architecture that adheres to the fundamental principles of cloud robotics. Thereafter the proposed SCMR framework architecture is explored, followed by the communication management and energy utilization of the framework. The chapter then concludes with the chapter summary, as depicted in Figure 3.1.

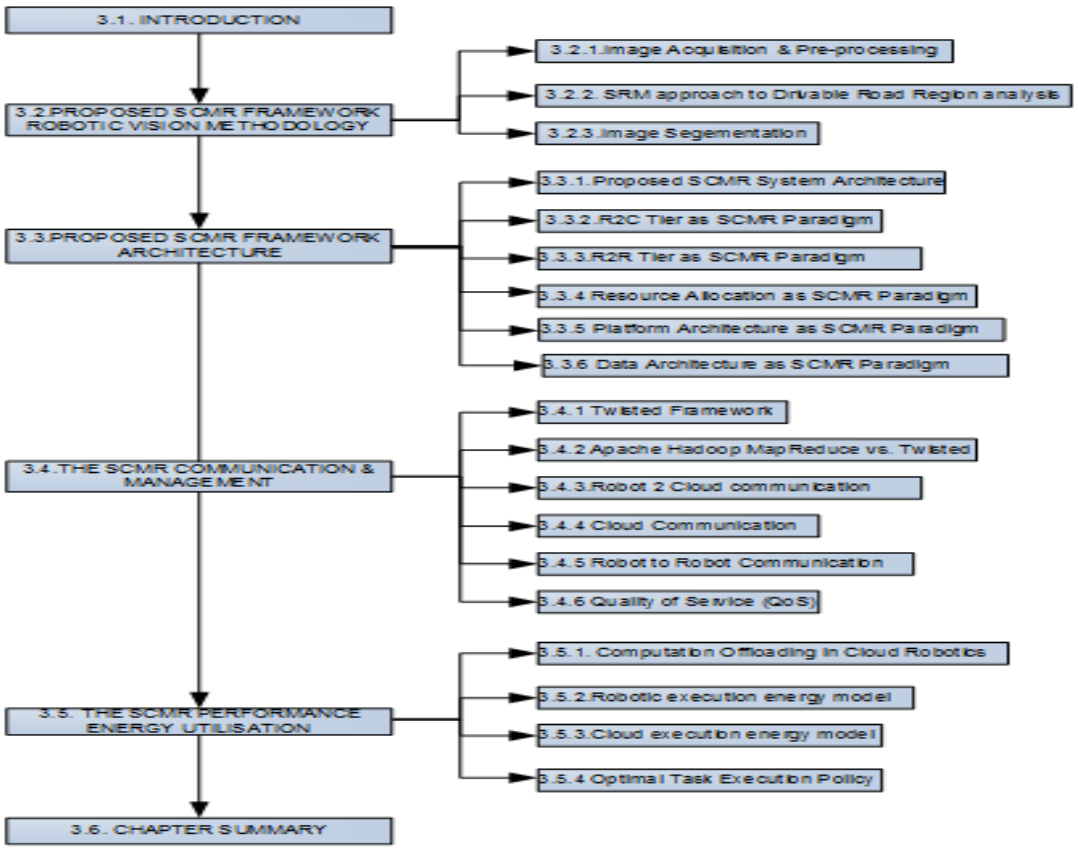


Figure 3.1: Graphical representation of Chapter 3

3.2 PROPOSED SCMR FRAMEWORK ROBOTIC VISION METHODOLOGY

This study was used to conduct drivability analysis for autonomous multi-robot teams in heterogeneous environments by leveraging the benefits provided by the robot cloud server to identify drivable regions from the images captured by the individual robots. The study makes use of a distributed architecture by considering the cloud server to have dedicated computers to perform the image processing. The drivable road detection computation is abstracted from the individual robots and performed in the cloud server, as shown in Figure 3.2. This section is therefore used to present and explain the system model for the cloud-enabled multi-robotics: drivable road detection system.

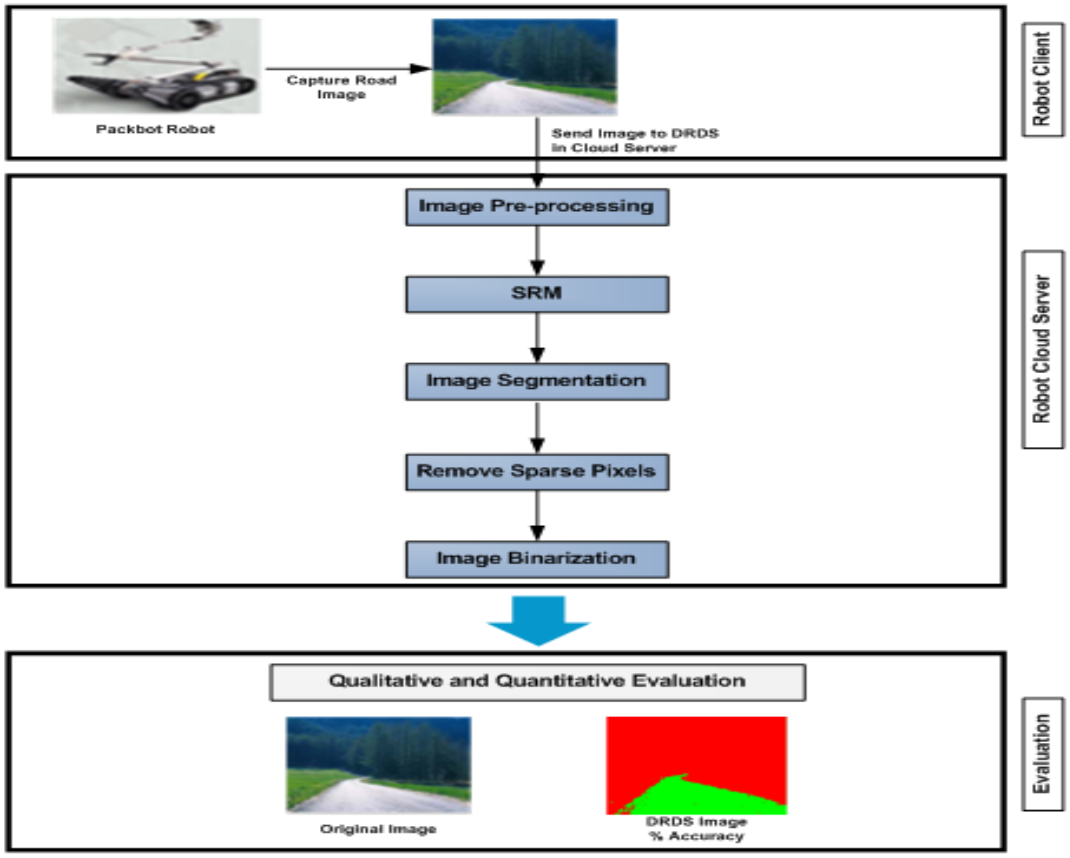


Figure 3.2: Schematic of the proposed system for cloud-enabled multi-robotics: Drivable road detection system

3.2.1 Image acquisition and pre-processing

Image acquisition is the process of finding a good data representation of images in a specific domain and these are subjected to available measurements. In this work the images were

acquired from publicly available road images found in accredited research papers in the domain of drivable road detection. Images for this study were also acquired locally using a camera to capture heterogeneous drivable road images, since the assumption is that the individual robots will be equipped with cameras to assist with their vision requirements. The images acquired to test the cloud-enabled multi-robotics drivable road detection system (CMR-DRDS) ranged from simple drivable road images with no obstacles to off-road images in desert terrains to underground mines and hallways with obstacles. This was done to ensure that the CMR-DRDS system was tested using a wide range of heterogeneous road images, since cloud-enabled multi-robotic teams could be found working in many different heterogeneous environments and the ability to navigate to a particular destination is important in ensuring that a task is completed successfully.

Image pre-processing is generally carried out on the raw data using highly sophisticated image-filtering algorithms [62] prior to data analysis in order to rectify or reduce the noise on a particular image, which is usually brought about by the imaging system. A popular technique to assist with increasing computational efficiency during image pre-processing is down-sampling. The application that will be used for image pre-processing generally determines the type of resolution for a particular image. In this work, during the pre-processing stage all the selected images where down-sampled to a 300x225 resolution. After down-sampling the next step is to filter out any noise in the down-sampled image before trying to perform SRM. The Gaussian filter is used as a 2D convolution operator and acts as a low pass frequency filter [59]. The 2D Gaussian filter function is formulated as:

$$G(x,y) = \frac{1}{2\pi\sigma^2}\, e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.1}$$

where the σ is the standard deviation of the distribution, which is assumed to be non-zero. The distribution of an image with mean (0, 0) and $\sigma = 1$ is illustrated in Figure 3.3.

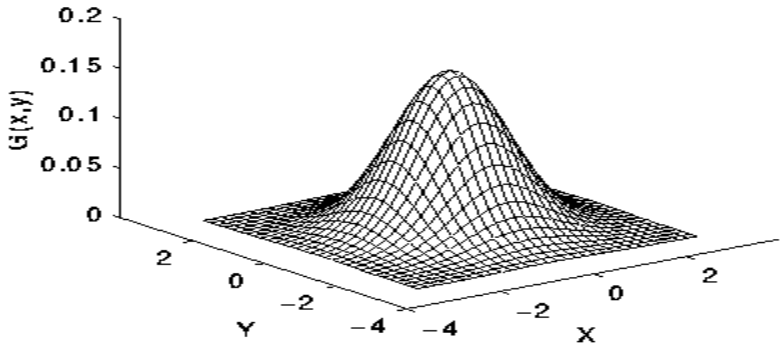


Figure 3.3: -D Gaussian distribution

The Gaussian smoothing filter uses the 2D distribution as a point-spread function to filter an image for noise by using convolution. The Gaussian function is then discretized and stored into discrete pixels using a convolution kernel. The float valued $3x3$ convolution kernel used in this study to approximate a Gaussian with $\sigma = 1$ is illustrated in Figure 3.4.

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

Figure 3.4: 3x3 Gaussian convolution kernel

In the Gaussian filter, as the distance from the kernels centre increases, the kernel coefficients diminish as the centre pixels have a higher weighting than those at the boundary. The kernel coefficient is dependent on the value of σ, as illustrated in Figure 3.3, where the sum of all the coefficients equates to $\sigma = 1$. Increasing the sigma σ value will result in an image becoming more blurred and for this study where the objective is to detect drivable road regions the standard deviation of $\sigma = 1$ is used to ensure that the edges of a particular road image are kept intact and not blurred beyond recognition for the robot. The main advantage of the Gaussian filter over other filters is that it is computationally fast without necessarily being computationally heavy, which is achieved by allowing the Gaussian kernel to be separable [59]. As a low pass filter the Gaussian filter also has the advantage of allowing for the removal of white noise at both high and low frequencies while still

preserving the edges of an image [59]. Figure 3.5 graphically illustrates the pre-processing of an image in the CMR-DRDS using the Gaussian filter with the following convolution matrix:

$$Gaussian_{matrix} = \begin{bmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{bmatrix} \quad (3.2)$$

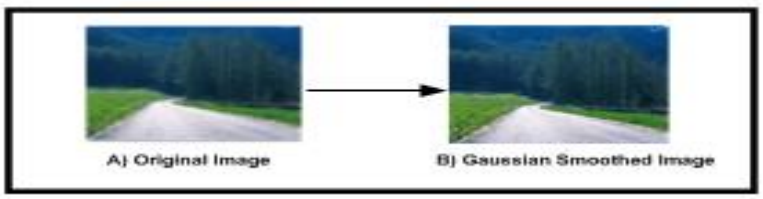


Figure 3.5: A) Original image, B) Gaussian smoothed image with noise removed and edge preserved

3.2.2 SRM approach to drivable road region analysis

Once the images have been smoothed using the Gaussian filter operator, the next step is to divide an image into sets of pixels with homogenous properties, also known as regions, and then iteratively grow the regions by combining them with other smaller regions that have similar homogenous properties. This technique is known as region merging and is widely used in the field of image processing. In this study the SRM technique proposed by Nock and Nielson in 2004 for segmenting an image into similar regions of colour or intensity is adopted [47]. The SRM technique is renowned for its statistical properties, such as homogeneity and separability, while it remains computationally efficient and robust and enables simplicity [63].

In this study a particular region was divided per pixel and a statistical test was applied to the bordering regions. The intensity differences of the bordering regions that are tested then indicate whether or not the region is adequately similar to be merged. The SRM processing of an image in the CMR-DRDS is performed on the cloud server to reduce the computational load on the individual robots. Figure 3.6 describes the flow of the SRM algorithm for drivability analysis in the CMR-DRDS.

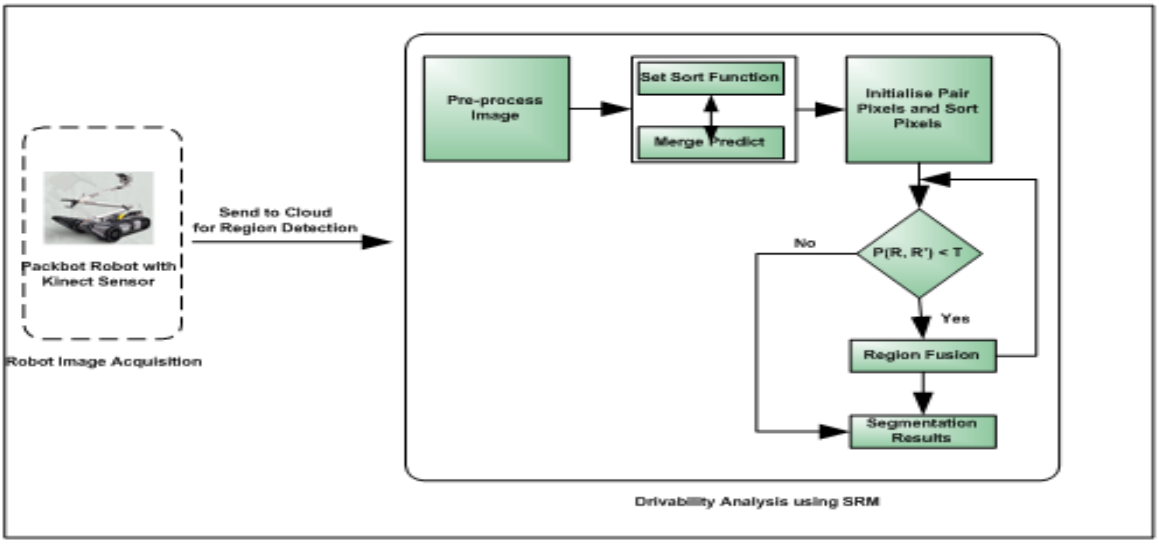


Figure 3.6: Drivability analysis using SRM in CMR-DRDS

There are two important criteria in the SRM algorithm. The first is the merging predict, as indicated in Figure 4, and the second important is the cluster Q. Together these parameters determine the total number of regions for the input image. The value of Q was introduced by [47] to control the complexity and coarseness of an image. An important advantage of the value of Q is the ability to use it as a tradeoff parameter to obtain a compromise between the results observed and the strength of the model [63]. In testing the SRM model with different values of Q, the value of $Q = 32$ gave the optimal result for image classification. Figure 3.7 illustrates the segmentation results of a road frame at different Q levels.

The SRM algorithm makes use of four-connectivity scheme, shown in Figure 3.8, to determine adjacent pixels that are relative to the centre pixel. The four-connectivity scheme is typically made up of 2 x m x n –m –n adjacent pixels, which are then sorted in ascending order and traversed in this order only once. The SRM sort function is illustrated in Equation 2.4 [47]. Thereafter the SRM algorithm takes every pair of pixels $(\mathrm{P}, \mathrm{P}')$ for a given set and performs the merging predicate. The SRM pseudo-code is illustrated in Table 3.1. In this study the focus is on the pixel region that forms regions of homogeneity at the base of the observed image I. This region forms the pixel region closest to the robot's vision and can therefore be used as the drivable road regions' as illustrated in the test cases presented in this study. This region is highlighted in yellow in Figure 3.8

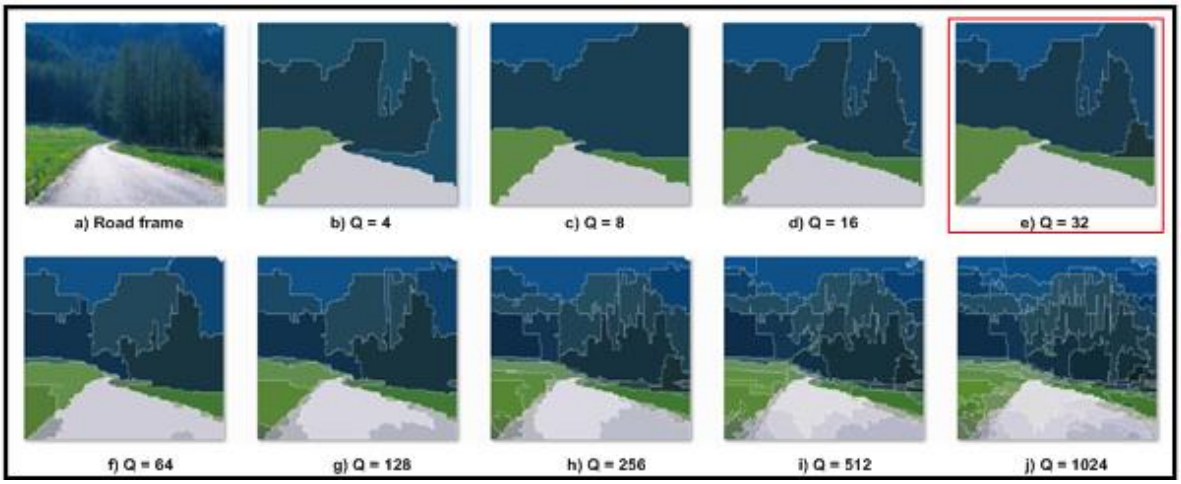


Figure 3.7: Segementation on a road frame at different Q Levels

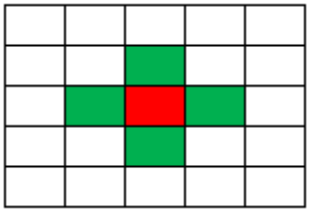


Figure 3.8: Four-connectivity scheme used in SRM

Figure 3.9: Pixel region of homogenity at the base of robot image *I*

Table 3.1: Psuedo-code for the SRM Algorithm [47]

Pseudo code for SRM algorithm
Step 1: Initialize image $\boldsymbol{I}$ and estimated segments $\boldsymbol{Q}$
Step 2: $\boldsymbol{D_I} = \{\boldsymbol{Four - Connectivity\ adjectent\ pixels}\}$
Step 3: $\overline{\boldsymbol{D_I}} = \boldsymbol{sort}(\boldsymbol{D_I}, \boldsymbol{f})$
While $\overline{\boldsymbol{D_I}} \neq \emptyset$
For i = 1 to $\lvert\overline{\boldsymbol{D_I}}\rvert$ do
Step 4: if $\left(((\boldsymbol{P}\left(\boldsymbol{R}_{(\boldsymbol{p_i'})}, \boldsymbol{R}_{(\boldsymbol{pi})}\right) == \boldsymbol{true}\right)\ \boldsymbol{and}\ \left(\boldsymbol{R}_{(\boldsymbol{p_i'})} \neq \boldsymbol{R}_{(\boldsymbol{pi})}\right)\right)$
then merge regions $(\boldsymbol{R}_{(\boldsymbol{p_i'})}, \boldsymbol{R}_{(\boldsymbol{pi})})$

3.2.3 Image segmentation

Image segmentation is one of the basic techniques of image processing and computer vision. It is a key step for image analysis, comprehension and description for robotic vision. The next few sections describe the different image segmentation methods used in the study for drivable road region detection on the proposed SCMR framework.

3.2.3.1 Region of interest selection

Immediately after SRM the search for the region of interest (ROI) begins through segmentation. In the CMR-DRDS the purpose of image segmentation is to identify the drivable road regions and the non-drivable road regions from the road frame submitted to the cloud server by the individual robot. The approach used in this study is to consider a particular region within the observed image as a drivable road region. A problem that is encountered by many segmentation algorithms is the calculation of an ideal threshold in an image that consists of the horizon or sky [64]. This problem is illustrated graphically in Figure 3.10 and shows that with images that contain the horizon or sky the detection of the drivable road region can become a challenge.

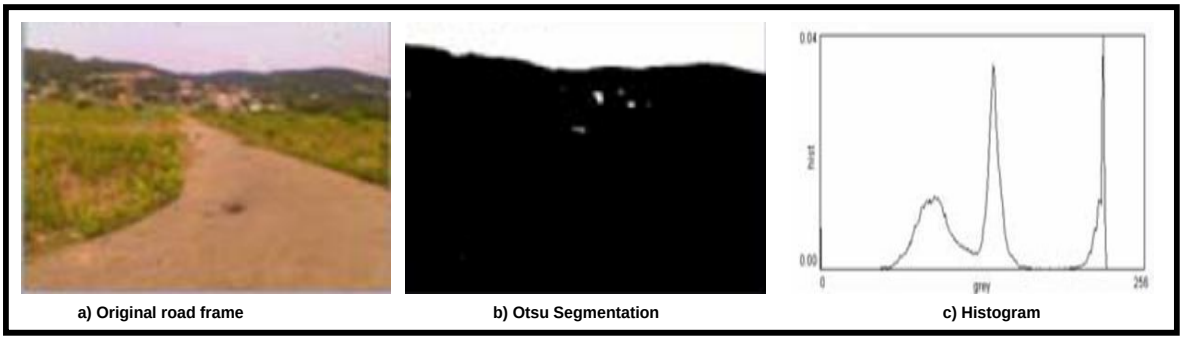


Figure 3.10: Image segementation on image containing horizon or sky [64]

In order to overcome this challenge, many systems have turned to only analysing the portion of the road ahead of the robot and assumed the absence of other obstacles from the road. This approach was adopted by the LAKE and SAVE autonomous vehicles relay, which demonstrated its success by autonomously manoeuvring vehicles on the highway [65]. The RALPH (Rapidly Adapting Lateral Position Handler) system used an alternate approach to select the ROI by only processing the portion of the image based on the result of a radar-based obstacle detection module [65]. Finally in [64], in a study used for real time road detection based on monocular vision, the observed image is divided into two parts. These are not necessarily two equal parts, but complementary sub-parts [64]. The rationale behind this segmentation is that humans use the information of an image closest to them for immediate decisions and use the information of an image on the horizon for future decisions [64].

In this study the approach used for selecting the ROI in order to detect the drivable road region leverages off the work done in [64]. The reason for this is to ensure that the wheel is not reinvented, but rather to improve on work done by others in the same field of study and to continue adding to the body of knowledge. Therefore, using this approach, the observed image is divided into two complementary parts known as the north-bound region and the south-bound region. The next step would then be to determine which of the two complementary images to select as the ROI. The assumption is that the robots will be equipped with individual cameras to capture road images and submit them to the cloud server for processing. Therefore the selected ROI would be the region closest to the robot's vision, which in this case is the south-bound region. Figure 3.11 illustrates the ROI selection through segmentation. The detection of

a drivable road region is achieved with greater precision and more effectively through segmentation when using the ROI as opposed to using the entire image.

Figure 3.11: ROI selection through segementation

3.2.3.2 Colour feature extraction

Once the ROI has been determined the next step in the CMR-DRDS algorithm is the colour feature extraction. Colour extraction is a feature used in image processing to identify objects in a given image frame. An image histogram is used to represent the number of pixels in an observed image graphically as a function of their intensity [50]. Figure 3.9(c) illustrates the histogram of a black and white image generated as output from the Otsu segmentation algorithm. The most common use of the image histogram is to determine the threshold of a particular image frame. If an image is found to be suitable for thresholding, then the corresponding histogram is called a bi-modal histogram because the pixel intensities are clustered around two separate values [50]. The actual threshold value for the observed image is found between the peaks of the histogram.

In this study the ROI is divided into nine complementary sub-images and the histogram is generated using the most dominant colour channel in the bottom centre sub-image, rather than creating the histogram using the entire image. This is described by Equation 3.3.

$$Cc = \arg\max_{\{R,G,B\}} (N_R, N_G, N_B) \tag{3.3}$$

where Cc represents the dominant colour channel. This technique was first introduced by [66] and later used by [67] with the assumption that the bottom centre of the image contains the road pixels most of the time. Figure 3.12 illustrates the sub-image used (pointed out using the

yellow arrows) to generate the histogram from the observed image. This histogram is then used as input to calculate the global threshold, which is discussed in the next section.

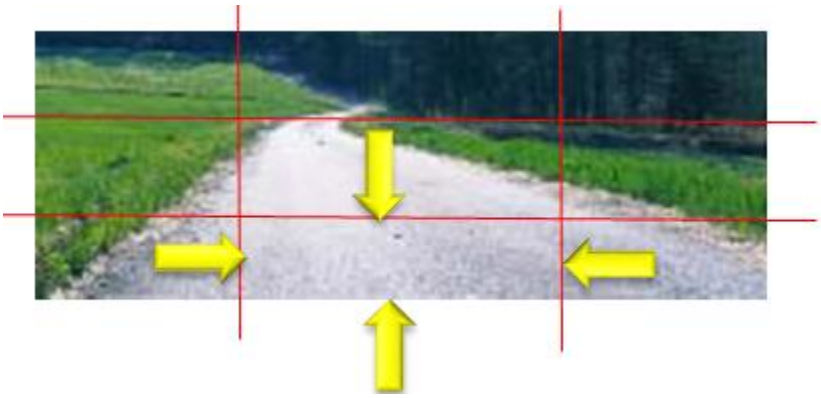


Figure 3.12: Image histogram generated using bottom center sub-image

3.2.3.3 Thresholding

One of the most commonly used image segmentation techniques is thresholding, which partitions an image into a background and a foreground. Image thresholds are divided into two groups: local and global thresholds [67]. The local threshold is determined by dividing a particular image into sub-images and for each of the sub-images a threshold is derived. A global threshold divides a particular image using only one threshold value. In this study the global threshold approach is used, which seeks the threshold associated with the region of interest, i.e. the robot's navigable area, as illustrated in Figure 3.12, instead of using a threshold of the entire image. The global thresholding method using an appropriate threshold T can be represented by Equation 3.4.

$$g(x,y) = \begin{cases} 1, & if\ f(x,y) > T \\ 0, & if\ (f(x,y) \leq T \end{cases} \quad (3.4)$$

The challenge, however, in image thresholding is the ability to determine the threshold of an observed image automatically. The basic idea behind automatic thresholding is the automatic selection of an optimal value for separating ROI from the background of an observed image based on their grey-level distribution [69]. In order to overcome this challenge, much work has gone into developing algorithms that can be used to determine the threshold of an observed image, such as the Otsu method [68], entropy-based methods [63], histogram and clustering-based methods [69].

The OTM, unlike other thresholding techniques, does not limit itself to a particular type of histogram of the image. This results in the OTM being able to be applied to unimodal, bimodal or multimodal histograms. The main disadvantage though of the OTM is its inability to deal with noise. This shortcoming is generally overcome by the use of image filters. In this study the histogram used to calculate the threshold of the observed image is a bimodal histogram, illustrating the peaks between drivable and non-drivable road regions and the Gaussian image filter is used to deal with the reduction of noise pixels in the observed image. Another advantage that OTM has over other thresholding techniques is its ability to overcome negative impacts caused by environmental variations; which is especially important for road detection [67]. Furthermore, the use of OTM in this study as the thresholding technique to determine drivable road regions is further vindicated by [70] and [71], as they regard OTM as one of the best techniques to be used in machine vision and [72] notes OTM as the most referenced thresholding technique in the field of machine vision.

The main characteristic of the OTM is the ability to obtain the maximum number of intra-classes variances of the observed image [68]. The thresholding process in OTM is seen as the partitioning of pixels of the observed image into two classes: C1 (object class) and C2 (background class). In OTM the optimum threshold is calculated by separating C1 and C2 so that their combined spread is minimal, while maximizing the separation between C1 and C2. The process of determining the threshold in OTM is iterative, which starts from an arbitrary value and ends when the change in the threshold value is insignificant, indicating that the optimum threshold value is close to being reached. In general the OTM is recursive and searches the maximization for the cases: $C1 = \{0,1,2,3, \ldots \ldots T\}$ and $C2 = \{T+1, T+2, T+3, \ldots \ldots N-1\}$ where T is the chosen optimal threshold and N the number of intensity levels of the image. The OTM searches for the threshold that minimizes the intra-class variance $\sigma_\omega^2(T)$ defined as a weighted sum of variances of classes C1 and C2. This is described in Equation 3.5:

$$\sigma_\omega^2(T) = q_1(T)\sigma_1^2(T) + q_2(T)\sigma_2^2(T) \tag{3.5}$$

where $\sigma_\omega^2(T)$ is the intra-class variance, $\sigma_1^2(T)$ is the intensity of the variance of the background pixels, $\sigma_2^2(T)$ is the intensity of the variance of the foreground pixels, $q_1(T)$ is the proportion of the background pixels and $q_2(T)$ is the proportion of the foreground pixels [68].

The class probabilities and class means for the background and foreground pixels are respectively estimated by Equation 3.6 and Equation 3.7:

$$q_1(T) = \sum_{i=1}^{T} H(i) \;\; and \;\; q_2(T) = \sum_{i=T+1}^{N} H(i) \tag{3.6}$$

$$\mu_1(T) = \sum_{i=1}^{T} \frac{iH(i)}{q_1(T)} \;\; and \;\; \mu_2(T) = \sum_{i=T+1}^{N} \frac{iH(i)}{q_2(T)} \tag{3.7}$$

The individual class variances for the background and foreground pixels in the OTM are then estimated by Equation 3.8 and Equation 3.9:

$$\sigma_1^2(T) = \sum_{i=1}^{T} [i - \mu_1(T)]^2 \;\; \frac{H(i)}{q_1(T)} \tag{3.8}$$

$$\sigma_2^2(T) = \sum_{i=T+1}^{N} [i - \mu_2(T)]^2 \;\; \frac{H(i)}{q_2(T)} \tag{3.9}$$

where H is the histogram of the selected channel from the bottom center sub-image selected as the ROI in Equation 3.4. The histogram generated from the bottom centre sub-image (Figure 3.11) is then used as input into the OTM to determine the global threshold of the observed image. The pseudo-code for the Otsu algorithm is presented in Table 3.2.

Table 3.2: Psuedo-code for the Otsu algorithm [68]

Pseudo-code for Otsu algorithm
Step 1: Estimate histogram and probabilities of each intensity level
Step 2: Set up initial class probability $\boldsymbol{q_i(0)}$ and class mean $\boldsymbol{\mu_i(0)}$
Step 3: Search all possible thresholds t = 1,2,3, … … .N; 1. Update class probability $\boldsymbol{q_i}$ and class mean $\boldsymbol{\mu_i}$ 2. Thereafter compute $\boldsymbol{\sigma_b^2(T)}$
Step 4: The optimal threshold $\boldsymbol{T}$ that corresponds to the maximum of $\boldsymbol{\sigma_b^2(T)}$

3.2.3.4 Sobel filter design

The Sobel filter is used as the edge detection algorithm in the drivable road detection system because of ability to detect edges in noisy images with little computation overhead required. The Sobel filter makes use of two 3x3 convolution masks and has inbuilt smoothing for noise

removal [62]. The convolution mask is slid over the image, enabling a set of square pixels in the image frame to be manipulated at a time. Each Sobel 3x3 convolution mask comprises a digital differentiator in one direction and a smoothing operator in the other direction [57].

Each Sobel mask uses mask coefficients in a weighted sum of value pixels (i, j) as the mask is slid over an image. The pixel values in the mask change as the mask shifts one position to the right until it reaches the end of the picture frame. Table 3.3 illustrates coordinates of the Sobel convolution matrix as they progress through the image frame.

Table 3.3: Coordinates of Sobel Convolution mask [50]

$(i-1, j-1)$	$(i-1, j)$	$(i-1, j+1)$
$(i, j-1)$	(i, j)	$(i, j+1)$
$(i+1, j-1)$	$(i+1, j)$	$(i+1, j+1)$

In applying the mask over a pixel that is in the first column of an image frame, the first column, last column, first row and last row of Table 3.3 cannot be manipulated by the 3x3 convolution mask [50]. This is overcome by replacing those values in the affected columns and rows by zeros. This technique is better known as padding. Using Table 3.3, the convolution operation to be performed on an image frame by the two 3x3 Sobel convolution matrices G_x and G_y can be represented as follows:

$$G_x[i,j] = Im[i-1, j+1] + 2 * Im[i, j+1] + Im(i+1, j+1) - [Im[i-1, j-1] + 2 * Im[i, j-1] + Im(i+1, j-1)] \quad (3.10)$$

$$G_y[i,j] = Im[i-1, j-1] + 2 * Im[i-1, j] + Im(i-1, j+1) - [Im[i+1, j-1] + 2 * Im[i+1, j] + Im(i+1, j+1)] \quad (3.11)$$

where Im is the original image frame and i, j are the mask coefficients in a weighted sum of value pixels. The pseudo-code of the Sobel edge detection algorithm is shown in Table 3.4.

Table 3.4: Psuedo-code for the Sobel edge detection algorithm [50]

Pseudo-code for Sobel edge detection algorithm
Step 1: Receive input image frame $\boldsymbol{G}$
Step 2: Read and load image frame with detected edges $\boldsymbol{E}(\boldsymbol{G})$
Step 3: Extract feature matrix $\boldsymbol{A}(\boldsymbol{G})$
Step 4: Compute $\boldsymbol{G_x}$ and $\boldsymbol{G_y}$ by applying the horizontal and vertical 3x3 convolution matrices
Step 5: Iterating through each pixel location calculate the gradient magnitude, $\|\mathbf{G}\| = \sqrt{\boldsymbol{G_x}^2+\boldsymbol{G_y}^2}$
Step 6: Return Edge featured image frame $\boldsymbol{E}(\boldsymbol{G})$

3.2.3.5 Removal of sparse pixels

Sparse pixels are generally referred to the set of pixels that appear as noise during image segmentation in images captured by the robot's camera. The actual definition of noise varies between scientific domains but the most commonly used definition of noise is unwanted signals [50]. In this study the term noise is referred to the unwanted signal (sparse pixels) in the images captured by the robot. These sparse pixels obstruct the autonomous navigation of robots and therefore the removal or minimization of sparse pixels is fundamental to the process of robotic vision.

The removal of sparse pixels occurs after the edges of a particular image have been detected. This is done by replacing the values of the pixels representing noise in the image by either a uniform value or zero. During the process of edge detection, the boundaries of a particular image are outlined by identifying areas of sharp contrast. Thus, after detecting the edges in the image, the process of removing the unwanted noise still remains, to ensure the robots are able to navigate autonomously to the desired location.

The study leverages some of the work done in [50] by applying some heuristics to remove the sparse pixels. A feature matrix is used and in each column-wise, bottom-up iteration, sparse pixels are replaced with a uniform value. The algorithm for removing the sparse pixels is illustrated in Table 3.5.

Table 3.5: Psuedo-code for the removal of sparse pixels [50]

Pseudo-code for removal of sparse pixels
Step 1: Receive Edge featured image frame $\boldsymbol{E}(\boldsymbol{G})$ with sparse pixels
Step 2: Extract Edge detected matrix $\boldsymbol{M}(\boldsymbol{G})$ of size $\boldsymbol{P} \times \boldsymbol{Q}$
Step 3: Iterate through $\boldsymbol{M}(\boldsymbol{G})$ using column-wise bottom-up approach.
Step 4: Replace all the values above a pixel value (deemed as noise) by a uniform value: 1. If the pixel value is zero (in instances of pixels located at edges), OR 2. If the average of the neighbouring value is greater than a defined threshold
Step 5: Repeat for all columns until sparse pixels are removed
Step 6: Return Edge featured image frame $\boldsymbol{E}(\boldsymbol{G}^{'})$ with sparse pixels removed

3.2.3.6 Image binarization

Image binarization is the process of identifying one of two possible values for each pixel. Generally the two colours used in image binarization are black and white, with a particular image split into the background and foreground. In this study the image captured by the robot is binarized using an optimal global threshold value and the pixel values above the threshold are deemed drivable, while the pixels below the threshold are deemed non-drivable. In addition, the two colours used in this study for the image binarization are green (drivable) and red (non-drivable), as illustrated in Figure 3.2.

Many binarization techniques can be used in the process of simplifying and unifying a particular image. The study makes use of the Otsu method to binarize the robot images because of its ability to overcome the challenge of image degradation [67]. The degradation of images is generally a result of a poor image source type, inadequate image acquisition and external environmental factors [64]. The first step in the process of image binarization is to convert the image into a greyscale image. Thereafter the threshold is determined using the Otsu method by creating the histogram of the particular image. Finally the image is binarized using the selected threshold. An overview of the binarization algorithm is illustrated in Figure 3.13

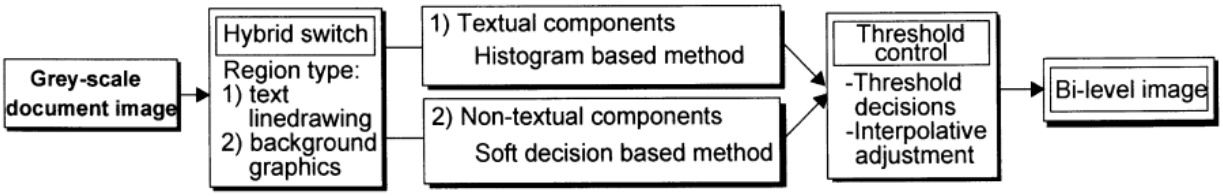


Figure 3.13: Overview of the binarization algorithm

3.3 PROPOSED SCMR FRAMEWORK ARCHITECTURE

This section describes architecture and design choices of the overall system architecture for the SCMR framework. It will also elaborate on the resource allocation, platform architecture, data architecture and communication architecture for the survivable cloud multi-robotic framework.

3.3.1 Proposed SCMR system architecture

The proposed SCMR framework consists of two tiers; the R2C tier and R2R tier. This is done with the intention to overcome the challenge of multi-robots that find themselves in heterogeneous environments with limited access to the internet and intermittent disconnections. The SCMR framework also differentiates itself from other cloud frameworks by making use of a robot leader, which is a clone of the physical cloud infrastructure. In many instances of teams that participate in sport, work or any other activity that is team-oriented, involving individuals working together towards a common goal, a leader is required to provide guidance, instruction, direction and leadership to the group of individuals for the purpose of achieving a key result or group of aligned results [73]. Similarly, the SCMR framework looks towards the robot leader to provide leadership to the individual robots in the R2R network tier when disconnections occur or access to the physical cloud infrastructure is limited.

The robot leader is also used to bridge the gap between the R2R network tier and the R2C network tier. In the R2C network tier the robot leader is updated with frequently requested data from the individual robots to ensure that the robot leader is synchronized with the physical cloud infrastructure in the event of disconnections. In the R2R network tier the robot leader serves as a clone of the physical cloud infrastructure and provides the individual robots with image processing, path navigation and path planning information through the creation of a

collaborative virtual ad hoc cloud. The system architecture of the SCMR framework is illustrated in Figure 3.14.

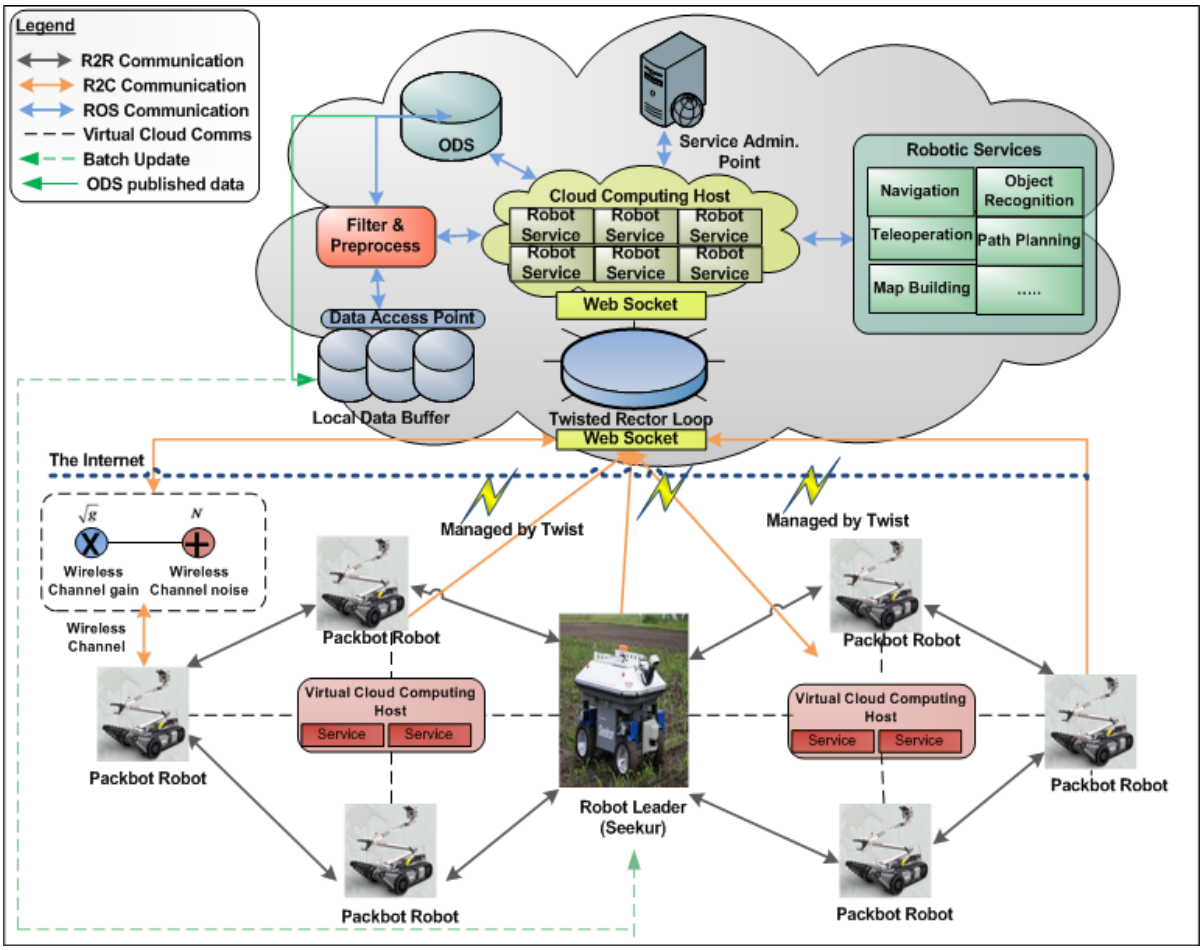


Figure 3.14: System architecture of survivable cloud multi-robotics framework

The main entities involved in the R2C system architecture are the robot clients (including robot leader), cloud cluster host (CCH), database, service administration point and robotic services. The SCMR framework makes use of the CCH as the central unit that takes care of all the jobs that come to and from the cloud, which are handled by the twisted framework. The twisted framework [74] is a framework for deploying an asynchronous, event-driven and multi-thread supported network system coded in Python. The CCH is set up to run any process that is a ROS [30] node, and all processes within the physical cloud infrastructure communicate with one another using the ROS inter-process communication. Having the well-established ROS protocol inside the cloud environment allows applications to run all existing ROS packages without any modification, and lowers the hurdle for application developers [6].

All robotic services in the SCMR framework, such as object recognition, path planning, map building, navigation, etc., are introduced through a service administration point with a web standard description language. Along with the cloud robotic services, the proposed framework also makes use of an operational data store (ODS), which is used to store all sensor data being transmitted by the individual robots in the multi-robot team. The data stored in ODS can range from the robot's geographical location to the robot's actual health condition. Finally a local data buffer is deployed for storage of the frequently requested data. Since activities of robots are usually regular, the same resource may be queried repetitively, thereby the number of queries that are required from the ODS. This will help improve the latency of requests and improve the real time response rate and reliability of response rate of robot requests, which are critical in a multi-robotic team environment. The data stored in the local data buffer is then downloaded by the robot leader in a scheduled batch job. This is done to ensure that the robot leader and the local data buffer are always synchronized, ensuring that in cases of disconnection the robot leader can provide the individual robots with the data and information they require to complete a particular task.

The proposed SCMR framework benefits are the following:

- It heterogeneous environments such as underground terrains that are accustomed to disconnections, the proposed framework allows the robot leader to serve as a clone of the physical cloud infrastructure and provide the individual robots with image processing, path navigation and path planning information through the creation of a collaborative virtual ad-hoc cloud.
- It allows for the creation of "flash-robots", which can be assembled quickly and cheaply by offloading the computation and data storage intensive tasks into the physical cloud infrastructure.
- The SCMR framework proposes a resource negotiation module that relieves the robots of the computation burden by handling the data request and data response for the individual robots and forming a bridge between the individual robots and the cloud infrastructure.
- It provides the ability to create an extensive library of skills or behaviours of the individual robots

- It provides the ability to leverage the computing capability of each individual robot in a multi-robot system,
- It enables collaborative decision-making by the robots.

The detailed discussions of the SCMR framework are presented in sections 3.3.2 to 3.3.6. One can see that the proposed SCMR has more advantages than the existing ones.

3.3.2 R2C Tier as SCMR paradigm

The greatest benefit of the R2C system architecture is in essence some of the major benefits already brought about by cloud computing. In the R2C tier the physical cloud infrastructure provides a pool of computational and storage resources that can be allocated elastically for real time demand by the multi-robot teams, without themselves having actually to carry the burden of storing or processing this data. The offloading of these computation-intensive tasks into the physical cloud infrastructure has resulted in the creation of "remote-brain" robots [11]. The R2C architecture is illustrated in Figure 3.15.

The first benefit that is provided by cloud storage of data is that it allows large volumes data to be unified and then prepared in a format that can be used by the other robots. The second benefit of storing this data in the cloud is brought about by the ability of data mining. Because of the large volumes of data that will be stored in a central location in the cloud, data mining can provide an extensive library of skills or behaviours that are related to task requirements and situational complexities, making it feasible to learn from the history of all cloud-enabled robots [11]. Other obvious benefits also include the cost reduction on the creation of robots, as most of the processing will done in the cloud, thus reducing the need for additional hardware and the power consumption required by individual robots.

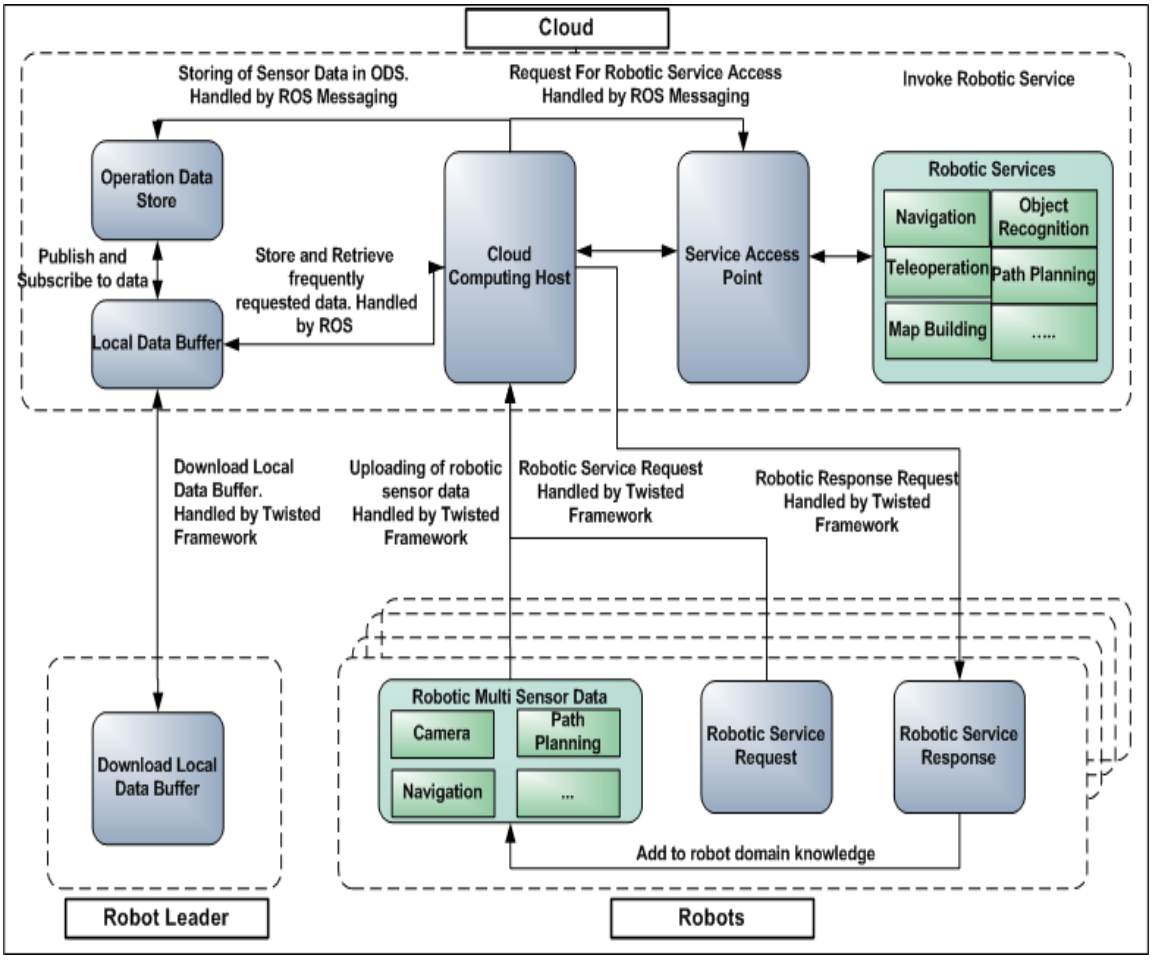


Figure 3.15: Robot-to-cloud tier

3.3.3 R2R tier as SCMR paradigm

In the event of cloud disconnections, the multi-robotic teams will communicate with one another via a wireless network to form a collective computing unit, which could also be viewed as the formation of a virtual ad hoc network. This is done at the R2R tier of the SCMR framework and provides many benefits to the multi-robot system. The R2R system architecture is graphically illustrated in Figure 3.16. In the R2R tier the robot leader, which is a clone of the physical cloud infrastructure, becomes the source of information and data for the individual robots. The ROS master node handles all the jobs that come to and from the robot leader. The robot leader also has a service access point, which is used to determine which services can be accessed by which individual robots. Along with the services, the robot leader also has a local data buffer that is downloaded from the physical cloud and stores the frequently requested data that can be used by the robots.

The communication protocol used in the R2R tier is the gossip protocol [75], which comprises randomized methods designed to transmit a message from a source to a destination without any explicit route-discovery mechanism [75]. The gossip protocol comes in a few variants, one of which is simply to broadcast the message to all robot neighbours, but this could lead to network congestion; another variant of the protocol is used when a robot randomly chooses neighbour robots that are within range and broadcasts that message to those robot neighbours. In the R2R tier the latter gossip protocol variant is used and the robots, after receiving the required information from the robot leader, will broadcast the information to the randomly chosen robot neighbours.

The first benefit of the R2R tier system architecture is the ability to leverage the computing capability of each individual robot in the multi-robot system by pooling all the robots within a specific range to form a virtual ad hoc cloud. The second benefit is that in the virtual ad hoc cloud that is created, information and data can be exchanged between the individual robots and the robot leader, which will enable collaborative decision-making by the robots. The last benefit is that the individual robots can now access and process information from the robot leader, which is a clone of the physical cloud infrastructure, when the individual robot is not in range of a cloud access point.

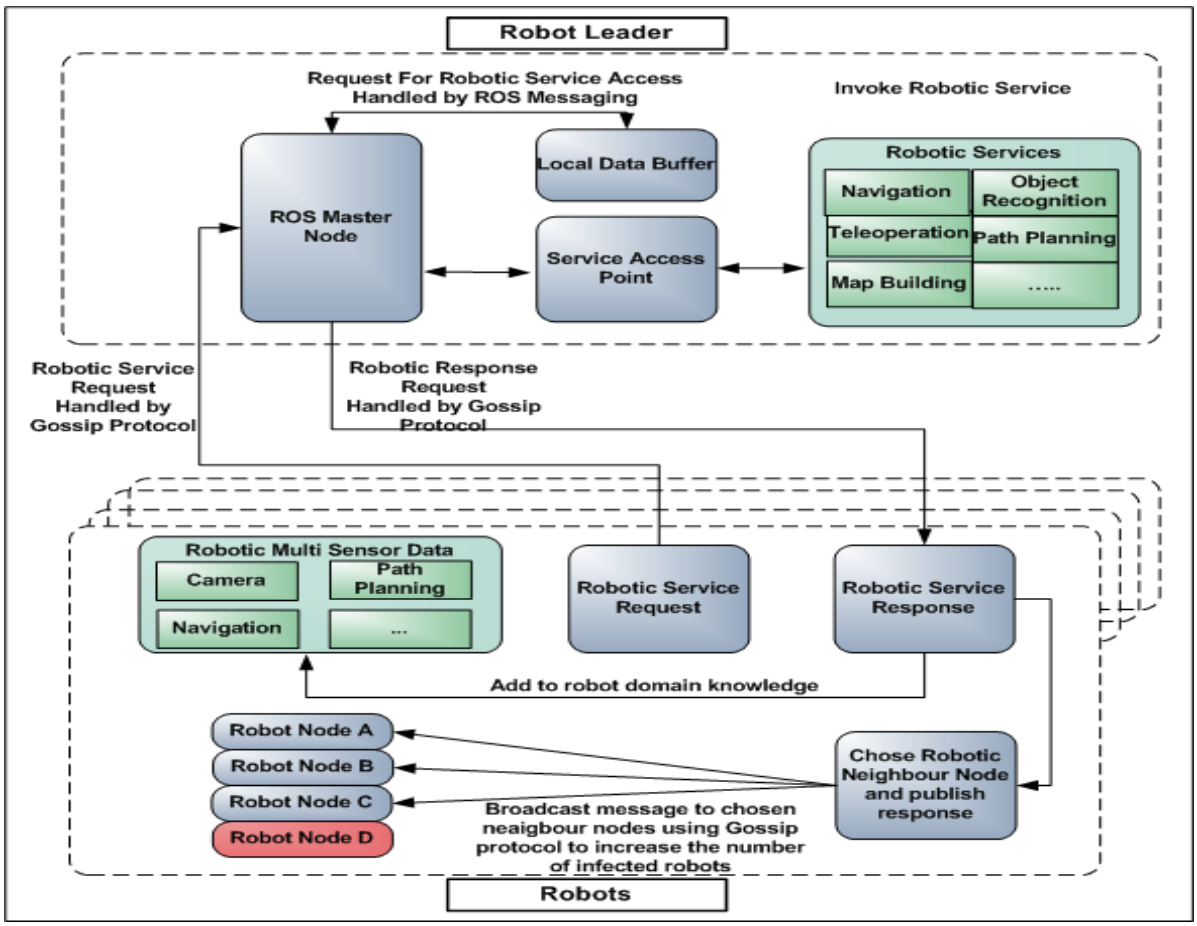


Figure 3.16: Robot-to-robot tier

3.3.4 Resource allocation as SCMR paradigm

A resource allocation system in cloud robotics is regarded as a mechanism that aims to fulfil any requirements of target applications [76]. Autonomous negotiation among multiple robots becomes a crucial problem in cloud robotic systems when they require the resources in parallel [4]. In a heterogonous environment that has a cloud multi-robot team working to achieve several outcomes, it is essential to take dynamic interactions into consideration for resource allocation. In the SCMR framework the resource allocation is divided into the R2C tier and R2R tier, as illustrated in Figure 3.17.

In the R2C tier the CCH is a server running data-retrieval management, as well as a data controller for handling parallel requests from multi-robot clients. It consists of the following major functionalities: resource allocation, resource negotiation, buffer management, data allocation, access control and queue management. The CCH then accesses the local data buffer

to retrieve data frequently used by the individual robots. The ODS stores all sensor information from the individual robots and publishes certain data to the local data buffer.

In the R2R tier, which handles resource allocation in cases of disconnection to the network, the robot leader acts as a CCH and provides the required resources to the individual robots. The robot leader makes use of the local data buffer that has been downloaded from the physical cloud infrastructure to provide the individual robots with the data they could potentially require. A virtual ad hoc cloud (computing unit) is also created by the R2R communication and resources are shared among the robots in this virtual cloud environment.

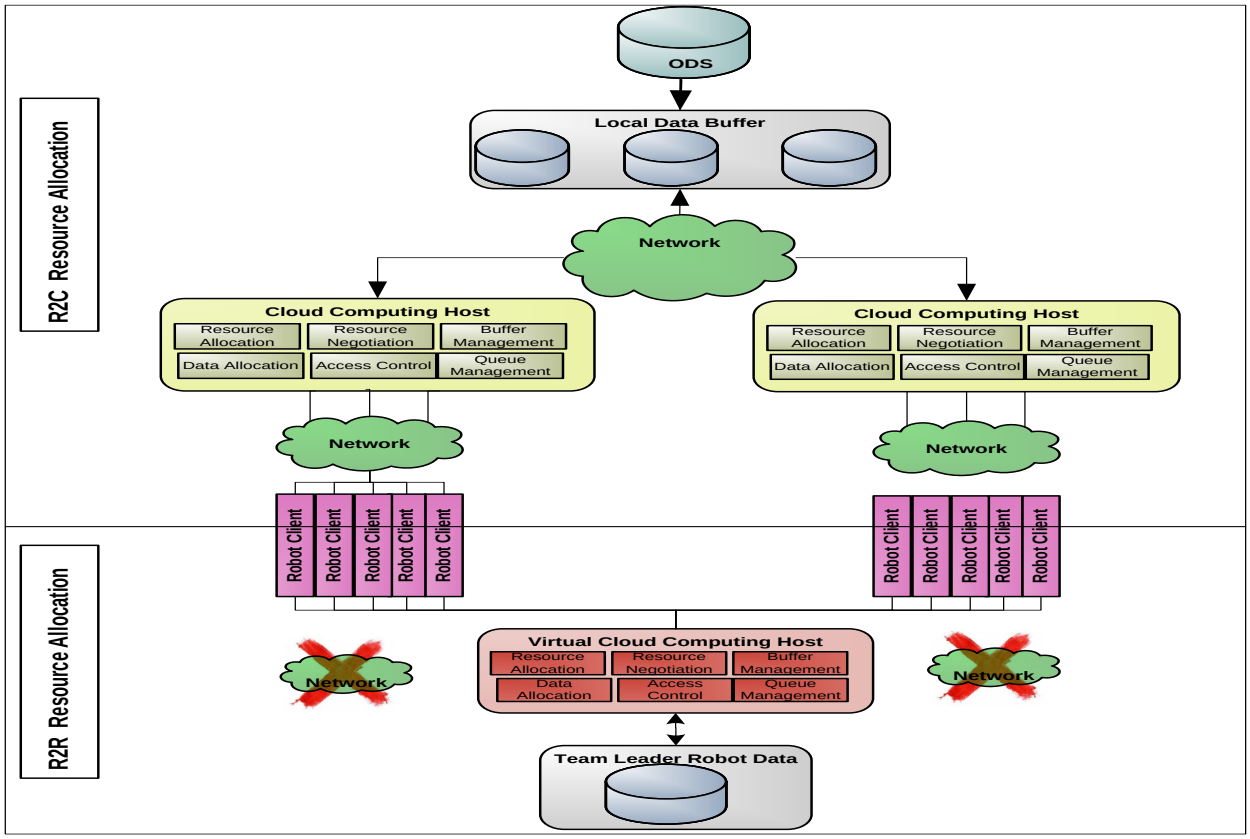


Figure 3.17: Resource allocation framework in SCMR divided into two tiers

3.3.5 Platform architecture as SCMR paradigm

The current cloud platforms, such as RoboEarth [7], DAvinCi [10] and UNR-PF [8], all use cloud computing as a data store and use web services to achieve service integration. These projects have also moved all the robotic functions from the robot platform onto the cloud platform, thereby obviating the need to have any services or data stored locally on the robot.

The challenge, however, is that multi-robot teams working in heterogeneous environments could find themselves being redundant and useless if connection to the network is terminated. The platform architecture proposed for the SCMR framework is a hybrid model, which consists of the platform architecture for the individual robots as well as the platform architecture for the robot leader. The proposed platform architecture is graphically illustrated in Figure 3.18.

The platform architecture for the individual robots leverages the complete capabilities and benefits that are brought about by cloud computing. The individual robots themselves have no applications, data or services stored locally and leverage the cloud for all of these services. The hardware abstraction remains unchanged and this consists of the robot drivers, camera and motion primitives. The robot leader platform architecture, however, is a bit different in the sense that it is a clone of the physical cloud infrastructure. The robot leader is equipped with the services, applications and data that are rendered in the cloud. In instances where individual robots in multi-robot environments find themselves without network access, they can leverage off this robot leaders platform architecture to complete a certain task.

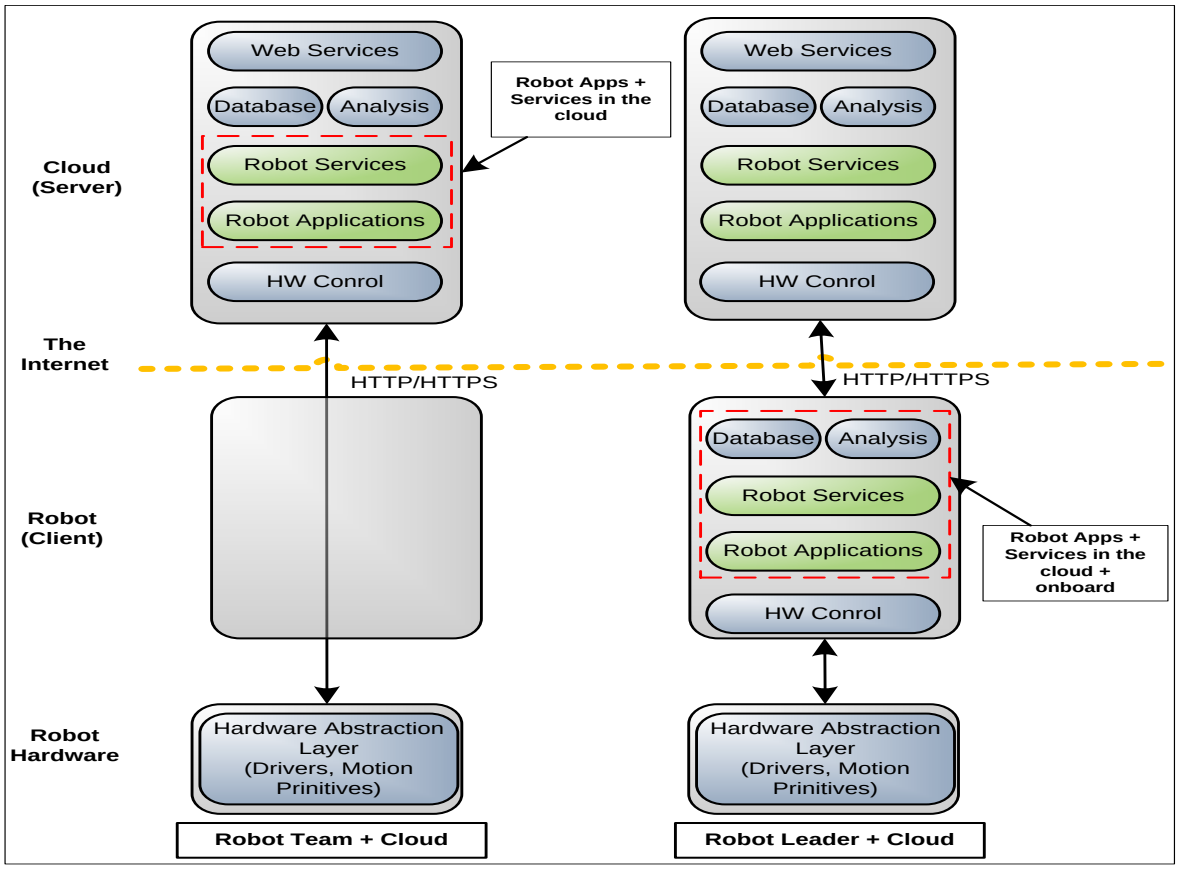


Figure 3.18: Platform architecture in the SCMR framework

3.3.6 Data architecture as SCMR paradigm

It is recognized that over time sizes of databases are growing substantially in view of the increasing amount of data being digitized for storage. In terms of cloud robotics the amount of information required to be stored in the cloud is also increasing [11]. An ODS is proposed as part of the data architecture for storing and gathering sensor data from the robots. For a multi-robotic team it is essential to ensure that the robots receive consistent data and that the data is highly available. Apart from the ODS, the other functionalities that make up the data architecture are the local data buffer, the filter and pre-process module, data access point (DAP) and the robot clients. The data architecture is illustrated in Figure 3.19.

All the sensor information retrieved from the robot clients is stored in the ODS. The data collected can also be used for reporting and analysis purposes. Using a publish/subscribe data publication model, the data from the ODS is published to different sources that would like

to subscribe to this data. Data that is requested from a robot client is handled through the twisted loop reactor, which obtains the data by invoking the filter and pre-process module. This module is used to access the local data buffer through a DAP and then return the requested data to the twisted loop reactor. The filter and pre-process module also o frequently requested data from the ODS and publishes this data to the local data buffer.

The concept of having a DAP over providing direct access is absolutely essential for several reasons, such as security, uniform interface, tracking, logging and error handling. The robot leader then subscribes to the data published by the local data buffer and has this data downloaded locally in a scheduled batch job. In cases where individual robots find themselves without network access, the data required is retrieved directly from robot leader. Using the gossip protocol, this data is in turn broadcast to chosen robot neighbours and this process is repeated several times, increasing the probability that the data will be relayed to all destination nodes.

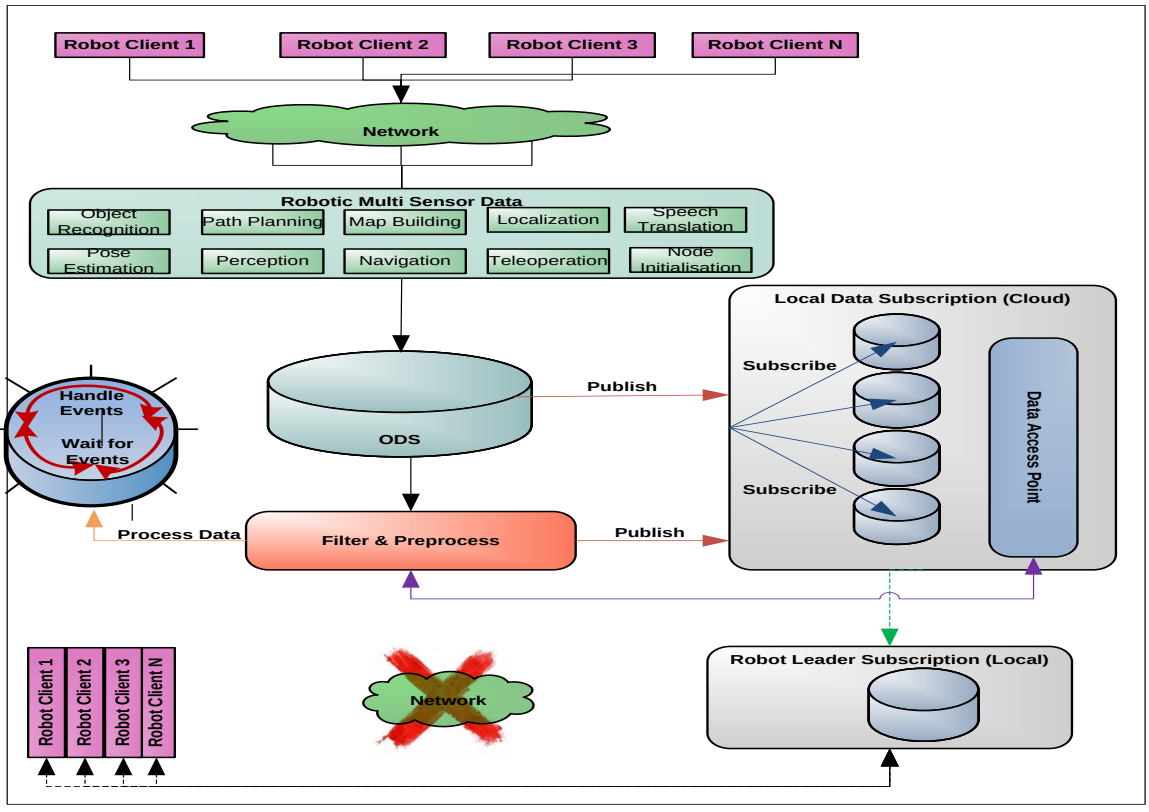


Figure 3.19: Data architecture in the SCMR framework

3.4 SCMR COMMUNICATION AND MANAGEMENT

The SCMR framework has its communication protocols divided into three parts. The first part is the internal communication protocol that occurs within the physical cloud infrastructure. This communication is handled by the ROS internal messaging protocol. The second part is the external communication protocol that defines the data transfer between the physical cloud infrastructure and the individual robots. This is defined as the R2C communication, as it is handled by the twisted-based network management protocol. The final communication protocol is the transfer of data and messages between the individual robots in the case of network disconnection. The transfer of data and messages between the robots will be handled by the gossip protocol. The sub-sections below discuss in detail the different communication protocols used in the SCMR framework.

3.4.1 Twisted framework

It is important for an SCMR framework to support a large number of concurrent TCP connection requests and it is therefore important to choose an appropriate network mechanism that will be fit for purpose. Taking into account the needs of the applications that make up the cloud multi-robotics framework, the twisted-based framework is chosen based on the fundamental architecture.

The twisted-based framework is implemented using Python and is a flexible asynchronous I/O network framework [74]. Twist is used for deploying asynchronous, event-driven and multi-thread network systems and consists of three basic objects: a reactor, protocols and a factory [76]. The primary function of the reactor is to execute the event loop and distribute events. The protocol handles every specific client-connecting request. The factory is responsible for instantiating protocols and storing the persistent configuration information. The entire procedure of the twisted-based framework is illustrated in Figure 3.20.

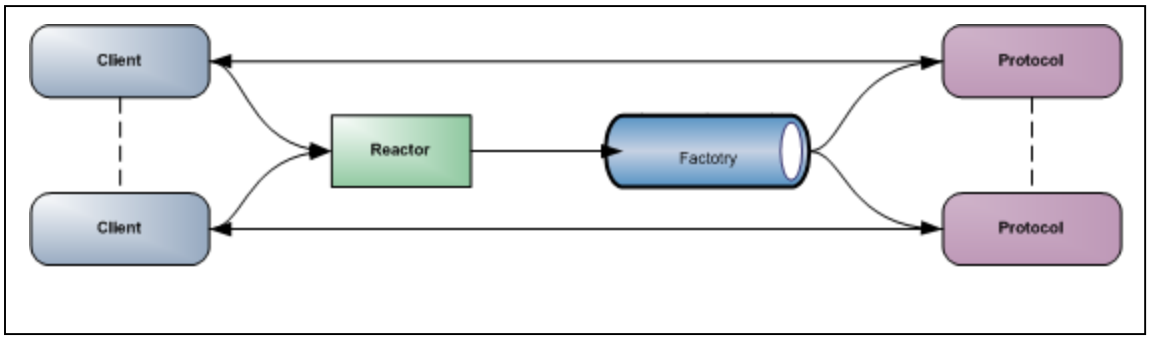


Figure 3.20: Twisted framework [74]

The twisted framework substantially simplifies the management of concurrent connections [76], which is important in a cloud multi-robot environment. The key reasons behind the robustness of the framework are attributed to the following three primary elements in the framework:

- **Reactor** - the reactor is the core of the event loop in the twisted framework. The reactor is the loop that is responsible for driving the applications used by the twisted framework [74]. The event loop is a programming construct that waits for and dispatches events or messages in a program. It works by calling some internal or external "event providers", which generally block until an event has arrived [76]. It then calls the relevant event handler. The reactor provides basic interfaces to a number of services, including network communications, threading and event dispatching.
- **Protocol** - the protocol defines the specifications for transmitting and receiving behaviours [74]. Functions regarding received data and sent data can be constructed following the predefined virtual function names. A protocol begins and ends its life with two predefined virtual functions: *connectionMade* and *connectionLost*, which are called whenever a connection is established or dropped [74].
- **Factory** - the factory is responsible for two main tasks: the creation of new protocols, and keeping global configuration and state. In addition to abstractions of low-level system calls, it also includes a large number of utility functions and classes, which facilitates the establishment of new types of servers [74]. Twisted also includes support for popular network protocols such SOCKETS, HTTP, HTTPS and SMTP, to name but a few.

3.4.2 Apache Hadoop MapReduce vs. Twisted

Apache Hadoop is an open-source software framework that is used for running applications on a large cluster built of specialized hardware and provides a reliable, scalable and distributed computing platform [77]. One of the main features of Hadoop is that it parallelizes data processing across many nodes in the cluster in order to speed up the computations of data [78]. One of the main functionalities in Apache Hadoop is the map/reduce framework [78]. Map/reduce is a computational paradigm that is used to divide a particular application into smaller fragments of work and each fragment of work is then executed and re-executed on any node in a particular environment.

Apache Hadoop has primarily been used in search and indexing of large volumes of text files; however; it has recently even been used in other areas such as machine learning, analytics, natural language search and image processing [10]. In terms of cloud robotics, projects such as the DAvinCi project have used map/reduce to implement the FastSLAM algorithm in order to show significant performance gains [10]. Other projects as well, such as the "Robot-Cloud" initiative, which is used to build low-cost heterogeneous robots, also make use of the map/reduce framework [6].

In a study conducted on the use of map/reduce in cloud robotics [79], the researchers indicated that map/reduce is actually a step backwards and not suitable for cloud robotics. In the study they indicated that map/reduce is not flexible enough for cloud robotics as a distributed computing and database, since robots need near real time response to solve problems in parallel [79]. They also indicate that in map/reduce, each of the N map instances produces M outputs files, each of which is destined for a different reduce instance. In instances where these files are required to be written to a local disk, the complexity increases exponentially.

In terms of processing algorithms, DeWitt and Stonebraker [79] also indicate that map/reduce is not efficient enough for algorithms such as SLAM and FastSLAM, which require computation on several robot clients in a cloud multi-robot environment. Finally DeWitt and Stonebraker [79] indicate that map/reduce requires a large number of disk seeks and access to the disks could eventually become a bottle-neck, slowing down the process. This is an aspect

that was not taken into consideration by the other cloud robotic projects, such as the DavinCi project that used map/reduce. In this study, however, the proposed SCMR framework makes use of the twisted-based network management protocol for the parallel management and communication management software platform.

3.4.3 Robot-to-cloud communication

In the proposed SCMR framework the robots will connect to the physical cloud infrastructure using the WebSockets protocol [32]. The protocol itself is based on the twisted framework [80]. The use of WebSockets will allow messages to be pushed to individual robots as opposed to a common server, which only pulls messages. ROS could also have been used for the R2C communication, as the ROS system provides task request and task responses among the nodes [30]. However, ROS does not provide support for queue management, especially asynchronous access for multiple tasks on a waiting list. It is therefore not adequate to support cloud multi-robotic teams that require executing real time tasks. The twisted socket communication is introduced as a complement to ROS and this is illustrated in Figure 3.21.

The host, which is the physical cloud in the proposed SCMR framework, and the robot clients are built on the twisted framework. The twisted program contains the main loop called a reactor and a call-back system. The system automatically launches a new thread for each client that attempts to connect to the network with an approved address and port. The Twisted loop reactor handles the individual threads through a WebSocket interface in the cloud, while the twisted factory is used to manage all the connections to the robot client reactor loops.

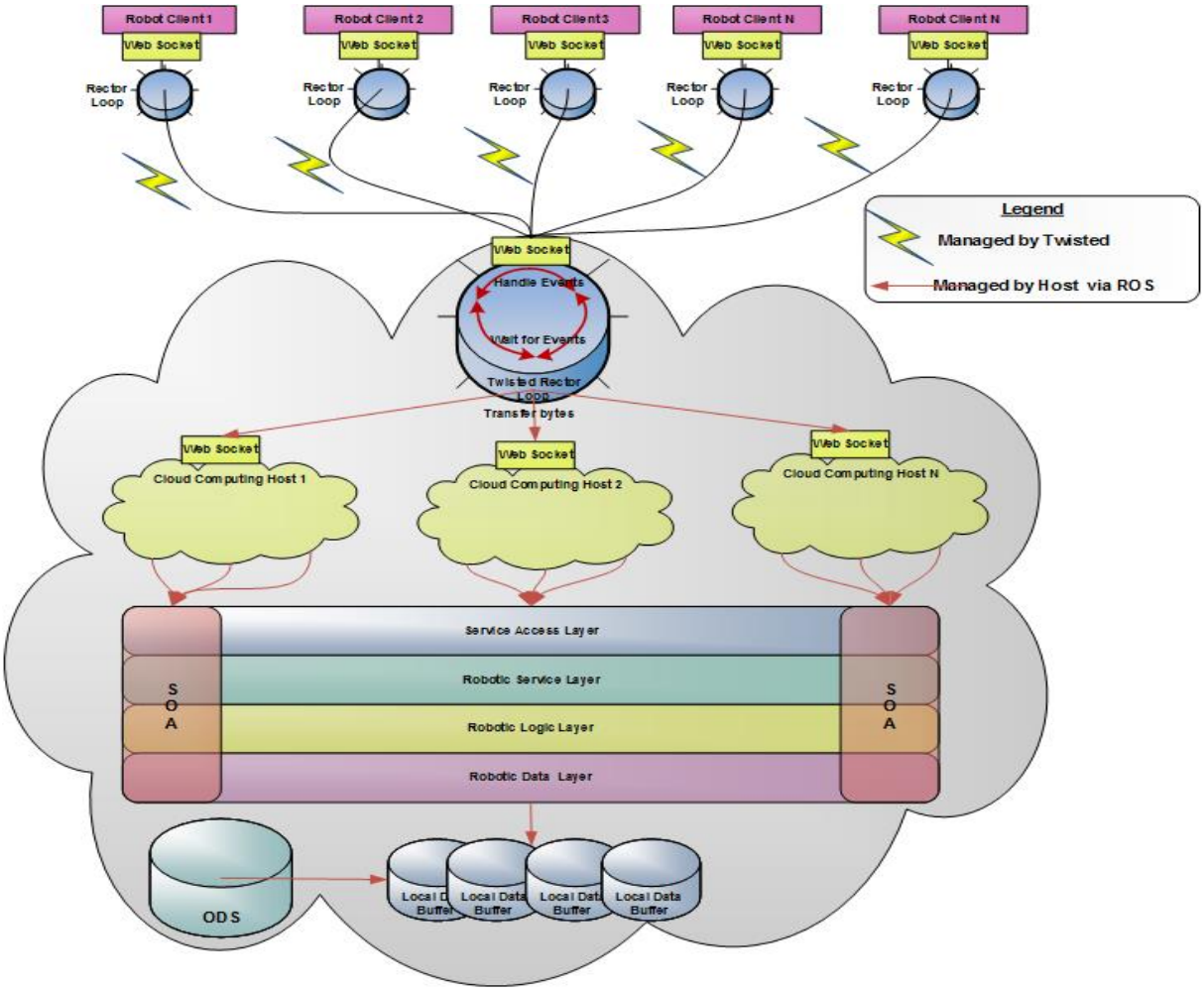


Figure 3.21: Robot-to-cloud communication in the SCMR framework

The principles of SOA are also introduced in the framework. All the robotic services available are loosely coupled and communicate with one another through standard interfaces and via standard message-exchanging protocol. The use of SOA for robotics has certain inherent advantages over the traditional methods of building a robot [81]. The main benefit is having a layer of common services with standard interfaces [81]. Each service can represent a component such as sensor, actuator or effector. The different layers that form SOA in the SCMR framework are the following [81]:

- Service access layer – This layer is used to authenticate service requests and determine which robotic services can be accesses by the different robotic clients.

- Robotic service layer – This layer is used to all common robotic services with standard interfaces
- Robotic logic layer – This layer is used to process all the robotic client requests such as object recognition, path planning and image processing. It provides an abstraction to the robotic services that are available.
- Robotic data layer – This layer is used to process data requests and provides an abstraction from executing SQL directly on the database.

3.4.4 Cloud communication

The communication protocol internal to the cloud in the proposed SCMR framework will be handled through the ROS messaging framework. The ROS messaging system is based on either a loosely coupled topic publish/subscribe or service-based model [30], as illustrated in Figure 3.22. In the publish/subscribe model, robots publish their messages on a named topic, then robots that want to receive these messages subscribe to the topic. A master node exposes the publishers to the subscribers through a service name. In the service-based model a robot pushes the messages onto another dedicated robot that has requested that data.

In the proposed SCMR framework the ROS messaging framework is used to send sensor data received from the robot clients to the ODS. A request for a particular service in the cloud multi-robotic framework is identified by a particular ROS message topic. The request is verified by checking if the service flag is enabled and if it is enabled, it will then look for the service that is requested. At this point the CCH in the framework will take over from the ROS master node and deal with the requested service through the twisted framework.

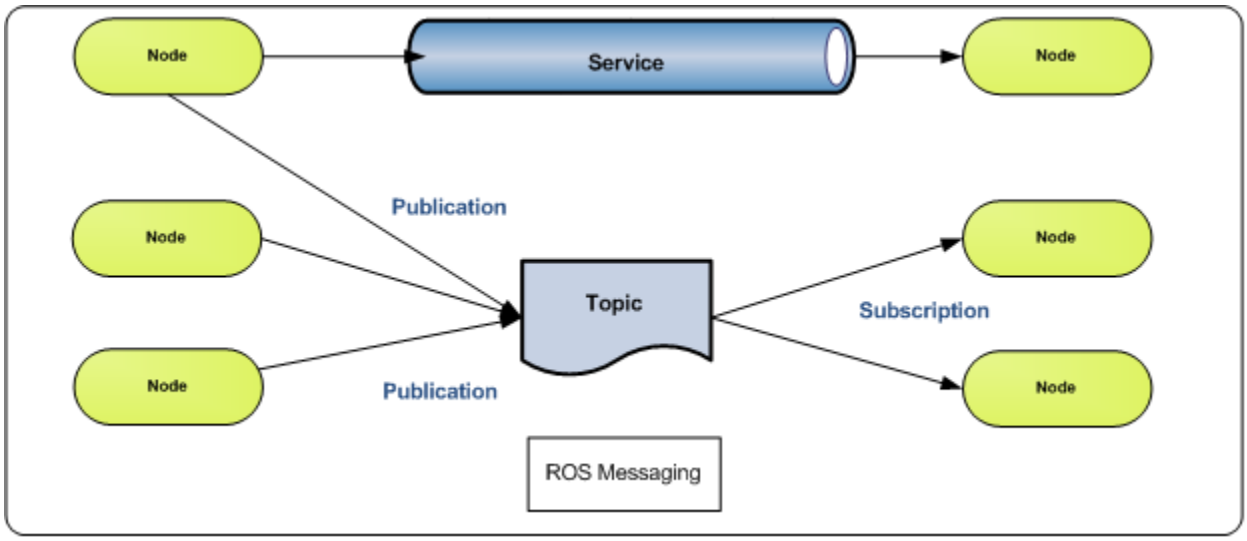


Figure 3.22: ROS messaging framework used in the SCMR framework for cloud communicaiton

3.4.5 Robot-to-robot communication

In a multi-robot team there are many scenarios in which robots are required to communicate with one another. The most obvious communication between robots in a multi-robot environment is to co-ordinate actions between them in order to accomplish a particular task. Another scenario is the ability of robots to share information with others, thus performing knowledge exchange between the robots. An example of this could be the need to share the map of a room that has not been explored by other robots. Finally, robots can share sensory evidence to enhance, de-noisify or reveal things about the world [11].

The communication protocol for R2R communication in the proposed SCMR framework is the gossip protocol [75]. The gossip protocol is a type of communication protocol used mostly in large-scale distributed systems. Gossip protocols have become increasingly popular in distributed application because of their simplicity, scalability and high reliability, even in constantly changing environments [75], which makes them suitable for highly dynamic mobile robotic networks. Gossip protocol involves periodic message exchanges between node pairs; this eventually results in information being spread throughout the system, which is similar to human gossiping. It is sometimes referred to as epidemic protocols because the information spread is similar to the spread of a virus.

The gossip protocols are also very simple to implement and require minimal additional computation and memory resources [75]. Figure 3.23 illustrates the gossip protocol principle and highlights the manner in which information is spread like an epidemic from one node to another until all the nodes have been infected with the information. The flow of information can be broken down as follows:

- Each node has some data associated with it and periodically gossips about that data with another node.
- Node $n1$ randomly selects a node $n2$ from a list of nodes known to it.
- $n1$ sends a message to $n2$ containing the data from $n1$.
- $n2$ sends back a response containing its data.
- $n1$ and $n2$ update their data set by merging it with the received data.
- The process is repeated for $n3$ and $n4$ until these nodes have also received the data.

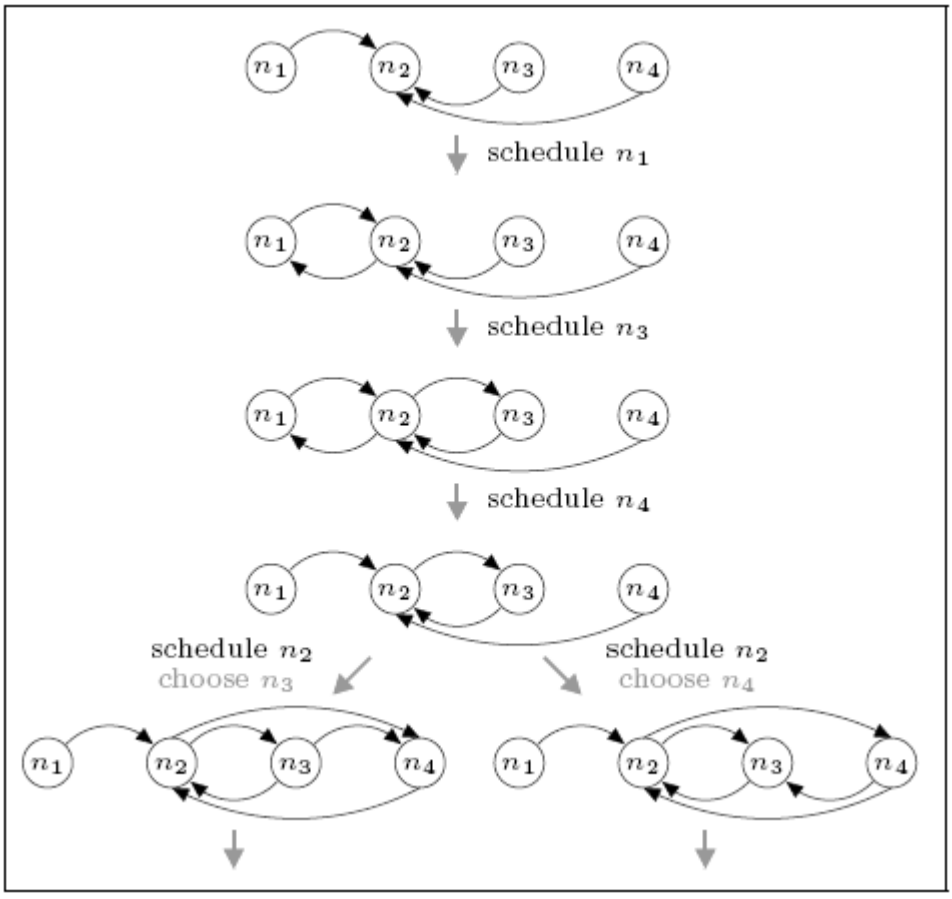


Figure 3.23: Gossip protocol used in the SCMR framework for R2R communicaiton

3.4.6 Quality of Service

A set of well-defined and accurate QoS standards is essential in order to validate a particular framework. In the case of the proposed SCMR framework this is no different. Generally, network-based applications use bandwidth usage as a primary factor in defining the QoS. This is attributed to the fact that most network-based applications are sensitive to bandwidth usage, as this affects their response. However, in cloud robotics, the bandwidth usage cannot be used as the only QoS factor. Instead, other factors such as processing capabilities or storage capabilities can be extended as factors to define QoS. In the proposed SCMR framework the following QoSs are selected as the primary criteria to assess the framework:

3.4.6.1 QoS factor - time to response

The time-to-response (ToR) is the time a robot client needs to receive a response after a request has been sent. It is represented by the following simple polynomial:

$$ToR = T_{Data\ Received} - T_{Request\ Sent} \tag{3.12}$$

3.4.6.2 QoS factor II - reliability of response

The reliability of response (RoR) is a measure of confidence that the retrieval data is free from errors. Its value is given in percent and calculated as follows:

$$RoR = \frac{\#\ Succeeded\ Requests}{\#\ Total\ Requests} \tag{3.13}$$

3.4.6.3 QoS factors

In the proposed SCMR framework multiple parallel client robot requests could be sent to the cloud and if these requests are not handled satisfactorily, this would result in a breach of QoS. It is therefore important to optimize the QoS in terms of fast response, reliable response and fairness of allocation for admission control and resource scheduling.

In any cloud robotic environment, as the data and the number of robotic services increases, the efficiency, reliability and fairness of the data retrieval become increasingly challenging. The resource scheduling therefore also becomes a challenge and it is important to formulate the scheduling problem with QoS constraints. In formulating the scheduling problem, the aim is

to assign each task to data retrieval while minimizing the total cost, where the cost is the time required to complete all tasks. Therefore, for n clients that could potentially share m resources, each task C_i consists of k_i parallel and dependent requests. Each resource R_j has a fixed price p_j, which is aligned to its expense and capacity. The number of requests a_{ij}then reflects the client C_i task allocated to a resource R_j. Therefore, the total cost of task C_i is defined as $u_i\,(a_i) = \omega_t \max t_{ij \epsilon ti} t_{ij} + \omega_e \sum_j e_{ij}$ [76]. The utility of the task C_i can therefore be represented mathematically by Equation 3.14 as follows:

$$u_i\,(a_i) = \frac{1}{\omega_t \max t_{ij \epsilon ti} t_{ij} + \omega_e \sum_j e_{ij}} \tag{3.14}$$

where t_{ij} represent the response time, ω_t represents the completion time of a task and ω_e represents the completion expense [76].

The basic data retrieval operation policy of the request negotiator implemented in the proposed SCMR framework is presented in Table 3.6. The main objective of the policy is to provide an index for ranking robot requests. The elements that are taken into consideration for the ranking of robot requests are the CPU, bandwidth and willingness to pay for each request. If a request arrives from a robot client the request negotiator will first filter the incoming data from a rank list, determine which records do not meet the minimum requirements and create a new list of candidate requests. The requests in the rank index are then responded to in parallel.

Table 3.6: Psuedo-code for the request negotiator scheduling algorithm [76]

Input: Willingness payment of request p_i and request ID i
Output: current_priority_list
1 BEGIN 2 define update_priority_list(p_i) 3 current_priority_list.append(p_i) 4 define is_lowest_priority(p_i) 5 current_priority_list.sort(p_i) 6 while i ≤ $n_{threshold}$ 7 update_priority_list(p_i) 8 is_lowest_priority(p_i) 9 return current_priority_list 10 END

3.5 SCMR PERFORMANCE ENERGY UTILIZATION

The biggest benefit that the phenomenon of cloud robotics brings to multi-robots working in heterogeneous environments is the ability to have them offload computationally intensive tasks to the cloud to have them executed. The SCMR framework provides a survival framework for multi-robot teams working in heterogeneous environments that face intermittent disconnections to the network. The framework allows individual robots to have tasks executed locally by a robot leader in the event of cloud disconnection, which would have not been possible previously unless the individual robots were burdened with powerful on-board computers. In a large environment this burden is unnecessary and expensive.

The challenge, however, would be to determine whether a task should be offloaded to the cloud for execution or executed by the robot leader. In order to overcome this challenge a unified framework is required that would be able to handle a list of complex offloading strategies. In determining the optimal offloading strategy, the most important factors that need to be taken into consideration are the amount of data exchanged and the delay deadline to complete a particular task, such as path navigation and image processing [11]. The offloading strategy also needs to determine whether it is more beneficial to have a task executed by a multi-team robot, given the presence of a cloud.

The concept of cloud robotics has led the way for more intelligent, efficient and yet cheaper robotic networks [8]. The challenge in this drive towards cheaper robots with “remote brains” in the cloud has led to robots being equipped with a limited supply of resources in computation, energy, bandwidth and storage [11]. In developing countries such as South Africa, the mining sectors, which are often heterogeneous environments, require these networked robots to assist in events such as disaster recovery. This could potentially require robots to be underground for long periods of time and a limited battery life could prove to be fatal and result in the multi-robotic team not completing a particular task.

The aim therefore is to develop an optimization framework to determine the optimal task execution strategies in the SCMR framework. The primary goal is to minimize the amount of energy consumed by a robot, under the constraint that a particular task needs to be completed

before a particular deadline. The reduction in the energy consumed by the robot is inversely proportional to the battery life of the robot and could result in the life or death of a miner stuck in an underground mine that requires urgent assistance from a multi-robotic team in the case of an explosion or an unforeseen event in the mine. The important trade-off is the energy consumed by the robot for executing a particular task using its on-board processing unit and energy used by the robot to send the information to the cloud or robot leader to be processed.

In the SCMR framework a particular task, such as path navigation or image processing, can be executed by the individual robots or in the cloud (i.e. the physical cloud or the virtual ad hoc cloud made up by the robot leader). In the stand-alone robot execution strategy the robot's computation energy can be minimized by optimally scheduling the clock frequency of the robot via dynamic voltage scaling (DVS) technology [82]. In the cloud execution strategy the amount of energy required by the robot to transmit the data to the cloud can be minimized by optimally scheduling the transmission data rate in a stochastic wireless channel [82]. In the cloud multi-robotic framework, the cloud execution strategy assumes a polynomial energy consumption model that transmits s bits of data across a wireless channel with a fading coefficient g that is proportional to s^n/g where n depends on the coding scheme. In the stand-alone execution model L bits of data are consumed to execute a particular robot task within a delay deadline of T and a clock frequency of f. Figure 3.24 illustrates the task execution in the two alternative modes discussed.

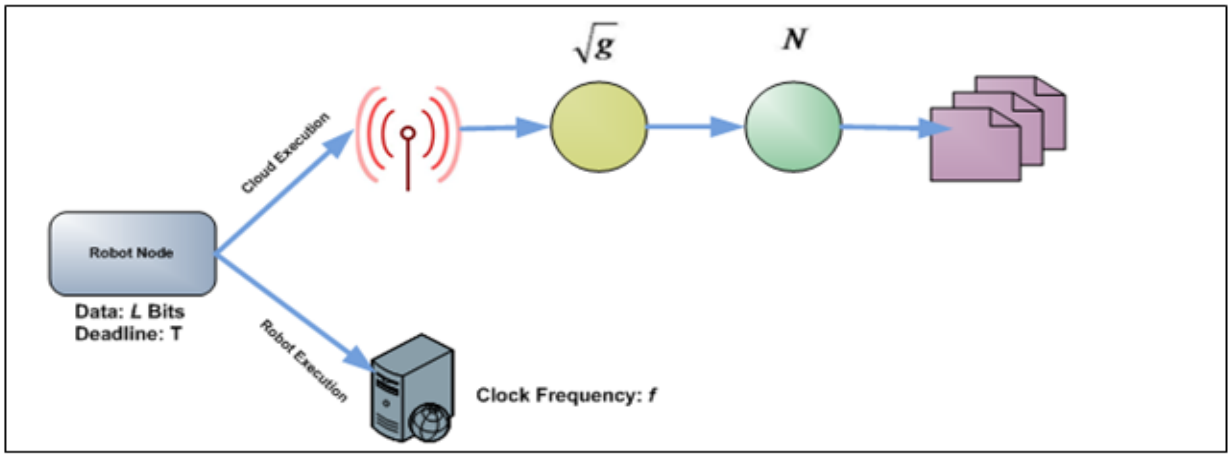


Figure 3.24: SCMR framework task execution modes [85]

3.5.1 Computation offloading in cloud robotics

In the context of cloud robotics, the tradeoff between the energy consumed by the individual robot and the energy consumed by the robot offloading the information to the cloud server is extremely important. It is therefore important to consider the energy consumed by individual robots (E_r) versus the energy required by a robot to receive and offload data (E_c). Taking this into consideration, it is possible to come up with a simple polynomial to reflect the total energy (T_e) consumed by a robot as [83]:

$$T_e = E_{proc} + E_{idle} + E_{trans} \tag{3.15}$$

where E_{proc} is represented by a polynomial expression as the total energy consumed by the robot while processing an instruction, E_{idle} is represented as the total energy consumed by the robot while it is idle and E_{trans} is the total energy consumed by the robot for transferring and receiving data.

3.5.2 Robotic execution energy model

The energy consumption required to complete a task by a robot is generally determined by the CPU workload. The workload, symbolized as W, is the number of CPU cycles required by the robot to execute a particular task. The workload (W) is also dependent on the size of the data (i.e. the number of bits) and the algorithm used to execute the particulate task (i.e. SLAM, FastSLAM). The use of (DVS in stand-alone robots for task execution can minimize the computation energy required by optimally configuring the clock frequency of the chip [82]. In CMOS circuits, the energy per operation denoted as ε_{op} is proportional to V^2, where V is the supply voltage to the chip [84]. In their study of CMOS circuits [84], it was observed that the clock frequency of the chip f is approximately linear proportional to the voltage supply, V, when the circuits were operated at a low voltage. Therefore, the energy per operation for the stand-alone robot can be calculated as:

$$\varepsilon_{op} = kf^2 \tag{3.16}$$

where k is the effective switched capacitance, depending on the chip architecture. As noted above in [84], the CPU on the robot can reduce its energy consumption by running more slowly (i.e. decrease the time it takes to complete a CPU cycle). This, however, is not a practical

solution, as a robot has to complete a particular task within a delay deadline (T), implying that the clock frequency cannot be low. Therefore, the robotic stand-alone execution that the clock frequency (f) is configured such that the energy consumed by the robot is minimized while still meeting the delay deadline of a particular task. This mathematical problem can be formulated as follows [82]:

$$\varepsilon_r^* = \min_{\varphi \epsilon \gamma} \{\varepsilon_{r(L,T,\varphi)}\} \tag{3.17}$$

where $\varphi = \{f1, f2, \ldots fw\}$ is the clock frequency vector that meets the delay deadline, γ is the set of all feasible clock frequency vectors and $\varepsilon_r(L, T, \varphi)$ is the total energy consumed by the standalone robot required to execute a particular task.

3.5.3 Cloud execution energy model

In order for a robotic task to be executed in the cloud, the energy required by the robot to send this request to the cloud is dependent on the size of the data being transmitted and the wireless channel mode. A robot application can be represented by $R(L, T)$, where L bits of data need to be transmitted to the cloud for a particular task to be completed before the delay deadline T. In determining the optimal cloud execution energy model for the cloud multi-robotic framework the energy transmission model adopted for this research is an empirical model similar to [85] and [86]. As illustrated in Figure 3.24 the task execution in the cloud is signified by a channel gain of g and a noise power of n. The energy consumed by the robot in transmitting s bits of data of a wireless fading channel within a time slot is represented mathematically as follows:

$$\varepsilon_{t(s,g,n)} = \lambda \frac{s^n}{g} \tag{3.18}$$

where λ represents the energy coefficient and n represents the monomial order. In attempting to reduce the energy consumed by the robot when transmitting the data to the cloud for task execution, one possibility is to reduce the number of bits transferred in a given time slot optimally, in response to a stochastic channel [85]. The challenge, however, is that by delaying the number of bits transferred to the cloud, the time period for executing a particular task will increase, which could mean that a task is executed outside the delay deadline time. Therefore, a middle ground has to be formulated that optimally transmits the data to the cloud by the robot, which minimizes the energy consumed while still executing the task before the delay

deadline. The optimal transmission energy for cloud execution can therefore be represented mathematically as follows:

$$\varepsilon_c^* = \min_{\theta \epsilon \gamma} \mathrm{E}\{\varepsilon_{c(L,T,\theta)}\} \quad (3.19)$$

where $\theta = \{s1, s2, \ldots sT\}$ represents the number of bits transferred in a time slot i that meets a delay deadline T, γ is the set of all feasible data schedules and $\varepsilon_{c(L,T,\theta)}$ represents the transmission of energy.

3.5.4 Optimal task execution policy

The optimal task execution policy context of cloud robotics is to choose here a particular task should be executed, the objective being to minimize the total energy consumed by the robot. However, in certain instances, such as cloud disconnections, the task will be executed by the robot leader, which is a clone of the physical cloud. In the proposed SCMR framework the optimal task execution policy is executed first before processing a robotic task, enabling the robot to know where the task should be executed. Therefore the optimal policy is determined by the following rule:

$$\begin{cases} Robotic\ Execution\ if\ \ E_r \leq E_c\ \ OR\ \ CLOUD\_DISCONNECTION \\ Cloud\ Execution\ \ \ if\ \ E_r > E_c \end{cases}$$

where a particular task is executed in the cloud (E_c) if the energy consumed by the cloud is less than or equal to the energy consumed by the robot. In all other instances, including cloud disconnections, the task is executed locally by the robot or robot leader (E_r).

3.6 CHAPTER SUMMARY

This chapter identified the research design and methodology used to evaluate the proposed SCMR framework. It also identified that in most current cloud robotic frameworks the issue of cloud disconnection is an ongoing challenge and introduced the proposed SCMR framework to overcome this challenge. The concept of a robot leader and the role it plays in instances of cloud disconnection was also described, together with the architecture, communication management and energy utilization of the proposed SCMR framework.

The importance of a robot being able to visually recognize and understand an object and thereafter have the intelligence to know what to do with the identified object was also described. This then led to the design of a drivable road detection system using the proposed SCMR framework. The design of the drivable road detection system makes use of key image-processing methods such as edge detection, thresholding and removal of sparse pixels. The resulting output of the system is a binary image indicating the drivable and non-drivable areas of the subjected picture frame.

Finally, the chapter concluded with illustrating the energy utilization on the SCMR framework. A common challenge faced by robots today is the limited energy utilization brought about by the amount of energy required to perform on-board processing in executing robotic tasks [11]. This chapter introduced an optimal task execution policy to overcome this challenge, which determined if a task should be executed in the cloud or locally on the robot, with the goal of energy conservation.

In chapter 4 the implementation of the proposed framework will be discussed, together with the tools used to build the prototype. Furthermore, the results obtained from the different experimental tests conducted on the proposed SCMR framework will be presented.

CHAPTER 4: IMPLEMENTATION AND EXPERIMENTAL EVALUATIONS OF SCMR FRAMEWORK ON DRIVABILITY ANALYSIS

4.1 INTRODUCTION

To validate the usefulness of the proposed SCMR framework, a prototype of the framework was developed and seven experiments were conducted. This chapter begins with the details of the prototype implementation, describing the software and hardware used. Thereafter, a more detailed explanation of the computation used to compute the energy utilization and QoS on the proposed SCMR framework is presented. The detailed results from each of the experiments conducted on the framework are then presented, followed by the chapter summary. A diagrammatic view of this chapter is presented in Figure 4.1 below.

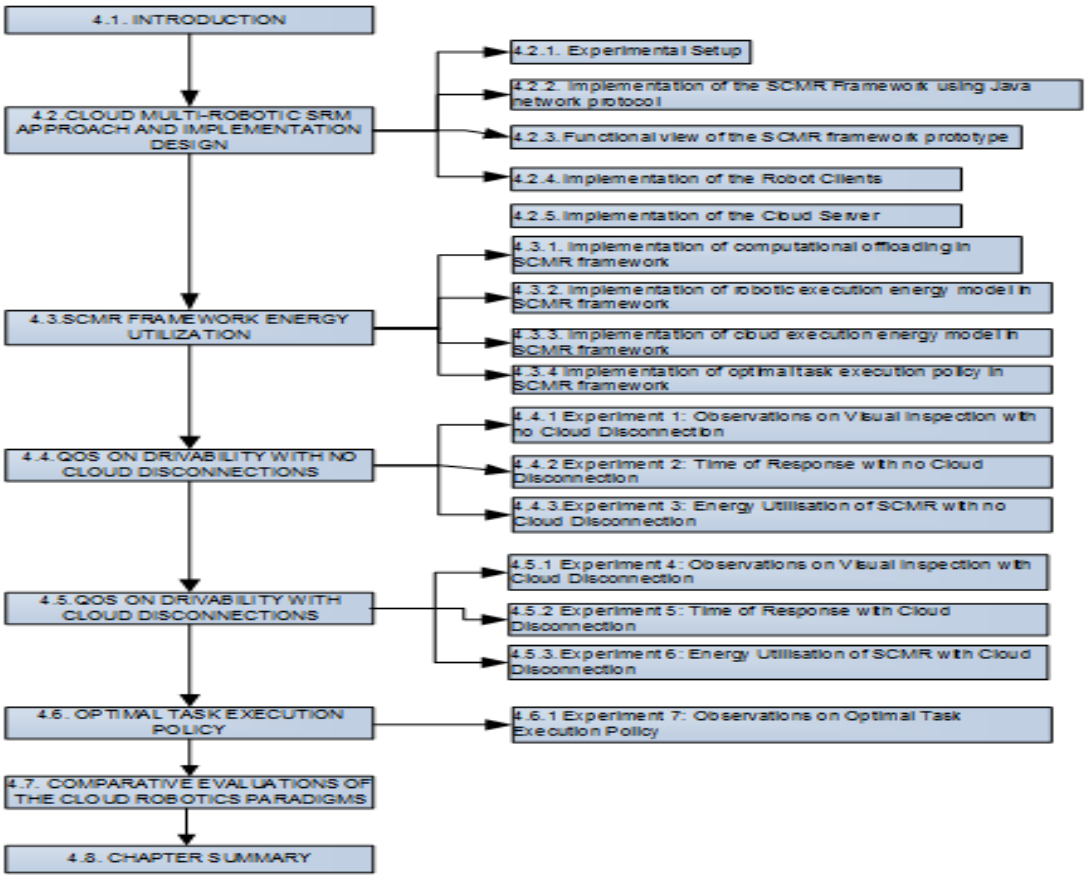


Figure 4.1: Graphical representation of Chapter 4

4.2 CLOUD MULTI-ROBOTIC SRM APPROACH AND IMPLEMENTATION DESIGN

4.2.1 Experimental Setup

In order to prove the concepts discussed in the study, a prototype of the SCMR framework was implemented using a client/server application built in Java. To make the SCMR prototype more adaptable, the following decisions were made on the implementation of the framework:

Hardware: A generic computer including an Intel processor and motherboard was used. The main components of the computer include:

- Intel® Core i7-3770 CPU @ 3.40 GHz – 3.90 GHz
- Four physical core processors
- 8.00 GB Random Access Memory
- 500.00 GB Local Hard Drive
- 120.00 GB Solid State Drive for Storing Operating System
- ATX 2.2 Power Supply
- 2 X 3.0 USB Ports
- Intel 4000 Graphics Card

Software: The SCMR framework was implemented on a 64-bit, Windows 7 Professional edition operating system.

Programming Language: The programming language used to develop a prototype in any study is an important aspect to consider. Much more flexibility can be added in the development of a prototype if it is supported by a language that has a sufficient number of skilled personnel and is used throughout the world. Therefore Java was the programming language used to develop the SCMR prototype. The prototype was built upon the Java Runtime Environment (JRE) and Java Development Kit (JDK) version 7 using the Eclipse Kepler IDE.

Image acquisition: The selection of the image frames to be used on the SCMR prototype is an important aspect in testing the robustness and accuracy of the proposed framework. It was important to ensure that the images were acquired from a wide range of heterogeneous

environments, since cloud-enabled multi-robotic teams could be found working in many of these environments and the ability to detect a drivable road region was essential to enable the robot to complete a specific task. Images for this study were therefore collected from other research studies in the field of drivable road region detection [24, 25], including images captured locally using a camera. The images used in the study ranged from simple drivable road images with no obstacles to off-road images in desert terrains, to underground mines and hallways.

4.2.2 Implementation of the SCMR framework using Java network protocol

The client/server application was developed using socket programming in Java, with communication between the client socket and server socket taking place through the stream communication protocol using TCP. The TCP protocol was chosen over the UDP protocol because of its reliability in ensuring that a given stream of data is delivered to the server through the connection socket. In the SCMR prototype each robot client initiates a TCP connection to the robot cloud server. When creating the socket connection, the robot client specifies the robot cloud server's IP address and port. The robot cloud server port used in this study was 2222. The robot cloud server used this port to listen for robot client connections. Upon receiving a robot client connection request, the robot cloud server creates a new socket for the client and binds a port number to it. A new port is created on the server because the initial port is used to listening for other robot client connection requests. The new port number is sent to the robot client, establishing a communication pipe for data to be streamed between the robot client and the robot cloud server. This process is commonly known as the "three-way handshake".

In the same way the robot cloud server accepts connection requests from other robot clients and creates a new port for each new connection. This process is completely transparent to both the robot clients and the robot cloud server. In order to ensure that the SCMR framework was scalable and robust enough to handle multiple robot client connection requests, a new thread was spawned on the server every time a new connection was accepted. Each robot client was also implemented using Java threads. Each robot client consisted of two threads – one thread to communicate with the cloud server and another thread to simultaneously read the input

stream of data to be sent to the server. In the next section the experimental results on the SCMR prototype are presented. Experimental tests were conducted on the framework in instances of cloud connection and cloud disconnection.

4.2.3 Functional view of the SCMR framework prototype

The proposed SCMR framework prototype is implemented to remove the burden of onboard image processing from the individual robots and move that processing overhead onto the cloud server. The prototype has each robot client containing an inbox and outbox image repository. The *outbox* repository is used to store the images that require SRM processing on the cloud server to detect drivable and non-drivable road regions. The *inbox* repository is used to store the images that are received from the cloud server after performing SRM and detecting which road regions are drivable and which are not. Figure 4.2 provides a functional overview of the proposed SCMR.

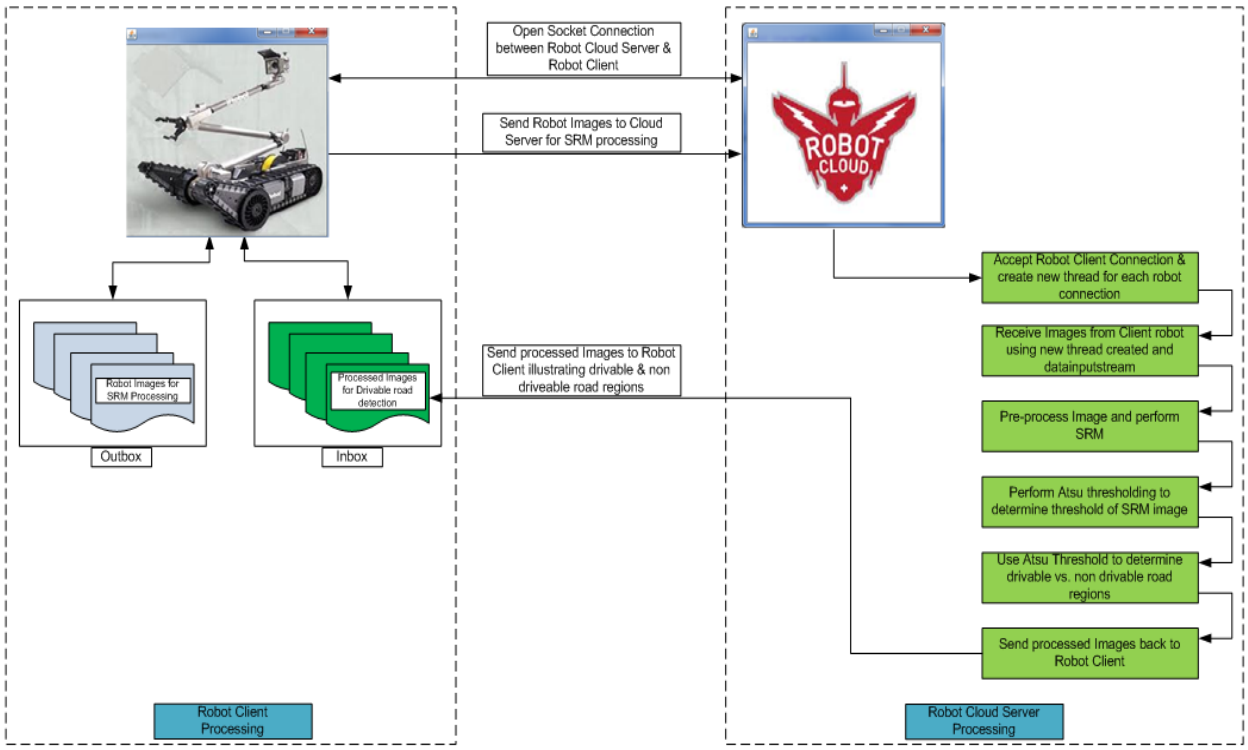


Figure 4.2: Functional view of the SCMR framework prototype

4.2.4 Implementation of the robot client

Each robot client initiates a call to the robot cloud server using a socket connection on a particular port that the server is listening to for connections. In the prototype the port used is *2222*. For every successful connection to the robot cloud server, a new thread is created on the cloud server to represent the robot client. Each thread (robot connection) on the cloud server is handled through the ***synchronized(this)*** method to ensure the threads are queued sequentially in FIFO (first in first out) and do not cause any deadlocks.

Using the client socket connection established between the robot client and the robot cloud server, the robot client applications streams the images in bytes to the robot cloud server and keeps the connection open to receive feedback from the cloud server. Figure 4.3 shows the robot client GUI used in the prototype and lists the actions that are performed when initiating a call to the robot cloud server.

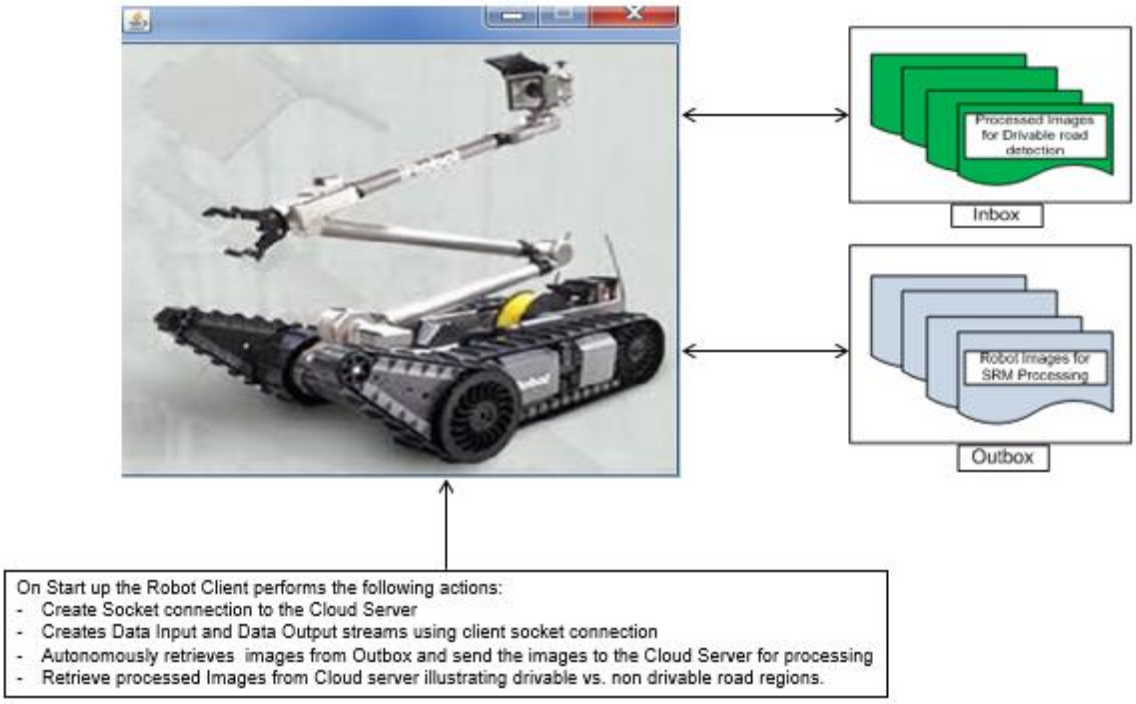


Figure 4.3: Robot client GUI

The robot client implementation makes use of a robot client ***helper class*** that is created to assist the robot clients with connections, disconnections and streaming of images to and

from the cloud server. Figure 4.4 is a code snippet of the helper class showing establishment of a socket connection with the cloud server (refer to Appendix A).

```
/*
 * Method to establish socket connection to the robot server from the robot client.
 */
public static Socket openRobotClientSocketConnection(String host, int port){
    /*
     * Open a socket on a given host and port.
     */
    try {
        clientSocket = new Socket(host, port);
        } catch (UnknownHostException e) {
          System.err.println("Don't know about host " + host);
        } catch (IOException e) {
          System.err.println("Couldn't get I/O for the connection to the host "+ host);
        }
    return clientSocket;
}
```

Figure 4.4: Robot client helper class: Socket connection

After the socket connection of the cloud server has been established, the next step in the robot client implementation is to instantiate the new ***DataInputStream***, ***DataOutputStream***, ***PrintOutputStream*** streams using the ***clientSocket*** object that was created when establishing the connection to the Robot Cloud Server. These streams are used to pass the data to and from the cloud server through the robot client. Figure 4.5 is a code snippet of the helper class showing the instantiation of these streams in the robot client implementation. (Refer to Appendix A for complete Robot Helper code.)

```
/*
 * Method to send file to robot server
 */
public static void sendFileToRobotServer (int intQ){

    /*
     * Open Robot Client Print Output Stream
     */
    try {
            clientSocket      = RobotHelper.openRobotClientSocketConnection(SERVER_HOST, SERVER_PORT);
            datainputstream   = RobotHelper.openRobotClientDataInputStream(clientSocket);
            dataoutputstream  = RobotHelper.openRobotClientDataOutputStream(clientSocket);
            printoutputstream = RobotHelper.openRobotClientOutputPrintStream(clientSocket);
```

Figure 4.5: Robot client helper class: Stream instantiation

Finally the files from the robot client's inbox repository are transported to the cloud server as bytes of data in an array size of 1024 for image processing, using the ***dataoutputstream*** object that was previously instantiated. Figure 4.6 is a code snippet of the helper class showing the transportation of the image locally from the robot to the cloud server for processing. (Refer to Appendix A for complete Robot Helper code.)

```
for (int k=0; k<Files.length; k ++){

    int filesize = (int) Files[k].length();
    byte [] buffer = new byte [filesize];

    FileInputStream fis = new FileInputStream(Files[k].toString());
    BufferedInputStream bis = new BufferedInputStream(fis);

    //Sending file name and file size to the server
    bis.read(buffer, 0, buffer.length); //This line is important

    System.out.println("Sending the following file to server = "+ Files[k].getName());
    System.out.println("File Size " + filesize + "B");

    dataoutputstream.write(buffer, 0, buffer.length);
    dataoutputstream.flush();

    System.out.println("File sent successfully to server");
}
```

Figure 4.6: Robot client helper class: Transporting image to cloud server

4.2.5 Implementation of the cloud server

In the proposed SCMR framework prototype the robot cloud server receives the streamed bytes of data from the robot client and performs the following actions in order:

- Create the images locally on the cloud server. (*This is done because they are sent to the cloud server in bytes of data.*)
- Pre-process the image and then perform SRM to merge similar regions (SRM image created at this point).
- Use the Otsu threshold algorithm to determine the threshold value for the SRM image that was created (Otsu black and white and Otsu histogram images created at this point).
- Using the Otsu threshold value received, the Otsu black and white image is used to reflect the drivable and non-drivable road regions. All the image pixels above the threshold are reflected in green (drivable road region) and all image pixels below the threshold are reflected in red (non-drivable road region).
- The RG (red green) image is then streamed back to the client robot using the client socket connection.
- The robot client receives the RG image in its inbox repository and using the processed image it illustrates to the robot which region it should drive on and which region it should not drive on.

Figure 4.7 shows the robot cloud server GUI used as part of the prototype and the actions performed by the cloud server.

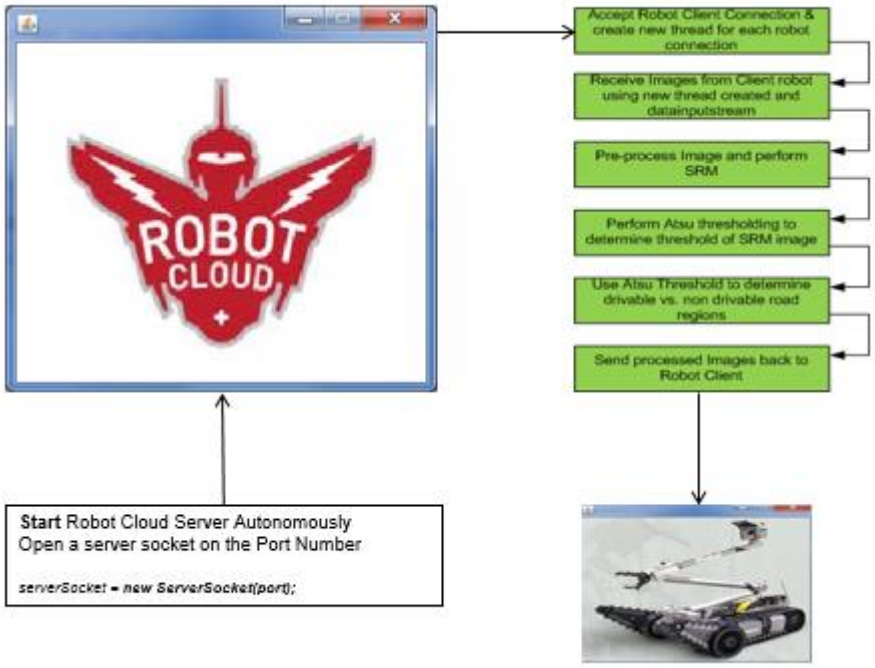


Figure 4.7: Robot cloud server GUI

The robot cloud server using the Java network protocol *persistently listens* on port 2222 for new robot client socket connections. Figure 4.8 is a code snippet showing the use of the Java network protocol to listen for robot client connections to create a socket connection in the cloud server. (Refer to Appendix B for complete Robot Helper code.)

```
/*
 * Open a server socket on the portNumber (default 2222). Note that we can
 * not choose a port less than 1023 if we are not privileged users (root).
 */
try {
  serverSocket = new ServerSocket(port);
  System.out.println("Robot Server started on port: "+port);
} catch (IOException e) {
  System.err.println(e);
}
```

Figure 4.8: Cloud server helper class: Server socket establishment

Upon establishing the server socket connection the cloud server creates a new **client socket and client thread** on the server for each and every robot client connecting to the cloud server. This is done to ensure that the socket for establishing robot client connections is always kept open. The code snippet below shows the creation of a new socket connection and client thread

for every new connection to the cloud server. (Refer to Appendix B for complete Robot Helper code.)

```
/*
 * Create a client socket for each connection and pass it to a new client
 * thread.
 */
while (true) {
  try {
    clientSocket = serverSocket.accept();
    int i = 0;
    for (i = 0; i < maxClientsCount; i++) {
      if (threads[i] == null) {
        (threads[i] = new clientThread(clientSocket, threads)).start();
        break;
      }
```

Figure 4.9: Cloud server helper class: Server socket establishment

A common challenge with multiple connections to a single cloud server is **deadlocks**. This occurs when a single thread waits forever for another thread to leave its critical section in a program. To overcome this challenge the prototype made use of the **synchronized(this){}** statements, which are mutually exclusive. This means that when one thread enters the **synchronized(this){}** statement it first verifies that another thread is not being executed in the **synchronized(this){}**statement block. In the event that it does find that another thread is being processed the **synchronized(this){}**statement then places the new threads on hold until the existing thread completes its execution. Figure 4.10 shows a code snippet from the cloud server helper class that makes use of the **synchronized(this){}**statement to execute critical sections in the program. (Refer to Appendix B for complete Robot Helper code.)

```
/* Welcome the new client. */
synchronized (this) {
  for (int i = 0; i < maxClientsCount; i++) {
    if (threads[i] != null && threads[i] == this) {
      clientName = "@" + name;
      break;
    }
  }
```

Figure 4.10: Cloud server helper class: synchronized(this){}statement

As the cloud server starts to receive the stream of data being sent from the individual robot clients, the image to be processed on the cloud server is recreated on the cloud server and stored in the cloud server's inbox directory. Upon receiving the image, the processing done to detect

drivable and non-drivable regions commences using image segmentation, Otsu, edge detection and image binarization. The code for this processing is illustrated in the appendix section of this study. The recreation of the image on the cloud server is illustrated in Figure 4.11 below. (Refer to Appendix B for complete Robot Helper code.)

```
for (int m=0; m<fileSize; m++){

    filelen = (int) sizes.get(m);

    output = new FileOutputStream(cloudServerRobotImages + files.get(m));
    dos=new DataOutputStream(output);
    bos=new BufferedOutputStream(output);

    byte[] buffer = new byte[1024];

    bos.write(buffer, 0, buffer.length); //This line is important

   System.out.println("File name ="+files.get(m).getName());

   while (filelen > 0 && (smblen = threads[i].datainputstream.read(buffer)) > 0) {
       System.out.println("filelen ........" + filelen);
       System.out.println("smblen" + smblen);
        dos.write(buffer, 0, smblen);
        filelen = filelen - smblen;
        System.out.println("filelen ........" + filelen);
        dos.flush();
        }
      dos.close();  //It should close to avoid continue deploy by resource under view
```

Figure 4.11: Cloud server helper class: Image recreation

Once the image on the cloud server has been processed to determine the most optimal Otsu threshold value, the cloud server is required to binarize the image. Typically most images are binarized into black and white for the background and foreground, but this study makes use of the red and green colour channels of the RGB colour type. Using the optimal threshold value, the image under study is binarized into red and green, with the green colour channel representing the drivable rod regions and the red colour channel representing the non-drivable road regions. The code snippet in Figure 4.12 illustrates the binarization algorithm used on the cloud server.

```
for (int y = 0; y < height; y++) {
    int color = img.getRGB(x, y);

    red = ImageOperations.getRed(color);
    green = ImageOperations.getGreen(color);
    blue = ImageOperations.getBlue(color);

        //System.out.println("Threshold : " + requiredThresholdValue);
        if((red+green+green)/3 < (int) (requiredThresholdValue)) {
            //finalThresholdImage.setRGB(x,y,ImageOperations.mixColor(0, 0,0));
            finalThresholdImage.setRGB(x,y,COLOR_RED);
        }
        else {
            //finalThresholdImage.setRGB(x,y,ImageOperations.mixColor(255, 255,255));
            finalThresholdImage.setRGB(x,y,COLOR_GREEN);
        }
```

Figure 4.12: Cloud server helper class: Image binarization

Finally the binarized image showing the drivable and non-drivable road regions needs to be sent back to the robot client. This is achieved using the data streams instantiated using the *clientSocket* object created on the cloud server. The data is sent back to the robot client using the port that was used during the socket connection creation. The code snippet in Figure 4.13 shows the streaming of the image from the cloud server to the robot client (Refer to Appendix B for complete Robot Helper code).

```
for (int k=0; k<RGBFiles.length; k ++){

    int filesize = (int) RGBFiles[k].length();

    byte [] buffer = new byte [filesize];
    //byte [] buffer = new byte [1024];

    FileInputStream fis = new FileInputStream(RGBFiles[k].toString());
    System.out.println("RGBFiles[k].toString() " + RGBFiles[k].toString());
    BufferedInputStream bis = new BufferedInputStream(fis);

    //Sending file name and file size to the server
    bis.read(buffer, 0, buffer.length); //This line is important
    System.out.println("buffer.length " + buffer.length);

    dataoutputstream.write(buffer, 0, buffer.length);
    dataoutputstream.flush();
    //dataoutputttream.close();
}
```

Figure 4.13: Cloud server helper class: Image sent back to robot client

4.3 SCMR FRAMEWORK ENERGY UTILIZATION

Energy consumption and energy efficiency in cloud computing and robotics is an area that has recently gained significant interest from the scientific community. The scientific research done on energy saving models in order to extend the battery life of applications by reducing energy consumption can be divided into the following four categories [21]:

- Develop the robots to include new technology semiconductors, which are becoming smaller in order to consume less energy. However, the drawback is that in order for a robot to perform additional functionalities, more transistors are required to ensure that the robot performs optimally. The increase in resistors is proportional to the increase in energy consumed by the robot.
- Individual components in the robot or the robot itself can be programmed to be in sleep mode or standby mode when they are idle or not required to perform any computationally intensive tasks. This will prevent the robot from wasting energy.
- Introduce energy optimal execution policies, which determine the optimal clock frequency of a processor in order to complete a particular task in a predefined time. The amount of energy consumed by the robot can be reduced by slowing down the clock frequency and increasing the execution time of a particular task.
- Remove the computation burden from the robotics all together. All computation that is required to be performed locally by the robot is offloaded into the cloud server and the energy that is consumed by the robot is only that of sending and receiving the required information to the cloud server. This approach to energy saving is also called *computation offloading* and is one of the focus points for the SCMR framework.

4.3.1 Implementation of computational offloading in SCMR framework

In the context of Cloud Robotics, the tradeoff between the energy consumed by the individual robot and the energy consumed by the robot offloading the information to the cloud server is extremely important. It is therefore important to consider the energy consumed by individual robots (E_r) versus the energy required by the robot to receive and offload data (E_c). Taking this into consideration, it is possible to come up with a simple polynomial to reflect the total energy (T_e) consumed by a robot as [21]:

$$T_e = E_{proc} + E_{idle} + E_{trans} \quad (4.1)$$

where E_{proc} is represented by a polynomial expression as the total energy consumed by the robot while processing an instruction, E_{idle} is represented as the total energy consumed by the robot while it is idle and E_{trans} is the total energy consumed by the robot for transferring and receiving data.

4.3.2 Implementation of robotic execution energy model in SCMR framework

The energy consumption required to complete a task by a robot is generally determined by the CPU workload, memory, cooling system, fans and other robotic components. In the proposed model the amount of energy consumed is based upon the CPU workload of the individual robots. In their study of energy efficiency for content-based image retrieval from a cloud [22], it was observed that the total energy consumed by a robot can be formulated as:

$$E_{proc} = P_c \times \frac{C}{S_r} \quad (4.2)$$

where P_c is the average computing power of the robot when it is busy, C represents the total number of instructions required by the robot for the computation and S_r represents the clock speed of the robot. The assumption is that a particular task can be performed locally by the robot without a cloud server.

4.3.3 Implementation of cloud execution energy model in SCMR framework

In order for a robotic task to be executed in the cloud, the energy required by the robot to send this request to the cloud is dependent on the size of the data being offloaded and the wireless channel mode. In determining the optimal cloud execution energy model for the SCMR framework, the energy execution model adopted for this research is an empirical model similar to [22]. The following assumptions were made in the research study with regard to cloud execution of robotic tasks:. The first assumption is that the execution file for the individual robots has been replicated in the cloud and therefore does not sustain any further energy costs. The second assumption is that the clock frequency of the CPU in the cloud is F times faster than the CPU of the robot. The energy consumed by the robot in offloading D bytes of data over a wireless channel with bandwidth B can be defined as:

$$E_{idle} = P_i \times \frac{C}{S_c} \quad (4.3)$$

$$E_{trans} = \left\lfloor P_u \times \frac{D}{B_u} \right\rfloor + \left\lfloor P_d \times \frac{D}{B_d} \right\rfloor \quad (4.4)$$

where P_i is the average computing power of the robot when it is idle, C represents to the total number of instructions required by the robot for the computation and S_c represents the clock speed of the cloud server, P_u is the uploading power over a wireless network, P_d is the downloading power over a wireless network, B_u is the uploading network bandwidth, B_d is the downloading network bandwidth and D is the number of bytes exchanged between the robot and the cloud server.

4.3.4 Implementation of optimal task execution policy in SCMR framework

The optimal task execution policy in the context of cloud robotics is to choose where a particular task should be executed, the objective being to minimize the total energy consumed by the robot [21]. According to Equation 4.4, it is possible to reduce the energy consumed by the robot by reducing the number of bytes transferred to the cloud server. However, the challenge with the delay in transferring the data to the cloud could mean that a task is executed outside an agreed-upon time period and therefore in breach of a service level agreement that could be put in place between the cloud service provider and cloud consumer. It is therefore important that a middle ground be formulated that will optimally decide whether a task is executed in the cloud or locally by the robot by calculating the amount of energy saved during a particular task execution. Equation 4.3 represents the amount of energy consumed while the robot is idle and Equation (4.4) represents the amount of energy consumed when transferring the data. Therefore the total amount of energy saved can be formulated as:

$$E_{saved} = P_c \times \frac{C}{S_r} - \left\lfloor P_i \times \frac{C}{S_c} \right\rfloor - \left\lfloor P_u \times \frac{D}{B_u} \right\rfloor - \left\lfloor P_d \times \frac{D}{B_d} \right\rfloor . \quad (4.5)$$

One of the assumptions is that the cloud server (S_c) is F times faster than the robot sever (S_r), i.e. $S_c = F \times S_r$. Therefore Equation 4.5 can be rewritten as follows:

$$E_{saved} = \frac{C}{S_r} \times \left\lfloor P_c - \frac{P_i}{F} \right\rfloor - \left\lfloor P_u \times \frac{D}{B_u} \right\rfloor - \left\lfloor P_d \times \frac{D}{B_d} \right\rfloor \quad (4.6)$$

where energy is saved if $E_{saved} > 0$. In such instances the offloading is beneficial in instances of heavy computation such as image segmentation with relatively low bandwidth communication.

4.4 QOS ON DRIVABILITY WITH NO CLOUD DISCONNECTIONS

In general the QoS is used to assess the performance of the SCMR framework. The QoS is used to illustrate the performance quality of the SCMR in detecting drivable road images while at the same time highlighting the optimal data required to fulfil a request. In the SCMR framework a request can either be fulfilled by the robot cloud server (in instances of no cloud disconnection) or by the robot leader (in instances of cloud disconnection). The SCMR implementation prototype for drivability analysis with no cloud disconnection is shown in Figure 4.14. This section is used to illustrate the performance of the SCMR framework with no cloud disconnections.

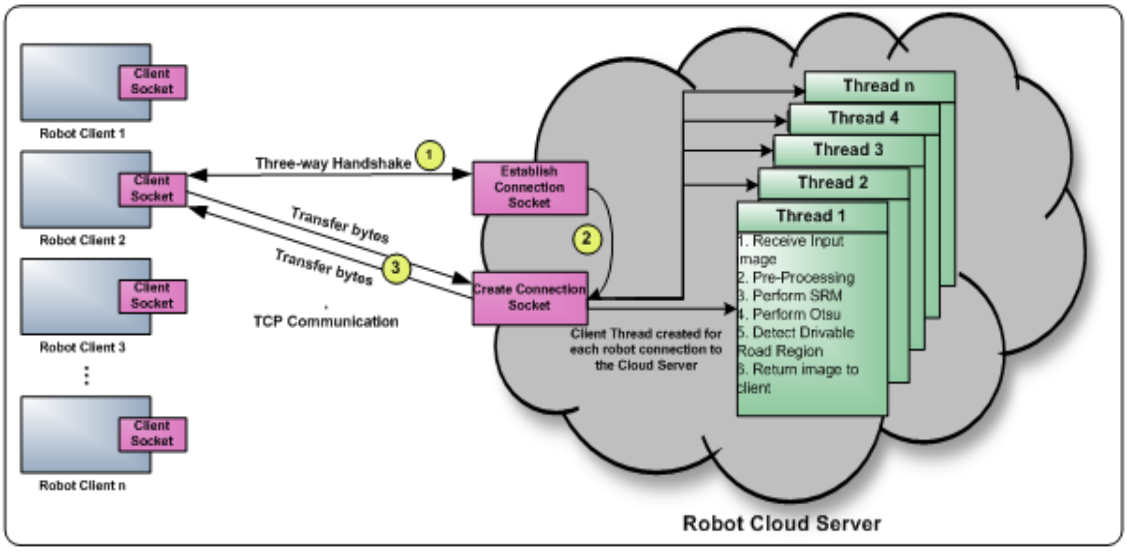


Figure 4.14: SCMR implementation with no cloud disconnection

4.4.1 Experiment 1: Observations on visual inspection with no cloud disconnection

The visual results obtained from the image frames submitted to the robot cloud server by the robot clients using the SCMR prototype are presented in this section. The images used in this experiment represent a wide variety of terrains and context. Table 4.2 contains the detailed comparative analysis of the images used in the visual inspection experimental tests, together with the image sets assigned to the different robot clients. Figure 4.15 shows the different

robot clients navigating on different terrains and the results obtained for drivability analysis using SRM on the SCMR prototype. The first row in Figure 4.15 consists of the original image frames. The second row presents the SRM filtered images, which consists of clusters that are generated for regions of homogeneity. In the experiment, after testing with different values of Q for SRM, Q=32 gave the optimal results for the image classification. The third row is the images created using the Otsu thresholding method while the last row is the RGB (red, green, blue) representation of the corresponding image frames. The red regions represent the non-drivable road regions while the lower green regions represent the drivable road regions. It is evident from the results that SRM has the ability to reconstruct regions with the same homogeneity that are closer to the robot's view. The results also illustrate that the regions closest to the robot's view are the drivable regions.

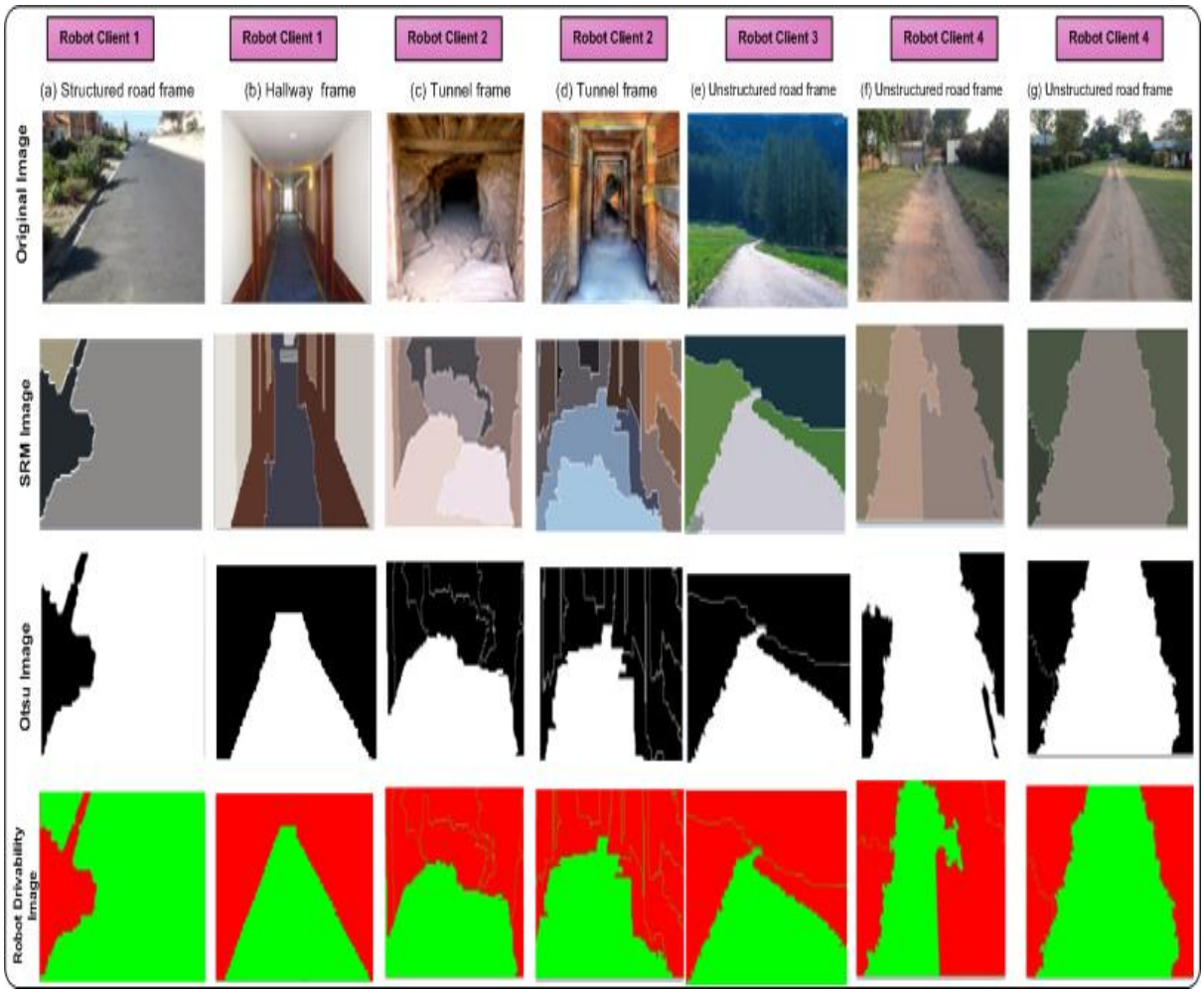


Figure 4.15: Qualitative results of SRM method on image frames with no cloud disconnection

4.4.2 Experiment 2: Time of response with no cloud disconnection

Most network-based applications use bandwidth usage as a primary factor in defining QoS [16]. This is attributed to the fact that most network-based applications are sensitive to bandwidth usage, as this affects their response. However, in cloud robotics, the bandwidth usage cannot be used as the only QoS factor [11]. Instead, other factors, such as processing capabilities or storage capabilities, can be extended as factors to define QoS. One of the main characteristics of cloud robotics is the ability to enable multiple robots to send parallel requests to the cloud sever for processing, with the goal of accomplishing a particular task or sharing knowledge. Therefore the ToR is a very important QoS in determining the performance of any cloud robotic framework and as such was selected as the primary QoS to be tested on the SCMR framework.

The ToR is the time required by the individual robot clients to receive a response from the robot cloud server after the request has been sent. The ToR is formulated as follows:

$$ToR = T_R - T_S \tag{4.7}$$

where T_R is the time the request was received and T_S is the time the request was sent to the robot cloud server.

In order to determine the ToR with no cloud disconnections on the SCMR framework, experiments were conducted using multiple robot clients. In the first experiment the images selected as part of the visual inspection (Figure 4.15) were first submitted one at a time to the robot server. Figure 4.16(A) illustrates the ToR against each image that was submitted to the cloud server, highlighting that the ToR increases as the size of the image increases. Since the images were submitted one at a time rather than as a stream of images, a connection to the cloud server had to be established for each picture being submitted to the cloud server. The ToR can therefore be greatly reduced if the images are submitted in a stream with the robot only being required to establish a connection to the cloud server once.

In the second scenario, four robot clients were used to submit requests to the cloud server to detect drivable road regions. Figure 4.16(B) illustrates the ToR for each of the client robots. The results show some correlation between the ToR received for the different robots. The first

image submitted to the cloud server for all the robots generally has a slightly longer ToR, which is brought about by the establishment of the socket connection. Thereafter the ToR is proportional to the size of the image being submitted. If the image is much larger in size compared to the other images, the ToR tends to increase gradually, as was the case in Robot 4's ToR which increased to 3 068 ms.

The results from this experiment demonstrate that the SCMR framework can be effective in alleviating the computational burden from individual robots in clearly determining drivable road regions using the SRM algorithm in cases of no cloud disconnection. In addition, the ToR response ranged between 2 400 ms to 3 100 ms for images ranging from 80 000 Bytes to 130 000 Bytes in size, which is sufficient time for robots to receive and process an image. In general each robot's request was queued on the server and because a new thread was spawned for each connection, each robot client achieved faster response times.

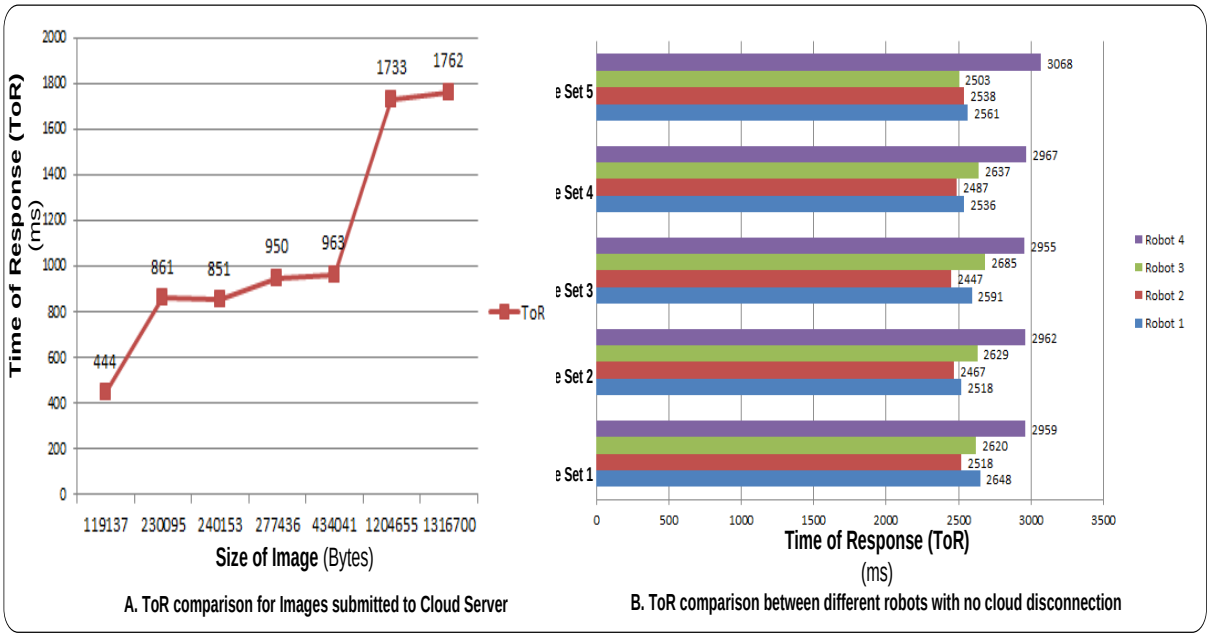


Figure 4.16: ToR with no cloud disconnections

4.4.3 Experiment 3: Energy utilisation of SCMR with no cloud disconnection

The next part of the experiment was to determine the amount of energy used on the SCMR framework with no cloud disconnections. Table 4.1 lists the energy calculation symbols and definitions used in the experiment. The client/server performance ratio *F* used in the

experiments was $\alpha = 300$ and using this ratio, the clock speed of the cloud server $S_c$was determined.

The total energy consumed by the robot when drivability analysis is performed using SRM on the cloud server is calculated when the robot is *idle* (E_{idle}) and when the robot is *transferring* (E_{trans}) data to the cloud server. The energy execution for cloud processing can therefore be formulated as:

$$E_{Cloud} = \left\lfloor P_i \times \frac{C}{S_c} \right\rfloor + \left\lfloor P_u \times \frac{D}{B_u} \right\rfloor + \left\lfloor P_d \times \frac{D}{B_d} \right\rfloor \quad (4.8)$$

where B_u is the uploading bandwidth and B_d is the downloading bandwidth used to send and receive image frames to the cloud server. Using the value of $n = 5$ as the number of computing instructions required for processing an image of 1 MB, Equation 4.9 can be populated as follows:

$$E_{Cloud} = \left\lfloor P_i \times \frac{5}{P_c \, X \, 300} \right\rfloor + \left\lfloor P_u \times \frac{1048576}{3014656} \right\rfloor + \left\lfloor P_d \times \frac{1048576}{3670016} \right\rfloor \quad (4.9)$$

where the bandwidth download speed is 28 Mb/s and the bandwidth upload speed is 23 Mb/s. Thereafter the energy used by the different robot clients was calculated. The robot clients used to test the energy utilization of the SCMR framework ranged from low end robots to high end robots with diverse processing power sets, as illustrated in Table 4.3.

Figure 4.17 (A) shows the energy utilization of the cloud server from different robot clients processing a 1 MB image frame on the SCMR framework. The energy utilization increases as the processing power of the individual robot client's increases. Figure 4.17(B) illustrates that as the bandwidth increases, the energy utilization of the cloud server exponentially decreases. This is attributed to the speed with which the image frame is transported to and from the cloud server. On average a 37% reduction is achieved among the different robot clients when the bandwidth is increased from 25 Mb/s to 70 Mb/s. This indicates that the network bandwidth B_d and B_u in Equation 4.9 plays a vital role in determining the energy utilization required to perform drivability analysis on the SCMR framework. Figure 4.17(C) shows the energy utilization of the cloud server from the different robot clients using the images selected as part

of the visual inspection, while Figure 4.17(D) illustrates the exponential power utilization of the cloud server brought about by the multithreading capability on the SCMR framework. For every new robot connection a new client thread was spawned and therefore the total energy consumption increased exponentially. The results indicate that as new robots are introduced onto the SCMR framework the energy utilization increases. The results also indicate that the energy utilization is dependent on the network bandwidth and the size of the images, which could be used to decrease the energy utilization on the cloud server.

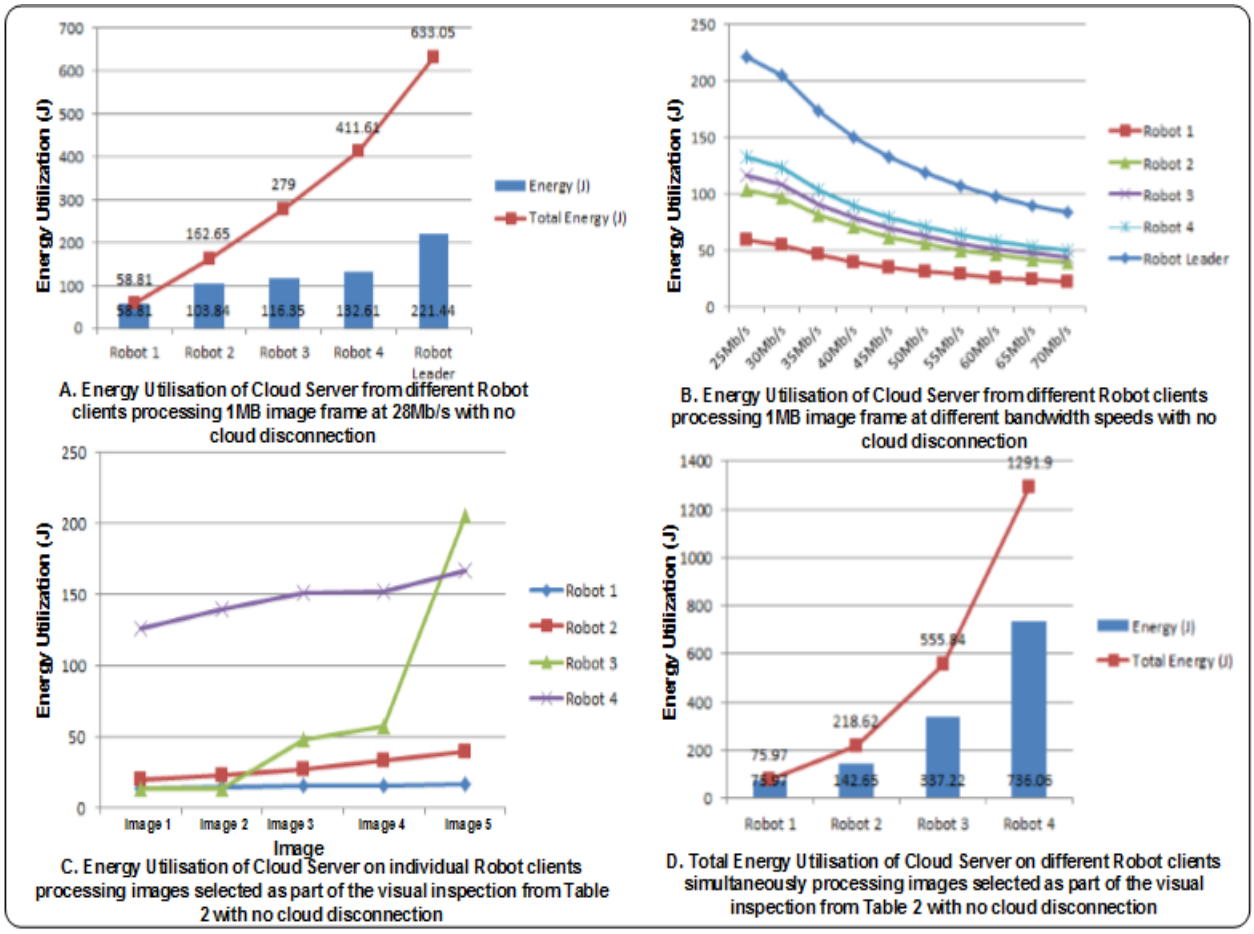


Figure 4.17: Cloud execution energy consumption

4.5 QOS ON DRIVABILITY WITH CLOUD DISCONNECTIONS

In this section the performance of the survivability aspect of the SCMR framework was tested. A stub was created to stop the cloud server from accepting connections in order to simulate the scenario of cloud disconnection. The SCMR framework was built to have a retry mechanism and after three unsuccessful error attempts at establishing a connection to the cloud server,

communication was transferred to the virtual cloud via the robot leader. Figure 4.18 illustrates the SCMR implementation with cloud disconnection.

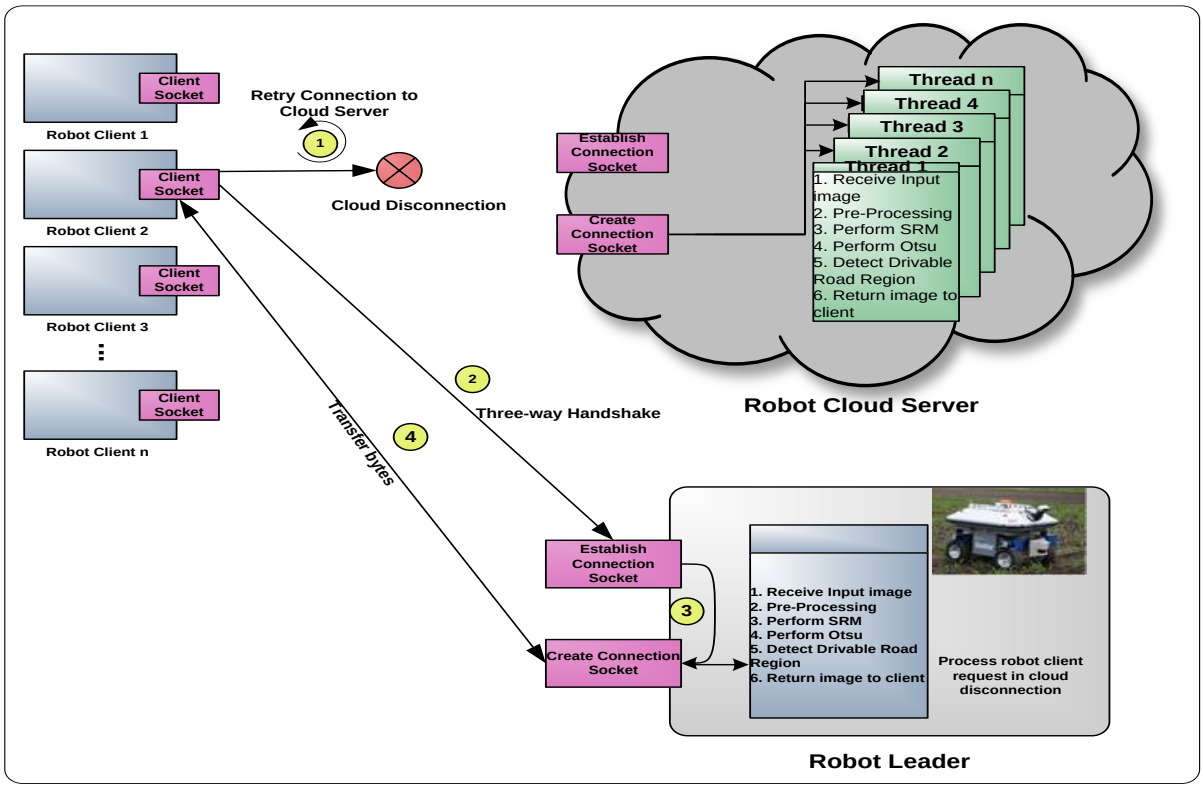


Figure 4.18: SCMR implementation with cloud disconnection

4.5.1 Experiment 4: Observations on visual inspection with cloud disconnection

In certain instances cloud-enabled multi-robotic teams could find themselves in environments where accessibility to the nearest cloud access point was not possible. Therefore the images used to test the visual inspection of the SCMR framework with cloud disconnection were of terrains where accessibility to a cloud access point could not be easily achieved. The images used ranged from deep underground mine tunnels or mountain caves to desert terrains. Figure 4.19 shows the visual assessment of the image frames processed by the robot leader using SRM on the SCMR prototype in instances of cloud disconnection. The first row in Figure 4.19 shows the original image frame that was submitted by the particular robot client. The second row shows the SRM filtered images with Q=32 and the third row represents the SRM images created using the Otsu thresholding technique. The last row illustrates the binarization of the Otsu image, which splits the image into drivable and non-drivable pixels. The red regions represent the non-drivable road regions, while the lower green regions represent the drivable

road regions. The results indicate that even in the event of cloud disconnection, the SCMR framework has the ability to use the SRM method to reconstruct regions of the same homogeneity. This ability enables the robot leader to perform successful drivability analysis in the event of cloud disconnection and is illustrated in Figure 4.19, where the region closest of the robots view is shown as drivable regions.

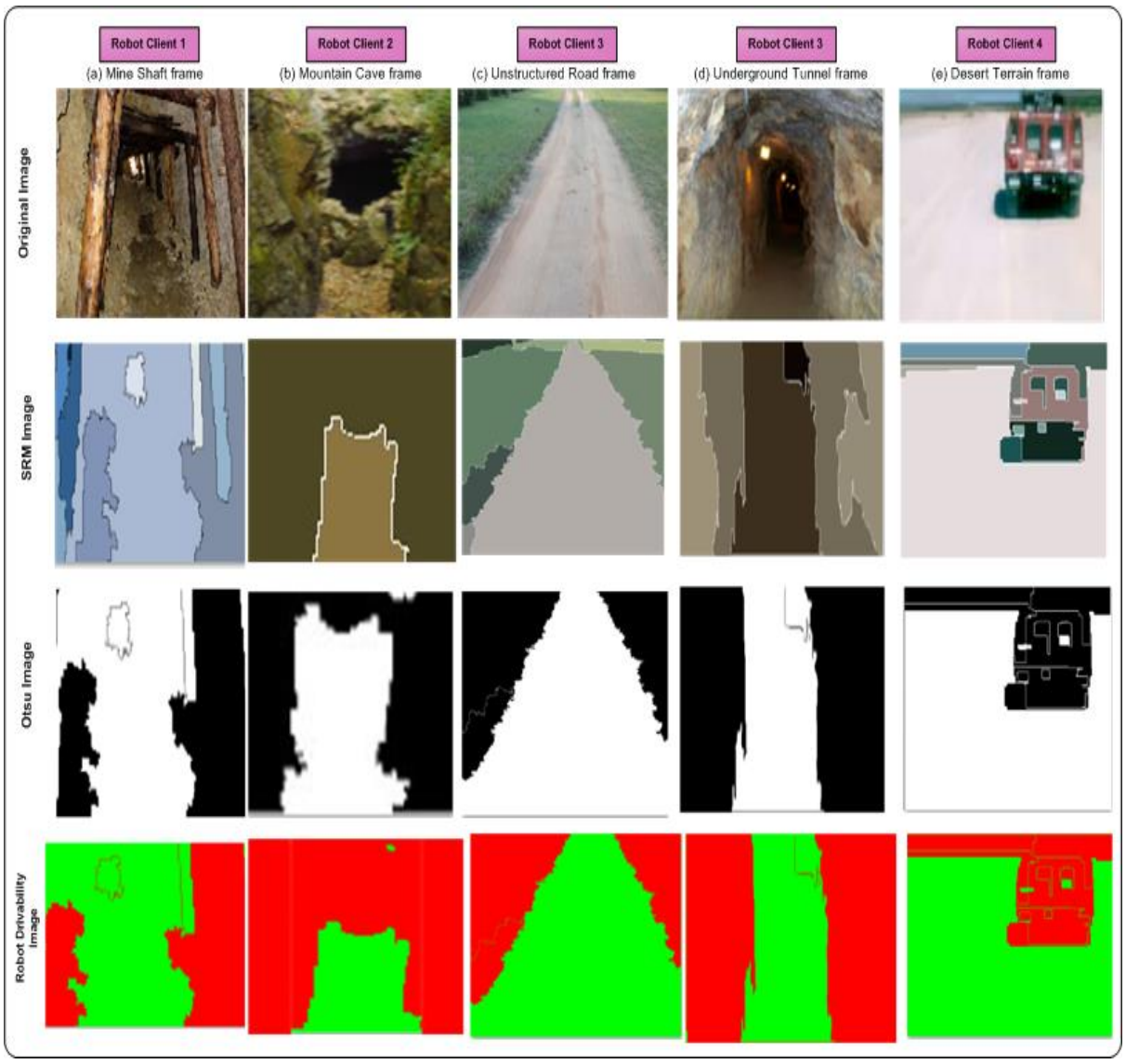


Figure 4.19: Qualitative results of SRM method on image frames with cloud disconnection

4.5.2 Experiment 5: Time of response with cloud disconnection

Upon determining that the SCMR framework can successfully perform drivability analysis in the event of cloud disconnections through the virtual cloud created by the robot leader, the next

step in the experiment was to validate the performance of the framework in cases of cloud disconnection. Multiple robot clients were used to submit requests to the cloud server and after three unsuccessful attempts at reaching the cloud server the request was routed to the robot leader. A comparison was drawn from the image set used to test ToR with no cloud disconnection and ToR with cloud disconnection. This comparison is illustrated in Figure 4.20(A), which shows a delayed ToR of $\approx 3\ seconds$. The ToR received for the different image sets submitted by the robot clients to the robot leader is illustrated in Figure 4.20(B), while the weighted average ToR between the image sets submitted with and without cloud disconnections is illustrated in 4.20(C). The results indicate that even though the drivability analysis is successfully performed by the robot leader, there is a measured delay in the ToR brought about by the cloud disconnection. This delay, however, should not affect the ability of a robot to complete a particular task. In general the results indicated that even without access to the cloud server the robot client could perform drivability analysis optimally.

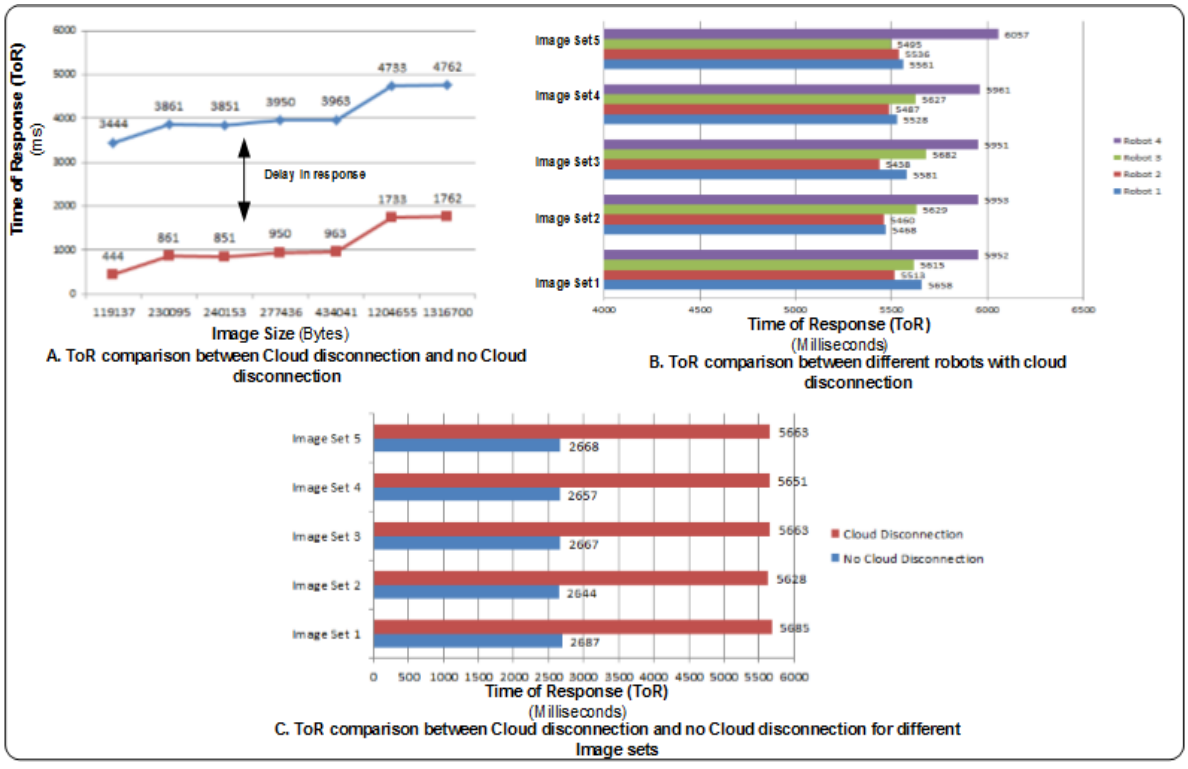


Figure 4.20: ToR with cloud disconnections

4.5.3 Experiment 6: Energy utilization of SCMR with cloud disconnection

The next step in the experiment was to determine the energy utilization of the different robot clients with cloud disconnection. The robot clients used in this experiment varied in processing power, as illustrated in Table 4.3. Using Equation 4.2, the power utilized on the SCMR framework during cloud disconnection was calculated. Figure 4.21 (A) illustrates the different energy utilizations of the individual robot clients on the SCMR framework. Figure 4.21 (B) illustrates the survivability aspect of the SCMR framework by comparing the total energy utilized during cloud connection and disconnection.

The graph illustrates the total energy utilized by the cloud server against the total energy utilized by the robot leader, highlighting the amount of energy saved by offloading computationally intensive processing to the cloud server. Finally, in the event of cloud disconnection, the energy consumption of the robot may be reduced to zero, in which case the robot leader takes over the drivability analysis calculations and the energy required increases again. Figure 4.21 (C) illustrates that as the individual robot clients are disconnected from the cloud server, their energy utilization drops, while the energy utilization of the robot leader grows exponentially, as it is required to act as the virtual cloud and perform the drivability analysis for the robot clients.

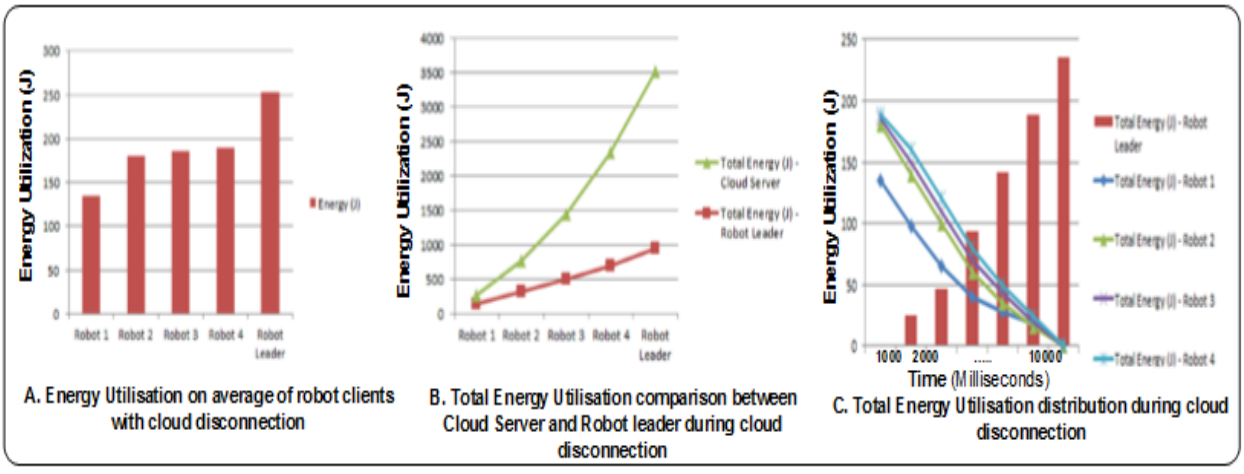


Figure 4.21: Robotic survavibility energy consumption

Table 4.1: Energy calculation symbols and definitions

Parameter	Operation
P_c	Power Computing
P_i	Power Idle
P_u	Power Uploading
P_d	Power Downloading
B	Bandwidth
C	Computing Instructions for SRM
F	Client/Server Performance Ratio
S_r	Robot CPU Clock Speed
S_c	Cloud Server CPU Clock Speed (*Offloading*)
D	Size of Image for SRM

Table 4.2: Comparing Drivability Analysis for the Different Robot Clients with and without Cloud Disconnection

				Cloud Processing				Robot Processing				R2C Processing	R2R Processing
Robot	Image Set	Image	Images Desc.	Original File Size (Bytes)	Upload - Start (ms)	Upload - End (ms)	Total Send Time (ms)	Download File Size (Bytes)	Download - Start (ms)	Download Time End (ms)	Total Recieve Time (ms)	Processing Time (ms)	Robot Leader Total Processing Time (ms)
Robot 1	Image Set 1	Image1.png	Tunnel Frame	240153	1399099623433	1399099623467	34	2236	1399099626080	1399099626081	1	2648	5658
Robot 1	Image Set 1	Image5.png	Mine Frame	253110	1399099623552	1399099623572	20	2836	1399099626089	1399099626090	1	2518	5468
Robot 1	Image Set 1	Image2.png	Unstructered Road	277436	1399099623468	1399099623492	24	2886	1399099626082	1399099626083	1	2591	5581
Robot 1	Image Set 1	Image4.png	Unstructered Road	287672	1399099623524	1399099623551	27	2807	1399099626086	1399099626087	1	2536	5528
Robot 1	Image Set 1	Image3.png	Hallway Frame	295630	1399099623493	1399099623524	31	2102	1399099626084	1399099626085	1	2561	5561
Robot 2	Image Set 2	Image6.png	Structured Road	197669	1399101193726	1399101193746	20	2433	1399101196264	1399101196264	0	2518	5513
Robot 2	Image Set 2	Image8.png	Unstructered Road	230095	1399101193780	1399101193800	20	2368	1399101196267	1399101196267	0	2467	5460
Robot 2	Image Set 2	Image9.png	Hallway Frame	274552	1399101193800	1399101193821	21	2296	1399101196268	1399101196268	0	2447	5438
Robot 2	Image Set 2	Image7.png	Mountain Cave	336791	1399101193747	1399101193779	32	3197	1399101196265	1399101196266	1	2487	5487
Robot 2	Image Set 2	Image10.png	Mine Shaft	401100	1399101193688	1399101193725	37	2847	1399101196262	1399101196263	1	2538	5536
Robot 3	Image Set 3	Image15.png	Underground Tunnel	119137	1399103167479	1399103167489	10	363	1399103170107	1399103170109	2	2620	5615
Robot 3	Image Set 3	Image14.png	Mine Shaft	123469	1399103167469	1399103167478	9	282	1399103170105	1399103170107	2	2629	5629
Robot 3	Image Set 3	Image11.png	Hallway Frame	434041	1399103167372	1399103167418	46	1552	1399103170103	1399103170103	0	2685	5682
Robot 3	Image Set 3	Image12.png	Mine Frame	514518	1399103167418	1399103167468	50	2371	1399103170104	1399103170105	1	2637	5627
Robot 3	Image Set 3	Image16.png	Underground Tunnel	1847819	1399103167489	1399103167648	159	2858	1399103170110	1399103170111	1	2463	5495
Robot 4	Image Set 4	Image20.png	Structured Road	1000270	1399103684149	1399103684153	4	509	1399103687111	1399103687112	1	2959	5952
Robot 4	Image Set 4	Image19.png	Desert Terrain	1103609	1399103684141	1399103684148	7	848	1399103687108	1399103687110	2	2962	5953
Robot 4	Image Set 4	Image21.png	Underground Tunnel	1194963	1399103684154	1399103684249	95	773	1399103687113	1399103687114	1	2865	5951
Robot 4	Image Set 4	Image18.png	Unstructered Road	1204655	1399103684039	1399103684141	102	3602	1399103687107	1399103687108	1	2967	5961
Robot 4	Image Set 4	Image17.png	Desert Terrain	1316700	1399103683888	1399103684008	120	3321	1399103687105	1399103687106	1	3098	6057

Table 4.3: Power parameters of the robot clients used in the SCMR experiment

	Robot Leader	Robot 1	Robot 2	Robot 3	Robot 4
CPU	Intel Core i7-3770 CPU	Intel Pentium(R)	Core 2 Quad Q8200S	Core 2 Quad E8500	Core 2 Quad Q9550S
Power Computing	177W	47W	83W	93W	106W
Power Idle	99W	25W	37W	38W	39W
Power Uploading	314.61W	83.54W	147.53W	165.31W	188.41W
Power Downloading	391.98W	104.08W	183.80W	205.95W	234.74W
CPU Clock Speed	3.5GHz	1.0GHz	2.3GHz	2.5GHz	2.83GHz
Memory	8GB	236.2MB	2GB	4GB	6GB
Hard Disk	500GB	35GB	100GB	180GB	200GB

4.6 OPTIMAL TASK EXECUTION POLICY

4.6.1 Experiment 7: Observations on optimal task execution policy

The optimal task execution policy is used to identify whether a task should be executed locally by the robot or offloaded into the cloud server with the intention of energy conservation. The total energy saved by a robot when performing drivability analysis on the SCMR framework was calculated using Equation (6), where energy is only saved if $E_{saved} > 0$. Figure 4.22 (A) shows the energy savings on the SCMR framework using different image sizes and different bandwidths. The experimental results indicate that energy is saved on the SCMR framework for offloading an image of 1 MB when the bandwidth is ≈ 25 Mb/s while offloading an image of size 3 MB only saves energy when the bandwidth is ≈ 70 Mb/s. This indicates that energy is saved when $\frac{D}{B}$ is sufficiently less than $\frac{C}{S_r}$ and the value of F is considerably large.

Figure 4.22 (B) shows the energy saved by the different robot clients at different bandwidth speeds. The experimental results show that for the robot leader, energy saving on the SCMR framework can be achieved at much lower bandwidth speeds owing to its greater on-board processing capabilities, while robots with smaller processing capabilities, such as Robot 1 used in the experiment, can only achieve energy savings at much higher bandwidth speeds. Therefore, for many low-cost robots, it makes sense to offload as many computationally intensive tasks as possible at high bandwidths, thus enabling them to save more energy, which could be used for computational tasks such as path planning. The results illustrated in Figure 4.22 (A) and (B) confirm the optimal task execution policy.

Figure 4.22 (C) illustrates the optimal energy execution model for drivability analysis on the SCMR framework. The model indicates that as the size of the image to be offloaded increases, the drivability analysis should be performed locally by the robot in order to conserve energy and if the computation required for drivability analysis increases, then the computation should be performed by the cloud server. The model also makes provision for the wireless bandwidth B parameter, which needs to be taken into consideration when deciding whether or not to offload the drivability analysis to the cloud server for computation.

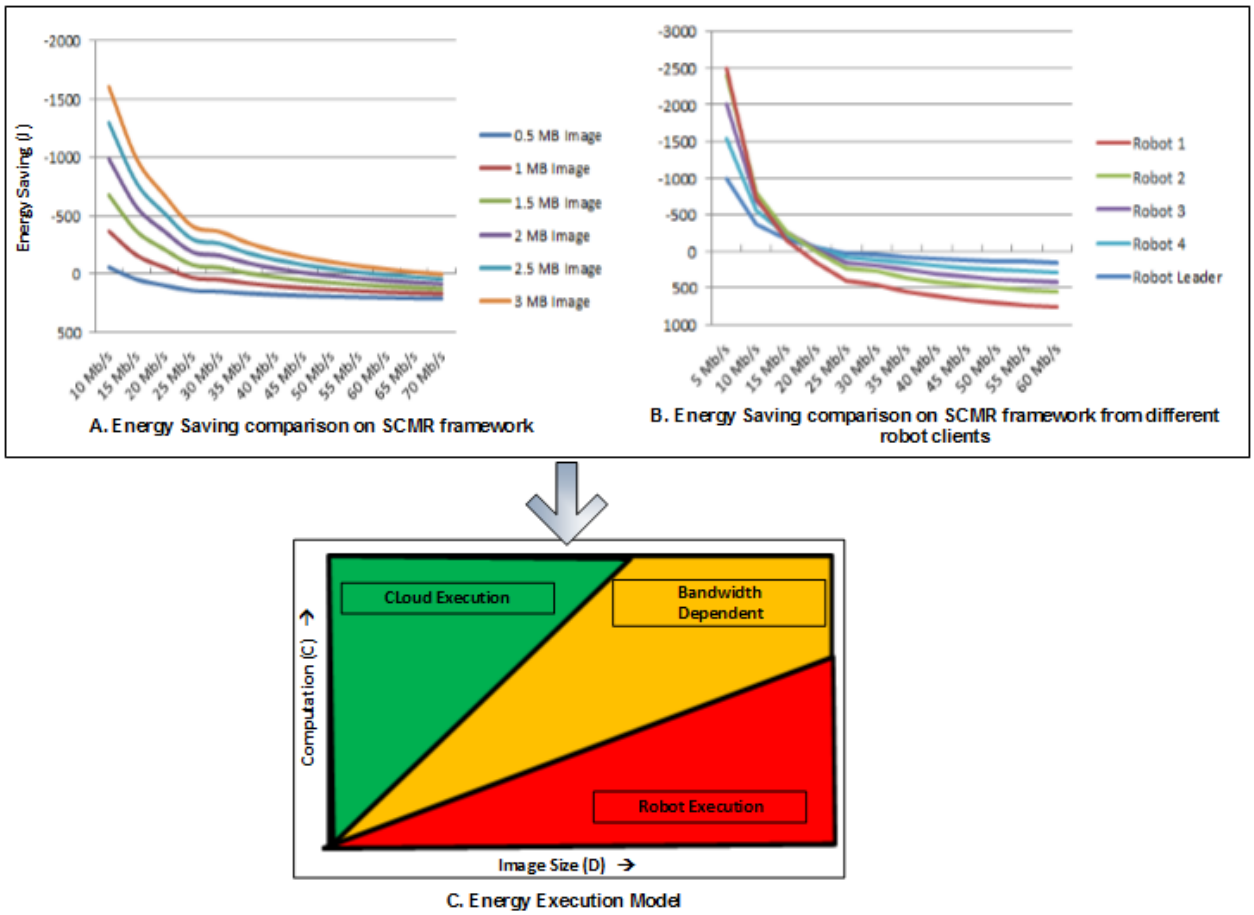


Figure 4.22: Energy saving and execution model

4.7 COMPARATIVE EVALUATIONS OF THE CLOUD ROBOTICS PARADIGMS

The three major attributes in any elastic cloud-based model are robustness, interoperability and mobility. The existing elastic cloud-based models presented in the study all exhibit different properties for these attributes. The SCMR framework is based on a cloud robotic model that leverages two tiers, the R2R tier and R2C tier, in order to manage cloud disconnections and has many benefits over the existing cloud robotic models.

In the R2R network model where the communication scheme is based on the randomized gossip protocol, suppose that node i randomly chooses node j to gossip with probability P_{ij}. In a worst-case scenario the probability is zero $O(N \log N)$ where N represents a node in the network and two nodes are not within communication range of each other and therefore not

neighbours. The communication delay in a R2R network, as illustrated in [20], can be calculated as $O(\log {}^{N}/_{\Phi})$, which is the delay for broadcasting a message from a single node to all nodes in the network. In the proposed cloud multi-robotic framework the R2R network is portioned into smaller teams, each of which has access to the cloud server. The maximum number of multi-robot teams can be represented as M. In the proposed cloud robotic model all robots have communication to the cloud server, indicating that $M = 1$. The fluidity of the network can be represented at minimum as $\frac{1}{M}$, which means that the communication delay can be limited to $O(M \log N)$. In a peer-based cloud robotic model, the assumption made by [11] indicates that a fraction α exists that connects to the cloud server for a given subset of nodes in a communication network.

Finally, in the R2C communication scheme, the worst case time required to communicate that a particular task needs to be executed is $O(M \log M)$, because the individual robots can offload the robotic tasks directly onto the cloud server for processing. Table 4.4 illustrates the worst-case delays for the different computing models discussed in this research study together with the major attributes, *robustness*, *interoperability* and *mobility*, that make up any elastic cloud-based model.

Table 4.4: Comparison of worst case communication delays for cloud robotic models

Model	R2R	R2C	Robustness	Inter-operability	Mobility
Peer-based	$O(\frac{1}{\alpha} \log N)$	$O(\frac{1}{\alpha} \log N)$	Low	Medium	Medium
Proxy-based	$O(M \log N)$	$O(M \log M)$	Medium	High	Medium
Clone-based	$O(\log N)$	$O(1)$	Medium	Low	Low
Proposed cloud robotic model for SCMR	***O(M log N)***	***O(1)***	***High***	***Medium***	***High***

The proposed cloud elastic model is aimed at being an optimal or near optimal model for multi-robot teams executing robotic tasks in heterogeneous environments with unreliable cloud connection. The proposed model itself is a hybrid model that includes execution by the individual robot (robot leader), collaborative execution by the multi-robotic team (packbot robots) connected to the network and cloud execution. The proposed model has the highest number of connections between the robot and cloud server and the highest mobility attributed by VMs being able to be instantiated anywhere in the physical cloud infrastructure, compared to other cloud robotic models.

4.8 CHAPTER SUMMARY

This chapter presented evidence in relation to the practically, scientific relevance, scalability and performance of the proposed SCMR framework. The first evaluation focused on visual observations of the drivable road detection images produced by the SCMR framework in instances of no cloud disconnection. The next part of the evaluation was to demonstrate the QoS of the proposed framework in terms of ToR and energy utilisation. Thereafter the survivability aspect of the framework was demonstrated by conducting experiments with connection to the cloud server not being made available. The last part of the evaluation was to validate the optimal task execution policy. This was used to determine efficiently where a particular robotic task should be executed, i.e. locally on the robot vs on the cloud server.

The results from the experiments indicate that significant energy can be conserved by the robot in deciding optimally where to perform the drivability analysis using SRM. This study found that at high bandwidth with no cloud disconnections the energy used by the packbot robot to offload an image to the cloud is greatly reduced compared to areas with low bandwidth. The results from the experiments provide a useful framework for cloud robotics that would enhance the ability of individual robots to execute robotic tasks such as path planning, drivability analysis, image processing and navigation in heterogeneous environments when connection to a cloud access point is not possible.

CHAPTER 5: CONCLUSIONS AND FUTURE WORK

5.1 INTRODUCTION

In this final chapter, a summary of the research study is provided by reconsidering the research questions and goals that inspired the study. In addition, the main contributions arising from the proposed SCMR framework are highlighted from both a theoretical and a practical point of view. Furthermore, the research limitations and challenges identified in the research study are highlighted. Finally, future research work that could address some of the identified challenges is also discussed.

As depicted in Figure 5.1, this chapter consists of five main sections. Section 5.1 provides the introduction to this chapter, while section 5.2 provides an overview summary of the research dissertation. Thereafter, section 5.3 reconsiders the research questions described in the first chapter of the dissertation and details the manner in which the research questions were answered to ensure that the objectives of the study have been achieved. The focus thereafter shifts to the practical implications and recommendations from the research study in section 5.4. Finally, section 5.5 concludes the chapter by providing the limitations of the study and highlighting future work.

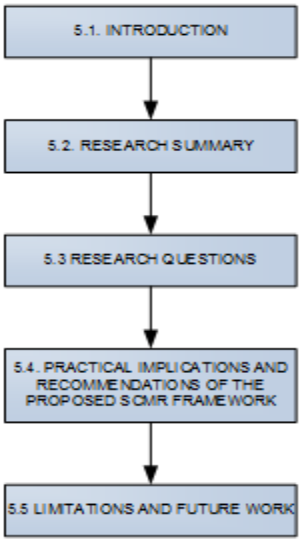


Figure 5.1: Graphical representation of Chapter 5

5.2 RESEARCH SUMMARY

The research study presented the design, implementation and evaluation of the SCMR framework for cloud robotics. The primary goal of the framework is to ensure that individual robots working in heterogeneous environments are able to execute robotic tasks in cases of cloud disconnection and also relieve robots of the computation burden in instances of no cloud disconnection by offloading the required data to the cloud server to be processed. The SCMR framework adopted the twisted network framework for handling all the requests to and from the central cloud host and made use of the WebSockets protocol for communication between the individual robots and the cloud server. In instances of cloud disconnection a virtual ad hoc cloud was created between the individual robots and the robot leader and the individual robots communicated with one another through the gossip protocol.

The flexibility of the SCMR framework was theoretically tested by simulating the power consumption of a low configuration packbot robot and comparing the energy conserved when performing drivability analysis using SRM with and without cloud disconnection. The results from the theoretical analysis indicated that significant energy can be conserved by the robot in optimally deciding where to perform the drivability analysis using SRM. This study found that at high bandwidth with no cloud disconnections the energy used by the packbot robot to offload an image to the cloud is greatly reduced compared to areas with low bandwidth. This study also found that the amount of energy conserved by the packbot robot substantially increases when offloading smaller images, compared to offloading larger images over a wireless network. Finally, the findings of the study confirmed the survivability benefit of the SCMR framework and also illustrated that the energy consumption of offloading an image for drivability analysis with and without cloud connection varies depending on a number of factors.

5.3 RESEARCH QUESTIONS

This section describes the manner in which the supporting research questions described in Chapter 1 (Section 1.4) were addressed to answer the main research question and accomplish the main objectives of the study. The main research question of the dissertation was phrased as follows: *How can a cloud multi-robotic framework be developed for surviving cloud*

disconnection over independent on-board processing by considering a team of robots' perception/vision, path planning and navigation? A review of the supporting questions (SQ) and objectives that were accomplished in the dissertation is offered below:

SQ1*: How can quality and reliable multiple drivable regions be simultaneously obtained with and without cloud disconnections in heterogeneous environments?*

Research from the literature review indicated that the visual capabilities of a multi-robotic team are extremely important in order to be able to navigate a heterogeneous environment successfully and that very little work has been done in this area, particularly on robotic vision in heterogeneous environments. This led to the adoption of the SRM technique to detect drivable road regions using the proposed SCMR framework. The methodology and design of using SRM for drivable road region detection were extensively documented in Chapter 3.

The cloud robotic model used for the proposed SCMR framework made use of a combination of the existing cloud robotic models in that it had a robot leader that acted as a clone of the cloud server and also had the ability to effect peer-to-peer communication between the robots and the cloud server. The differentiating factor between the proposed model and the existing models was that it made use of the R2C and R2R paradigm that was introduced to overcome the challenge of cloud disconnection.

The main objective of the study was to investigate and develop an SCMR framework that would be able to handle cloud disconnections in heterogeneous environments. In addressing SQ1, the design requirements presented in Chapter 3 were taken as input to develop a prototype of the proposed SCMR framework. The design requirements provided a basis for developing a solution that would not only address the challenge of cloud disconnection but also ensure that the solution is future-proof in terms of scalability, extensibility, reusability and interoperability. The details of the prototype implementation and evaluation were presented in Chapter 4 of the dissertation.

Once the development of the prototype had been completed using the Java network protocol, a client/server application listening on port 2222 was used to simulate requests to the cloud server to detect drivable road regions. Four different robot clients were used, each having varying processing power. The robot clients used images acquired from a wide range of heterogeneous environments to send to the cloud server. The detailed results of these visual inspections are shown in Chapter 4. The findings from the experimental tests in Experiment 1 and Experiment 4 show that SRM has the ability to reconstruct regions with the same homogeneity that are closer to the robot's view. The results also illustrate that the regions closest to the robot's view are the drivable regions.

Thereafter the next step was to evaluate the worst-case communication delays between the different cloud robotic models using the three major attributes in any elastic cloud-based model, which are robustness, interoperability and mobility. The modelling of the theory in terms of the worst-case communication delays was done using the Big-O notation. In the analysis of the worst-case communication delays it was found that in the R2R network the communication delay can be modelled as $O(\log {}^{N}/_{\Phi})$, while in the R2C network the communication delay can be modeled as $O(M \log M)$. The detailed results from the evaluation of the worst-case communication delays for the different cloud robotic models, including the model used for proposed SCMR framework, are detailed in Section 4.7 of Chapter 4.

SQ2*: How is QoS optimized in terms of fast response, optimal task execution policy and time of response within the cloud multi-robotics framework?*

In this supporting question, the goal was to determine the manner in which the proposed SCMR framework can be optimized to meet its QoS requirements in terms of TOR and ROR. The analysis indicated that a set of well-defined and accurate QoS standards is essential in order to validate a particular framework.

The analysis of different QoS factors revealed that most applications use bandwidth usage as the primary factor. However, in the field of cloud robotics this cannot be the only important factor considered when defining a set of QoS factors to validate the proposed framework.

Taking into consideration the requirements for the proposed SCMR framework, the QoS factors were defined using Equations 3.12 and 3.13 in Chapter 3.
Thereafter the proposed SCMR framework was validated using the defined set QoS standards in Experiment 2 and Experiment 5 to determine its efficiency and reliability, to ensure the frameworks met its design requirements. The findings from the scientific experiments indicated that ToR response ranged between 2 400 ms and 3 100 ms for images ranging from 80 000 Bytes to 130 000 Bytes in size, which is sufficient time for robots to receive and process an image. Furthermore, the results validated the proposed SCMR framework by demonstrating that the framework can be effective in alleviating the computational burden from individual robots in clearly determining drivable road regions using the SRM algorithm even in the event of cloud disconnection.

SQ3*: To what extent does the level of complexity analysis of the developed framework efficiently reduce the computational energy consumed overheads based on on-board processing and cloud mechanisms?*

In determining an approach that aims to deal with the computational energy overheads of the proposed SCMR framework, it was first important to understand a number of issues, such as What are the current energy consumption challenges faced in cloud robotics? How do these differ from the challenges faced in cloud computing? What are the existing inhibitors and opportunities in cloud robotics? What components contribute to the energy consumption of processing robotic tasks in a cloud robotic framework? What are the characteristics of these components?

In order to answer these questions, a comprehensive literature review on the related work and technologies in the scientific field of cloud robotics was performed and together with the dissertation's problem space and identified research objective, this formed a valuable compass.

The supporting question (SQ3) was covered by providing a comprehensive review of literature on existing cloud robotic frameworks and the limitations and shortcomings that currently exist in those frameworks. This then allowed for the fundamental building blocks for the design,

development and implementation of the proposed SCMR framework. The identified key building blocks were divided into the system architecture, platform architecture, data architecture, R2R and R2C paradigms, resource allocation and communication management. The main characteristics of the building blocks of the SCMR framework were presented in Chapter 3.

Once the fundamental building blocks of the proposed SCMR had been devised, it was important to understand the manner in which these components would be utilized to build a framework that would efficiently reduce the computational energy consumed overheads of the robots. Furthermore, it was important to understand the robotic tasks that would be executed on the proposed SCMR framework to determine the energy consumption of a robot in executing a particular task.

Thereafter an optimal task execution policy in Section 3.5.4 was formulated, which was used to identify whether a task should be executed locally by the robot or offloaded into the cloud server with the intention of energy conservation. The results from Experiment 3 and Experiment 6 indicate that as the size of the image to be offloaded increases, the drivability analysis should be performed locally by the robot in order to conserve energy and if the computation required for drivability analysis increases, then the computation should be performed by the cloud server. The experimental results in Section 4.6 indicate that energy is saved on the SCMR framework for offloading an image of 1MB when the bandwidth is ≈ 25 Mb/s, while offloading an image of size 3 MB only saves energy when the bandwidth is ≈ 70 Mb/s.

The energy execution policy also makes provision for the network bandwidth that needs to be taken into account when deciding where to execute a robotic task with the intention of energy conservation. The findings from the study showed that at higher bandwidth the energy conserved by the robot was higher than at lower bandwidth when offloading a robotic task to the cloud server for execution.

5.4 PRACTICAL IMPLICATIONS AND RECOMMENDATIONS OF THE PROPOSED SCMR FRAMEWORK

In Chapter 1 (Section 1.6) the primary contributions emanating from the research study were discussed. In this section the focus is mainly on the practical implications and recommendations that become apparent from the implementation of the proposed SCMR framework. The following are the noteworthy practical implications, contributions and recommendations emanating from the dissertation:

- Fundamental building blocks that comprise cloud robotic architecture, services and infrastructure together with their characteristics.
- Essential requirements and guidelines for designing and developing a unified survivable cloud multi-robot framework.
- A fully functional SCMR framework for addressing some of the challenges identified in existing cloud robotic models.
- An optimal task execution policy for determining whether a robotic task should be executed locally by the robot or offloaded into the cloud server with the intention of energy conservation.
- Knowledge generation for system engineers, practitioners and academics as a reference guide to understand alleviation of robotic computational intensity and survivability of cloud disconnection models.
- Systematic presentation of a worked scenario on energy consumption of SCMR as applied to MRS drivability analysis for engineering practices.

The results from the study also provide a practical and useful framework for cloud robotics that would enhance the ability of individual robots to execute robotic tasks such as path planning, drivability analysis, image processing and navigation in heterogeneous environments when connection to a cloud access point is not possible.

5.5 LIMITATIONS AND FUTURE WORK

The overall objectives of the study set out in Chapter 1 of the dissertation were accomplished. However, the proposed SCMR framework could be improved upon by addressing some of the

limitations identified during the design, implementation and evaluations phases, as well as providing future recommendations.

The SCMR framework prototype developed for this dissertation is intended to demonstrate proof-of-concept implementation, rather than the actual realization of a fully-fledged cloud multi-robotic team environment executing specific robotic tasks. Furthermore, it should be noted that the prototype was implemented and tested only in the Java environment and not in other environments such as Microsoft .NET.

It should also be noted that other cloud robotic concepts such as resource sharing, machine sharing and learning and robotic hardware are beyond the scope of this study. In addition, the area of security measures for protecting robotics systems against malicious intent has been well researched within the scientific community and has therefore not been considered as part of this dissertation.

The findings from this research study provide a reference guide to understand the different cloud robotic models and provide a framework that can be extended to environments such as underground terrains and mines for future research in order to make the existing models and frameworks more efficient, scalable and reliable. Future work can be done on conserving the robots' energy using different offloading schemes and possibly also looking at the compression of images and pre-processing of images prior to the robot offloading the data to the cloud server.

The proposed framework was limited to performing the robotic task of drivable road region detection using SRM to assist with the robotic vision capabilities of the individual robot clients. Future work can be done in extending the robotic tasks performed on the proposed framework to include tasks such as path planning, map building and object recognition.

REFERENCES

[1] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in Proc. IEEE Int. Conf. Robotics and Automation, May 2010, pp. 2308 –2315.

[2] B. Siciliano and O. Khatib, "Springer Handbook of Robotics," in Proc. Springer, 2008.

[3] B. Dhiyanesh, "Dynamic resource allocation for machine to cloud communications robotics cloud", Nehru Institute of Engineering and Technology, Kovai, 2010.

[4] K. Goldberg and B. Kehoe, "Cloud robotics and automation: A survey of related work," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2013-5, June 2013. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-5.html. Accessed on 06 Aug 2013.

[5] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for adhoc networks," in Multi Topic Conference, 2001. IEEE INMIC2001. Technology for the 21st Century. Proceedings. IEEE International, 2001, pp. 62 - 68.

[6] R. Doriya. P. Chakraborty and G. C. Nandi, "Robot-cloud: A framework to assist heterogeneous low cost robots", Robotics & Artificial Intelligence Laboratory, Indian Institute of Information Technology, 2012 International Conference on Communication, Information & Computing Technology (ICCICT), Oct. 19-20, Mumbai, India

[7] G. M. Hunziker-Dominique, "The roboearth cloud engine." [Online]. Available: http://doc.roboearth.org/rce. Accessed on 06 Aug 2013.

[8] M. Sato, K. Kamei, S. Nishio, and N. Hagita, "The ubiquitous net-work robot platform: Common platform for continuous daily robotic services," in System Integration (SII), 2011 IEEE/SICE Int. Symp., December 2011, pp. 318 –323.

[9] N. Fernando, S. W Loke and W. Rahayu, "Mobile cloud computing: A survey," Department of Computer Science and Computer Engineering, La Trobe University, Australia, 2013, pp 7-28.

[10] R. Arumugam, V. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. Kumar, K. D. Meng and G. W. Kit, "Davinci: A cloud computing framework for service robots," in Robotics and Automation (ICRA), 2010 IEEE International Conference on, may 2010, pp. 3084–3089.

[11] H. Guoqiang, W. P. Tay, and W. Yonggang, "Cloud robotics: Architecture, challenges and applications", IEEE Network, 26(3):21, 2012.

[12] M. Waibel, M. Beetz, 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, "Rapyuta: The RoboEarth Cloud Engine", IEEE,18(2), pp. 69–82, 2011.

[13] Hadoop map reduce framework [Online]. Available: http://hadoop. apache.org/hdfs/. Accessed on 14 July 2013.

[14] E. Eade and T. Drummond, "Scalable monocular SLAM," IEEE Int. Conf. on Computer Vision and Pattern Recognition, 1, pp. 469–476, 2006.

[15] NAO Robot [Online]. Available: http://www.aldebaran-robotics.com/. Accessed on 14 July 2013.

[16] D. Babu and P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," Journal of Applied Soft Computing, 2013, pp 2292 – 2303.

[17] V. Namboodiri and T. Ghose, "To cloud or not to cloud: A mobile device perspective on energy consumption of applications," Department of Electrical Engineering and Computer Science, Wichita State University, 9(4):566–581, April 2010.

[18] R. Buyya, C. S. Yeo and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities." Department of Computer Science and Software Engineering, 2009, pp. 9.

[19] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" IEEE Computer, 43(4), April 2010, pp. 51-56.

[20] N. Cao, "Secure and reliable data outsourcing in cloud computing," Technical Report, Worcester Polytechnic Institute, Department of Electrical Engineering and Computer Science, 2012, pp 20-34.

[21] H.E. Schaffer, "X as a service, cloud computing, and the need for good judgment", IT Professional, 11(5), Sept-Oct. 2009 pp. 4 - 5.

[22] Cloud Computing and SOA, Geoffrey Raines, [Online]. Available: http://www.mitre.org/work/tech_papers/tech_papers_09/09_0743/09_0743.pdf. Accessed on 17 July 2013.

[23] Y. Jadeja and K. Modi, "Cloud computing - Concepts, architecture and challenges," International Conference on Computing, Electronics and Electrical Technologies [ICCEET], 2012, pp 3-4.

[24] C. Pelletingeas, "Performance evaluation of virtualization with cloud computing," Technical report, Edinburgh Napier University, 2010.

[25] R. Griffith, A. D. Joseph R, Katz A. Konwinski G, Lee D. Patterson A, Rabkin I, Stoica M, Armbrust, A. Fox and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing" [white paper], 2009.

[26] H. Aliakbarpour, P. Freitas, J. Quintas, C. Tsiourti and J. Dias, "Mobile robot cooperation with infrastructure for surveillance: Towards cloud robotics," Institute of Systems and Robotics, University of Coimbra, 2012.

[27] P. J. Vashi, "Cloud robotics: An emerging research discipline," Technical report, Intelligent embedded Systems, Mälardalen University, 2013.

[28] B. Dhiyanesh, "Dynamic resource allocation for machine to cloud communications robotics cloud", Nehru Institute of Engineering and Technology, Kovai, 2010.

[29] C. Hsu and F. Lian and C. Huang and Y. Chang. "Detecting drivable space in traffic scene understanding." Proceedings of the IEEE International, 2012.

[30] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in ICRA Workshop on Open Source Software, 2009.

[31] M. Sato, K. Kamei, S. Nishio, and N. Hagita, "The ubiquitous network robot platform: Common platform for continuous daily robotic services," in System Integration (SII), 2011 IEEE/SICE Int. Symp., Dec 2011, pp. 318 –323.

[32] A. Staranowicz and G. L. Mariottini, "A survey and comparison of commercial and open-source robotic simulator software," Technical report, ASTRA Robotics Lab, Dept. of CSE, University of Texas at Arlington, 2011, pp. 3-9.

[33] Y. Chen, Z. Du, and M. García-Acosta, "Robot as a service in cloud computing," in 5th IEEE Int. Symp. Service Oriented Syst. Eng. (SOSE), 2010, pp. 151–158.

[34] K. Kamei, S. Nishio, N. Hagita, and M. Sato, "Cloud networked robotics," IEEE Network Mag., 26(3), pp. 28–34, May–Jun. 2012.

[35] V. Trifa, C. Cianci, D. Guinard, "Dynamic control of a robotic swarm using a service-oriented architecture", Proceedings of International Symposium on Artificial Life and Robotics. Beppu, Japan, January 2008.

[36] P. Tröger, A. Rasche, F. Feinbube, and R. Wierschke. "SOA meets robots - A service-based software infrastructure for remote laboratories" In Proceedings of the 2nd International Workshop on elearning and Virtual and Remote Laboratories, Germany, February 2008, pp.57--62.

[37] L. Vasiliu, B. Sakpota, and Hong-Gee Kim. "A semantic web services driven application on humanoid robots." In Software Technologies for Future Embedded and Ubiquitous Systems, 2006 and the 2006 Second International Workshop on Collaborative Computing, Integration, and Assurance. SEUS 2006/WCCIA 2006. The Fourth IEEE Workshop on, page 6 pp., April 2006.

[38] N. Fernando, S. W Loke and W. Rahayu. "Mobile cloud computing: A survey," Department of Computer Science and Computer Engineering, La Trobe University, Australia, 2013.

[39] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg. "Cloud-based robot grasping with the Google Object Recognition Engine.". Submitted to IEEE International Conference on Robotics and Automation, 2013.

[40] C. R. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "LabelMe: A database and web-based tool for image annotation," International Journal of Computer Vision, 77(1-3):157–173, October 2007.

[41] Willow Garage. Personal Robot 2 (PR2), [Online]. Available: www.willowgarage.com. Accessed on 15 June 2013.

[42] M. Tenorth, A. C. Perzylo, R. Lafrenz, and M. Beetz, "The RoboEarth language: Representing and exchanging knowledge about actions, objects, and environments," In IEEE International Conference on Robotics and Automation, number 3, IEEE, May 2012, pp. 1284–1289.

[43] J. J. Green and K. Hlophe and J. Dickens and R. Teleka and M. Price, "Mining robotics sensors", International Journal of Engineering and Advanced Technology (IJEAT), 1(4), April, 2012, pp. 8-15

[44] A. Hemami and F. Hassani, "An overview of autonomous loading of bulk material," In 26th International Symposium on Automation and Robotics in Construction, ISARC, November 2009, pp 405–411.

[45] A. Boeing, T. Bräunl, R. Reid, A. Morgan and K Vinsen, "Cooperative multi-robot navigation and mapping of unknown terrain," Air Force Research Laboratory, under agreement number FA2386-10-4024U.S, 2011.

[46] C. Yinka-Banjo, O. Falola and I. Osunmakinde, "Autonomous multi-robot behaviors for safety inspection under the constraints of underground mine terrains," Ubiquitous Computing and Communication Journal, 2010.

[47] R. Nock and F. Nielsen. "Statistical region merging," IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(11), November 2004, pp. 1-7.

[48] X. Wang and J. Wu, "Remote sensing image segmentation based on statistical region merging and nonlinear diffusion," 2nd International Asia Conference on Informatics in Control, Automation and Robotics, 2010.

[49] M.E. Celebi, H. A. Kingravi, J. Lee and W. Stoecker, J.M. Malters, H. Iyatomi, Y.A. Aslandogan, R. Moss and A.A. Marghoob, "Fast and accurate border detection in dermoscopy images using statistical region merging," TexasWorkforce Commission James

A. Schlipmann Melanoma Cancer Foundation and NIH (SBIR 2R44 CA-101639-02A2), No. 16, 2006, pp. 1-10.

[50] O. E. Falola, "Minimisation of sparse pixels in colour feature extraction: An assistive pre-processing strategy for drivable region detection for robotic vehicles," African Institute for Mathematical Sciences (AIMS), 2010.

[51] J. Minguez and L. Montano, "Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios," Journal of Robotics and Autonomous Systems, 52, 2005, pp. 290–311.

[52] A. V. Nefian and G. R. Bradski, "Detection of drivable corridors for off-road autonomous navigation", Technical report, Application Research Laboratory, Intel Corp., Santa Clara, CA, 2012.

[53] A. M. Neto, A. C. Victorino, I. Fantoni and J. V. Ferreira, "Real-time estimation of drivable image area based on monocular vision," IEEE Intelligent Vehicles Symposium (IV), Gold Coast : Australia, 2013.

[54] C. Khare and K. K. Nagwanshi, "Image restoration technique with non linear filter," International Journal of Advanced Science and Technology, 39, 2012.

[55] P. Milanfar, "A tour of modern image filtering," Technical report, Electrical Engineering Department, University of California, Santa Cruz CA, 95064 USA, 2011

[56] A. Buades, B. Coll, and J. Morel, "A non-local algorithm for image denoising," CVPR, June 2005, pp. 60–65.

[57] O. Falola, I. Osunmakinde and A. Bagula, "Supporting drivable region detection by minimising salient pixels generated through robot sensors," Council for Scientific and Industrial Research (CSIR), 2011.

[58] J. F. Canny. "A computational approach to edge detection". IEEE Trans. Pattern Anal. Machine Intell.,PAMI-8(6), 1986, pp. 679-697.

[59] M. Raman and A. Himanshu, "Study and comparison of various image edge detection techniques," International Journal of Image Processing (IJIP), 3(1), pp. 1-11, 2009.

[60] J. Matthews, "An introduction to edge detection: The Sobel edge detector," [Online]. Available: at http://www.generation5.org/content/2002/im01.asp. Accessed on 17 June 2013.

[61] L. G. Roberts. "Machine perception of 3-D solids" ser. Optical and Electro-Optical Information Processing. MIT Press, 1965.

[62] R. Maini and H. Aggarwal, "Study and comparison of various image edge detection techniques," in International Journal of Image Processing, 3(1), May 2010, pp. 10 –14.

[63] O. Falola, I. Osunmakinde and A. Bagula, "Autonomous robots' visual perception in underground terrains using statistical region merging," School of Computing, College of Science, Engineering and Technology, University of South Africa, South Africa, 2012.

[64] A. M. Neto, and L. Rittner, "A simple and efficient road detection algorithm for real time autonomous navigation based on monocular vision," Proceedings of the IEEE, 2006.

[65] M. Bertozzi, A. Broggi and A. Fascioli, "Vision-based intelligent vehicles: State of the art and perspectives", Robotics and Autonomous Systems 32, 1-16, 2000.

[66] I. Ulrich and I. Nourbakhsh, "Appearance-based obstacle detection with monocular colour vision", Proceedings of the AAAI National Conference on Artificial Intelligence, 2000, pp. 866-871.

[67] A. M. Neto, L. Rittner, N. Leite, D. E. Zampieri, R. Lotufo and A. Mendeleck, (2007), "Pearson's correlation coefficient for discarding redundant information in real time autonomous navigation systems", Proceedings of the IEEE MSC 2007.

[68] N. Otsu, "A threshold selection method from gray-level histogram", IEEE Transactions on Systems, Man, and Cybernetics, 1978,9, pp. 62-66.

[69] J. Zhang and J. Hu, "Image segmentation based on 2D Otsu Method with histogram analysis", IEEE Computer Society, 2008 pp. 104-114.

[70] P. K. Sahoo, S. Soltani, and A. K. C. Wong, "A survey of thresholding techniques", Computer Vision Graphics Image Processing, 41, 1988, pp. 233-260.

[71] U. S. Lee, Y. S. Chung and H. R. Park, "A comparative performance study of several global thresholding techniques for segmentation", Computer Vision, Graphics, and Image Processing, 1990.

[72] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation", Journal of Electronic Imaging; 13, 2004, pp. 146-165.

[73] P. Urcola and L. Montano, "Adapting robot team behavior from interaction with a group of people," IEEE/RSJ International Conference on Intelligent Robots and Systems September 25-30, 2011.

[74] G. L. Zadka, "The twisted network framework," [Online]. Available: http://twistedmatrix.com/user/glyph/ipc10/paper.html. Accessed 15 July 2013.

[75] D. Shah, "Gossip algorithms," in Foundations and Trends in Networking, 3(1), 2008.

[76] L. Wang, M. Liu, Q. H. Max and R. Siegwart, "Multi sensor Data Retrieval in Cloud Robotic System" Department of Electronic Engineering, The Chinese University of Hong Kong, Autonomous Systems Lab, ETH Zurich, Switzerland, 2012.

[77] D. Borthakur, "The hadoop distributed file system: Architecture and design," [Online]. Available: http://hadoop.apache.org/common/docs/r0.18.0/hdfs_design.pdf. Accessed 29 July 2013.

[78] Apache.org, "Hadoop map reduce framework," [Online]. Available: http://hadoop. apache.org/hdfs. Accessed on 29 July 2013.

[79] D. J. DeWitt and M. Stonebraker, "Mapreduce: A major step backwards," [Online]. Available: http://www.hpts.ws/papers/2009/session10/shekita.pdf. Accessed on 29 July 2013.

[80] I. Fette and A. Melnikov, "The WebSocket Protocol, RFC 6455," [Online]. Available: http://tools.ietf.org/html/rfc6455. Accessed 30 July 2013.

[81] Y. Chen, Z. Du, and M. García-Acosta, "Robot as a service in cloud computing," in 5th IEEE Int. Symp. Service Oriented Syst. Eng. (SOSE), 2010, pp. 151–158.

[82] J. M. Rabaey, "Digital integrated circuits," Prentice Hall, 1996.

[83] K. Kumar and Y. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" IEEE International Symposium on Circuits and Systems, 2010.

[84] T. Burd and R. Broderson, "Processor design for portable systems," Journal of VLSI Singapore Process, 13(2), 1996, pp. 203-222.

[85] J. Lee and N. Jindal, "Delay constrained scheduling over fading channels: Optimal policies for monomial energy-cost functions," IEEE International Conference on Communications (ICC), Dresden, Germany, June 2009.

[86] J. Lee, and N. Jindal, "Energy-effcient scheduling of delay constrained traffic over fading channels," IEEE Trans. Wireless Communications, 8(4), 2009, pp. 1866-1875.

APPENDICES

Appendix A – Robot Client Helper Class

The following is a snapshot of the Java code for the Robot Helper class from the SCMR prototype implementation, which was used to manage the robot client connections to the cloud server.

RobotHelper.ja... | RobotCoudServ... | RobotCloudServ... | ImageOperatio... | SRM.java | OtsuBinarize.j... | ThresholdMana... | CollectionAcc... | CreditNoteDeta...

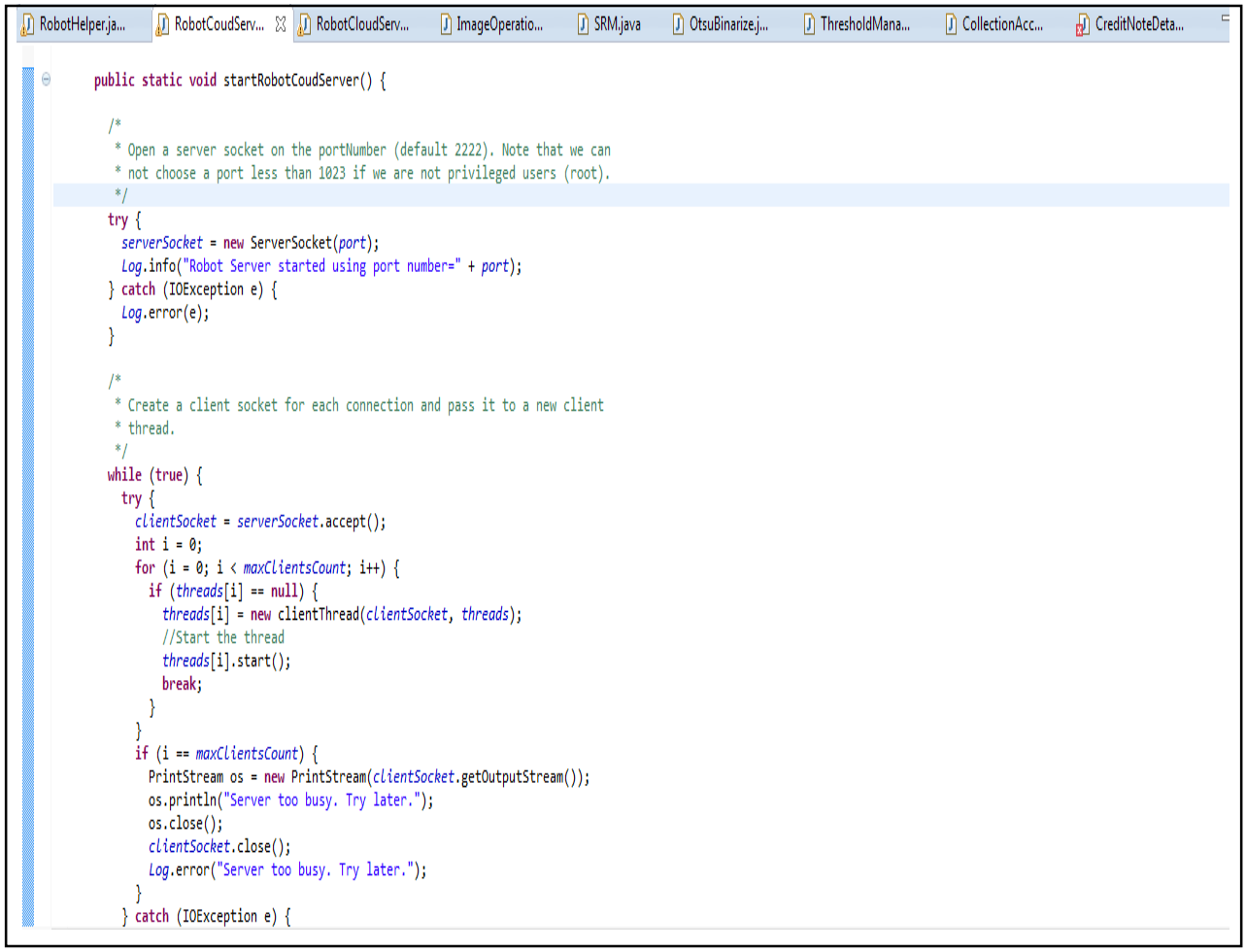
```
/*
 * Method to establish socket connection to the robot server from the robot client.
 */
public static Socket openRobotClientSocketConnection(String host, int port){
    /*
     * Open a socket on a given host and port.
     */
    try {
        clientSocket = new Socket(host, port);
        } catch (UnknownHostException e) {
          System.err.println("Don't know about host " + host);
        } catch (IOException e) {
          System.err.println("Couldn't get I/O for the connection to the host "+ host);
        }
    return clientSocket;
}

/*
 * Method to establish data output stream for robot client to send buffered data to the robot server
 */
public static DataOutputStream openRobotClientDataOutputStream (Socket clientSocket){
    /*
     * Open Robot Client Data Output Stream
     */
    try {
        dataoutputttream = new DataOutputStream(clientSocket.getOutputStream());
        } catch (IOException e) {
          System.err.println("IOException " + e);
        } catch (Exception e) {
          System.err.println("Exception "+ e);
        }
    return dataoutputttream;
}

/*
 * Method to establish data input stream for robot client to receive buffered data from the robot server
 */
```

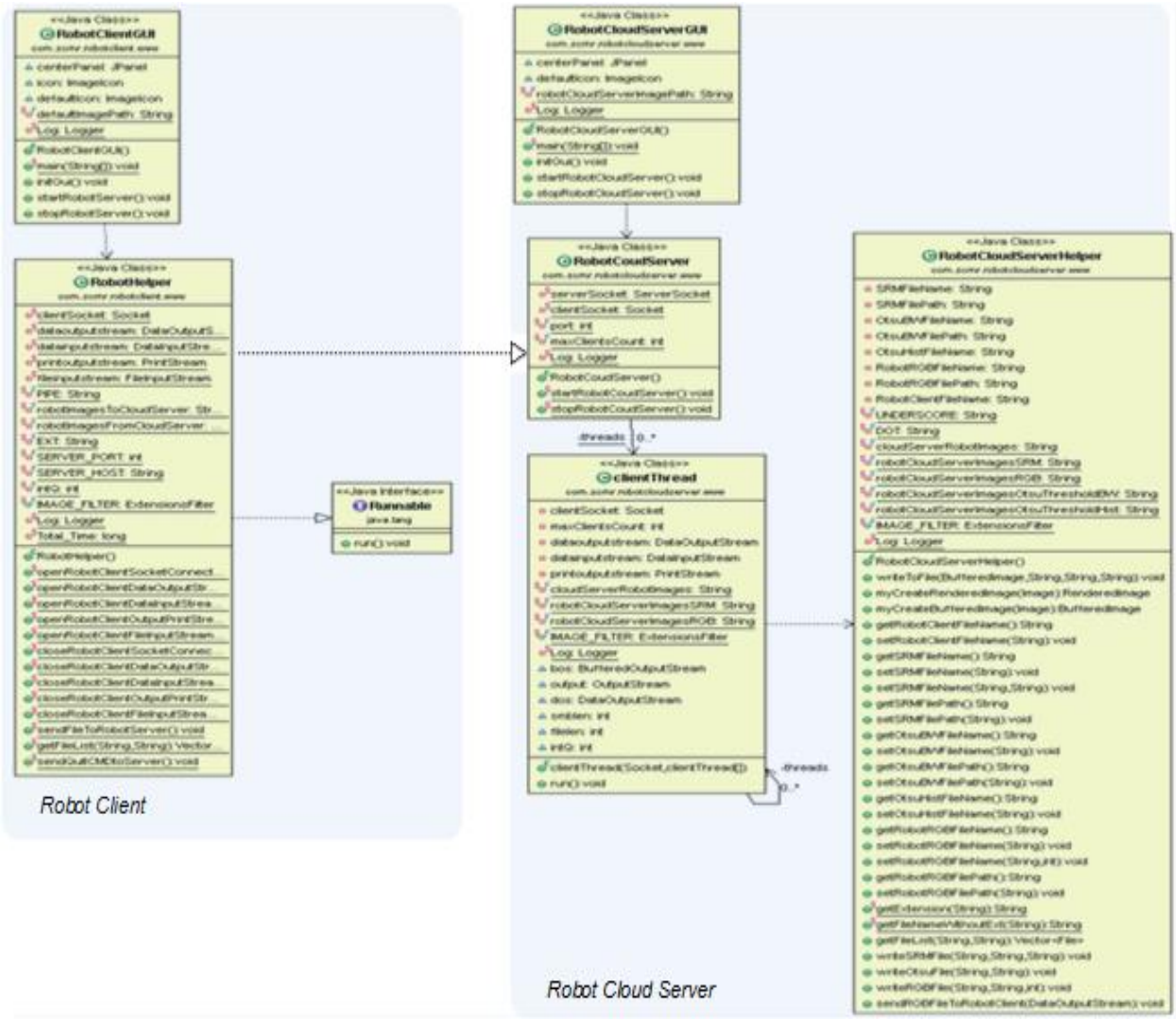
Appendix B – Cloud Server Client Helper Class

The following is the Java code for the Cloud Server Helper class from the SCMR prototype implementation, which was used to process the multiple robot client requests.



Appendix C – Proposed SCMR Framework Modelled in Unified Modelling Language

The following is a snapshot of the proposed SCMR framework in a UML diagram generated using Eclipse, which was the IDE used to develop the prototype framework.



Appendix D – Log4J Logging on the Proposed SCMR Framework

Apache Log4J was used for logging onto the proposed SCMR framework prototype. The log files were stored locally on the application. A new log file is created daily for the logging of the client/server application. The naming standard for the log file is: /SCMR/LOG/SCMR.'yyyyMMdd'.log. The properties file for the log4j is illustrated below:

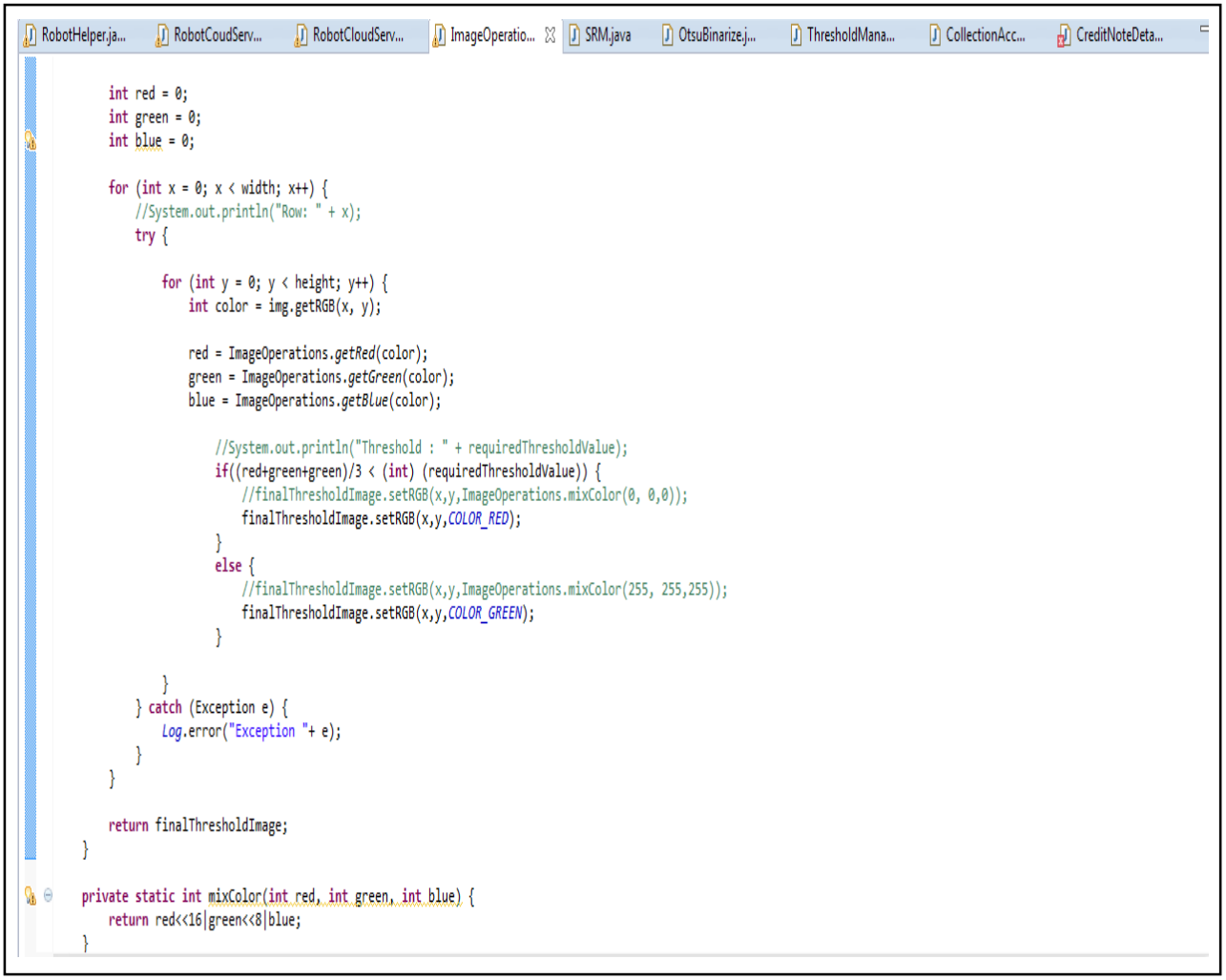
```
# Root logger option
log4j.rootLogger=INFO, FileAppender, stdout

# Direct log messages to stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyyMMdd HH:mm:ss.SSS} %5p %m>%n

# Direct log messages to a log file
log4j.appender.FileAppender=com.scmr.SCMRLog.www.DailyFileAppender
log4j.appender.FileAppender.File='E:/SCMR/LOG/SCMR.'yyyyMMdd'.log'
log4j.appender.FileAppender.BufferedIO=false
log4j.appender.FileAppender.Append=true
log4j.appender.FileAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.FileAppender.layout.ConversionPattern=%d{yyyyMMdd HH:mm:ss.SSS} %5p %m>%n
```

Appendix E – Image Binarization Class

The following is the Java code for the Image Binarization class from the SCMR prototype implementation, which was used to indicate the drivable and non-drivable road regions using a given threshold.



Appendix F – SRM Class

The following is the Java code for the SRM class from the SCMR prototype implementation, which was used to perform the statistical region merging on images submitted to the cloud server by the robot clients.