CCRP: A Novel Clone-Based Cloud Robotic Platform for Multi-Robots

Hamza Aagela

Supervisor: Dr. Violeta Holmes

A thesis submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

University of Huddersfield

United Kingdom

August 2019

Copyright

i The author of this thesis (including any appendices and/or schedules to this thesis) owns any copyright in it (the "Copyright") and s/he has given The University of Huddersfield the right to use such copyright for any administrative, promotional, educational and/or teaching purposes.

ii Copies of this thesis, either in full or in extracts, may be made only in accordance with the regulations of the University Library. Details of these regulations may be obtained from the Librarian. This page must form part of any such copies made.

iii The ownership of any patents, designs, trademarks and any and all other intellectual property rights except for the Copyright (the "Intellectual Property Rights") and any reproductions of copyright works, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property Rights and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property Rights and/or Reproductions.

Dedication

To my parents, sisters and brothers, my Dear wife and my kids who provided me with endless support.

Acknowledgements

Though all things are possible. Therefore, I give special thanks to my God for giving me the ability and persistence to accomplish this Research. I am grateful to my supervisor Dr. Violeta Holmes for her precious support during my PhD journey.

I would like to express my most sincere gratitude to my dear wife Aisha Al Debri for the countless support and the encouragement, where she was my source of motivation that allows me to complete this research. Also, I would like to thank all my family for their continuous support, especially to my Father 'Yousif' and my Mother 'Sabah', my lovely kids 'yamin', 'Arkan' and upcoming ones.

I would like to thank all of my friends for their supports, specially, my colleague 'Dr. Josh Higgins'. Finally, I would also like to thank the School of Computing and Engineering for providing all the needed facilities and equipment to conduct this research.

Abstract

Recently, the cloud computing paradigm has evolved from various research fields. A new path of research, cloud robotics, has emerged which allows robots to inherit the enormous computing and storage capability of cloud. Advances in cloud computing technologies, networking, parallel computing and other evolving technologies, and the integration with multi-robot systems, make it possible to design systems with new capabilities.The main advantages of cloud robotics are in overcoming the limitations of on-board robot computing and storage capabilities and in improving energy efficiency. Nevertheless, there is a lack of cloud robotics frameworks that can provide a secured environment for multi-robot application. The implementation of a robust cloud robotic platform capable of handling multi-robot applications has been shown to be challenging.

This research proposes a novel Clone-based Cloud Robotic Platform architecture (CCRP) which assigns a Virtual Machine (VM) clone of each individual robot's operating system in the cloud, enabling fast and efficient collaboration between them via the cloud's inner-network. The system utilises Robot Operating System (ROS) as a middleware and programmable environment for robot development. This model is using the OpenVPN as a communication protocol between the robot and the VM, which provides considerable enhancement for the security and additional network for the system to allow multi-master ROS deployment. The Quality of Service (QoS) for the system has been tested and evaluated in terms of performance, compatibility and scalability via compari-

son study, which examines the CCRP performance against a local system and a proxy-based cloud system.

Two case studies have been deployed for different robot scenarios. Case study 1 was focused on a navigation task which includes the process of mapping and teleoperation implemented in Google public cloud. The real time response has been examined by using the CCRP to teleoperate the NAO and Turtlebot robots. A response time and video streaming delays were measured to assess the overall QoS performance. Case study 2 is composed of a face recognition task performed using the CCRP in a private cloud on an Openstack platform. The objective of this task was to evaluate the system ability of running the tasks in the cloud effectively and to assess the collaborative learning capability. During the CCRP development and deployment stages an optimization study was conducted to determine optimal parameters for data offloading to the cloud and energy efficiency of a low-cost robot.

The result of the CCRP performance evaluation proved that it is capable of running on a public and private cloud platform for self-configuring and programmable robotic systems, as well as executing various applications on different robot types. The CCRP is facilitating the improvements to QoS performance, compatibility and scalability and is providing a secure cloud computing environment for on-board robots.

List of Publications

Journal Publications:

1. Aagela, H., Holmes, V.: *"Novel clone-based cloud robotic model to overcome limitation of the multi robots QoS."*, Robotics and Autonomous Systems, Elsevier (Under Review)

2. Higgins, J, Al-Jodi, T, Aagela, H., Holmes, V.: *"Inspiring the Next Generation of HPC Engineers with Reconfigurable, Multi-Tenant Resources for Teaching and Research*", Journal of education, SAGE Journals. (Under Review)

Conference Publications:

1. Aagela, H., Holmes, V.: *"Collaborative Cloud-based Face recognition approach for Humanoid robots."*, EMiT19, ISBN: 978-0-9933426-2-6, University of Huddersfield, Huddersfield, April 2019, UK.

2. Aagela, H., Holmes, V.: *"Cloud Robotics-Based System for Robot Teleoperation."*, EMiT19, ISBN: 978-0-9933426-2-6, University of Huddersfield, Huddersfield, April 2019, UK.

3. Al-Aqrabi, H., Hill, R., Aagela, H., Holmes, V.: *"Securing Manufacturing Intelligence for the Industrial Internet of Things."*, Springer, ICICT2019, Brunel, London, UK.

4. Aagela, H., Al-Jodi, T., Holmes, V.: *"Web-based Wireless Wake-on-LAN approach for Robots.",* IEEE, ICAC'18, Newcastle University, Newcastle upon Tyne, September 2018, UK.

5. Aagela, H., Al-Nesf, M., & Holmes, V.: *"An Asus_xtion_pro based indoor MAPPING using a Raspberry Pi with Turtlebot Robot."*, IEEE, ICAC'17, University of Huddersfield, Huddersfield, September 2017, UK.

6. Aagela, H., Holmes, V., Dhimish, M., & Wilson, D.: *"Impact of Video Streaming Quality on Bandwidth and Face Recognition Accuracy in Humanoid Robot NAO."*, ICC '17, March 2017, Cambridge, UK.

Poster Publications:

1. Antoniades, A., Aagela, H., Holmes, V.: "Black-Box/Tracking System for Drones using *LoRa"*, ISBN: 978-0-9933426-2-6, University of Huddersfield, Huddersfield, April 2019, UK.

Contents

List of Figures

List of Tables

Nomenclature

2D	two dimensions
3D	Three Dimensions
α	Cronbach's alpha
API	Application Programming Interface
CCRP	Clone-based Cloud robotics Platform
CD	Contact Dynamics
CPS	Cyber-Physical System
CRI	Clone ROS Images
FPS	Frames Per Second
FR	Face Recognition
GCP	Google Cloud Platform
IaaS	Infrastructure as a service
JSON	JavaScript Object Notation

LAN	Local Area Network
M2C	Machine to Cloud
Nc	Common Network
PaaS	Platform as a service
PCA	Principal Component Analysis
PKI	Public Key Infrastructure
QoS	Quality of Service
REST	Representational State Transfer
ROS	Robot Operating System
RTT	Round-trip delay time
SaaS	Software as a service
SDN	Software Defined Network
SLAM	Simultaneous Localisation and Mapping
SPSS	Statistical Package for the Social Sciences
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VM	Virtual Machine
VPN	Virtual Private Network
Wi-Fi	Wireless Fidelity

Chapter 1

Introduction

1.1 Overview

Cloud robotics as a concept is a relatively new paradigm which opens various research opportunities in the robotics field. Certainly, the remote support for computing, accessibility and the availability of storage create an attractive research area that works on enhancing the on-board robot capability for processing complex tasks. Whilst networked robotics and web support in robotic systems can be dated to the end of the 1990s, the first practical cloud robotics project, RoboEarth, was introduced at the middle of 2009 (Waibel et al., 2011). The concept of cloud robotics is gradually earning research interest due to the fast growing cloud computing technology. The term of Cloud computing could refer to a distributed virtualised computing environment, or "cloud service", that is utilised in order to enhance the performance of a robot accomplishing complex tasks. For example, drawing a 3D model for an object is one of the

current challenges for on-board robots, or to find the path direction in random areas (Doriya, Chakraborty, & Nandi, 2012a). (Agüero et al., 2015) stated that the use of a parallel system within the cloud is required in order to provide the robot with real-time operation. All the contemporary cloud providers (Azure, Amazon, Google Cloud Engine) can provide such massively parallel cloud systems. In terms of the cloud service architectures available, there are three different types, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Each one of those services has its own features and implementation, where each cloud service comes with a certain level of management and control.

Indeed, the impact of cloud on several computer systems and services drives the transition of systems architecture and business models to be cloud-based rather than on-board or local network-based services. Therefore, the robotics field has a potential to inherit the advantages provided by the cloud such as low maintenance cost, scalability, large storage and high computational power. Generally, the on-board robot is still limited in terms of computational power and storage. Therefore, the need of offloading intensive computational processes to an external computing system become higher, which is able to process the robot sensor data and feed an action or result back to the robot, as well as managing the process of storing and sharing knowledge. As shown in Figure 1.1 the key components of cloud robotics infrastructure consist of distributed compute and storage resources, which are provided to various type of robots in order to help them in performing robot tasks in the cloud side or to connect with other robots.

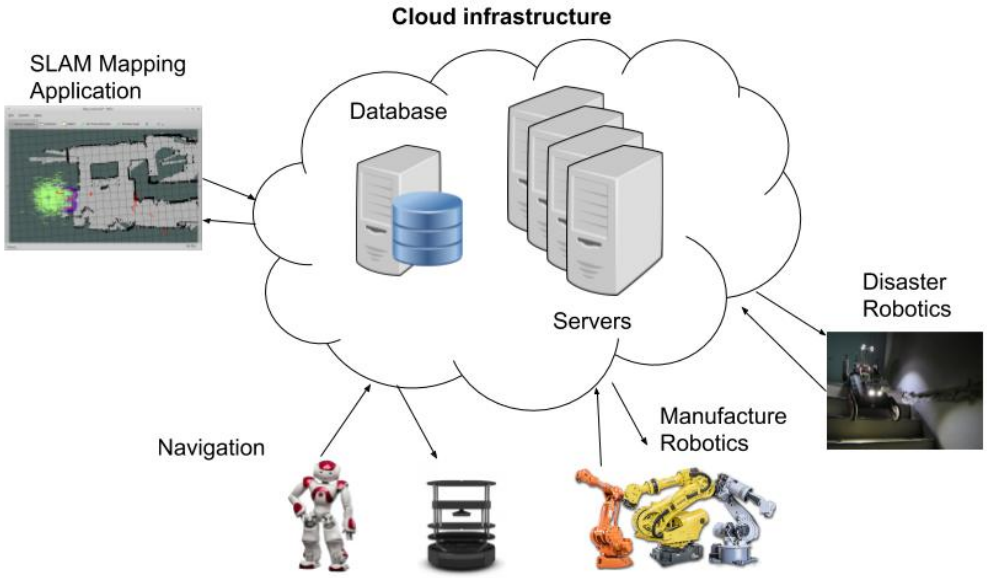


Figure 1.1: General High-level overview of cloud robotics system architecture and applications

(Hu, Tay, & Wen, 2012)

The environment of multi-robots profits the most from the cloud robotics concept as it allows the collaboration in more scalable way, where the robot is not restricted by its geographical location. Cloud services could be utilised as a source of robot knowledge and facilitate sharing and obtaining data between each other. This open the opportunity for robots to learn from other robot experiences, for instance, by collecting data from surrounding environments, sharing conditions, related control policies and path plans. Moreover, they can share the data on the outcome performance, providing potential for the cloud to improve the performance of the robot and automation systems around the world (Doriya et al., 2012a).

The concept of cloud robotics is gradually reaching its peak. With respect to contemporary literature, the RoboEarth project released a "world wide web"

serving robots from a huge distributed database and repository, as shown in Figure 1.2. This provides a Cloud Robotics infrastructure that includes a centralised database to keep knowledge produced by humans or robots and transfer it into a machine-readable format. The database kept in the RoboEarth knowledge base contains various elements, such as maps, object recognition models, locations and task knowledge (action recipes and manipulation plans) (Waibel et al., 2011).

The RoboEarth has a Cloud Engine called Rapyuta which is designed and developed to provide a cloud computation environment for on-board robots to offload some of their tasks with minimal configuration. The Rapyuta platform provides access to the RoboEarth knowledge repository and database (Hunziker, Gajamohan, Waibel, & D'Andrea, 2013).

Due to the high demand for cloud resources as a service provider, the computer industry has a direct influence and different application-service modules are available that serve robotic technologies. For example, DAvinCi is implementing SLAM (Simultaneous Localisation And Mapping) using a Software-as-a-Service (SaaS) cloud model (Arumugam et al., 2010).

A project called CORE also utilises a SaaS cloud model for proposing a solution for object recognition. Rapyuta, described above, uses Platform-as-a-Service (PaaS) to create the centralised sharing of information as a cloud architecture for robotics applications.

The concept of Cloud Robotics comes with challenges and limitations that are facing the system developers and researchers. These challenges are divided

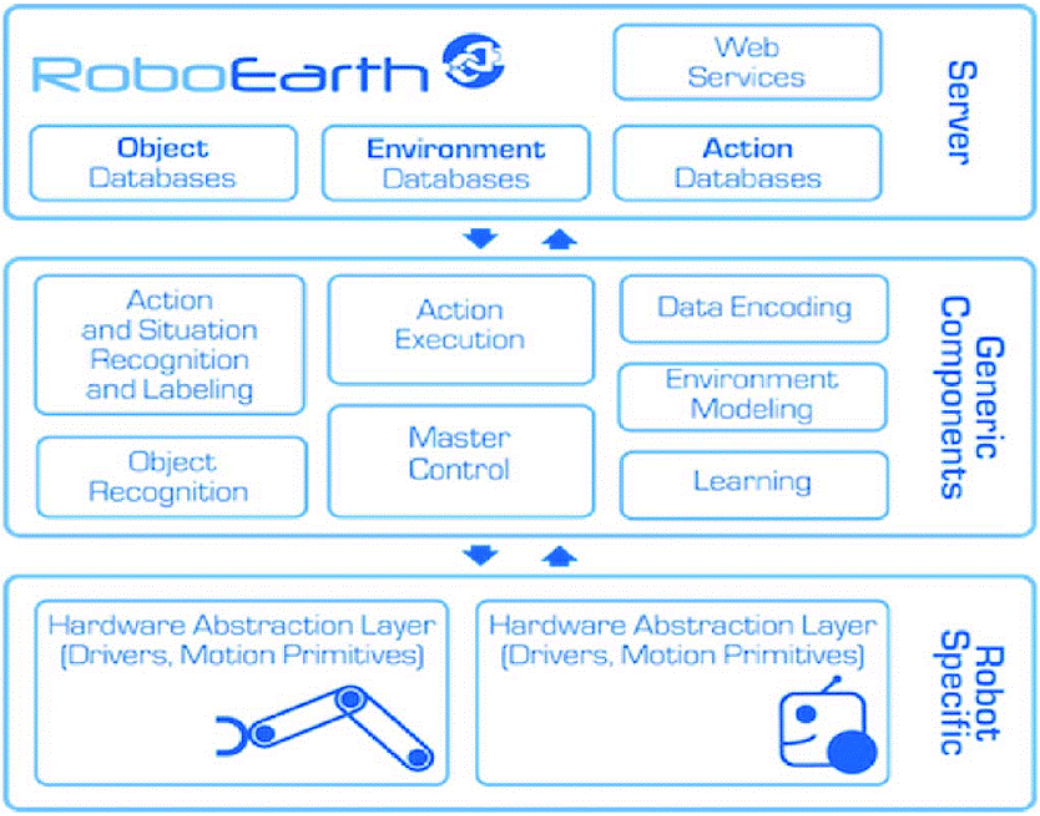


Figure 1.2: The RoboEarth system architecture
(Waibel et al., 2011)

into several categories and outlined below:

1.1.1 Communication

The network interconnect between one or more robots and a cloud service is a major factor in any cloud robotics system, which has a potential impact on the performance of the systems that will not only depend on the processing power, but also on the network speed and the ability to provide real-time communications, the network speed should provide a network latency less than 200 ms in order to be capable to support a real-time applications. In addition,

the selection of a suitable network protocol for communication with the cloud service is a critical factor (Hu et al., 2012).

1.1.2 Heterogeneity of robots

The challenge of sharing knowledge amongst robots that have various hardware components and physical form factors leads to common problems in robot planning, such as sharing the robot. If a robot shares a model plan with other robots, the plan might not be practical for the other one to utilise if it has different capabilities (Kehoe, Patil, Abbeel, & Goldberg, 2015).

1.1.3 Knowledge representation

Sharing knowledge between various robot platforms will require that knowledge to be well-defined with a standard data representation which is understandable from any robot (Doriya et al., 2012a). A representation of the data has to provide the robot with a clear format and message protocol. The robot data can be varied in term of its type such as, maps, sensor data, trained object or face images and so on.

1.1.4 Brainless robots

This problem happens when the network connection breaks down. In this case, the robot turns back to being "brainless" unless it has a default behaviour activity once the connection lost (Kehoe et al., 2015).

1.1.5 Security

The concept of Cloud Robotics inherits all the cloud security issues such as an access authorisation, network access between the robot and the cloud, Encryption and so on, where the cloud virtual machines are vulnerable for most hacking activity via its public IP address. In addition, the privacy of the data that is shared by the robots and stored in the cloud is considered to be a major problem as the robotics data can be sensitive. (Kehoe et al., 2015), (Ren, Wang, & Wang, 2012).

There are several points that need to be considered in terms of offloading the data or process to the cloud, such as communication, security, latency and knowledge sharing, and each one of these elements has its own challenges. Thus, in this study the focus is to establish a multi-robot environment cloud robotics approach that tries to overcome current framework limitations. This work develops and examines a new clone-based cloud robotics model that allows various robot platforms to offload intensive processes to a cloud platform, and allows the robot to share and store data and knowledge on the cloud. The major driving forces behind the need of developing a new cloud robotics framework will be carefully analysed based on various aspects: compatibility, availability, license restrictions, robot cooperative learning, ability of sharing knowledge, and network performance.

1.2 Motivation

- The key motivations of this research are driven from the resource limitation of Humanoid robot NAO, in terms of the processing power and storage capacity as well as the limited power resources. According to (Wang, Liu, & Meng, 2012), the data storage is a key problem with the robot, as physical limitations determine the maximum amount of storage capacity that a mobile robot could carry on-board. Increasing the amount of the storage will require increasing the speed of the computational power of the robot, thus, consuming considerable amount of limited battery power.

- The idea of offloading some of the robot tasks to external cloud resources becomes a feasible alternative. However, (Kehoe et al., 2015) stated that the network connection is a key challenge that can face a cloud robotics implementation for any mobile robot, where the communication delay can prevent the robot from obtaining adequate real-time performance.

- Most of the current cloud robotic systems are designed to work with either certain robot types or only supporting a single robot environment. Therefore, the current multi-robot framework is limited in terms of availability, are often proprietary, not available to the researchers and do not support existing robotic systems.

- The currently available ROS-based framework for multi-robot collaborative applications is complex to configure and has serious limitations given the high barrier of entry to be used for this style of task collaboration.

These are the factors which motivate this research study to investigate a self-configurable robust cloud robotics solution that can provide users with single/multi-robot cloud environment, which should be suitable and compatible with all ROS distributions.

1.3 Research Aim and Objectives

The aim of this research is to design a novel clone-based cloud robotics generalised architecture that is compatible with Robot Operating System (ROS) based robots and ROS distributions. This architecture will provide a secure, collaborative and configurable system to allow the robots to offload their onboard computational and memory intensive tasks onto a single/multi-robot cloud computing environment.

Objectives

- Devise the network design and the key feature of the CCRP framework, and profile network performance to enhance the system optimisation.
- Design and deploy multi-robot system in CCRP framework.
- Evaluate the performance of a single and multiple robot population vs a proxy-based ROS-based cloud robotics platform. In addition, assess the performance of the system with practical robot applications by designing and developing two case studies.

- Train the systems to acquire knowledge of robot environment and perform an intelligent task such as face recognition in single and multi-robot systems and navigation in the unknown environments.

1.4 Organisation

The following chapters of this thesis are organised as follows:

Chapter 2 outlines the background and related work , starting with an overview of Cloud Computing, providing information about the deployment methods and cloud service models types. This chapter provides a brief overview of the robot technology and some of its applications. The currently available cloud robotic architectures are identified. The second part of this chapter investigates the previous work and related Cloud Robotics frameworks in terms of their architecture and functionality. Finally, it concludes with critical evaluation of the given related work and summary.

Chapter 3 presents the Robot Operating System (ROS) and breaks down some of its important concepts and structure. In addition, it demonstrates the multi ROS master concept and its unique architecture. The chapter closes with a discussion of using ROS in the cloud.

Chapter 4 discusses the utilised research methodologies and research approach. In addition, it defines the main hardware and software requirements for the research that involves providing an overview of each robot used within this study. This chapter describes the design of the case studies as well as providing

information of the collected data. Finally, it will show the adoptive technique in how the evaluation process of the proposed system performance.

Chapter 5 discusses the network profiling process, the experimental setup of the face recognition system and provides three case studies. Finally, it evaluates the procedure of developing a low-cost robot and analysis of its performance.

Chapter 6 introduces our novel cloud robotics solution, CCRP framework, giving an overview of the framework and providing detailed explanation of the architecture and its network interconnect. Finally, it will demonstrate a structure of the developed API manager.

Chapter 7 discusses the performance of CCRP in terms of the network latency and the QoS performance. In addition, it presents a security evaluation of the system, and the performance of the API manager through analysis of the test results.

Chapter 8 discusses the application of real-time cloud-based navigation application as a case study of using the CCRP. This consists two major parts - the mapping and teleoperation processes - which are devised by explaining the experiment setup and the system design. Finally, an evaluation of the navigation application performance is presented.

Chapter 9 introduces a novel collaborative cloud robotics face recognition (FR) application and an evaluation of the FR application performance.

Chapter 10 provides a conclusion of the accomplished work in this research, summarises all the contributions of the work and includes answers for the given

research questions, identifying opportunities for future work to continue this area of study.

Chapter 2

Background and Related Work

2.1 Introduction

This chapter presents an overview of research related background topics as well as the state of the art of currently available solutions for multi-robots cloud environments. The background research involves several concepts, such as cloud computing paradigm. In addition, it defines some of the related robotic technologies, and the concept of cloud computing and cloud robotic. An overview of the existing work in the area of the cloud robotic and a critical appraisal of recent publish work is presented, from which the research gap can be identified.

2.2 Cloud Computing

The concept of cloud computing has emerged as the definitive paradigm for development of services on-demand and virtualisation technologies. The cloud computing terms are well-defined. The National Institute of Standards and Technology (NIST) states that *"Cloud is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable resources (e.g. Networks, servers, storage, applications, and services), that can be rapidly provisioned and released with minimal management effort or service provider interaction"* (Peter & TIM, 2010).

Another definition by (Jamsa, 2011) states that cloud based systems are characterised by the ability to dynamically and transparently scale up or down virtual resources in terms of size, such as to add or remove processors and disk space, charging the customer whose solution resides in the cloud only for the resource that is consumed. The cloud providers supply the customer with abstractions of computational resources and other services that developers can utilise to implement complex systems without managing physical infrastructure.

The cloud computing deployment methods can be categorised into three methods. First one is Public Cloud, where a third-party company develops a commercial system and provide its cloud services over the internet for the public as a subscription plan or pay-as-you-go. The second type is Private Cloud, which is only available for authorised users who are part of the business or organisation. This concept can be expressed with a LAN (Local Area Net-

work) and WAN (Wide Area Network). The distinction between these cloud deployment methods is not binary, and hybrid cloud deployment methods can be accomplished with a blend of orchestration between the private and public cloud.

2.2.1 Cloud Deployment Models

Public Cloud can be defined as a third-party computing utility or service that is provided to users over the Internet. The end-user is charged based on the usage of storage, CPU cycles and network bandwidth. The key difference between the public cloud and private cloud is that the users are not responsible for any management or maintenance, where the user data and services is hosted in the provider data centres and it is their responsibility for the management and maintenance of the physical infrastructure. Most public cloud vendors promise to deliver high QoS only and assurances on the availability and performance of the provided services. This provides key motivation to research and examine the performance of the QoS with regards to public cloud computing. The public cloud service can be provided in various service form that is explained in Section 2.2.2.

There are several well-known public cloud vendors and the availability of their technologies at scale drives the path research is taking within the paradigm of Cloud Computing. In 2006, Amazon introduced a Public Cloud service to the industry, which provides IaaS and PaaS architectures. The architecture is advertised as two separate projects: the first one is the Amazon Elastic Compute Cloud (EC2) and the second one is Amazon Storage Service (S3).

Then, other company such as Google, IBM, and Microsoft adopted a similar approach in developing their own cloud system with unique features (JoSEP et al., 2010).

Additionally, as a result of the closed source exclusive nature of public cloud, there are some concerns in terms of user security and privacy in terms of hosting a sensitive data and applications by third party service provider that cannot be audited in the public domain. Therefore, some organisation and business go toward developing their own cloud system as private cloud.

Private Cloud is a specific deployment method of cloud computing that establishes a cloud-based system in which only specified users can access and utilise. The definition of the private cloud has a passionate debate, but typically private cloud can be seen as a highly virtualised data centre that is sited within an organisation or business network and provides services for only the authorised customers (Furht, 2010). There are a number of private cloud systems that can be used for deployment that apply various technique and technologies, such as, Openstack, VMware, IBM Cloud Private, Microsoft, Mirantis, Oracle Private Cloud and RightScale. One perceived advantage of using the private cloud is that it solves the privacy issue, allowing the owner to benefit from a great deal of the cloud features, without sharing the data or process with a third party cloud providers.

OpenStack is an open-source software that developed as an associated work among several developers, who merged several Cloud Computing technologists in order to build a cloud platform which can be used to build both public and private cloud infrastructure. The project was initiated by Rackspace and

NASA then it became popular between the cloud developer community, who are working together to establish a highly scalable open source Cloud platform (Sefraoui, Aissaoui, & Eleuldj, 2012).

The OpenStack software stack is composed of three projects that provide functionality for the formation of private clouds. (Rosado & Bernardino, 2014) These projects are as follows:

- Nova is the main compute unit within Openstack, which is responsible for handling the VMs, providing access for the virtual resources through an API.
- Swift is responsible for the OpenStack Object Store tasks, where it allows storage and retrieval of data via an API. Swift is designed to provide robustness and availability for the entire data storage.
- Glance manages VMs and Images. Its image services include various features such as discovering, recording, and recovering VM images. In addition, it has an API, which permits querying of VM image metadata and recovery of the actual image.
- Neutron is responsible for providing networking as a service among virtual network interface devices and real ones. In addition, it is responsible for managing the components and activities related to network function virtualisation.

The Image Service offers the capability of creating snapshots for recording and reproduce VM images. The various Storage APIs and object storage provided by OpenStack is used as a backend for the Image Service, which can be used

to spawn multiple VM instances from template images to create a replaced of the original VM.

2.2.2 Cloud Service Models

This section provides an overview of the main cloud service model categories that can be used as classifier of cloud capabilities and usability, based on the performance, level of control involved and the type of the services provided. The cloud service models are defined as follows:

Software-as-a-Service (SaaS)

SaaS is defined as a supplier for a remote software through a cloud system to users, who have no influence in the infrastructure and usually subscribes to the utility-based charging model of payment. It can provide an online software alternative solution installed and run on cloud remote server, which is typically installed and executed on a local machine. The Microsoft Office 365 and Slack can be used as an instance of the SaaS application (Mell, Grance, et al., 2011).

Platform-as-a-Service (PaaS)

PaaS is defined as an extra layer of abstraction in top of a virtualised infrastructure that act as a provider that offers a software platform. Which is providing the users with tools such as, libraries, programming languages, and frameworks. In order to allow them to design and develop their personal

applications. Examples of PaaS, such as, Amazon web services "AWS" Elastic Beanstalk and Microsoft Azure, include the development of applications that are fully operated on the cloud.(Keller & Rexford, 2010),(Mell et al., 2011).

Infrastructure-as-a-Service (Iaas)

IaaS is defined as a provider that supplies a virtualised computing environment on top of the physical resources, as a virtual machine with various computer capability such as, CPU, GPU, RAM and storage disk. The users have a full control of the cloud resource in the same way as a local machine with more flexibility, but typically cannot utilise the abstractions offered by other cloud models.(Mell et al., 2011), (Serrano, Gallardo, & Hernantes, 2015).

The current market for cloud services is providing various kind of services based on virtualisation. However, it does not explicitly provide a direct solution for robots nor does it offer the appropriate tools to build a robotic application. Therefore, there is a need to provide a solution that utilise the cloud resources to build a framework for a developer to create software which can meet the requirements of robotics applications, and enhance the compatibility with existing robotics framework.

2.3 Robots

A robot can be defined as a machine developed in a cross-disciplinary engineering and science branch including mechanical and electronics engineering, computers, information system (Tzafestas, 2013), IT and other industries. Robotics models, builds, operates and utilises robots as well as the computer systems to control them, provide sensory feedback and process information. The main reason of developing robots is to create a machine which could replace humans and perform complex tasks. Robots can be divided into various form and categories based on the robot appearance and functionality, such as humanoid robots, drones, industrial robots, and underwater robots. (Austin, 2018), (Bogue, 2016). Each robot type is unique in their features and applications. The robot's applications can vary based on the task specification, where some robot task may only require a single robot process, or it can be a companion of robot multi-processes to accomplish single task.

2.3.1 Simultaneous Localisation And Mapping (SLAM)

The Simultaneous Localisation And Mapping (SLAM) within the robots entails complex computational problems that are focused on dealing with building a map of an unknown environment, which is able to track down the location of the robot in the same map (Ghani, Sahari, & Kiong, 2014). Mapping is one of the most common robot tasks where the robot needs to create constant map for the surrounding environment by utilising its sensor data, such as ultrasound and camera laser. The robots are responsible of processing sensor data

as well as storing real-time data in RAM, in order to create an accurate map. This indicates that most methods cannot make consistent maps for huge areas, as a result of the higher computational power and storage required (Tuna, Gulez, Gungor, & Mumcu, 2012). Low cost robots with low on board capability will struggle to fulfil the required tasks. There was a solution proposed by (Mohanarajah, Hunziker, D'Andrea, & Waibel, 2014), who suggested a solution that offloads the SLAM process to a cloud service to overcome the lack of resource availability in the local system. In addition, the generated map can be shared with other robots. Although the cloud can solve the lack in resources, the network latency can have negative impact of the SLAM performance (Ali, Hammad, & Eldien, 2018). However, as the controls sent to the actuators requests by only how they want to change position and move, with high network delay it does not necessarily represent how the robot is moving (Rehman, 2013).

2.3.2 Robot Teleoperation

Although teleoperation robot task was one of the first robot's applications since 1950's, it is still an vital research field until now, which is posing many challenges to industrialists, scientists and researchers. The remote robot manipulation is a major task needs complicated perceptions, decisions and actions to be taken by human operator or autonomous system, utilising the available environment and robot sensors data (Small, Lee, & Mann, 2018). Nowadays, there are several teleoperation systems still depend on a human operators, however, the inclusion of full or part automatic control is now increasing and

become common. Yet, automation rarely completely replaces human operators, because of sensitivity of certain teleoperation tasks, where the current technology is not yet capable to replace fully the actions and intuition of the operator. The real-time robot teleoperation applications have been implemented in various fields, such as transportation, underwater exploration and telesurgery (Song et al., 2018). These teleoperation applications require a very reliable system that can securely control the communication between a robot and the operator, and response to the operator's actions in real-time. However, modern applications need larger processing capability and memory, much more than the resources available in most of robotics system. (Aagela & Holmes, 2019a)

2.3.3 Robot Vision

Robot vision processes visual data from the environment using a combination of camera hardware and computer algorithms. For example, a 2D camera can be installed on a robot system that detects an object for the robot. The use of a 3D stereo camera could be a more complex example to guide a robot to mount rollers on a moving vehicle. The robot vision can be utilised for different robot task such as face or object detection and recognition.

The robot Face Recognition (FR) application is a software based on a computer vision algorithm for the automated detection, identification or verification of an individual from either a digital picture or a video. This is generally performed by humanoid robot, which is comparing targeted facial features of a detected person to a pre-processed facial dataset. It is used usually in secu-

rity systems; however, it became an important task for any humanoid robot. Software for facial recognition is based on the ability to recognise a human by measuring various features of a face. There are several FR approaches which have been developed, for example, Linear Discriminant Analysis (LDA) (Li & Yuan, 2005), and Principal Component Analysis (PCA) (Ismail et al., 2011), (Jolliffe & Cadima, 2016), which are implemented in various studies. The performance of these methods has shown to be acceptable, yet, they have limitations of being computationally intensive (Bolotnikova, Demirel, & Anbarjafari, 2017).

Object recognition is a computer vision technology in order to identify one or more objects in videos or photos. The object recognition process is typically a result of utilising a deep learning algorithm, where it allows the robot to obtain a visual detail about the detected object such as the object colours, name and so on. The process of object recognition can be done either locally (H. Liu, Li, Xu, & Sun, 2018), or by using external computing resource (Cheng, Bier, & Mostafavi, 2017). However, this robotics task can be very challenging in on-board robot as it requires intensive processing power and huge storage, in order to keep all the trained object images. Moreover, the use of the external cloud resources can happen over API, where there are a number of API solution that can be linked with robot to perform object recognition process, such as Google Cloud Vision API and Azure Computer Vision API. (Hosseini, Xiao, & Poovendran, 2017).

2.3.4 Multi-Robot and Multi-Area Management

(Kumar, Pattnaik, & Pandey, 2017) stated that multi-robot and multi-area management are a key issue in most of the current cloud robotics system; managing the cloud resources allocation as well as the network between the cloud to cloud resources (C2C) or between the robot to the cloud (M2C) are challenging issues. In terms of the multi-area management issue, to provide services for the robots, which runs over a wide area and several geographical locations, the cloud robot needs a mechanism to share the data about each area, for example, maps, landmarks and so on. This information could be static like the maps or could be dynamic like the robot location, obstacles, and objects that have the possibility to change their location constantly. It is a general belief that multi-robot systems have more advantages than single-robot systems. The key motivations of emerging multi-robot solutions stem from the real world applications, where most single robot system struggle in dealing with complex tasks effectively. In addition, a multi-robot system can provide parallelism as well as redundancy (Fazli, Davoodi, & Mackworth, 2013).

2.4 Cloud Robotics

The idea of control the robot remotely over the network started at the end of 1994. The first robot was linked to the web by web-based controller that use the browser to remote control the robot (Kehoe et al., 2015). Then, (Inaba, 1997) worked on a project titled “remote brained robots” that defined the advantage

of distant computing in robot management. In 2001, the IEEE Robotics and Automation Society created the Practical Group on Networked Robots and introduced a series of workshops (Doriya et al., 2012a). The RoboEarth cloud project was introduced at the end of 2009. It was designed to be a global network of robots, which is allowing the robot to share data, information and knowledge with other robots. Moreover, the robot could learn the behaviour based on the environment. The RoboEarth research group developed several cloud services and designed a cloud robotic network (Waibel et al., 2011). In 2010, James Kuffner has introduced the term *Cloud Robotics*, which become the standard of any cloud application that is used by robots and has been agreed by many academic groups, as well as the IEEE transactions on automation science and engineering (Kuffner & Robots, 2010).

According to (Waibel et al., 2011) the MyRobots project introduced the idea of a social network for the robots, by applying the same concept of traditional social networks for humans, where robots can cooperate with each other, share the current state and information from its sensors. The RoboEarth and RoboBrain dataset were planned to be dynamic with the updated that comes from connecting robots (Hu et al., 2012).

(Saxena et al., 2014) stated that the ability to "combine knowledge from the Internet sources with finer detail about the physical world" will provide a robot with the significant amount of knowledge, which could help the robot learn about its environment. The project proposed a robot query library, which offers a set of ready retrieval functions. The project tries to create a network of knowledge, such as map information and object details, which could be used

by the robots.

The "Lightning" framework proposes a system for Shared Robot Learning by indexing path from a number of robots over time, utilising the Cloud platform for parallel planning and path modification. These systems could be extended to worldwide networks to ease shared path planning, together with traffic path. In addition, sharing the information could develop the abilities of robots with limited compute resources (Kehoe et al., 2015).

A new cloud robotic paradigm concept expands the idea of networked robotics, suggesting a new research area called Cloud Networked robotics (Kamei, Nishio, Hagita, & Sato, 2012). In addition, (Manzi et al., 2017) improved the cloud robotics service by introducing the concept of Cloud Service Robotics as "the integration of different agents that allows an efficient, effective and robust cooperation between robots, smart environments and citizens".

(Manzi et al., 2017) developed a web service application to serve a robot called KuBo with ROS middleware. The aim of this study was to increase navigation and speech recognition capability for the on-board robot KuBo by creating a smart environments based application that follows the software as a service (SaaS) cloud model. Therefore, the robot uses those services as an external source of knowledge. However, the robot is using its own resource to run the navigation process. Other researchers also utilised the external cloud computing resources to speed up the processing time for intensive robot tasks, for example SLAM algorithms, navigation, object recognition and video and image processing (Gouveia, Portugal, Silva, & Marques, 2015), (X. F. Liu, Shahriar, Al Sunny, Leu, & Hu, 2017).

A generic cloud robotics solution should aim to move the computational robot tasks to a cloud resource via the network using a unified communication protocol, in other words, the offloaded data and information from linked robots must have a unified structure, in order to allow the robots to interact with each other. In the multi-robot cloud robotics environment, the robot system is able to learn and share the experience of other robot behaviour. However, it is seeming that any single robot is isolated, and lacks learning capabilities. Moreover, the programming environment should be familiar for the developer to all the mass adoption and existing robot algorithm and software can be used.

2.4.1 Cloud Robotic Connection Models

In this section, the currently available machine-to-cloud M2C cloud robotic connection models are identified. According to (Hu et al., 2012), There are three main machine to cloud M2C connection models,which are: peer-based model, proxy-based model and clone-based model. Each model has its own feature that defines the level of robustness, interoperability and the mobility. The three elastic models Were define as summarised in Figure 2.1:

a) **Peer-Based Model** - in this model one machine connect to a VM in the cloud, which consider as an additional computer unit. The process could be executed in both the robot or cloud side.

b) **Proxy-Based Model** - in this model, one VM works as an edge device (proxy) for a number of robots, this VM has a single link to each robot.

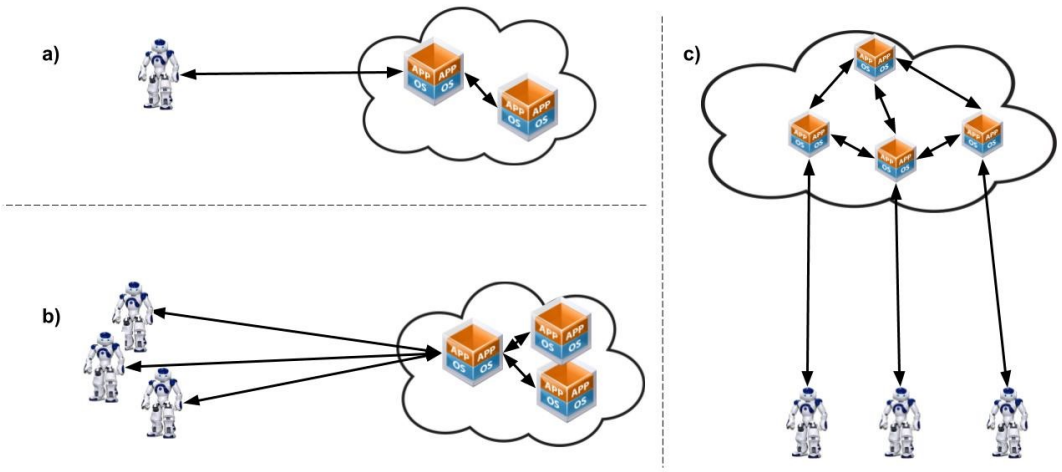


Figure 2.1: Overview of cloud robotic computing models
(Hu et al., 2012)

c) **Clone-Based Model** - in this model each robot links to a VM that works as a clone image in the cloud. The connection between the robot and the cloud use M2C peer-to-peer network. Moreover, a group of clone images form of an ad-hoc M2M network inside the cloud environment.

Each of these cloud robotic computing models demonstrates a different level of robustness in its network links and QoS, as illustrated in Table 2.1. The shown level is based on model capability of offloading intensive process to the cloud. Robustness defines the strength of network connectivity between a robot or a set of robots and the cloud infrastructure.

The clone-based model obtains the highest number of links between the robots and the cloud infrastructure, therefore, it is the most robust model. On the other hand, the proxy-based model has the lowest robustness in regards of network availability. QoS defines the quality of the provided service by the

Model Type	Robustness	Interoperability	QoS
Peer-based	Medium	Low	High
Proxy-based	Low	High	Low
Clone-based	High	Medium	High

Table 2.1: Comparisons of different cloud robotic models (Hu et al., 2012)

cloud or the response time that is needed to complete an offloaded task. The peer-based and the clone-based models have the highest QoS, because of two factors: network and resource availability. While the proxy-based and the gateway-based model as the lowest QoS, whereas the network and the resource are shared by multi robot.

2.5 Related Work

Since 2010, there were several cloud robotics frameworks that were developed in order to investigate the cloud robotic paradigm. In this section, the existing multi-robots ROS-based frameworks will be explored to analysis their capability and determine some key features.

2.5.1 DAvinCi Framework

This work was motivated by regular use of computer complex tasks, for example computer vision (object or facial recognition) and mapping, in the DAvinCi framework outlined in Figure 2.2. The information of the sensors is sent to a controller, which is made active on request to each robot and enables a task

that requires multiple robots to contain sensor information (Roth, Livingston, Blair, & Kolonay, 2010). DAvinCi can be classified as PaaS and can provide tasks in the cloud environment to create a global map. The system consists of a server which connects the robot to Hadoop cluster. The server is running ROS master node in a server. ROS messages are embedded into requests / responses for the robot / server communication using the Hypertext Transfer Protocol (HTTP). For data storage, the Hadoop Distributed File System (HDFS) runs MapReduce for parallel processing. Note that parallel algorithms are crucial for processing speed. (Arumugam et al., 2010) The data is divided into blocks to simplify the parallel processes in addition to managing file storage. As already mentioned, the ROS message system is used to connect the DAvinCi server to robots. On the server side, ROS subscriber nodes function as HDFS clients are used to collect these messages as well as push the data to the parallel file system to execute Map Reduce tasks.

The authors present a Map Reduce model fitting grid FastSLAM algorithm, simulating the implementation of the one, two, 4 and 8 cluster nodes. The speed is raised noticeable with less time for numerous particles in the four and eight cluster nodes, which shown the algorithm is parallel.

2.5.2 REALcloud Framework

The REALcloud framework is a cloud service Platform, which offers the REALabs platform. The public Internet access to this platform is validated and robotic services are available after a validated access. Robotic applications on the REALabs server are further developed, which also offers a greater pro-

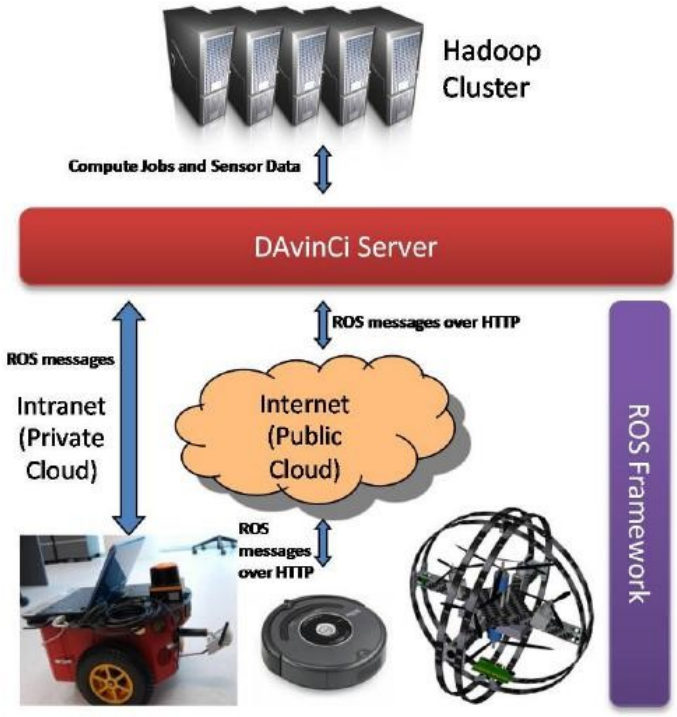


Figure 2.2: Architecture overview of the DAvinCi (Roth et al., 2010)

cessing power than a single or on-board computer and access to specialised devices such as GPUs and FPGAs. This capability is accessed from an integrated cloud environment to support a wide range of robotic applications (Mahmood & Saeed, 2013).

The platform consists of four main cloud-connected packages. The Integrated package includes HTTP server which is running on mobile robots. The treatment capacity of these robots is expected to be low, so robots only combine certain fundamental process while the task is carried out completely by the cloud. All security control, proxy and network address translation is handled in the Protocol Handler package. The programming interfaces are available to the users as an APIs. Finally, the system and resource access are conducted through the management package.

Figure 2.3 demonstrates REALcloud Framework structure, and each client is limited to the use of their own VM during direct connections to the virtualised REALcloud environment. Since it is harder for a VM to use another VM's resources. The above-mentioned packages are more obvious here: client packages operate on client side while the controlled packages and Protocol Handler operate on the server side. (Mahmood & Saeed, 2013).

2.5.3 RobotCloud Framework

RobotCloud has been designed to permit the cheap robots to run computation intensive tasks on a cloud resources. Figure 2.4 outlines regular working procedure of the RobotCloud architecture. which is Integrated into the ROS

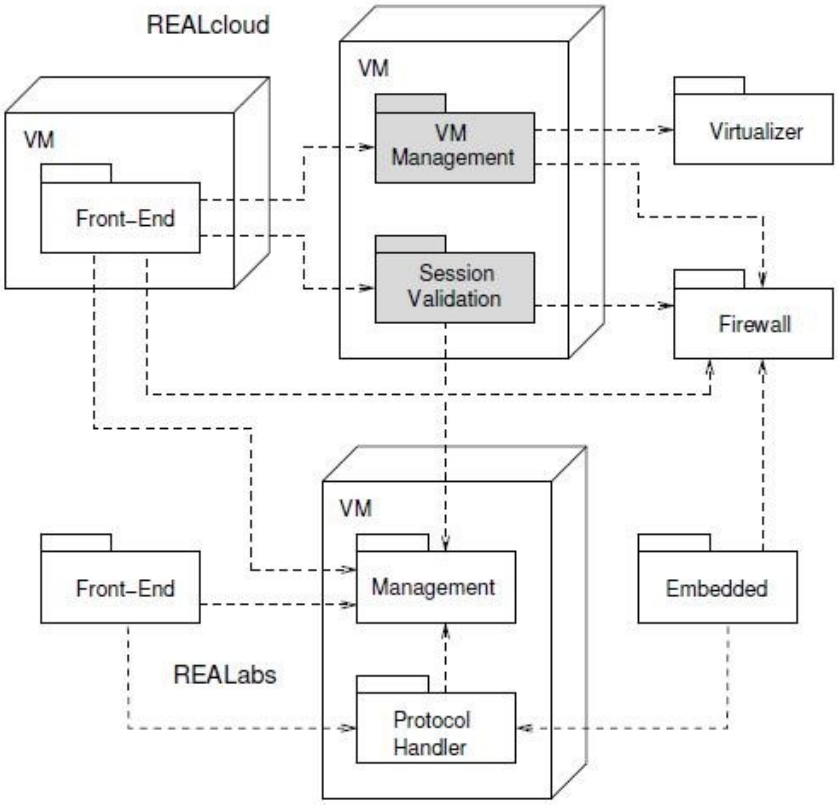


Figure 2.3: The schematic diagram of the REALcloud Framework, Reproduced from:

(Mahmood & Saeed, 2013)

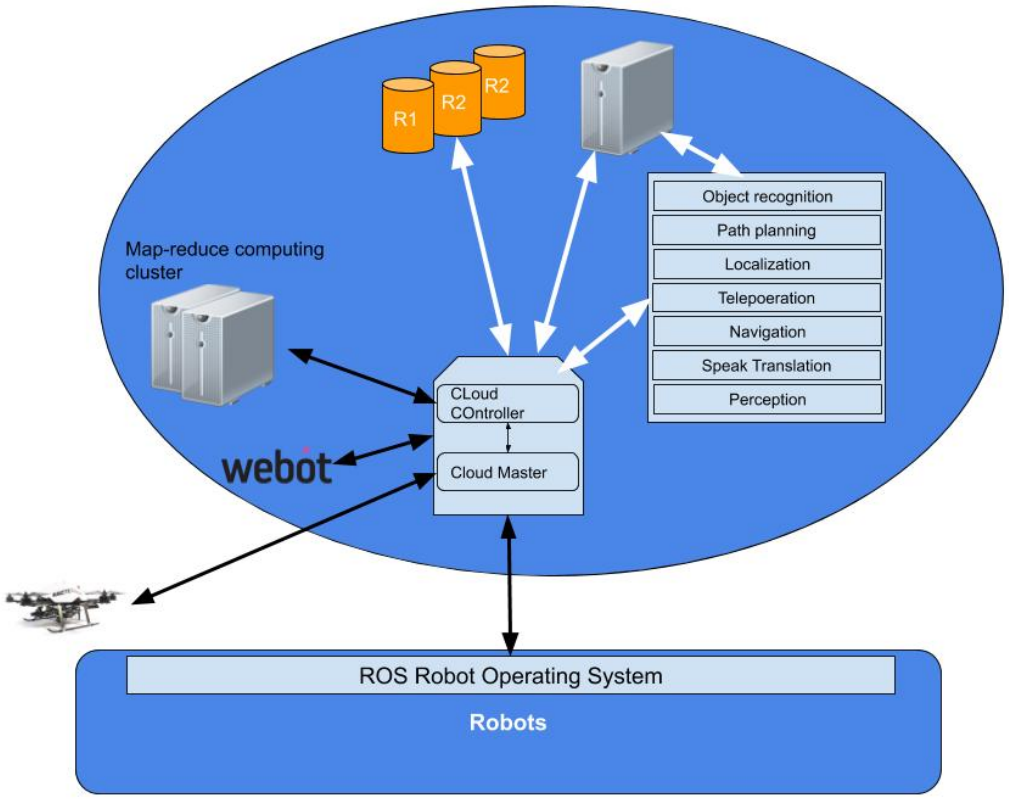


Figure 2.4: The schematic diagram for RobotCloud framework architecture, Reproduced from

(Doriya et al., 2012b)

Master Node as a central unit (Cloud Controller) to manage the entire system. The unit transmits all requests for service to the Service Management Point to check for permission. Registering / removal will then check the availability of the service, which will put the request in the queue if not available. The Cloud Controller is notified when permission is granted to the service is starting to response to the robot requested. A Map Reduce computing cluster and robotic services for example, path planning, and map building, which are services supplied by the RobotCloud system (Doriya, Chakraborty, & Nandi, 2012b).

2.5.4 Rapyuta Framework

It is an open-source Framework that was designed to provide the client with gateway to the RoboEarth knowledge-based repository and provide additional processing power to the linked robots (Hunziker et al., 2013). The Rapyuta platform is known as Cloud Engine for the RoboEarth. as a result of additional processing power provded by the cloud, which can handle the processing power required to interact with RoboEarth. In addition, it assist the linked robots to offload robots tasks in a similar way as other cloud robotic frameworks and to present themselves as a cloud computing platform (PaaS) especially designed for applications with multi-process high bandwidth robotics (Mohanarajah et al., 2014).

Rapyuta is a ROS compatible environment that utilises web socket-based communications protocols to allow the connection of ROS robots, as well as other browser machines. The network connection straight can be considered as limitation for the performance of the system in contrast to other cloud computing frameworks. (Mohanarajah et al., 2014).

The computer systems are Linux containers that provide a virtualisation environment that enables a safe and scalable separation of processes from different robots. The containers allow memory and processing time to be distributed while maintaining application speed, as if the process were not isolated. This method offers clever ways to address the problems of cloud computing security and scalability. As the Rapyuta is using ROS, therefore, ROS inter process communicates easily between all processes within one container.

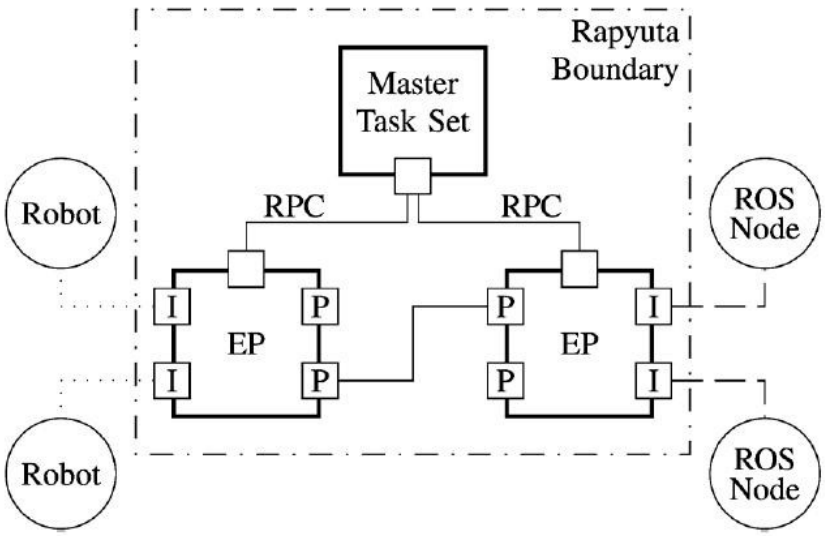


Figure 2.5: Infrastructures of the Rapyuta platform. Reproduced from (Mohanarajah et al., 2014)

The communication protocols are handled by Endpoint processes as shown in Figure 2.5. This Rapyuta was implemented to balance the complexity and latency of the system in the communication between processes. Interfaces for each VM are used by use of the WebSockets protocol to communicate between a Rapyuta internal and an external process (Mohanarajah et al., 2014).

The core task sets are several functionalists which are grouped as systems management processes. Each of the four core tasks in a standard case comprises a process that creates a PaaS. The ROS Master node is the key unit that manage link amongst robots withing the Rapyuta environment.

The system of the Rapyuta function is centrally organised. The component network provides a layer of abstraction that the other components take advantage of. The user is a person who links his robots to the system. The

LoadBalancer administers machines running in a computer environment. And the distributor distributes incoming links to the endpoint processes available (Mohanarajah et al., 2014).

2.6 Evaluation of Related Work

The Cloud robotic systems presented in the previous section have various positive features that can be used in enhancing this work. However, only one out of the four given multi robot framework is available to the public, which is only Rapyuta. Yet, the DAvinCi project and RobotCloud framework have a smaller scope, mostly relying on a map reduction cluster for computing and both running on a single ROS Master Node. Since the transition of a multi-robot system into the cloud is at the heart of this work, these qualities offer them some kind of flexibility.

The REALcloud framework has similar issues because it does not provide the communication mechanisms needed to create a multi-master environment to the best of knowledge. While a multi-robot system does not require a master ROS node on each robot, the strategy is to be promoted, as it offers robust and reliable advantages that are essential when a distributed strategy is in mind. Chapter 3 provides further information on the concept of ROS Master.

DAvinCi Framework

The DAvinCi framework uses an HTTP-based communication is a protocol with proven results which has been established. While HTTP needs inspection, real-time task can only be performed in a cloud environment. In addition, it allows efficient pluralisation by using a Hadoop cluster for running the algorithms in parallel. However, the DAvinCi system is sharing a one ROS master node operating on the cloud that adopt the proxy-based model. Therefore, it is less robust and reliable as a result of the high potential for the single node to fail. Furthermore, applications must be coded using Map Reduce patterns, therefore decreasing applications flexibility and introducing a higher barrier to entry than the other solutions. Finally, it is not open source and the design of the system does not allow for separation of procedures and security.

REALcloud Framework

As with the DAvinCi framework, REALcloud is also based on an HTTP protocol. Even though inspection is necessary for HTTP, the cloud environment does not provide real-time task capability and there is no ability to utilise onboard computational resources in addition to the cloud resources in a hybrid manner. However, the virtual environment and session validation offers an additional security layer.

Robot Cloud Framework

The Robot Cloud Framework also combines a Map/Reduce cluster with processing and storage. It adds additional services for service management, security layer and resource management mechanisms. However, similarly to DAvinCi and REALcloud, only a single ROS master node is running on the server, which means less strength and reliability due to the failure point on the node. Robot Cloud is also not available under an open source license.

Rapyuta Framework

The Rapyuta framework uses HTTP protocol in addition to WebSockets, which are the basis for more efficient communication compared to HTTP alone. It utilises containers which is an easy way to distribute computational resources, improving the scalability. Finally, it is open source, allows for easy access to the RobotEarth repository and allows for the architecture of multi-master environments. However, Rapyuta only supports old versions of both ROS distributions and Linux distributions. This limits the compatibility of the software in a heterogeneous environment and makes it difficult to debug the multi-master environment, especially when containers is deployed in different VM. Where the traffic has to be routed via the cloud local network. In addition, automation support is poor.

2.7 Gap in knowledge

This research investigates a number of existing solutions that have been conducted to address the challenge of creating a ROS-based multi-robot cloud robotics development environment. However, those solutions are limited in terms of integration, compatibility and availability. Most of the designed frameworks are designed to target specific applications. Considering this, the research gap this research attempts to address is the unavailability of a unified multi-robot cloud robotics framework, which can support any standard ROS-based application and can run in various cloud environments. The security is not considered as a major factor in the contemporary work.

This research intends to explore the feasibility of integrating a two layer approach to security to enhance the authority session. The collaboration process between robots in a multi-robot environment can be a challenge. Therefore, as a test application for the proposed solution a collaborative FR approach is to be designed and implemented. Such a collaboration system would allow a robot to recognise a previously unseen face, based on data that has been learned by another robot. A robust, secure, self-configurable, compatible and scalable cloud robotics approach, as per our current knowledge obtained from previous study, has not been implemented and hence these factors motivate us to develop a single framework. To the best of my knowledge, there is no research explore the optimisation of the network and hardware, while using a cloud robotics system. Furthermore, this research attempts to contribute knowledge and examine the impacts of the data quality in the network latency performance, and how it relates to the robot application performance. A new

hardware optimisation study will be devised to address this gap in knowledge while utilising cloud robotics systems.

2.8 Summary

This chapter provided an overview of the cloud computing concept. Then, it explained the some of the robots application that involved within this research. In addition, it discussed the cloud robotic topic and the current state of this concept by looking into its current clanging and most popular deployment methods. A comprehensive investigation into existing cloud robotics systems, as well as the robots that could be used during the experiment. In addition, investigations into a multi robot and multi environment real-time connection for mobile robots.

Finally, this work has been inspired by the Rapyuta framework which implements ROS multi-master concept, where the communication is handled by the framework also naturally leads to the construction of multi-master architectures. However, it has number of limitations as highlighted in this chapter. The compatibility and sustainability issues are the main factor of designing this work which can serve as an alliterative solution. Therefore, the design of the solution presented in this thesis attempts to provide a high level of flexibility and scalability to multi-robot cloud environments.

Chapter 3

Robot Operating System

3.1 ROS Overview

The Robot Operating System (ROS) project introduced at Stanford University in 2000 was started as a platform to support only two robots: Personal robots' (PR) program and Standford AI Robot (STAIR). In 2007, a company called Willow Garage contributed significant resources to creating the system and contributed to develop robotic packages. Further resources and expertise were provided by countless developers and study within the scientific community. The ROS was released under a BSD open source license and steadily attracted more specialists. Gradually, it has become one of the most popular robotics framework for researchers and developers. ROS was moved to be a global open source project, including the maintenance and main development activities, in 2013. ROS is effectively implemented by huge number of users worldwide. The ROS is defined as a development environment for robotic

functions, which provides off-the-shelf robot applications and programmable robot environment.

ROS is an acknowledged license framework for open source code reuse that enhance the productivity for the robot operating system. The system maintenance and distribution are two main ROS components. In terms of the code development, ROS is split into two sections: firstly, the essential part of ROS, developed and maintained at the Willow Garage laboratory. It has most of the fundamental tools for remote computing, as well as the whole ROS distribution and core elements of the software. The second part is the "Universe", which is the world-wide scope of the code developed and maintained by the global ROS community. Generally, the Universe includes libraries, software dependencies and hardware definitions. These define the hardware architecture as well as a low level of control to manage the linked devices, regardless if the device is a robot or an individual sensor linked to a computer or smartphone.

In terms of OS, ROS support mainly Linux Operating system such as Ubuntu and Debian, however, ROS hydro works with several other OS such as Android and Microsoft Windows. ROS provides support for a wide range of hardware, sensor and actuators. It defines the hardware architecture as well as a low level of control to manage the linked devices, regardless if this was a robot or an individual sensor linked to a computer or smartphone.

In order to run the ROS packages, at least one ROS master should be active in the network as Figure 3.1 shows, the simple ROS connection system that allows the ROS node to send and receive messages from other nodes. Once the node has been registered on the master the message could be sent directly

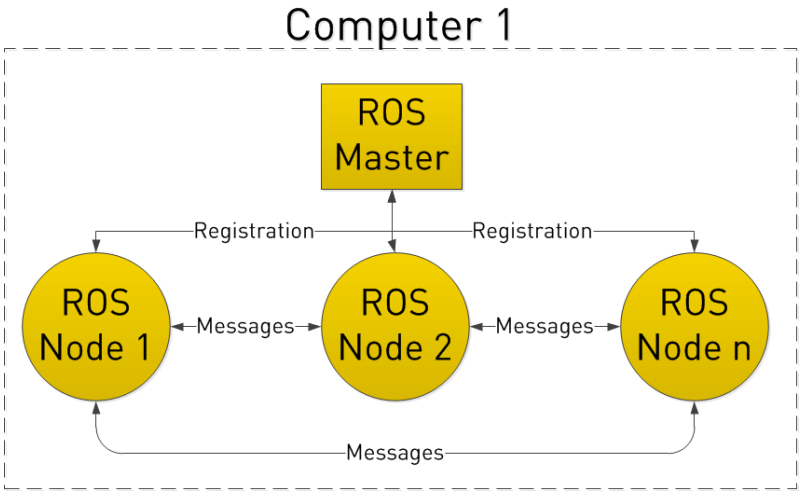


Figure 3.1: The ROS basic system
(*ROS : Intro To The Robot Operating System — Robohub*, 2013)

between the nodes.

ROS has many elements that require an operational structure to control the packages. ROS code is structured in packages. Where each Package comprise a number of nodes, dependency, configuration files, libraries, and so on. The aim of using a package is to offer a structure that is use easily to enable the reuse of the software. ROS has flexibility to work with ready algorithms that can be utilized for improving the robots performing complex tasks. SLAM is an example that used for robot mapping and navigation (da Silva, Xavier, do Nascimento, & Gonsalves, 2017).

3.2 ROS Structure

The ROS Framework has several code manageable nodes, messages, services, tools and library files. There are a number of key concepts in the ROS file

system: the package, the stack and the depository. Packages include ROS framework. Packs contain nodes, libraries of dependencies, setup files, software from third parties, and so forth. The purpose of a package is to offer an easy-to-use structure for easier software deployment. The appropriate stack is a package collection with a complete set of features. The Structure of ROS can be divided into three level of concepts as follows:

3.2.1 File System Level

The File system is the basic level and an essential building block in order to run a function in ROS, as shown in Figure 3.2. This function contains structures of folder and files to be executed. Each of individual folder has its unique functionality, consisting of six types:

1. **Packages** The ROS core construction block is called a package. The package includes the minimum requirement of content for a ROS program to run, such as configuration files, nodes and libraries.

2. **Manifests** A manifest presents essential package data, for example, licenses, repositories, and dependencies.

3. **Stacks** Several packages which are merged and interact is referred to as a stack.

4. **Stack manifest** Provides the essential stack data, such as dependence and licenses.

5. **Message** is available in several data formats that can be sent between

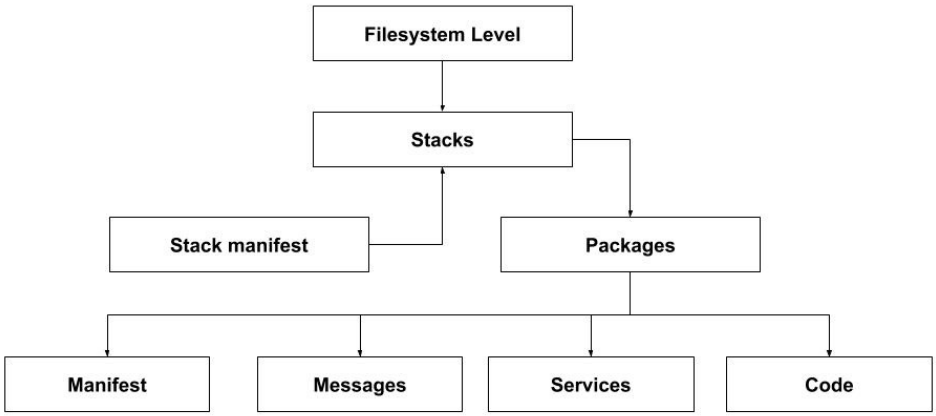


Figure 3.2: File system level Structure. Recreated from (Jusuf, 2016)

nodes.

6. **Services** determine the data structure for reacting to the request and response.

3.2.2 Computation Graph Level

The computation graph can be represented as a point-to-point (P2P) connection form of ROS data processing. The software executes all the data and its processes will be linked via a P2P network. As shown in Figure 3.3, this level mainly consists of a number of key concepts:

- **Master** The Master deals with the name and identification for each connected node. In addition, it works as a search engine when the nodes looking for other nodes.
- **Node** It is a process that runs a task. When several nodes can be exe-

cuted simultaneously, the P2P connection involved can easily be drawn as a graph. The nodes interconnect with the rest of the nodes by sending messages.

- **Message** all the communication between nodes is sent and received via messages. There are several types of message, for example, Integer (Int), floating-point, and string. In addition, the user can create a customized type of messages in ROS. The process of exchanging messages is performed using a publish/subscribe model.

- **Topic** Nodes publish messages into a topic. A node is linked to the source of the data only if the node is subscribed to topic. The topic-based publish and subscribe concept is really flexible.

- **Parameter** Parameters offer the ability to edit the setting of the nodes whilst they are interacting with each other.

- **Service** If feedback or response is required from a node, then a service is a more efficient ROS method to use, as a topic on its own will not be enough in this case. In addition, services allow interaction between different nodes. The name of the service has to be unique.

- **Bag** it is used to store and allow play back of data. It is suitable for developing robot software that requires collecting data, which may be stored in a buffer to be processed.

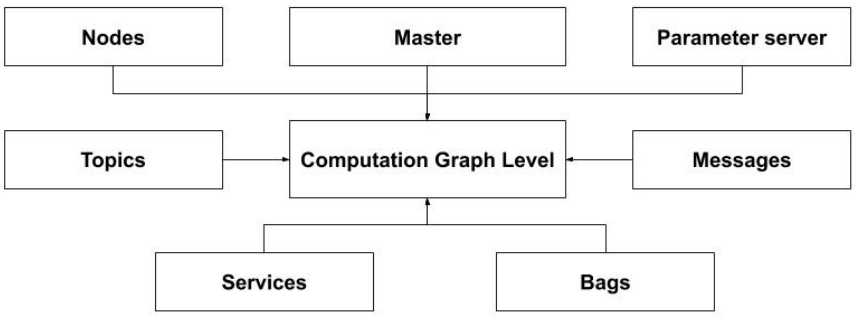


Figure 3.3: The Computational Graph Level. Recreated from (Jusuf, 2016)

3.2.3 Community Level

The ROS Community level is composed of resources for ROS that support each community member, allowing the discussion of ideas and exchange of robot applications and knowledge. These are defined as follows:

- **Distributions** ROS distributions are set of versioned meta-packages of ROS that available online. There are twelve versions of ROS so far, which are outlined in Table 3.1. ROS distributions are comparable in their role to Linux distributions, where they provide an easy way to install and setup a group of software and packages as well as support them with steady maintenance for each version.

- **Repositories** ROS depends on a joined network of code repositories, as various organisations and research group would be able to develop and release new robot software projects.

- **Wiki** The ROS Wiki is the main website that stores and documents most of the information about ROS distributions and software. As any

ROS distribution	Release date	Supported OS
Box Turtle	March 2010	Ubuntu 8.04 to 10.04
C Turtle	August 2010	Ubuntu 9.04 to 10.10
Diamondback	March 2011	Ubuntu 10.04 to 11.04
Electric Emys	August 2011	Ubuntu 10.04 to 11.10
Fuerte Turtle	April 2012	Ubuntu 10.04 to 12.04
Groovy Galapagos	December 2012	Ubuntu 11.10 to 12.10
Hydro Medusa	September 2013	Ubuntu 12.04 to 13.04
Indigo Igloo	July 2014	Ubuntu 14.04 LTS
Jade Turtle	May 2015	Ubuntu 15.04, macOS, Android, Windows
Kinetic Kame	May 2016	Ubuntu 16.04, macOS, Android, Windows
Lunar Loggerhead	May 2017	Ubuntu 17.04, macOS, Android, Windows
Melodic Morenia	May 2018	Ubuntu 18.04, macOS, Android, Windows

Table 3.1: ROS Distribution Releases
(*ROS Distributions — ROS*, 2018)

other Wiki users can contribute and upload their own documentation.

- **Answers Blog** ROS Answers is a forum that allow users to ask questions, where others can contribute their answers. The blog is other forum that provides a consistent stream of updates, news articles and so on.

ROS has evolved from the initial Box Turtle release, where many of the stacks in the ROS system had their initial 1.0 release, which required intensive user testing and on-robot testing. As the distribution becomes more stable, the release cadence has slowed down which has allowed robot developers to use the ROS platform with a high degree of confidence in terms of performance, reliability and sustainability.

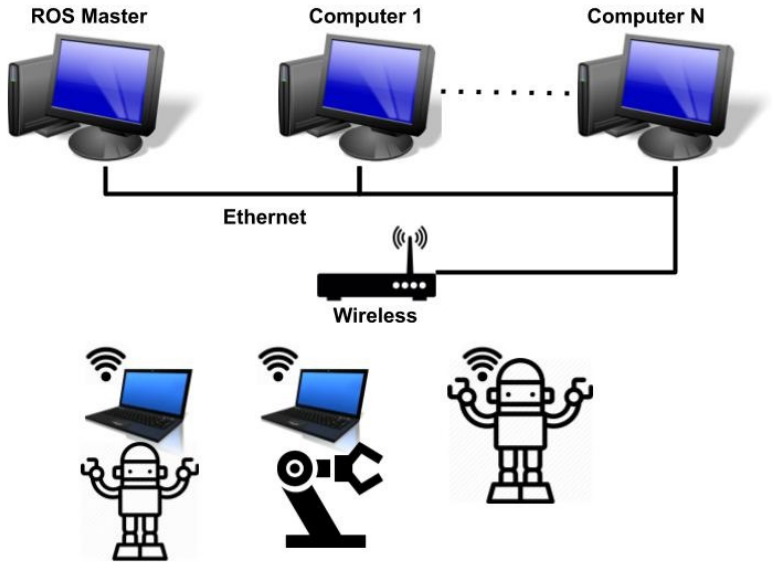


Figure 3.4: Regular hardware arrangement of a single master ROS system. Reproduced from
(Juan & Cotarelo, 2015)

3.3 Single ROS Master

The general network architecture for the ROS framework is designed for single ROS master implementation, where a single ROS master is responsible for managing all the connections among the nodes. The ROS master and the nodes can both run in a single computer or multi-computers in the same network, as shown in Figure 3.4. Each robot or computer must have at least one connection method to the network, either Ethernet, wireless or both in order to be part of a ROS environment (Juan & Cotarelo, 2015).

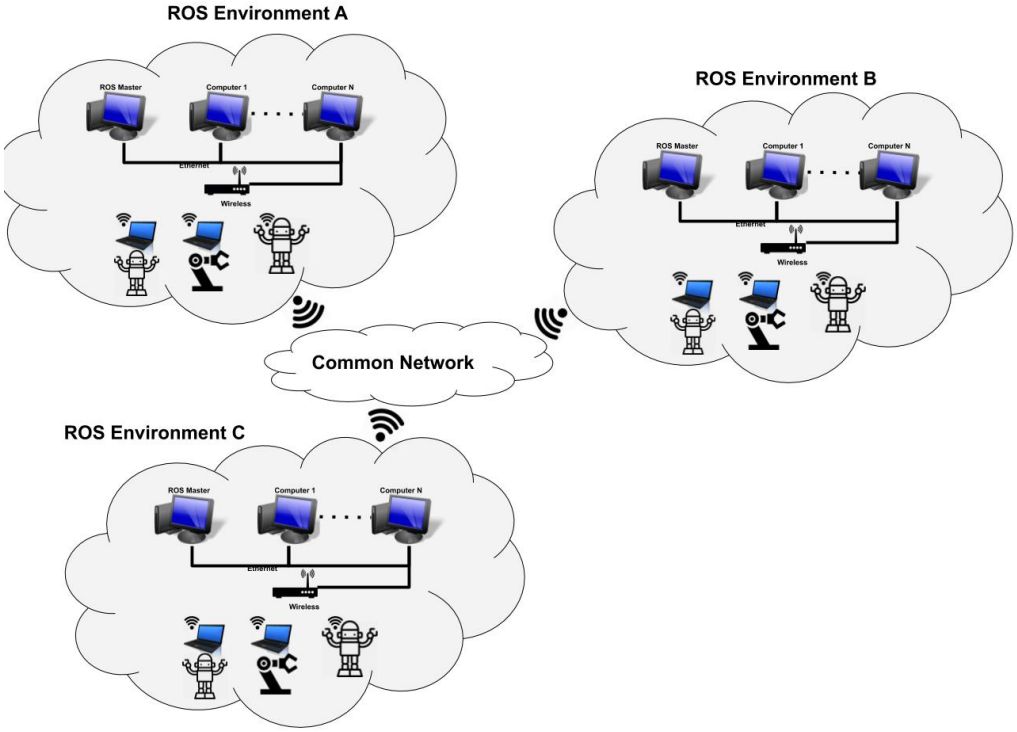


Figure 3.5: Hardware arrangement of multi-ROS master sub-systems collaborating with each other. Reproduced from
(Juan & Cotarelo, 2015)

3.4 Multi ROS Master

The idea of sharing knowledge between various ROS environments open a new challenge for the ROS community, where generally each ROS environment is controlled by a single ROS master as mentioned in pervious section. However, in 2015 a project called Multi ROS master `multimaster_fkie` proposed a solution for creating collaborative multi-ROS environments. As shown in Figure 3.5, the project utilises an additional network as a common network that allows each ROS master to share some or all of their ROS topics, actions or services among other ROS environments (Juan & Cotarelo, 2015).

Since `multimaster_fkie` does not interfere with ROS node launching in other ROS environments, there is no need to share the public keys between ROS master nodes, it enables password-free access to shared published topics from one ROS master to other.

In the case of using `multimaster_fkie`, all the computers, smartphones and robots in each ROS environment will share their selected topics via the ROS master in their environment, which will have access to the ROS multi-master system via the common network.

The major limitation of this framework is apparent when the multi-master system communication is implemented via wireless interfaces, due to power issues and interference. This has nothing to do with the framework itself, but special care should be taken when using wireless connections. However, it has shown reliable performance with a wired connection.

The `multimaster_fkie` have several advantages in terms of security, scalability, ease of installation and ease of use. The operator of each ROS master is able to select the allowed topic to share and update, which keeps the local ROS environment synchronized with the other ROS environments. This multi-master solution is continually supported since the "Indigo" ROS framework up to the latest distribution "Melodic". Therefore, after critical analysis and testing for the capability of this project to be moved to cloud network, it was selected to be part of the proposed solution in this thesis to create collaborative environment between the robot clone cloud images. The system limitation identified is not applicable when you consider that the connections between ROS-masters will only occur in the cloud side, where it has strong

network interconnection infrastructure.

3.5 Summary

In this chapter, the ROS framework was discussed, where a contextual overview was given about the ROS system and its development history. In addition, the ROS structure - divided into file system, computational graph and community levels - was explained. The ROS single master architecture is defined and it is identified how this limits the communication scalability in a multi-robot scenario. Finally, an existing architecture for multi-master ROS deployment identified in related work is discussed in more detail. The suitability of this architecture is analysed within the context of this project, which aims to create a collaborative environment between ROS systems where communication occurs in the cloud, rather than directly between robots. This architecture will be incorporated in the design decisions of the system outlined in Chapter 5. The following Chapter describes the methodology that has been applied within this research.

Chapter 4

Research Methodology

4.1 Introduction

The aim of this chapter is to define the research methodologies chosen for this work, and the reasons for selecting a particular methodology. Detailed hardware and software specifications for "robot test-bed" are explained. In addition, the methods of data acquisition and data analysis techniques used for this research are outlined. The design of the case studies and the experimental work undertaken is presented, highlighting the need for a mixed method approach and well defined scope in order to evaluate the system within the context of the cloud robotic field which is growing in the size and number of applicable scenarios and circumstances.

4.2 Research Methodology

This work adopts the usage of an empirical research methodology to accomplish its objectives, which basically means *"An investigation that produces results based on direct observation rather than theory and preconceived models"* (Paradies, 2006). This method was selected since this study is designed to examine the defined issues in a real robot as mentioned in section 1.2 instead of the use of simulated environments.

Firstly, an inductive research method was utilised to build a cloud robot system that is capable of handling some of the offloaded robot common tasks in single robot environment. The idea of the research has been established by applying a bottom-up approach, where the theory of current CCRP was gradually improved from observing the performance of various cloud robotic deployment models. In addition, a deductive research method was utilised in order to endorse and validate our theory. The design of a novel framework solution was accomplished to address the issues stated in Section 1.2 and fulfil the objectives defined in Section 1.3, and ultimately answer the posed research questions. The CCRP system was tested and evaluated via two case studies that enabled validation of the performance of our system. A quantitative research was applied in this study through gathering a large dataset that is going to be collected from various experiment and case studies, obtained and analysed using quantitative tools.

Initially the focus is put on developing the CCRP system with both private and public clouds, and testing if the system was compatible with different in-

frastructures. Following the chronological order of activities undertaken during this research, next an optimisation study is conducted in order to determine the range of parameters to be used in the case studies when comparing the CCRP to other approaches. The next step was to observe the system performance while it is running a fixed time task, in order to evaluate the performance of our approach in contrast to multi-robot proxy-based model cloud robotic system. Finally, the proposed solution system was tested and evaluated in a sequence of representative robotic case study scenarios, which will result in a number of recommendations for future research and development for the CCRP.

4.3 Research Approach

The topic of the cloud robotics is relatively new, first suggested in 2009 by the RoboEarth project (Waibel et al., 2011). This project opens the door for a number of research that focused on applying various kind cloud robotic model and framework that stated in Chapter 2, in order to improve the on-board robot performance. The models for cloud robotics solution have been inspired by the existing systems identified during an appraisal of published work that is devised and developed further through innovation and new insight in multi-robot cloud robotic environment, with respect to our robotics lab capability in terms of hardware and software.

The existing solutions will continue to be assessed via literature published in academic papers, journals and books. Thus, in a field where the lack of academic resources is acknowledged, the development and resulting analysis

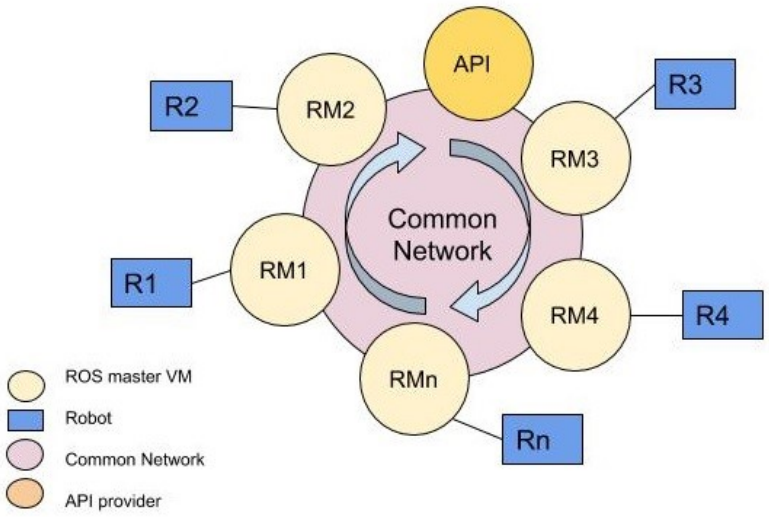


Figure 4.1: General CCRP Methodology

will contribute a major part in this research.

This study proposes a new multi-robot cloud robotic framework that deploys the clone-based cloud robotic connection model that allow the on-board robot to offload process and receive control commands or feedback. The proposed model is developed based on the analysis of the existing model and a novel solution created in order to overcome their limitations. As shown in Figure 4.1 the general CCRP methodology as each robot (R) connected to a ROS master VM (RM). The connection between the master nodes done via the common network. Also, the system suggests the use of configurable API manager to allow access external resources.

The key issues in the current generation of cloud robotic systems are communication and latency. This problem grows with increasing number of robots that must communicate with the cloud system. Therefore, our solution tries to enhance the usability and availability for the cloud provider by using multi

ROS core system. This technique is going to allow each robot to utilise the cloud machine by itself without sharing the resources. The clone-based model provides the system with high level of robustness and resource availability; therefore, it is expected that it can improve the communication performance. However, the robots will still be able to share data and communicate with other robots that runs in a different ROS environment via the Multi-master network layer "Common network", which is going to be defined in Chapter 5. The CCRP is thus designed to provide sustainability, where the CCRP can be deployed with most recent ROS distributions and subsequent robot applications.

4.4 Experimental robots

The research was limited in terms of real robot selection as there was only three type of robots within the University of Huddersfield robotic lab namely: the NAO robot, Turtlebot 2 robot and AR. Drone 2.0. Two out of the three available robots were used: the NAO robot, Turtlebot 2 robot, which explored in more details in this section. Other robots were considered as ideal robot solutions for the research experiment, such as the humanoid robot "Pepper".

4.4.1 Humanoid robot NAO

In 2008 the Aldebaran NAO Robot have been introduced in its final form as Figure 4.2 shown, which designed to resemble the human appearance. It is

one of the most common robots accessible for education. The robot weight about 4.5 kg and above half a meter in height. The robot has a wide collection of sensors and actuators, which allow it to interact with the surrounding environment. it is equipped with a set of sensors (e.g., tactile touch and four microphones). There are two VGA CMOS cameras are equipped with the Nao's head and speakers as shown in Figure 4.3. it has two connection method, which are: Wi-Fi and Ethernet. In terms of CPU, it comes with an X86 AMD GEODE 500 MHz CPU. (Ismail et al., 2011)). It has numerous features that allow it to perform tasks where human-interaction is a key element, such as face recognition and speaking.

In regards of software, NAO is provided with the NaoQi 2.0 software that runs on Linux OS. The software manage the access to all the NAO sensors, sending the control signal to actuators, sensors, control the Wi-Fi network. NaoQi is capable to perform functions in both parallel and serial. NaoQi's functions have bindings for both C++ and Python programming languages. (Aagela, Holmes, et al., 2017)

The NAO robot has two cameras as shown in Figure 4.3 the Field of view of NAO video cameras. The cameras support various video qualities and resolutions as shown in Table 4.1, with a maximum frame rate up to 30 fps. The top camera is used for recording and video streaming during the experiment.(Ismail et al., 2011).

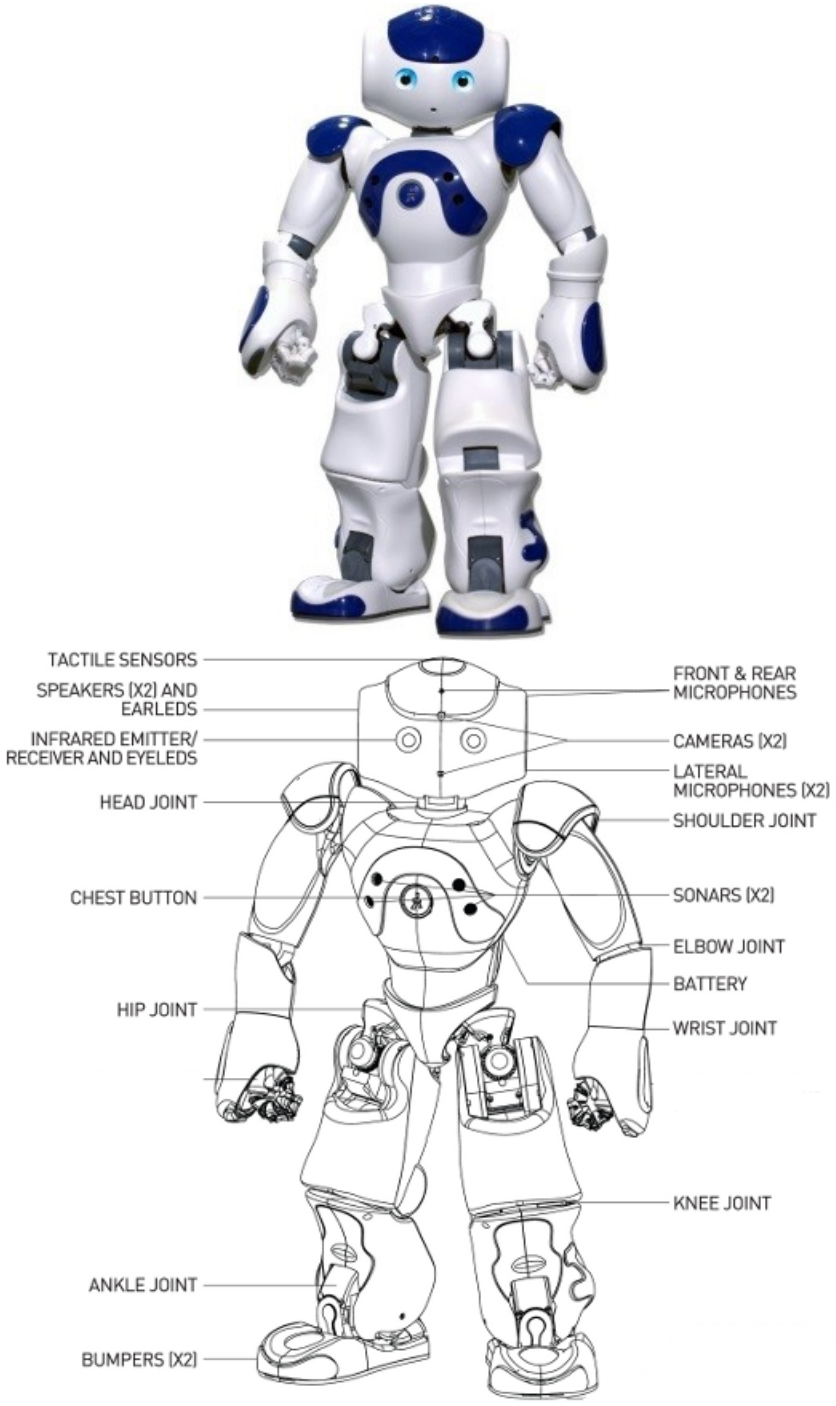


Figure 4.2: NAO H21 Robot design (Aldebaran.com, 2018)

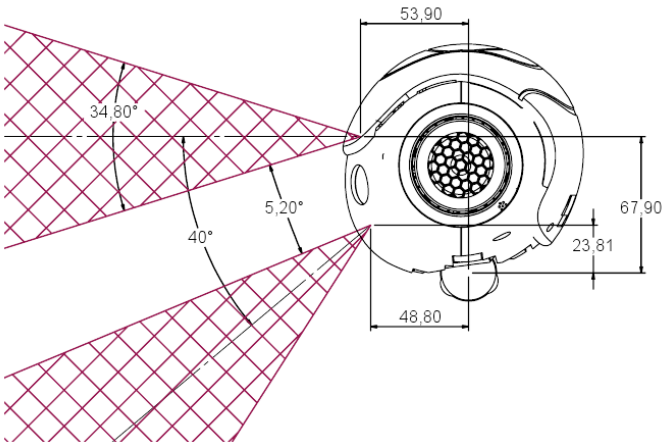


Figure 4.3: Field of view of NAO video cameras (Aagela, Holmes, Dhimish, & Wilson, 2017)

Video Quality	Resolution
K4VGA	1280x960
KVGA	640x480
KQVGA	320x240
KQQVGA	160x120

Table 4.1: Video Quality Provided By NAO Robot (Aagela, Holmes, et al., 2017)

4.4.2 Turtlebot Robot

TurtleBot robot is a wheeled robot that was built by Willow Garage (Garage, 2011) which is supported by ROS. It has a various kind of sensors fitted on its mobile-base that allows for the robot to perform several task such as, navigation, mapping, and path planning on unknown environment. The hardware part includes several parts as shown in Figure 4.4 such as Microsoft 3D sensor Kinect or 3D sesor ASUS Xtion Pro. It has a wheeled Mobile base and NiMH Battery Pack that provides 3000 mAh capacity (TurtleBot2, 2014).

The TurtleBot operation and tasks can be tested by both the real robot or a simulated one provided in ROS, using the Gazebo simulator together with Rviz 3D visualisation software (Zamora, Lopez, Vilches, & Cordero, 2016). Together these offer the most suitable way to inspect the TurtleBot's features and capability.

4.5 Cloud Platforms

This section provides information about the cloud platforms used to conduct the research activities. The research selects two Cloud computing platforms, which are public cloud (Google Cloud Platform - GCP) as well as private cloud (Openstack).

Nowadays, Google Cloud Platform is one of the most significant and growing in the public cloud computing market. It offers users numerous remotely accessible services to design and develop various applications from just a website to

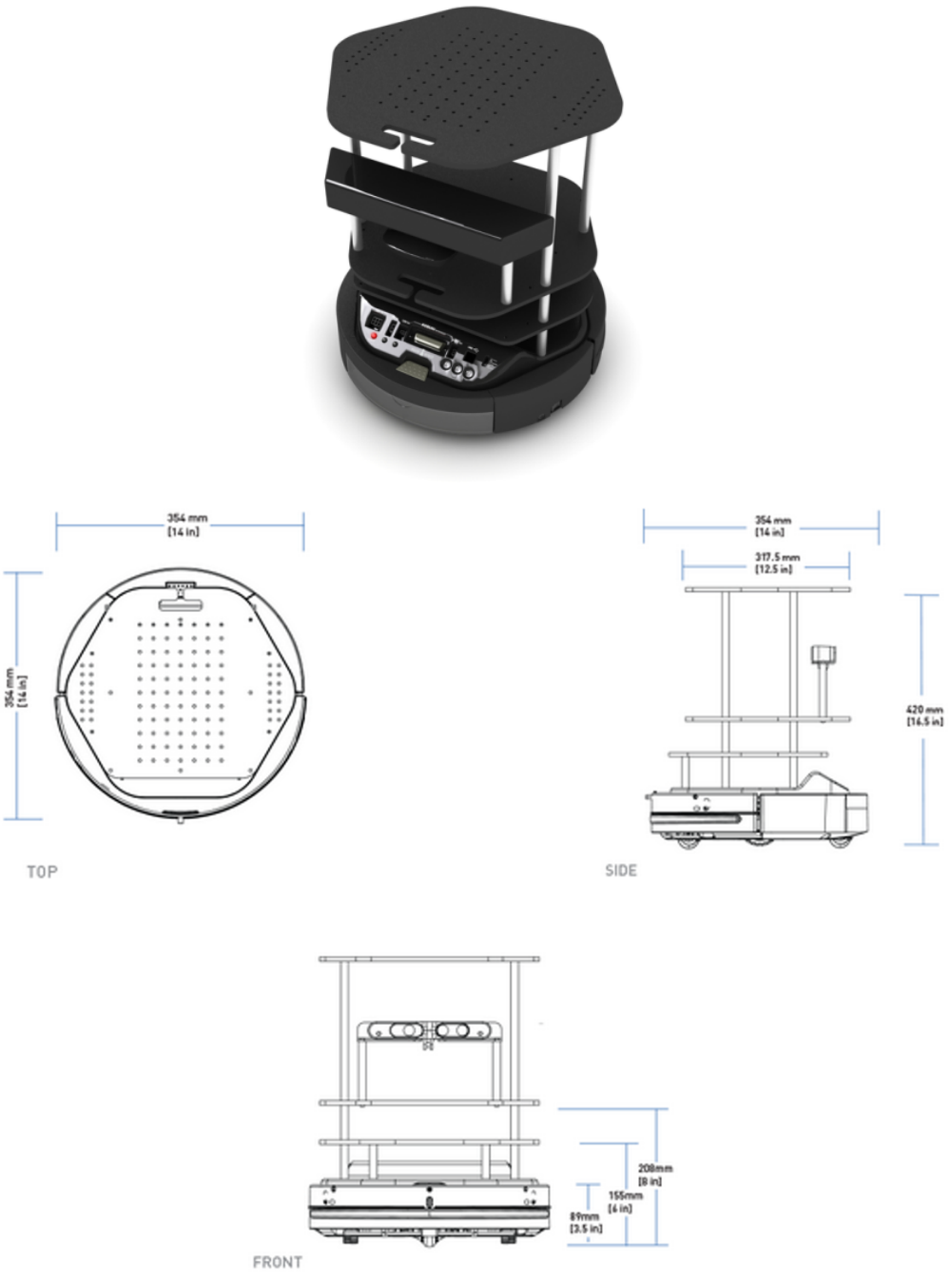


Figure 4.4: Turtlebot Robot and schematic diagram (TurtleBot2, 2014)

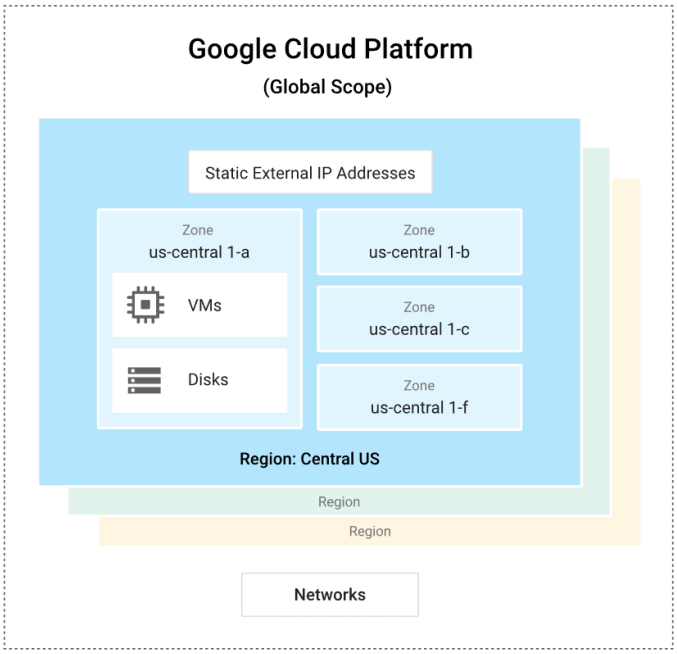


Figure 4.5: Google cloud platform compute engine global scope (*Google Cloud Platform Overview*, 2018)

complex worldwide distributed services. GCP provides a wide range of cloud service such as compute engine, app engine and storage (Navale & Bourne, 2018).

The selection of GCP was based on various aspects, first of all, it has a large world-wide region, which allow users to select the physical deployment location of their VMs in various zone and regions, as shown in Figure 4.5. This choice relative to the device accessing the cloud services can significantly impact the network latency. In addition, GCP provides full configurable network environment which is very important for our system (*Google Cloud Platform Overview*, 2018).

Figure 4.6: University of Huddersfield OpenStack Private Cloud fabric

4.5.1 OpenStack

In this research OpenStack is used to create a local private cloud platform within the University of Huddersfield network infrastructure. The reason for configuring our own cloud platform is to prove the compatibility of our proposed system with the private cloud environment where no external public IP provided. In addition, our system attempts to enhance the privacy and security, therefore, using private cloud platform can be a first option for organisations which are concerned of sharing their data with a third-party company. The deployment process of OpenStack can be done via various methods, in this research the OpenStack cloud was designed and developed using multi-node Devstack installation, for which the steps of implementation can be found in Appendix B. In terms of the fabric of the system, it includes one cloud controller and two compute and storage nodes, shown in Figure 4.6. This system is used to host a number of VMs for the a robot clone images used in the experimental work in this research.

4.6 Case Studies

The field of cloud robotic is growing and the proposed system tries to be a general solution for single and multi-robot environments. However, evaluating the performance in terms of all possible scenarios and circumstances is not feasible. Therefore, this work designs two case studies to assist the CCRP in two key factors: the ability of supporting various kind of robots with real-time response time and low latency, which should be less than 400 ms to be considered as acceptable real-time according to (ITU, 2003). In addition, to provide the ability to share knowledge amongst several robots in multi-robot secure environment, and the capability of offloading the regular robot task to the cloud.

4.6.1 Real-time robot teleoperation and mapping

This case study merge between the process of teleoperation and the mapping, where both tasks usually needed before the robot is be able to navigate in unknown environment. As mention in section 2.4.1 the mapping task require substantial amount of RAM with minimum one Gbyte and high processing power, which are available via the cloud. Also, the teleoperation process is allowing the operator to navigate around the unknown area to create a map for the surrounding environment. The objective of this case study is to analysis the real-time performance and the capability of both robots "NAO and Turtlebot" and compare it with the local ROS scenario. Which consist of three major tasks:

1. Teleoperation
2. Offload video
3. Create Map

The scenario of this case study is defined to suit our robotic lab capability and the available resources. The main task is to create a full map for the room and teleoperation between two points with obstacles.

4.6.2 Collaborative Face Recognition for Multi-robots

In general, the face recognition task for most of the Humanoid robot is a key ability that should be performed in order to react with different response according to the person's identity. This case study designed to measure FR Confidence rate and response time performance of the multi-ROS master and multi-robot environment for the proposed system. As here the scenario is using two NAO robot to perform learning and recognition faces task for 10 people.

4.7 Data Collection and Analysis

The data is collected in various parts throughout this research by observing performance and outcome of a number of experiments at the University of Huddersfield robotics laboratory. The data collected by two main methods: firstly, by using network analysis advance tools such as Wireshark and Nload as well as using ROS time measurement tools. There are a several occasions

where the data is going to be collected during this study. The second method is by observing the outcome of executed tasks and gather the result. The collected data are kept in an Excel Files (.XLSX), in order to perform some data analysis. The data sample for each experiment is defined with more details in each related section.

Initially, before analysing the data, the collected data have been checked in case there are any corrupt results or errors in the values. Some of the data was plotted onto a graph for distribution analysis to discover and outline the vital relationships. Other data was processed before plotting as the plotted graph was for data averages value. Subsequently, an observation was made to determine the desirable result of the generated graphs. Finally, a compaction evaluation looked into the difference in the performance between the measurements of local ROS performance and measurements of CCRP into the cloud.

4.7.1 Reliability and Validity

One of the most important factor is assist research outcome is to look into the reliability and validity of the collected data, this part is presenting the methods of how the reliability and validity of the gathered data was appraised.

Reliability

The reliability of the obtained data was evaluated by examining the variable Cronbach's alpha (α) that was used to determine the data consistency, a soft-

ware called Statistical Package for the Social Sciences SPSS have been used to calculate the value of α, which has to be can be define as a formula (1) shows, which use the number of data items and the average inter correlation between the items (Vaske, Beaman, & Sponarski, 2017).

$$\alpha = \frac{N.\bar{c}}{\bar{v}+(N-1).\bar{c}} \quad \ldots\ldots\ (1)$$

- N is the number of data items
- c-bar is the average inter correlation between the items
- v-bar is the average variance

The formula (1) suggests that if the number of data items increases the α will increase. Also, if the average inter-correlation between items is low, therefore, the α will also be low. As the average inter-correlation rises, α increases as well.

(Gliem & Gliem, 2003), (Vaske et al., 2017) debated that the level of the reliability of any collected data can be obtained by calculating α, and to consider collected data as reliable source, the acceptable level should be between 0.65 to 0.80.

Moreover, there is other factor should be considered when measuring the reliability is the fact that the study is conducted with an actual robot not a simulated one. Therefore, the obtained data was measured repetitively to improve the reliability level.

Validity

The validity of the gathered data was weighed by comparing the outcomes of the experiment in local system performance with normal ROS environment with the result obtained from the proposed system.

4.8 Performance Evaluation

The process of evaluating the performance was divided into several stages, in order to determine the system usability, compatibility and performance capability as a full stack cloud robotics solution. The selected case studies are going to be used to benchmark the performance of the proposed solution by stressing out the network and the system, in order to identify the limitation and bottleneck for the new cloud robotics approach. Moreover, the performance of the task functionality and the level accuracy both were observed, and measured by analysing a set of factors, such as round trip time (RTT) via the cloud, response time and accuracy level "confidence". The system is going to be deployed with various Linux and ROS distributions in order to verify the compatibility and functionality of CCRP.

Confidence

This metric measurement is useful in identifying the performance of the robot task, such as object recognition and navigation. It represents the probability of the accuracy of the recognised object or face. The evaluation metrics for

measuring the performance are related to object and face recognition tasks. The outcome of those robot tasks has two possible results: succeeded or failed to perform the given task. The following formulae show how to define the success and failure rate:

- **Success rate** percentage of the overall succeeded tasks, defines the level of the reliability of the running the given task.

  $Successrate = SucceededtasksTotaltasks * 100$ .....(2)

- **Failure rate** opposite of the success rate.

  $Failurerate = 100 - Successrate(\%)$ .....(3)

Response Time

The service response time average is also an important factor that needs to be measured. It is defined as the overall time required for the system to react to an action or event. There are a number of variables involved to obtain the average response time. This includes the network latency and the processing time, buffering or queuing time. The response time determines the performance of the given service. The impact of this time was different based on the nature of the task. There are some tasks, such as object recognition, where a firm deadline is required to be met in order to run the received action.

Response Time = Network Latency + Processing Time .....(4)

4.9 Summary

This chapter outlined the research methodologies that were applied to fulfil the research approach aim and objectives, demonstrating the process of the research approach. The robot equipment used in this research was defined in addition to the cloud platforms utilized. A brief summary of the designed case study was provided and the reason behind the selection, in order to address the key issues of multi-robot communication and latency. The methods used to collect data are defined along with the analysis technique. It is recognized that the potential scope of a general purpose multi-robot cloud solution encompasses an ever increasing range of scenarios and applications. Therefore, two key applications have been defined which are - teleoperation and mapping, and face recognition - and incorporate an optimisation study into the method to determine the range of parameters which should be used when comparing the CCRP system to other solutions. Finally, the evaluation performance of the system was discussed. The following chapter describes the design and the architecture of our proposed system.

Chapter 5

Clone-based Cloud Robotics Platform (CCRP)

5.1 Introduction

This chapter will describe our novel cloud robotic platform design and implementation. The CCRP allows the full stack deployment of ROS-based multi-robot systems on a virtual cloud environment. The main objective is to develop a standalone, self-contained and re-configurable clone VM, which will allow single robot or multi-robots to be connected, providing increased computational power and storage to the on-board robot's resources. In addition, this platform is designed to enable the robots to connect to external cloud services via the API manager. Finally, it provides a secure communication between robots in a collaborative environment.

As shown in 5.1, CCRP provides a cloud VM per robot, each with the associated ROS applications installed within it. Each VM is responsible for handling off-loaded processes from a robot, managing API requests, and providing the robot with additional storage. This approach defines a model that allows a ROS workstation or edge device, that would typically serve the robot locally, to be migrated to the cloud.

In addition, the CCRP proposes a network architecture using a Virtual Private Network (VPN) and a ROS bridge to establish secure connections between the robot and its clone image in the cloud. In a multi-robot collaboration scenario, this model utilises the multi-ROS master that permits robots to "see" each other via an additional network layer - the "common network". The multi-ROS master allows a clone image executing a master for a particular robot to share some of the topics with another clone master image. Therefore, robots can collaborate with others by publishing or subscribing to the topics via the common network.

From a technical point of view, this model establishes a cloud robotics solution to increase the mobility and interaction abilities of the robot, where the collaboration between multiple robots does not depend on the geographical location of the robot.

This chapter includes a general overview of the CCRP architecture, followed by a description of the design network architecture that is the backbone of the CCRP architecture. In addition, API manager is described. Finally, the low level implementation detail of each component is explained. The following publication have arisen from my research detailed in this thesis: *Novel clone-*

based cloud robotic model to overcome limitation of the multi robots QoS by Aagela, H., Holmes, V (2019).

5.2 CCRP Architecture

A general overview of the CCRP architecture is presented in Figure 5.1. Unlike other multi-ROS master solutions, the CCRP provides the user with a fully configurable full stack clone-based cloud robotics platform that is compatible with most ROS recent distribution. Therefore, the user can build a customised CCRP system based on their needs. In addition, it provides a dynamic scaling capability as well as providing additional layer of security by using both VPN and Public Key Infrastructure PKI.The CCRP utilise set of existing software elements and link them together. As shown in Figure 5.1, the CCRP architecture is spanning over two sides, the cloud side and a client side, where each side has its own configuration.

The cloud side contains of several Clone ROS Images (CRI) that run inside a Virtual Machine (VM). Each CRI must be created within a Linux based OS (supporting ROS distributions on Ubuntu 14.04 or above). It includes a number of software components as shown in Figure 5.2; those components are defined as follows:

- **OpenVPN server** a containerised secure network with VPN software network tunnelling solution that works in both client and server sides (Feilner, 2006).

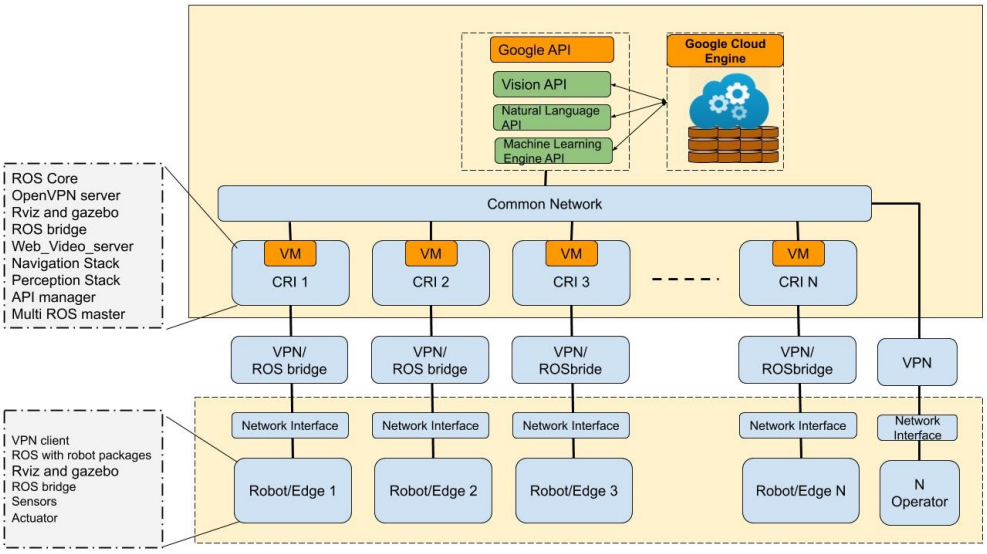


Figure 5.1: CCRP Architecture

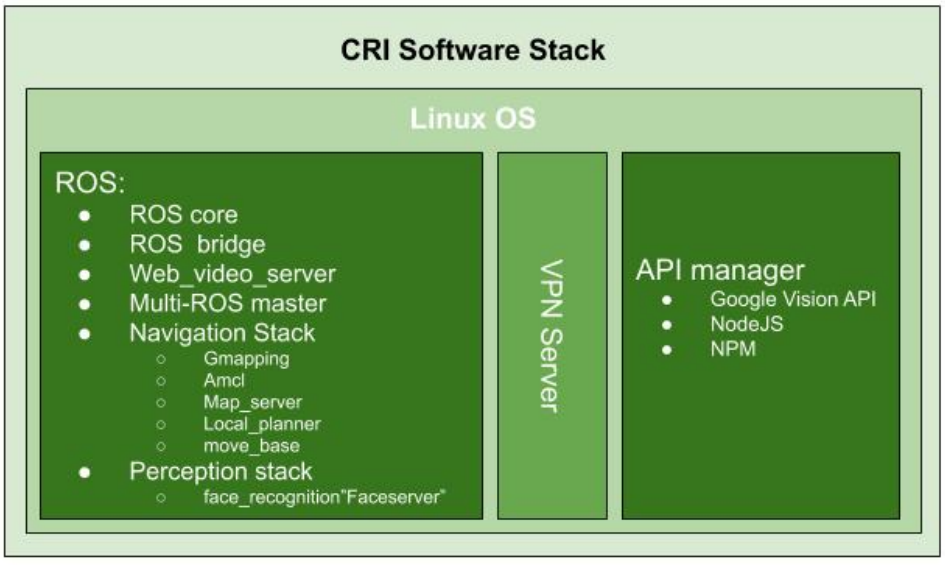


Figure 5.2: CRI software stack

- **ROS core** a number of nodes and associated software responsible for initializing the ROS based system. One ROS core must be launched to establish a ROS environment (Quigley, et al, 2009, May).
- **Navigation Stack** a collection of software that support the robot to navigate in its environment, such as, teleoperation, mapping, localization, path planning and navigation (Quigley, et al, 2009, May).
- **Perception stack** the perception ROS package is a ROS holding all the perception associated applications, such as object detection, face recognition, and so forth. Also, the important libraries for example, PCL, OpenCV and others (Koubâa, 2017).
- **API manager** a server-based application that provides the robot with access for a set of external APIs such as Google vision API and others.
- **Multi ROS master** a key feature in this architecture; built on ROS software and allows the process of sharing services and topics amongst several ROS environments (ROS masters).

The client side of this architecture can be composed of three components: robot, edge or operator. If the client-side acts as the robot and edge, the installation of ROS will be based on the robot type. If it is an operator, the installation will include some visualisation tools and exclude the robot ROS packages. The CCRP integrates a number of key elements that need to be configured, described as follows:

- **VPN client** used to connect the robot to the VPN server that is hosted on the CCRP.

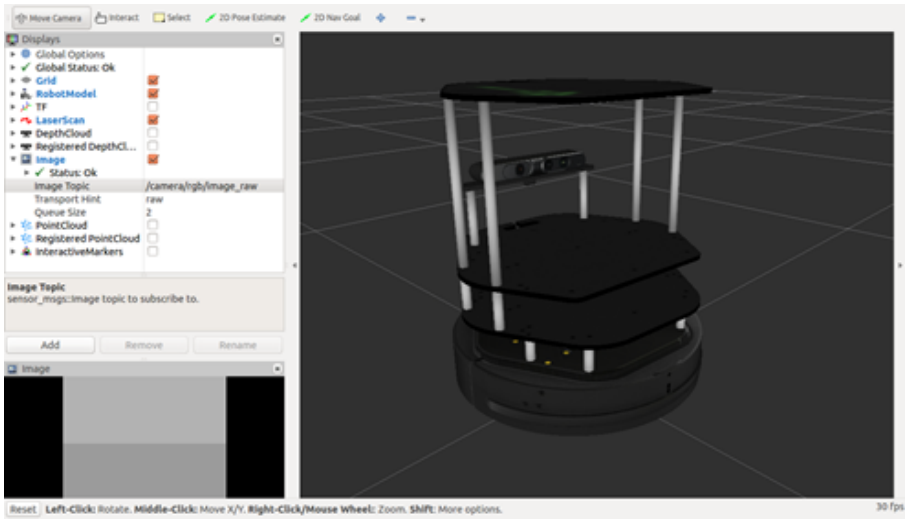


Figure 5.3: The RVIZ visualiser application

- **ROS with robot packages** each robot has its own ROS libraries and software (robot and edge only).
- **ROS bridge** a tool used to establish a link between the ROS nodes and ROS master.
- **Rviz and Gazebo (operator only)** used for displaying both sensor data and state information from the ROS.

The operator components Rviz and Gazebo are used to view the robot model and current position, video streams from onboard cameras, 3D model and any map that might be created. In addition, the software is able to deal directly with robot sensor topics, such as infrared distance measurements, camera data and sonar data, as shown in Figure 5.3.

The connection between the CRIs is using the local cloud network, which is utilised as a common network to share topics between robots via the respective clone images. In addition, the cloud side provides the user with an API

manager application that can be used to connect to any API in order to access external services.

An innovation in the CCRP model requires each client to be connected via the VPN tunnel and have their PKI register with the respective CRI before establishing an authorised link between the client and the cloud. Once the link is available, clients use ROS bridge to discover and connect to the ROS master.

The clone-based model was selected to increase the robustness, security and reliability of the system. This architecture allows the full potential of utilising IaaS platforms to deploy a cloud robotics application with a high level of flexibility and elasticity of implementation. Moreover, the design of CCRP architecture supports robot mobility, where the networking configuration is separated from local environment, and each robot will connect to the cloud provider over the external network. This addresses the lack of travelling distance in a regular ROS environment. In addition, it allows the process of sharing knowledge between robots regardless to the geographical location.

The VPN network that runs within the CCRP system is expected to introduce some communication overhead, which is analysed together with the impact of the physical location of the datacenter. The proposed system is likely to slightly increase the network latency between robot/edge and their controller node CRI. In terms of system performance, the impact of this can be longer response time as demonstrated in Chapter 6. This impact will heavily rely on the physical distance between the client and its CRI datacenter, and the resulting network performance.

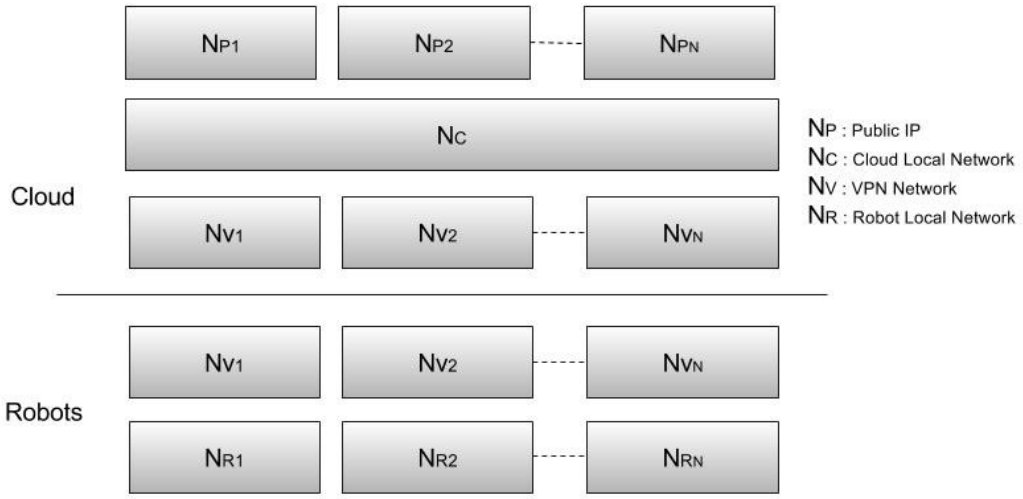


Figure 5.4: The CCRP Network Architecture

5.3 Network Architecture

In this section, the network architecture is described, where in our approach the network architecture plays a major role in determining the overall system performance of any cloud robotics system. The design of the network has been made based on the challenges identified from previous work: compatibility with ROS local environments, providing secure connections between the robot and the cloud, and supporting at least three subnet networks which implement the common network for multi-master ROS. Therefore, a VPN based interconnect was selected to provide the additional virtual subnet network. OpenVPN has been implemented to establish connectivity between the local network on the client side with the one provided by the cloud.

As Figure 5.4 shows, the network architecture is divided into four different network schemes that run in both cloud sides and Robot/Edge sides. The first network *NP* is the public IP network or the internet network, each of the

clone instance VM will need a public IP that will be assigned by the cloud provider. The second network *NC* is the cloud local "Common Network" where each of the instances within the same cloud environment has its own local IP, accessible by other instances in the same domain. This network is used as the common network for the multi-master ROS, which establishes a link between individual ROS environment to share their selected topics. The third network *NV* is the VPN network. This is the key network that allows the Robot/Edge to connect to its respective clone image as well as adding a secondary virtual subnet network for the VM, which is going to be used by the local ROS environment. Finally, *NRN* is the local network for the robot or the edge, which could be any internet enabled network, such as, LAN, Wi-Fi and Cellular Mobile network (3G or 4G).

This network architecture is applied for each CRI, therefore, robots will not be visible directly to each other, where each robot only can access its CRI via the VPN network. Thus, the security level is enhanced, however, there must be no conflicts between the topics shared by the robots. The Transmission Control Protocol (TCP) protocol used to offload and download the traffic over the VPN network provides a connection-oriented network with lowest packet loss in contrast to use of User Datagram Protocol (UDP).

5.4 API Manager

The API manager is an additional feature added into the CCRP system to allow the system to access external Internet resources, which can potentially

expose the robot to access huge datasets and advanced algorithms. In addition, it can minimise the size of the CRI for tasks that require a huge dataset, such as trained data for object recognition, by not requiring each CRI to store the data locally in order to perform the function.

The designed API manager suggests using a public cloud API, such as Google Object recognition or Azure object recognition. The API manager used as a web application that subscribes to robot data, then a request will be submitted to the API service provider. Once the result is received, it will be published as a new topic. The CCRP design demonstrate the usage of the Google vision API to obtain information about an unknown object which has not been trained to recognise. The feedback of the API will return with a list of match set. The best match will be taken after filtering matches. The API manager application is implemented in each CRI individually.

5.5 Security

Distributed system architectures such as cloud computing, together with the emergent Cyber-Physical System (CPS) architectures of robotics clouds, present significant challenges for security and privacy (Ren et al., 2012). The wider distribution of cloud nodes and the extent and nature of the data collected, transformed and exchanged between robots and their clones within clouds are potential system vulnerabilities. In the cloud computing domain, authentication permits the integration of various robots deployed in various contexts. However, cloud robot applications are exposed to security and privacy threats

by virtue of exploits, eavesdropping, distributed attacks, viruses, malware attacks, and other known attacks upon cloud computing architectures (Al-Aqrabi & Hill, 2018).

In a multi-robot environment, the authentication frameworks cannot be static. The business systems running on clouds are dynamic and hence the authentication interactions need to be dynamic, as well ((Al Aqrabi, Liu, Hill, & Antonopoulos, 2014),(Sotiriadis, Bessis, Antonopoulos, & Hill, 2013). For robotic clouds, there is the additional interactions between the cyber and physical systems to take account of, which is inherently dynamic as well. This is compounded by the fact that the system must manage the authentication of multiple parties within a multi-robot cloud environment.

There are many factors that could indicate the level of the security of public cloud systems, such as encryption techniques, the number of open ports, rule-based access control and password complexity. The CCRP addresses the security issue by utilizing a two-level authentication technique: the first level is VPN authentication using username and password. The second level is a Pre-shared key (PSK), public key infrastructure (PKI) between the robot/edge and its clone. As a result, the level of the security between the robots and cloud is significantly increased, meaning that only an authorised robot will be able to connect to the cloud services.

The process of sharing knowledge between the robots is managed by the multi-master ROS application where the robot will share only the authorised topics that are selected by the robot's operator. Specifically, in a complex application there is a need to securely delegate access control mechanisms to one or more

parties, who in turn can govern methods that enable multiple other parties to be authenticated in relation to the services that they wish to consume. The data privacy in the world of the cloud computing is a major concern for customer with sensitive information (Ren et al., 2012). The CCRP try to address this issue by demonstrating the compatibility of the CCRP with a small-scale private cloud platform, which can be provide a virtualised environment and it operated only locally within an organisation network.

5.6 CCRP Process Handler

This section is explaining the procedure of the way our system handles different kind of processes. Initially, the client side is responsible for initiating the connection to the CRI: first the VPN client connects to the VPN server using generated script, which will require the client to provide a username and password for authentication.

The process of offloading the robot task from the on-board robot to the cloud starts by checking the availability of ROS master from the robot side. Once the robot is connected to its CRI, then the cloud application deals with the robot request via its topics. A robot sends a request message to a cloud endpoint through a ROS bridge. The CRI, through the cloud endpoint node, will reply with the response for the robot request, which is usually a task needs to be executed or a data need to be stored.

Finally, the process of sharing knowledge is happening between the CRIs, where each CRI has a unique machine name that is used to look up other

ROS master within the common network by implementing the multi-master discovery mechanism.

It allows CRIs to share datasets to other robot's topics. Therefore, the authorised CRI can obtain shared resources over two main methods: the first method is to obtain a live data via subscribing to targeted CRI topics. The second method is to pre-share data between the CRI using techniques that work on merging the new obtained data from one system with others. a method of merging was used to merge the new obtained files which is provided in the ROS distribution to create a synchronised unified knowledge dataset among CRIs, by using the "merge_bag.py script.

5.7 Implementation

This section covers the process of deploying the CCRP, which consist several steps and include several elements. Therefore, the installation process of the system is divided into number of subsections.

5.7.1 Cloud Setup

The CCRP system can be hosted in a cloud VM of the operators choice. In this research, The CCRP was implemented within two cloud platforms, Google Cloud Platform (GCP) and our local OpenStack cloud. The process of setting up the VMs were similar in both cases. There are essential configurations that need to take place with the cloud virtual environment, first, deploying the

appropriate operating system OS such as, Ubuntu or CentOS. In case of GCP the process of building the VM the user should select the zone and region of the physical datacenter.

The right amount of resources must be allocated (CPU, RAM and storage) depending on the robot's requirements. Next, configuring the network policy rules and cloud local network setting to open the needed protocols and ports (the default setting for most cloud providers restricts access via a firewall).

Finally, a floating IP address is assigned to the VM as an external way of communication - this IP can be a public IP address when a public cloud platform is used or a local IP address in an organization that deployed its own private cloud within the local network.

5.7.2 ROS Installation

The ROS installation is a key process of the deployment of CCRP, which is explained in detail in Appendix C, both ROS Indigo and kinetic were used in this research, because of the compatibility with all essential libraries for the available robots in our lab.

Once the ROS is installed in both side of the network, the Network Identification between the robot/edge and its clone must be defined by populating the IP address and hostname by using the command line in the Linux terminal and edit the "/etc/hosts" local system file, which is done manually. As the CRI host has three IP address that is Public IP address, Local cloud IP address

and the VPN address. The VPN address is used for the ROS environment network setup. The following commands used to setup the IP and hostname configuration for the ROS in the cloud side.

```
export ROS_IP="VPN IP address of the cloud side"
export ROS_HOSTNAME="Hostname of CRI"
```

Similarly, for the client side, the VPN IP address and hostname will be configured, but, an additional configuration is required in order to define the IP of the ROS master.

```
export ROS_IP="VPN IP address of robot or edge side"
export ROS_HOSTNAME="Hostname of robot or edge side"
export ROS_MASTER_URI=http://"VPN IP of CRI":11311
```

In the robot/edge side, there is extra robot related packages which need to be installed and those configurations will vary based on the type of robot. In this research there is a need of installing two software stack that are going to be used for the experiments conducted in Chapters 7 and 8. The first package is the navigation stack package that can be installed for most ROS distributions by using the following command:

```
sudo apt-get install ros-"ROS distribution"-navigation
```

The second package for face recognition case study that must be installed from source, where a workspace folder named `catkin_ws` must be created and the

following commands has to be applied:

```
cd ~/catkin_ws/src
git clone https://github.com/procrob/procrob_functional.git
  --branch catkin
cd ~/catkin_ws
catkin_make
source ~/catkin_ws/devel/setup.bash
```

One of most important packages to deploy multi-robot environment for CCRP system is the Multi-master ROS. The installation for the `multimaster_fkie` system will take place once the ROS environment configured in the CRI. The following command used for the installation, which again can be used with various ROS distributions:

```
sudo apt-get install ros-ROS_distribution-multimaster_fkie
```

Once the multi-master software installed in all CRIs then the operator must run a discovery script to allow the application to search and find the other ROS master nodes. In each CRI, the operator must also define which topic to share within the multi-master environment.

In addition, the time synchronization between the ROS master and connected nodes is a critical factor in the communication, which needs to be always synchronized. ROS utilizes a software named Chrony to schedule the Network Time Protocol (NTP) updates.

5.7.3 VPN Setup

The OpenVPN application is utilized to create the VPN connection. The reasons of using this particular software are that it provides a full manageable VPN web-based solution, where the user can create and manage users and edit the setting by web dashboard. The server and client sides require two different installation steps for the OpenVPN as follows:

Server Side

- Install the OpenVPN software type
- Creating new user that to be use by the client via the dashboard
- Edit the SSL library to use the OpenSSL protocol
- The method of accessing the private subnet of the server side must use routing technique instead of NAT as shown in Figure 5.5

Client Side

On the client side, the OpenVPN software is installed and the VPN profile is downloaded over the external IP for the CRI. It can then be used to start the VPN connection as follows:

```
sudo openvpn --config config.ovpn
```

Figure 5.5: The VPN settings on the openvpn dashboard

Once the above command is used a username and password will be required to allow the connection to be established.

5.7.4 Scalability

The process of deploying the CCRP is not complicated, however, to reproduce a large number of CRIs can be time consuming for a cloud application, thus, the deployment time can impact the scalability of the system. Therefore, the snapshot was used as a method for CRI redeployment. The snapshot allows

a user to create a customised image of the VM in certain state, hence, the operator can create a snapshot of the fully configured CRI that can be quickly used to recreate a CRI VM when needed.

5.8 Summary

The concept of the CCRP system was outlined in this chapter. Firstly, the architecture of the CCRP was explained that include a breakdown of the essential element in the system. Then the network architecture of the system was defined. In addition, the API manager was explained. Followed by a discussion about the security that implemented within the CCRP system. Moreover, a brief explanation of the process handler of the CCRP system was provided. Finally, the method of implementation was given that provide a clear specification of the deployment processes of the CCRP.

The CCRP framework introduces the novel integration of various technologies applied to multi-robot cloud systems. A VPN is utilised for cloud to robot connectivity, and the interconnection model is devised to allow multi-master ROS interoperability between both public and private clouds using a single unified network architecture. This provides flexibility to generally deploy many types of robot communication typologies, where previous solutions have focused on narrow use cases. This reduces the operator effort required to take advantage of cloud offloading and efficient common network communication regardless of the geographical distance and connectivity limitations (such as availability of public IP addresses).

Chapter 6

Cloud Robotics Optimization

6.1 Introduction

This chapter presents a number of experiments that aim to profile the network performance and analyse the relationship between the data quality and accuracy level of robot face recognition task. The results from these experiments are used to define the parameters for the CCRP evaluation in Chapters 7 and 9, in terms of video stream resolution, bandwidth, Frames Per Second (FPS), and the effect this has on the accuracy of the FR algorithm. The second part considers the optimisation of a low cost and low power consumption robot that still able to perform some of the robot complex task such as creating 3D model for objects and maps of the surrounding environment, which will be used to evaluate the cloud offloading capability of CCRP in Chapter 8.

Indeed, the network connectivity and performance are major issues that face

any cloud robotics implementation, therefore, this research considers the performance of the network and studies the impact of the network delay on the outcome of the robot tasks on the edge device or the cloud platform. The following publications have arisen from my research detailed in this thesis: *Impact of Video Streaming Quality on Bandwidth and Face Recognition Accuracy in Humanoid Robot NAO* by (Aagela, Holmes, et al., 2017)and *An Asus_xtion_pro based indoor MAPPING using a Raspberry Pi with Turtlebot Robot* by (Aagela, Al-Nesf, & Holmes, 2017).

6.2 Network Profiling

The network profiling is conducted in this research to analyse the impact of the offloaded sensor data quality such as the video and the accuracy level that is obtained from a robot's task, in order to determine the most optimal data quality that can be used with low bandwidth requirement. In this research the NAO robot was used with Facial Recognition (FR) activity, in order to define the optimal video quality by defining the relationship between the data quality - which sets the size of the needed bandwidth - and the accuracy rate for the wireless and wire transmission medias. The design of the experiment was established to connect between the robot with a local workstation PC, as shown in Figure 6.1. The PC runs both Choregraphe and Monitor software to receive the NAO sensor data via a local network.

A comparison study was conducted for different network media, such as Ethernet and Wi-Fi. In addition, four different resolutions (KQQVGA, KQVGA,

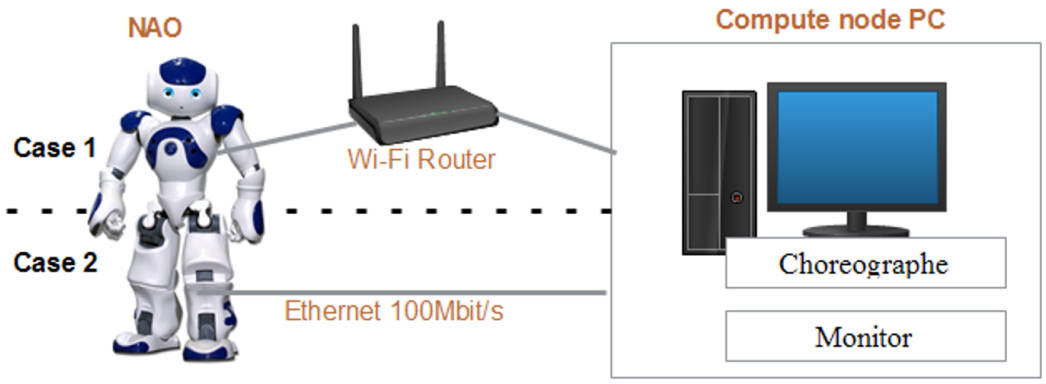


Figure 6.1: Network architecture for profiling cases
(Aagela, Holmes, et al., 2017)

KVGA, K4VGA) for video streaming will be considered. The bandwidth consumption was measured based on a defined Frames Per Second (FPS) which is sent from the NAO robot to the Edge Node and Cloud. The key Diversity between black and white video quality and colour video quality will be explained. In addition, black and white and colour video modes will be evaluated to identify any potential reduction is bandwidth. To complement this, the accuracy of the face recognition algorithm for each stream, and the robot learning algorithm, will be considered. Finally, the best video resolution and mode will be selected based on the observed experiments and discussion (Aagela, Holmes, et al., 2017).

6.2.1 Experimental setup

The experiments are conducted using two pieces of software: Choregraphe and Monitor. Monitor was utilised in the first experiment to configure the video streaming settings. This includes the video resolution / quality, FPS

of the video stream and to switch between black and white and colour video modes. Choregraphe is used to program and control the movements of the NAO robot based on a timeline of actions that must be carried out, as shown in Figure 6.2. This illustrates the sequence of actions and behaviours that the face recognition and learning process is composed from.

Face recognition behaviours are composed through the connection of face tracking components and basic awareness, that give NAO the ability to perform movements in order to follow a face that must be recognised by the algorithm. The face recognition algorithm to examine the detected face is wired after the face detection box. The lighting conditions in the room were found to affect the recognition result. Therefore, to ensure reliable results are obtained, the face recognition tasks have been performed using the same lighting setup and the same distance from NAO's camera to the face being tested. The NAO robot used to conduct the experiment was monitored using two tools: NLOAD and Tomato. Tomota v.1.28 runs on the Wi-Fi router to monitor bandwidth at the physical layer. NLOAD runs on the Linux command line to monitor bandwidth from the software perspective. These were used to measure the performance characteristics of the network link between the robot.

6.2.2 Face Recognition Algorithm

The Choregraphe-supported Face Recognition Algorithm is the incremental Principal Component Analysis (PCA) Algorithm (Luthffi, 2011), which is responsible for analyzing the identified face to obtain the facial characteristics. The algorithm passes through a Python code used to compare captured infor-

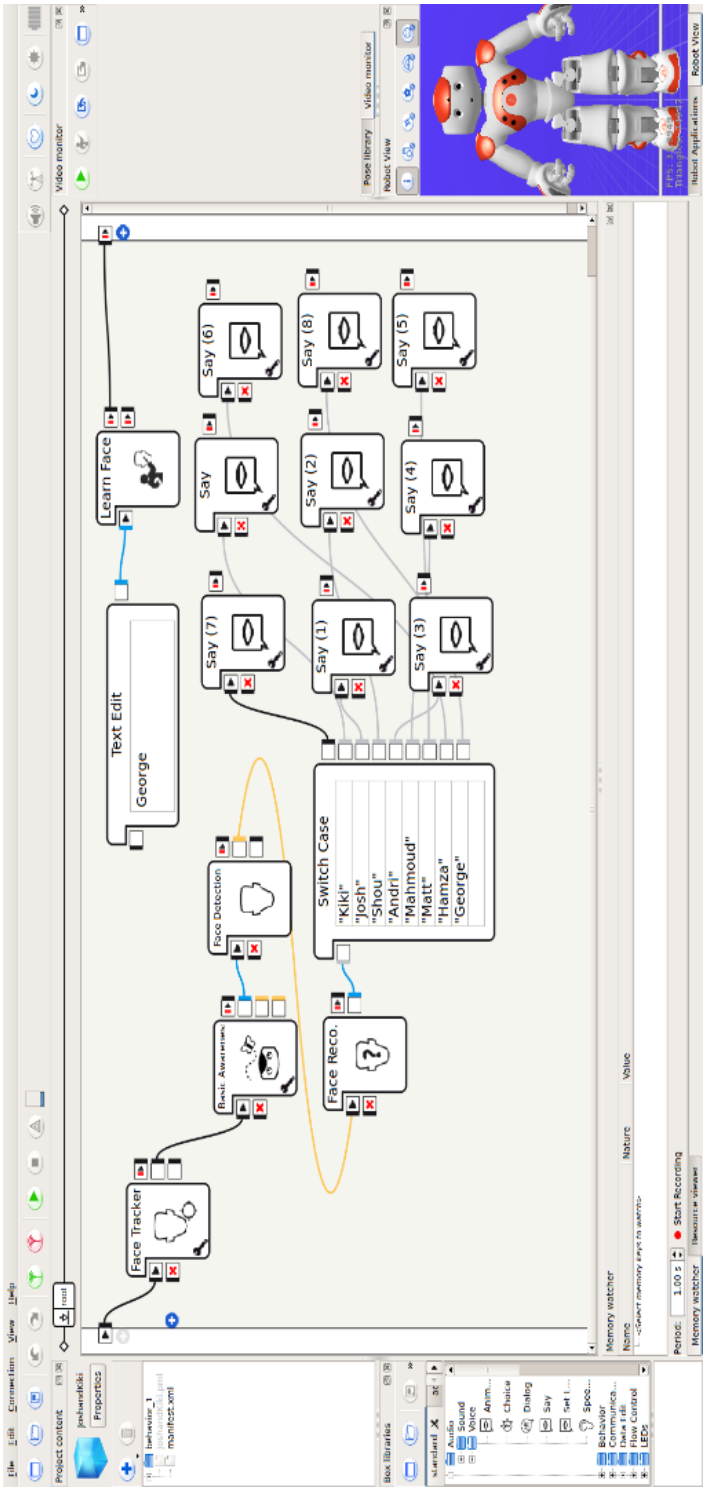


Figure 6.2: Sequential behaviours that required to perform FR and face learning processes using Choregraphe software
(Aagela, Holmes, et al., 2017)

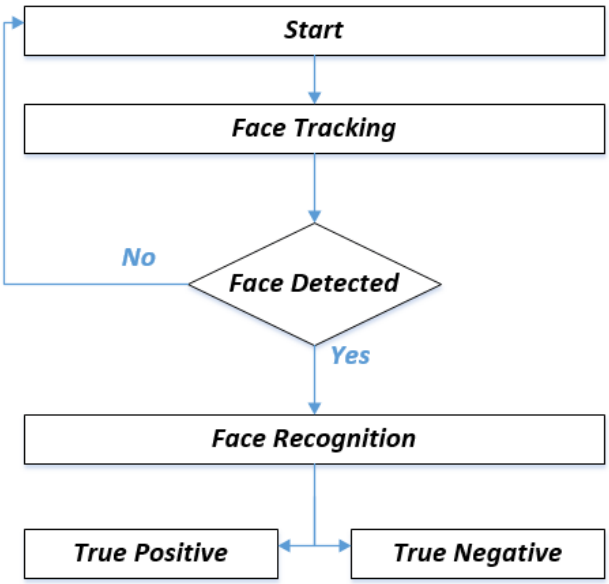


Figure 6.3: FR process stages to perform Face Detection and Recognition with NAO robot
(Aagela, Holmes, et al., 2017)

mation with the dataset of accessible faces. This process is outlined in Figure 6.3.

6.2.3 Experimental Cases

The following sections describe the results of the 3 experimental cases for the video streaming and FR optimisation.

6.2.3.1 Video Streaming over Wi-Fi

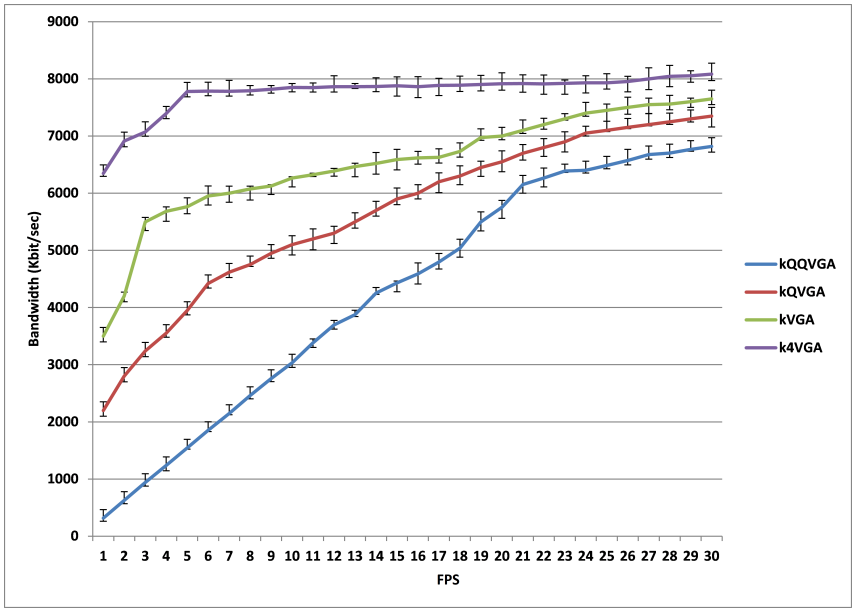
In this experiment the NAO is connected to a wireless network using Wi-Fi to test its performance capability with various video streaming quality. Figure

6.4a indicates the bandwidth for colour video stream transmission using Wi-Fi connection in bits per second, while Figure 6.4b indicates the bandwidth for black and white video quality. In both instances the highest bandwidth is comparable to 8283 Kbps using four distinct video quality modes. This outcome demonstrates that in Wi-Fi transmission technology the video stream bandwidth is constrained.

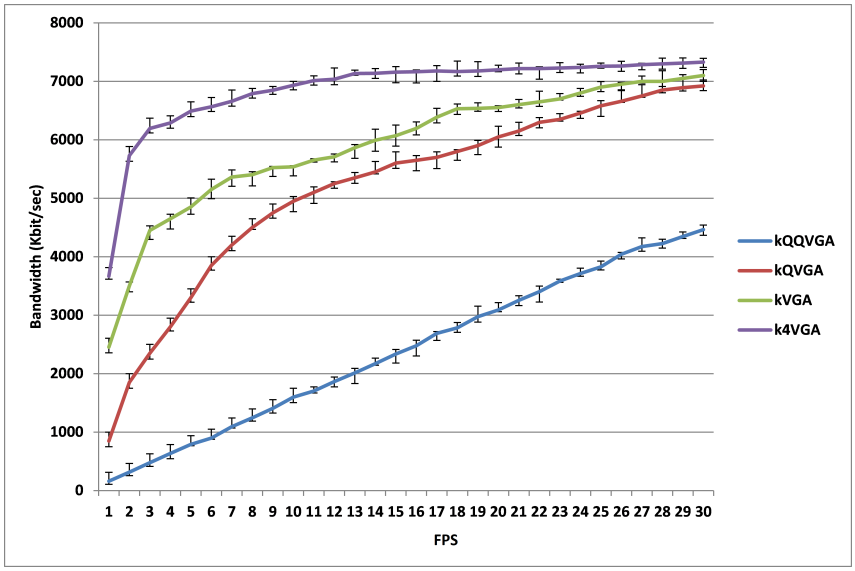
The bandwidth of the colour video requires about double the bandwidth compared to the black and white mode. For example, using the KQQVGA quality video stream, for the first 26 frames the colour video bandwidth remains double the black and white bandwidth. As shown in Figure 6.4a, when 1 Mbps bandwidth is being utilised, this results in transmission of only 7 KQQVGA video quality colour frames.

6.2.3.2 Video Streaming over Ethernet

The objective of this experiment is to further evaluate the performance of the video qualities. where the bandwidth was constrained with the use of the Wi-Fi, which was limited to 8.2 Mbps. The maximum data rate of 81.4 Mbps using Ethernet as shown in Figure 6.5a, four distinct video qualities are used for the colour video stream. The highest data rates were required by KVGA and K4VGA resolutions. which were stayed constant after a certain amount of frames as a result of the bandwidth limit for the connection media. However, the data rate will continue to increase as the frames number increases for both KQQVGA and KQVGA video qualities; this is true for both the colour video stream and the black and white video stream as seen in Figure 6.5b.

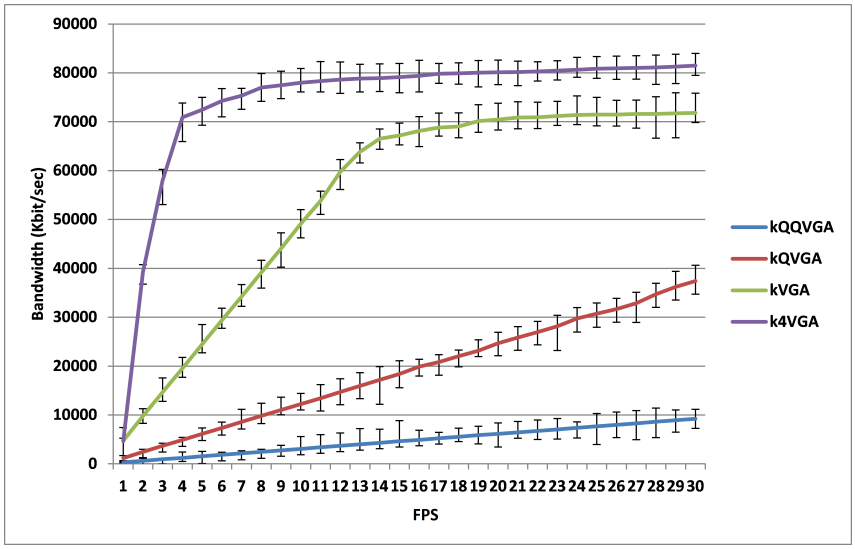


(a) Colour

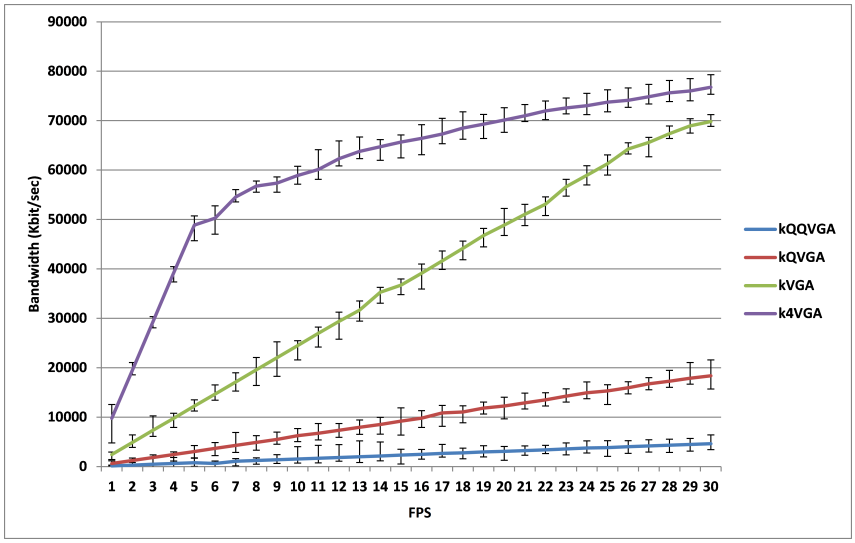


(b) Black and White

Figure 6.4: Wi-Fi video streaming
(Aagela, Holmes, et al., 2017)



(a) Colour



(b) Black and White

Figure 6.5: Ethernet video streaming (Aagela, Holmes, et al., 2017)

The primary distinction between experiments in Case 1 and Case 2 is that streaming over Wi-Fi data is limited to up to 8.3 Mbps.However, using Ethernet, it is about 82 Mbps. In addition, the use of the black and white video streaming required less data rate in contrast to the coloured video, as shown in both Figures 9.4 and 9.5.

6.2.3.3 Face Recognition Accuracy over Wi-Fi

In this experiment, the NAO robot has been used to examine the level of precision of the face recognition task using four distinct video qualities. This experiment investigates the effect of the video stream quality (frames per second and resolution) on the accuracy of face recognition. Designed in Choregraphe, the face recognition model is defined in Figure 6.2. It has been applied to eight recognised faces that have been previously captured. For each human face, the information recorded and the name of the individual are added to the model.

The experiments were constructed to detect a person's face using dynamic video streaming. As detailed in Figure 6.2, the robot will initially track any face within its field of vision. If a face is detected, the face recognition process will start working on recognise the individual by comparing it a match between any of the saved faces. Then it will begin tracking any possible face motion again ready for the next recognition. The accuracy of face recognition is calculated by using the machine learning formula outlined in (5), where true positive indicates a correct recognition of the human face and true negative indicates a wrong recognition of the human face.

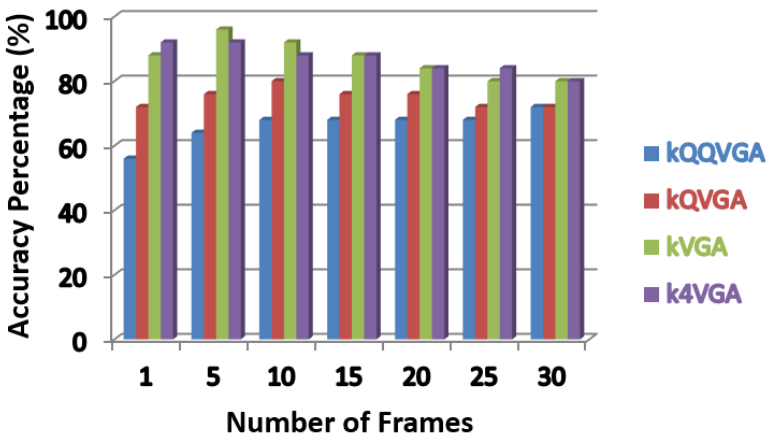


Figure 6.6: Face recognition accuracy rate for different video quality Colour video streaming

$$\text{Face recognition accuracy} = \frac{\text{True Positive+True Negative}}{\text{Total Number of Iterations}} \quad ..... \quad (5)$$

The number of iterations made for each video quality streaming in this case study is 100, and in each experiment the positive and negative detections of a human face are added to the face recognition formula (5). Figure 6.6 shows that with KVGA video quality and 5 fps, the maximum accuracy level in recognising the human face is 96 percent. However, using KQVGA video quality and 1 fps, the minimum accuracy level is closer to 56 percent. In addition, the black and white video streaming face detection accuracy shown in Figure 6.7 still has a high detection accuracy rate compared to the colour video stream. Using K4VGA, the highest detection accuracy level is 92 percent and the minimum detection rate using KQVGA is 46 percent. This is the only result that falls below the minimum acceptable accuracy rate.

It is clear that the FPS has a significant impact on the face recognition performance and overall accuracy. As shown in Figures 6.6 and 6.7, the accuracy

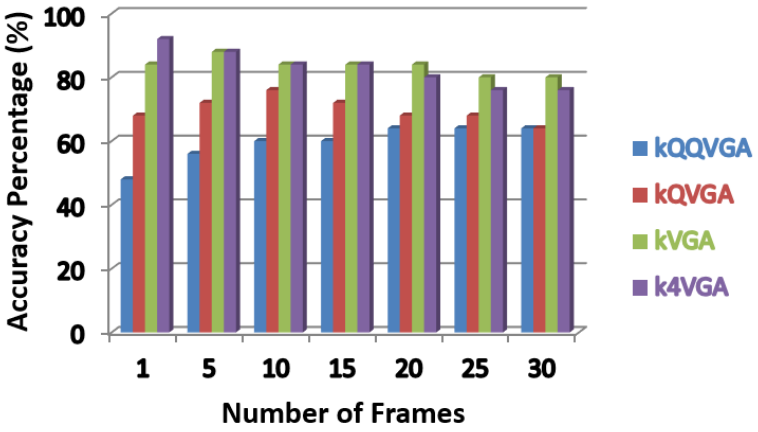


Figure 6.7: Face recognition accuracy rate for different video quality Black and white video streaming
(Aagela, Holmes, et al., 2017)

level for KQVGA video quality increases when the amount of frames is increased. This is due to the rise in bandwidth consumption on the robot side and low buffering rate. However, as the amount of frames rises, the accuracy rate for face recognition will reduce for K4VGA. It can be concluded that there is very high bandwidth consumption and buffering rate for this video quality.

KQVGA and KVGA's face detection accuracy will only improve for the first 10 frames. The detection rate will subsequently continue to decline or remain at the same percentage level. The average detection rate for different video qualities was calculated from the results acquired in this experiment. Table 6.1 demonstrates that for the KVGA colour video quality, the best possible face detection rate acquired from the experimental results is 86.86%. Having said this, it was found that the Black and White detection rate differs only by 3% which suggests it could be a suitable candidate for reducing the network

Average Detection Rate (%)	Video Quality			
	KQQVGA	**KQVGA**	**KVGA**	**K4VGA**
Colour	66.29%	74.86%	86.86%	86.81%
Black and White	59.43%	69.71%	83.43%	82.86%

Table 6.1: Average Face Recognition Detection Rate (Aagela, Holmes, et al., 2017)

requirements whilst maintaining accuracy.

6.3 Low Power and Cost Robot

This section explains the process of optimising the design of a low cost robot. The motivation was from the one of the key objectives of the cloud robotics concept, which is to reduce the need of expansive equipment on the on-board robot and move the heavy process to the cloud. This can allow a low cost robot platform to perform a complex task that needs high computational power and large storage.

The robot design is utilising the Raspberry Pi 3 with the Turtlebot 2 robot base as an alternative for the default compute unit which is typically a laptop. The reasons of using the Raspberry pi as a replacement were that the Raspberry pi 3 has a Wi-Fi capability that allow the robot to be mobile and has got better specifications as mention in section 6.3.1 in contrast to other single board computer, however, the Raspberry Pi 3 has limitation on its integrated RAM, therefore, a hot swappable RAM technique has been applied that allow the Raspberry pi to used some storage from the main SD card in order to cove this limitation. As Figure 6.8 shows, the Raspberry Pi is mounted on the

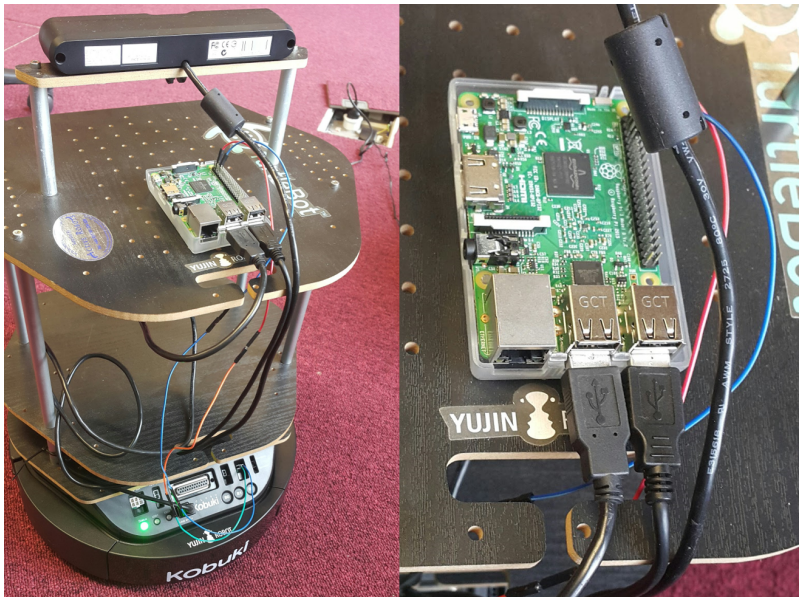


Figure 6.8: Utilising a Raspberry Pi as a main computer unit for Turtlebot II (Aagela, Al-Nesf, & Holmes, 2017)

top of the robot and connected to the sensors via USB cables. Moreover, this robot platform now has a single power system, where the Raspberry Pi uses the robot battery via available power output in the Turtlebot base as shown schematic diagram in Figure 6.9, in contrast to the laptop which requires a second charging point. In general, the use of the Raspberry Pi proved its usability and compatibility with the Turtlebot 2 robots. In addition, by installing the ROS, the robot gets access to a wide range of the desirable robot functionality as ready to use packages and software with minimal on-board footprint.

6.3.1 System Specification

This approach was planned to offer an affordable and effective indoor mapping system that can be used remotely with ROS in the cloud, mainly in dangerous

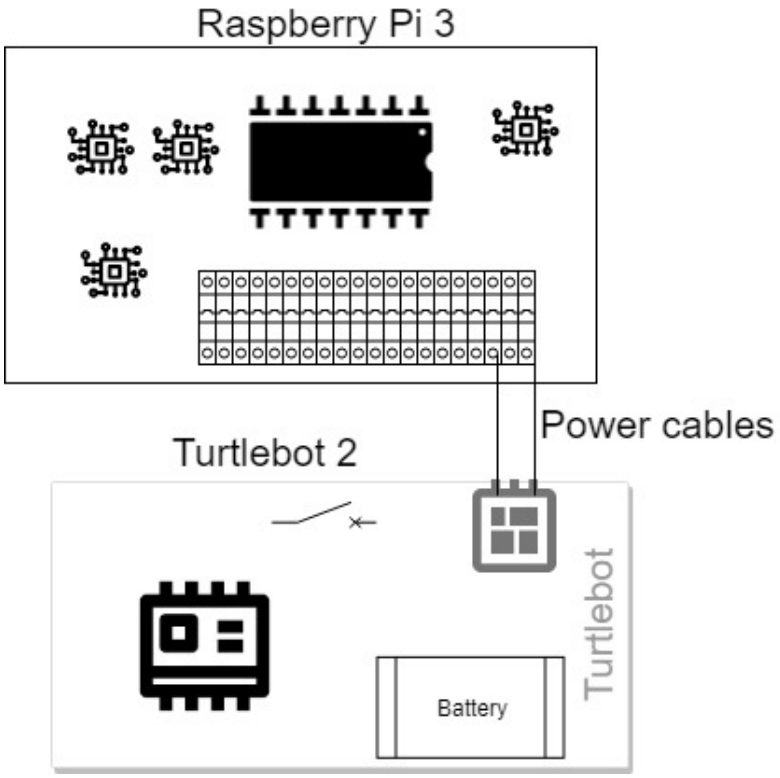


Figure 6.9: Schematic Diagram for Raspberry Pi powered by Turtlebot II base (Aagela, Al-Nesf, & Holmes, 2017)

environments for human operations. The key components of the proposed system are firstly, the Raspberry Pi, which is a single board computer that has most of the computer components built-in a small board. The Raspberry Pi 3 uses a 1.2 Ghz ARM processor and 1GB of RAM, (Aagela, Al-Nesf, & Holmes, 2017), which is used during in this research. The Raspberry Pi 3 runs on Ubuntu 16.04 Operating System that has been installed in a 32 GB Micro SD card. Moreover, ROS has been installed with its basic packages. The Raspberry Pi 3 connects to the Turtlebot base and the Asus xtion pro Sensor via USB cables. The Asus xtion pro is mounted in the middle of the robot.

A simplified diagram of the network architecture that is used by the robot is detailed in Figure 6.10. The robot is connected to a master node via a local Wireless-G network with a bandwidth of 54 Mbps Wi-Fi connection (Aagela, Holmes, et al., 2017). In order to configure ROS environment over our network, the master node PC and the Raspberry Pi were provided with private IP addresses as well as the hostname, which is important for the ROS communication system. Basic functionality tests, such as teleoperation, were successful demonstrating that the robot system can be used to evaluate the CCRP framework.

6.3.2 Power Consumption

In terms of the power saving and power consumption, the consumed power between the Raspberry Pi and laptop is contrasted, which has an ARM CPU and Intel quad core i5 processor respectively. According to (Anwaar & Shah,

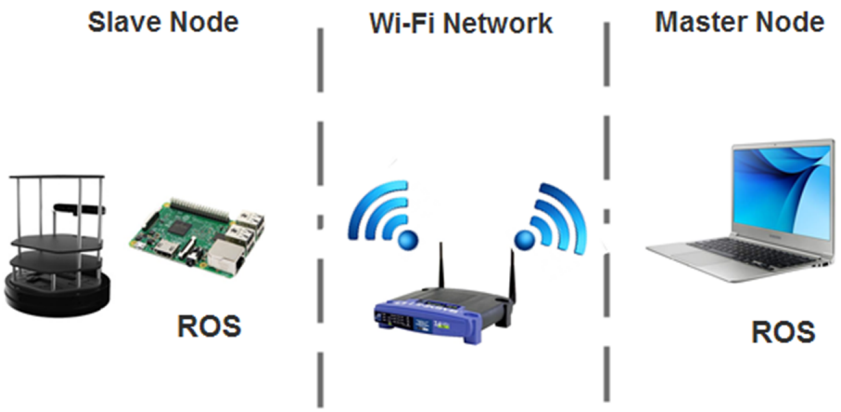


Figure 6.10: Low cost robot network architecture
(Aagela, Al-Nesf, & Holmes, 2017)

2015), the Raspberry Pi 3 only requires an average of 48 Kilojoule/day in contrast to the laptop which requires an average of 1050 Kilojoule/day, as shown in Figure 6.11. Therefore, the native TurtleBot compute unit consumes about 20 times more than the Raspberry Pi proposed system, allowing the optimised robot to operate for longer time with lower maintenance cost and simplified connectivity.

6.4 Summary

Firstly, this chapter explained the value of the conducting an optimisation evaluation analysis for network profiling and the on-board robot resources requirement before implementing any cloud robotics case studies. This assists in defining the optimal requirement robot and the cloud system, providing acceptable range of parameters for subsequent evaluation. The results indicate that there is a significant impact of the video streaming quality and network

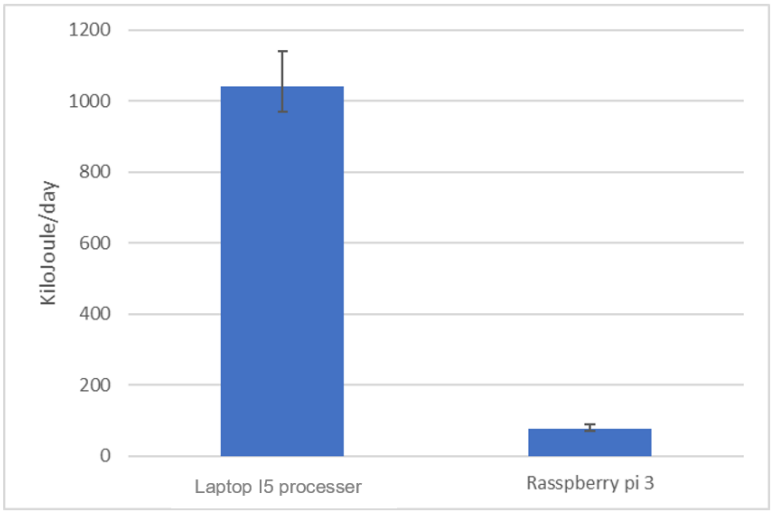


Figure 6.11: Low cost robot power consumption in Kilojoule comparison (Aagela, Al-Nesf, & Holmes, 2017)

buffering on the face recognition process. The finding proves that it is possible to enhance the network performance and consume less network bandwidth by selecting the right video streaming quality based on the robot task. Secondly, a quantitative measure of the video qualities types and the frame rate impact on the network bandwidth consumption on the distributed robot application is presented. An understanding of this impact will inform the development of real-time cloud robotics applications that run on the CCRP platform. Finally, the optimisation study considers the robot hardware, where a solution for utilising low cost computational resources on-board the robot which can support offloading to the cloud is proposed. The use of the Raspberry Pi 3 proved its usability and compatibility with the Turtlebot II robot, which uses ROS as its main framework and is supported on the low power ARM processor utilised by the single board computer. Moreover, our approach can save on-board robot considerable amount of power. The following chapters detail

the in-depth evaluation and implementation of real life case studies using the CCRP framework.

Chapter 7

CCRP Evaluation

7.1 Introduction

This chapter provides the performance evaluation of the CCRP that demonstrates the improvements of the clone-based model over the proxy model in developing multi-robots cloud robotics application. Indeed, the QoS is a major issue that face any cloud robotics implementation, therefore, this research considers the performance of the system and studies the impact of the response time on the overall system performance in both proxy-based model and clone-based model in a multi-robots ROS environment. The evaluation process includes, measuring the network latency performance and a test application measure the response time for both Clone-based model "CBM" and proxy-based model "PBM". Therefore, there are several tools and techniques that will be used to assess the system performance.

7.2 Network Performance

Although most of the cloud applications desire real-time communication, the latency can have a major impact on achieving it. In order to measure the network latency effectively for the CCRP, the round trip time (RTT) was measured to analyse the network latency between the robot/edge side and the public cloud that have been tested with two different geographical destinations as well as with and without the use of the VPN. The network latency usually occurs because of various reasons such as, type of communication media (wire/wireless), the distance between the sender and the receiver, and network protocols. The use of IaaS cloud service model allows us to carry out the measurement in two regions. Unlike the previous work, the CCRP integrates the VPN as a part of the network, therefore, the analysis of VPN integration is important to measure the additional network overhead that is added between the robot and the respective cloud instance. The Wireshark tool was used to measure the RTT, the result was taken from a 100 sample and calculating the RTT latency average. As shown in Figure 7.1, the RTT measurement for communications between the robot and the VMs runs on Google Cloud Platform across two datacenters, one runs in Europe EU region and other in United States US region. The result shows that the use of the VPN has miner impact on the RTT for both cases, however, there was dramatic increase on the RTT from around 50ms for EU server to about 180ms US server. Note that the robot/edge system is located at University of Huddersfield, United Kingdom.Where in the cloud robotics field the network latency is measure factor in determining the overall application performance, therefore, the selection

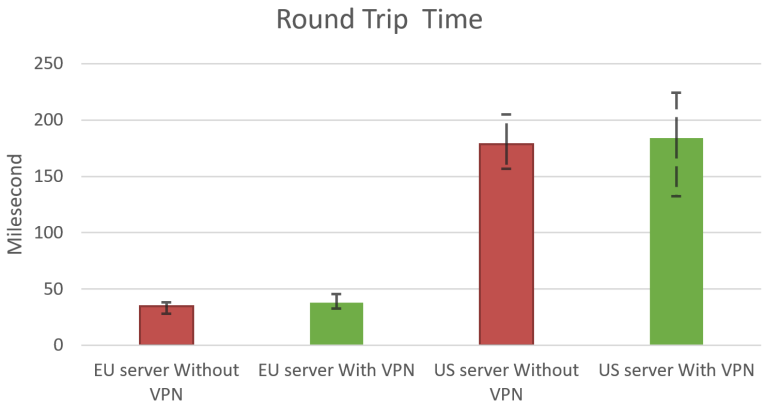


Figure 7.1: RTT measurement between robot and cloud VM with two servers, EU server and US server and with and without VPN

of the physical host location should be carefully considered.

7.3 Quality of Service and Application Performance

A Cloud solution was always advertised as a trade-off between computer resources (computational power and storage) and the bandwidth of the network. This exchange was proven to be effectively positive, as nowadays the network medias are capable of providing a high bandwidth and high availability for Internet connection. However, the network latency has a negative impact on the QoS and the application performance, as the sent data will be disrupted from its normal sequence. Nevertheless, the overall QoS performance defined by measuring the response time, which include the network latency and the processing time. In addition, the impact of network congestion that may occur when the server is busy and needs to hold data in a buffer to be processed

later can lead to packet loss and a reduction in endpoint user QoS, negatively impacting application performance.

The process of controlling the amount and size of the data that needs to be offloaded is a key factor to improve the QoS of any cloud robotics system. According to the results obtained from our optimisation study, the low quality of sensor data can be effective in terms of providing an acceptable accuracy rate for a robot task like face recognition with low bandwidth requirement. As a result, assisting the use of sensor data quality with this model is going to be essential in order to raise the awareness of importance of the network optimisation to control the use of the available bandwidth and reduce network latency.

7.3.1 Experimental setup

In order to evaluate and validate the performance of the proposed system, the system will be exposed to several kinds of testing that cover the best and worst use case for the system. A critical factor of the performance of offloading robot processes to a cloud platform is depends heavily on the response time, which is influenced by the network latency and processing time delay. The response time is going to be essential factor to analyse the impact of increasing the number of the linked robots in a multi-robot's environment. There are a three real robots (two NAO and one Turtlebot) and seven simulated Turtlebot robots, which are configured with ROS environment and linked to CCRP as well as to proxy-based model with one ROS master. For the simulated robot, both Gazebo and Rviz were used to create and control the virtual robots in

local VMs that are hosted in Virtualbox software. The test task was designed to execute a code that runs in fixed times 100ms and 500ms delay to simulates a robot tasks that consume various processing time to define the impact of the processing time on the overall performance of both scenarios. A ROS code has been developed to establish an application that consume fix time, the following code was used to control the execution time.

```
1 double secs =ros::Time::now().toSec();
2 ros::Duration d(0.5); // 0.5 equal to 500 ms
3 secs = d.toSec();
```

7.3.2 Experimental results

This section illustrates the outcome of the response times of both "CBM" and "PBM" with the designed time delay scenarios. Each one was measured by averaging the result of using each system with different number of robots from 1 to 10 robots, simultaneously. Initially, as shown in Figure 7.2 the result of running the 100ms delay application that displays both models were around neck and neck while using up to two robots at around 150ms. Then, the PBM gradually increase to reach approximate 270ms in case of using 10 robots, while the result remains steady with the CBM approach at around 153ms.

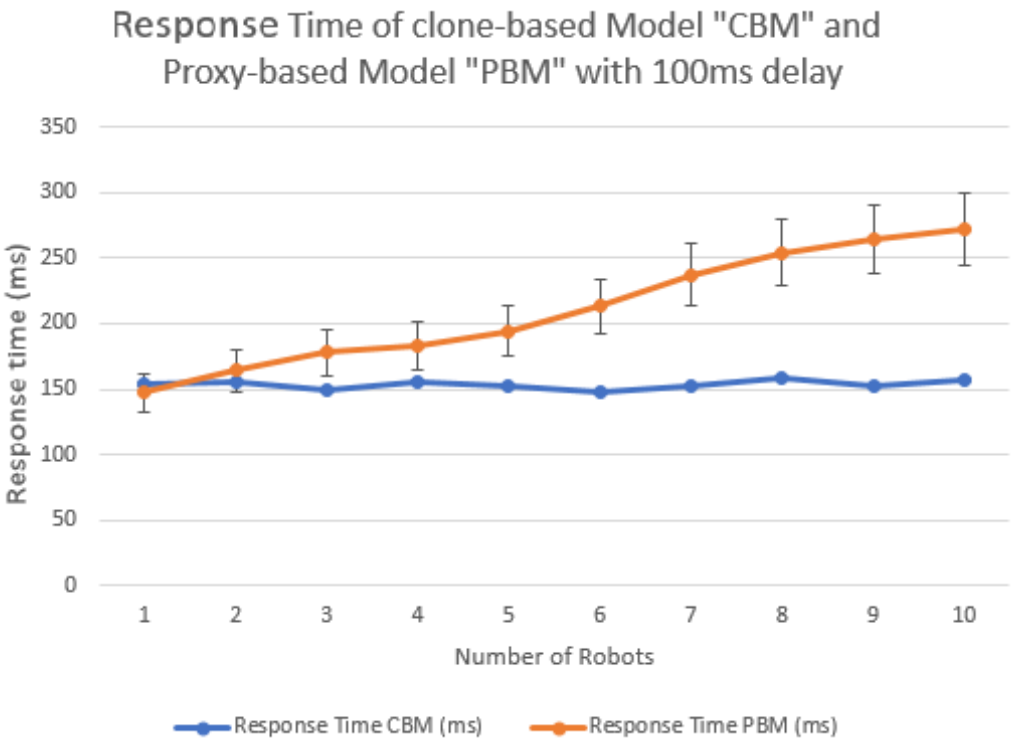


Figure 7.2: Response time of Clone-based model "CBM" and proxy-based model "PBM" with 100ms delay

In the second scenario, where 500ms processing time is applied as shown in Figure 7.3, similarly to the previous scenario the result was very close at approximately 550ms between the two model in case of using one or two robots. However, the results rise significantly by around 2 second for the PBM whilst 10 robots was used. In contrast to CBM, which shows a linear result that remain steady regardless to the number of used robots. Therefore, the CCRP system which adopted the use of CBM will provide a higher QoS and thus stable application performance, where the number of robots linked to the multi-robots' environment will not impact the performance.

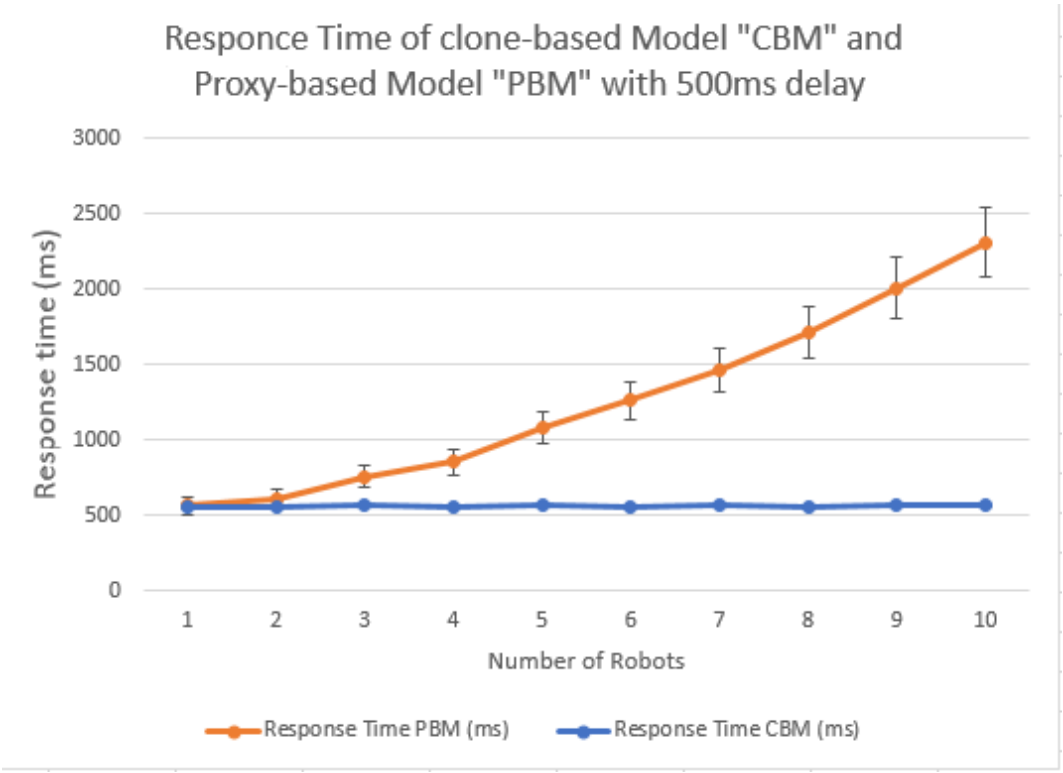


Figure 7.3: Response time of Clone-based model "CBM" and proxy-based model "PBM" with 500ms delay

7.4 API Manager Performance

It is one of the essential robotic task to recognise objects in the surrounding environment. However, this task could be challenging for the on-board processing capability of robots with limited capacity. Thus, the research suggest to use the Google Vision API to perform the object recognition by the developed API Manager. The object recognition process requires access to a huge image dataset that could be used to identify the object details. Table 7.4 shows the object recognition algorithm that has been deployed in the system.

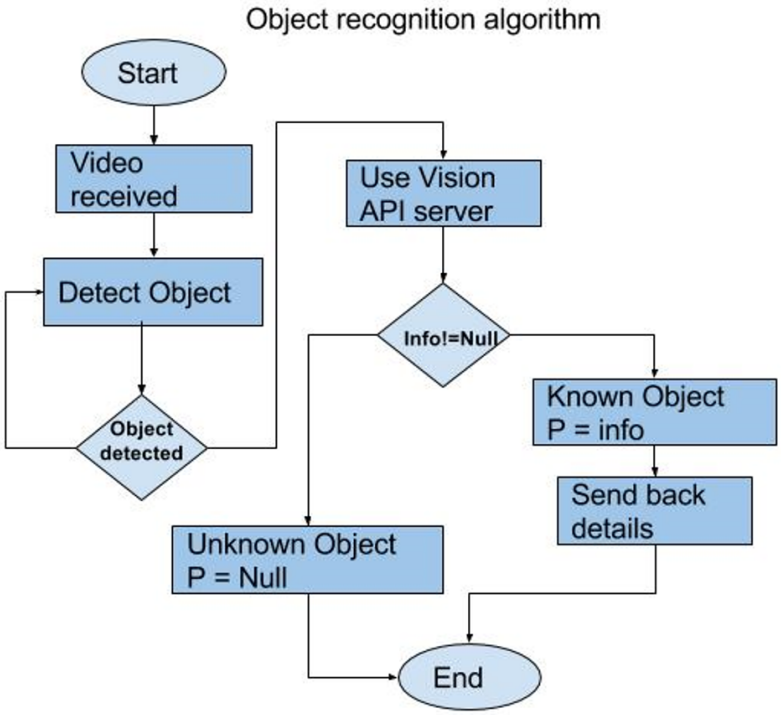


Figure 7.4: Object recognition algorithm

The object of this use case is to test the performance of the developed API manager and test the compatibility and usability of the external API in real-time applications. Table 7.1 illustrates the object recognition confidence rate for number of objects by using the Google Vision API. The outcome of this test was approximately 90% confidence. Therefore, it could be considered as an accurate solution for object recognition.

In addition, in this scenario the humanoid robot NAO will need to say the recognised object, thus, the API manager is going to use the Text-to-speech Google API to covert the name of the object to an audio, then send the file to the NAO.

Object Name	Confidence $\mu \pm \sigma$	Failure rate %
Disk	0.81 ± 0.06	3
Chair	0.89 ± 0.08	-
Human	0.93 ± 0.06	-
Door	0.86 ± 0.09	7
Laptop	0.93 ± 0.03	6

Table 7.1: Outcome of the Google Vision recognition service during the experiment testing the API manager

Following the previous performance tests, the API manager was evaluated by measuring the response time for the object recognition services that uses the Google Vision API for various image qualities have been examined. As Figure 7.5 shows the object recognition response time with four video qualities, the result clearly demonstrates the different response time that is needed based on the picture quality, where increasing the offloaded picture quality will increase the response time correspondingly. The minimum response time that aims to be obtained with the lowest quality will be just over 200 ms. However, the impact of reducing the image quality on the obtained accuracy level was considered as suggested in our optimisation study. Based on the testing result, the KQVGA was selected to be the optimal image quality for this experiment.

The developed API manager has been designed to control the request event between the robot and the cloud. In this test the Google Vision API was used to recognise a number of objects within the lab environment. Table 7.1 illustrates the level of the conference rate μ with the error rate σ that produced by repeating the test 50 times for each object. In addition, the table shows the percentage of the failure rate, when the outcome of the API failed in the

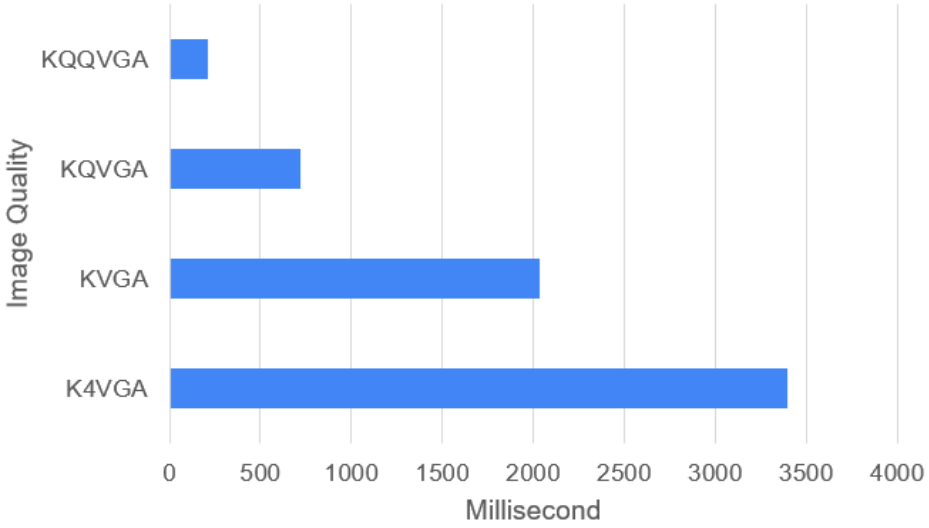


Figure 7.5: The object recognition response time with four video qualities

recognition process.

As a result of this test give a high level of confidence approximately 90%. Thus, with the right configuration the API manager is capable of utilise the Google Vision API to allow the robot recognise object with high accuracy and response time at around 700 ms. Based on collected results, it can be considered as a acceptable real-time interactions with humanoid robot.

7.5 Summary

This chapter analyses the QoS and performance of the CCRP system. The evaluation process consisted of several experiments. The network performance was examined by measuring the RTT for the network traffic between the robot and the cloud sides. The objective of this measurement was to identify the expected network delay in the process of offloading a data or task to the

cloud. Moreover, it delivers a critical analysis of the impact of cloud datacenter location and how it can minimise the network latency significantly. Moreover, it was proven that the use of VPN did not add drastic overhead delay on the transmitted traffic.

The second part aimed to evaluate the general application performance and the QoS of the CCRP demonstrated promising and stable performance on the given scenarios. The evaluation of the developed API performance and how it operates with some of the result that showed the performance of the API manage on running the Google Vision API was acceptable and in practice can provide response time of around 700ms which is reasonable for "real-time" interactions with a humanoid robot. The overall evaluation results showed that the CCRP capability is effective in keeping the stability while running simulated robots tasks.

Chapter 8

Real Time Robot Teleoperation and Mapping

8.1 Introduction

This chapter discusses the first robot case study application which utilizes the CCRP in a real world robot field. Chapter 4 signposted the use of the teleoperation and mapping tasks with the CCRP. Nowadays, the mobile robots become more popular with a perceived a great improvement in technology. However, many obstacles prevent the robots to be used widely in our daily activity, such as robot cost and hardware and software performance in operating tasks, such as mapping, teleoperation and path planning. The real-time control is an important robot application, where the robot is going to be listening to the tele-commands the operator sends and respond accordingly.

In this chapter the real-time teleoperation is used as a case study for two reasons, first of all, it is a vital application for robots as well as it can define the level of the real-time performance of the system. In terms of how fast and accurate the robot will respond to the given command. In addition, there is a need to offload the video from the robot to display it on the operator side and that will cost network bandwidth that can impact the performance. The objective of this case study is to examine the system capability to run a real-time remote-control action as well as testing the network behaviours between the robot and the controller application to identify the impact of the latency on the real-time teleoperation.

Robot mapping focuses on giving the robot the ability to learn about the surrounding area and is essential for mobile robots in order to be able to perform localisation and the path planning (Hamzeh & Elnagar, 2015), (Aagela, Holmes, et al., 2017). Therefore, creating a virtual environment for the surrounding area of the robot that could be used as a map would be vital task, which is needed to be used by another application that could be useful in operations like military, search and rescue and smart cities and so forth. Sending the robot into rough missions and exploring unknown places that could be dangerous for a human to explore means that the robot itself will be in danger, therefore, the cost of the robot that could perform such an operation will be a vital factor. Hence, decreasing the cost of the hardware and the software for the robot is desirable. Mapping an indoor environment will require a laser scan, a mobile robot base and compute unit with wireless communication. Thus, the robot will gather the depth information by the Asus Xtion pro 3D

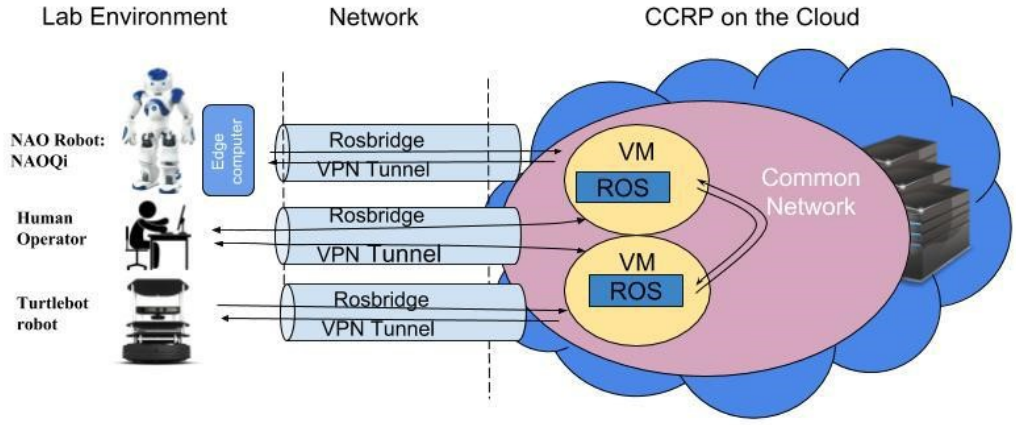


Figure 8.1: Teleoperation and mapping CCRP System architecture (Aagela & Holmes, 2019a)

sensor as stated in various implementations. For instance, creating a map for indoor environment. Localising and mapping within indoor environments is actually vital for remote controlled robots. There are several approaches for the mapping that have been researched which utilise laser scanners (Rehman, 2013).

The motivation for this work is to assess the network performance and functionality performance of the teleoperation and mapping processes with the CCRP system for heterogeneous robots in the real-time. As shown in Figure 8.1, the teleoperation and mapping CCRP system architecture allows the used robots to be linked via a network with respective VMs in the cloud. Similarly, an operator can connect directly to the required robot via its CRI, in order to control the targeted robot. The following publications have arisen from my research detailed in this thesis: *Cloud Robotics-Based System for Robot Teleoperation* by (Aagela & Holmes, 2019a).

8.2 Robot Teleoperation

There were number of other related work that considers a teleoperation platform for heterogeneous robots. (Hoffman, 2016) presented the humanoid robot OpenWoZ software that is implemented as a web service running on ROS, and a cloud-backed multi-platform client. The OpenWoZ server uses representational state transfer (REST) protocol to manage the connection requests from a number user simultaneously. A system proposed by (Magyar et al., 2017), named CoWoOZ was developed as a cloud-based teleoperation software, which can be used via a web dashboard to manage and control the robot. (Small et al., 2018) stated that the teleoperation task is an important part of other robotic tasks, for instance, grasping, mapping and navigation and so on. The existing cloud-based teleoperation platforms for controlling robots, which were proposed in reviewed publications, show that the research focus is on developing environment for the targeted types of robots and tasks.

The CCRP cloud-based teleoperation model is displayed in Figure 8.2, where operator will sending teleoperation signals using the CCRP, the ROS teleoperation package has additional autonomous correction feature that used to synchronise the state between the operator side and the robot environment. The response of the operator send back to the robot actuators that act accordingly. The robot sensors were constantly used to send the feedback such as video stream from a robot camera, and Odometry data that determines the robot's pose and next move. The experiments here used both the Turtlebot and the NAO robots. Note, NAO only used with the teleoperation task due to lack of mapping capability.

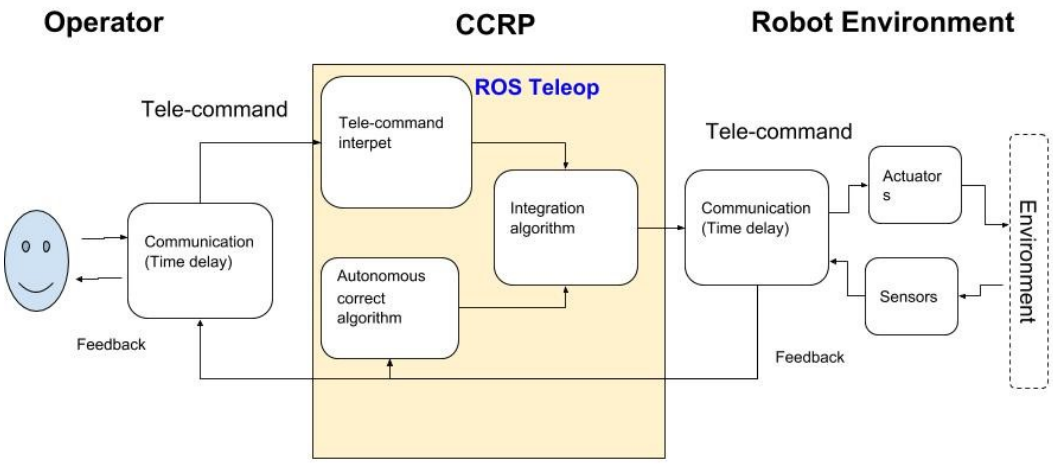


Figure 8.2: CCRP Teleoperation model
(Aagela & Holmes, 2019a)

The idea of the model is to extend the distance between the robot and its operator, where the robot should receive a tele-command and act accordingly. The cloud role is to analyse the tele-command by using the tele-command interpret software, which works as translator between the operator and the robot actuators. In addition, the autonomous correct algorithm integrates with the received command to filter the unexpected action based on the sensor status or changing in the environment. There are a number of supported controller pads that could be used with ROS, including the keyboard, Xbox controller and touch screen.

8.3 Mapping

The process of creating a map for unknown area is considered as continuous process even at times when the robot is steady, the laser scans that are received

from the Asus xtion pro that produces 30 depth images per second. The basic process of SLAM can be summarised in a workflow shown in Figure 8.3, where map generation is a continuous process initiated by using the depth data collected from the robot's sensors, which can define the distance and the shape of the surrounding walls, people, objects, etc.

Then, the position of the robot is estimated and saved it at certain point of time. Finally, the local map will be updated by drawing a simulated 2D map of the floor layout.

8.4 Experimental setup

In this case study, the capability of CCRP running the teleoperation and mapping tasks in multi-robot environments were examined. An operator can remotely connect to the targeted robot via its robot CRI. The experiments were conducted by using a number of hardware and software elements. The used hardware were: NAO robot and Turtlebot robots as shown in Figure 8.4, In addition, a laptop for NAO robot work as edge computer and Raspberry pi for Turtlebot robot that used as compute unit. The navigation stack which include the teleop package was installed for both Turtlebot and NAO in their CRI.

A fundamental teleoperation movement for both robots are applied in indoor environment. As shown in Figure 8.5, the teleoperation activity is designed so that the robots are required to move from point A to B and avoid an one obstacle placed after 1.5m from point A. The Wireshark and rostopic

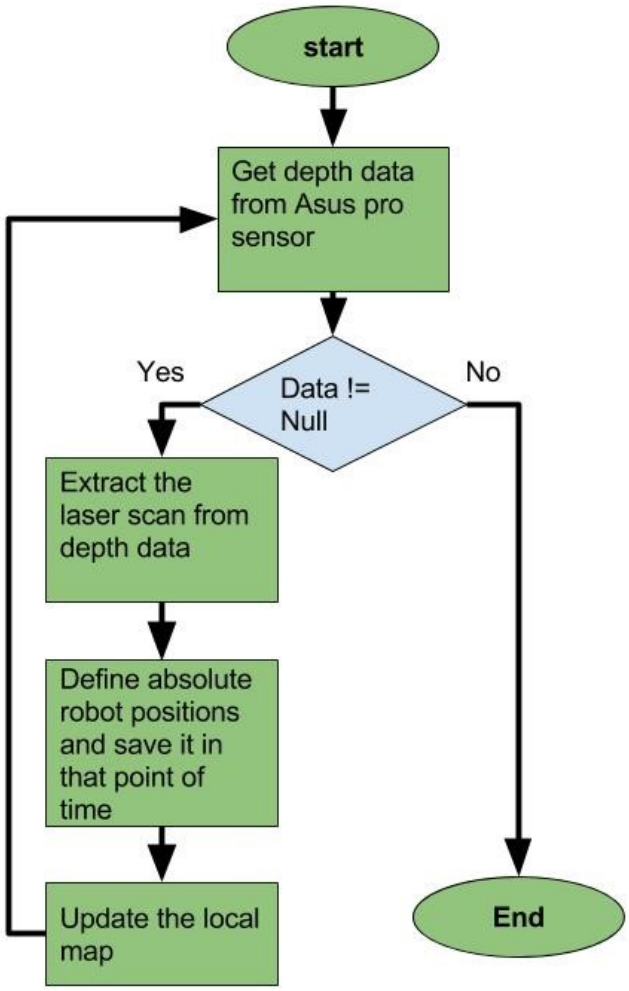


Figure 8.3: Workflow for the mapping process on the cloud (Aagela & Holmes, 2019a)

Figure 8.4: The experimental robots: A) Turtlebot robot B) NAO Humanoid robot

(Aagela & Holmes, 2019a)

delay/bandwidth 'bw' tools were used to measure the network performance between the operator and robots, it is important to recognise that there are two side of the connection, between robot to cloud and robot to the cloud.

Normally, the depth information that is received through the 3D sensor, in our case the Asus sensor, is converted into a laser 12-format data on a real-time basis, which is deemed as the initial data that is needed by the entire process. Thus, to be able to begin the generation of the first local grid map as well as the first pose of the robot, where the middle of the map gives the provision for the initial location of the robot, an application called Gmapping is used which initiates the process of generating a local grid map for limited area (Balasuriya et al., 2016), (Aagela, Al-Nesf, & Holmes, 2017). The process of creating the map requires the robot to move around the room and scan it by using the 3D

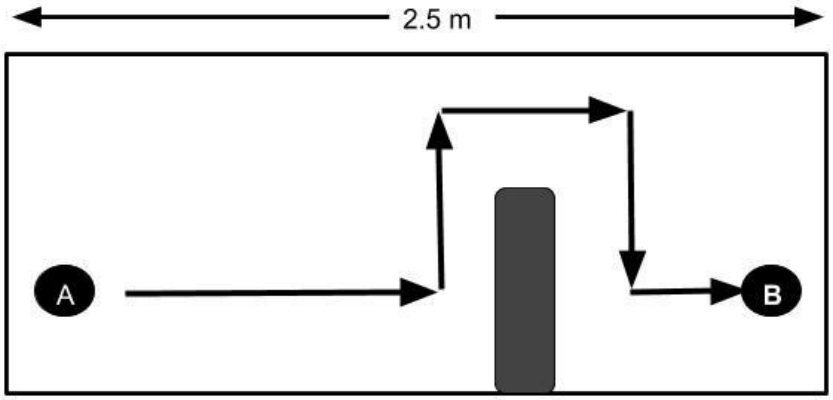


Figure 8.5: Robots are required to move from a point A to point B avoiding an obstacle placed 2.5m from their initial position
(Aagela & Holmes, 2019a)

sensor. The scan has been conducted whilst the robot is moving at a speed of 25cm/s forward, which is considered an average speed given the Turtlebot robot which is capable of a maximum speed of 50cm/s. The Asus xtion Pro sensor has the ability to provide depth information by the use of an IR beam that is reflected from the facing wall against the robot. In this test the design system is linked to CCRP system to allow the remote process for the map to be done by the cloud.

8.5 Experimental results

To evaluate the CCRP teleoperation algorithm performance, the robot response delay, video transmission delay and response time delay were measured, by processing the gathered data from 30 experiments for each robot. In addition, to evaluate the capability of the low cost robot in producing an accurate map for the surrounding environment by comparing it with a map created by

the native system.

8.5.1 Transmission Delay

As shown in Figure 8.6, the video transmission delay was about 150ms, where the result were similar in the performance for both robots. The response delay represents a time delay between sending a tele-command signal to receiving the control message in the robot side. It was on average at about 250 ms for Turtlebot robot and 273ms for NAO robot. The NAO robot shows slightly slower in response time. However, it took around 70s compared to the Turtlebot which takes only around 25s to accomplish the given task. The difference in time here was due to the difference in robots movement speed. The results of the experiments illustrated that the CCRP cloud robotics system was effective in handling teleoperation and mapping tasks, and can enable a real-time remote control for different kind of robots in multi-robot environments (Aagela & Holmes, 2019a).

8.5.2 2D Mapping

As shown in Figure 8.7a, the low cost robot defined in Chapter 6 was able to create a 2D map for the room. In contrast, the same experiment was repeated with using the laptop and with local ROS system instead as Figure 8.7b displays. The indoor map task for unknown environment was challenging for the Raspberry Pi on board the Turtlebot, where the generated map had unclear edges for some of the room walls compared with the map generated by the na-

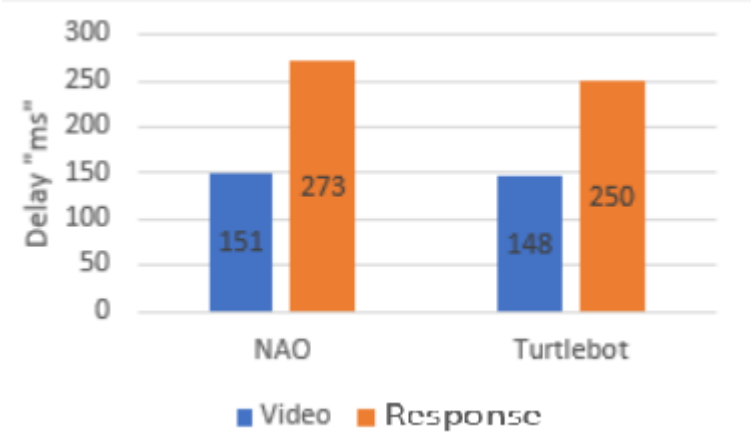
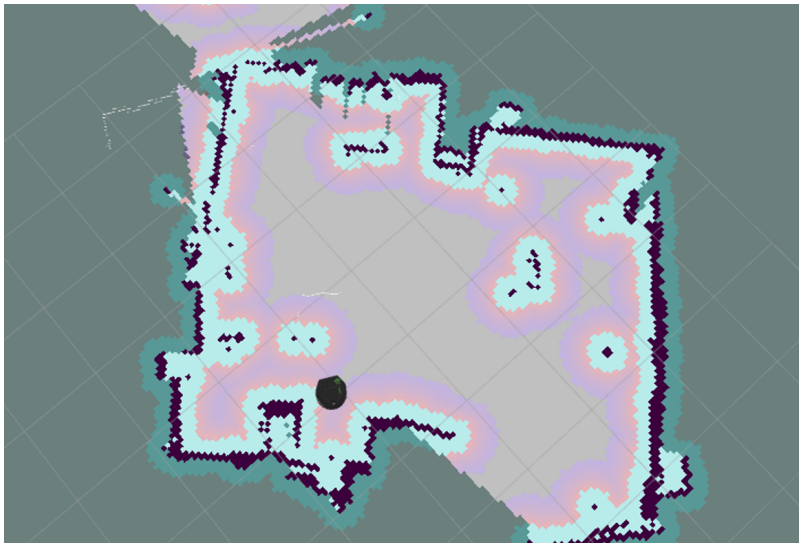
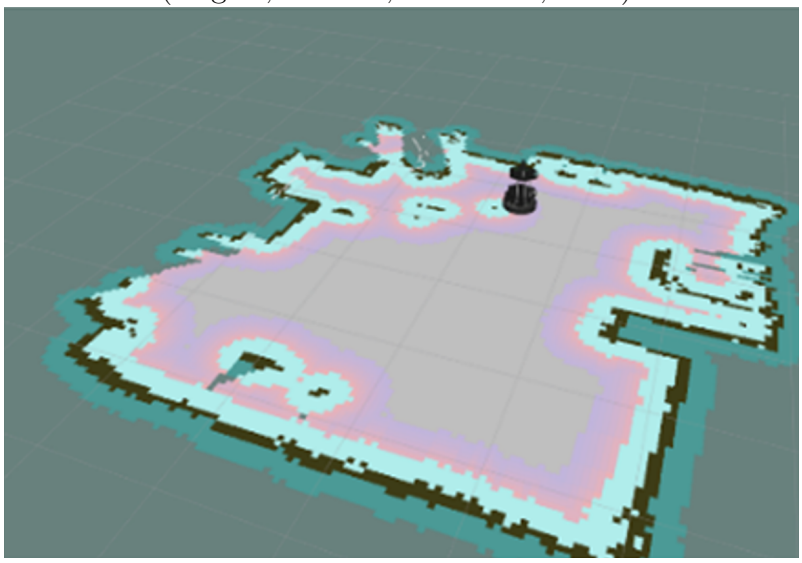


Figure 8.6: Experimental results of the delay time in millisecond for the response time delay and the video delay
(Aagela & Holmes, 2019a)

tive robot resources, which had more accurate and sharper for wall edges and objects position. However, the first result was still enough to perform navigation process on the obtained map. The mapping process depends on defining the location of the robot according to the last time pose that the was done by the scanner sensor in the previous map, this process can fail for a number of reasons, particularly when the robot start rotating or moving with a high speed. This issue occurs because it needs more processing power to process the generated map grid as well as the network latency. Thus, the proposed solution was sensitive for the robot speed and movement. It is suitable to run the SLAM process while moving the robot with slow speed up to 30cm/s to obtain an accurate 2D map (Aagela & Holmes, 2019a) as the speed ranged from 5cm/s up to 30cm/s provided a clear map that can be used for the navigation task. however, speed from 35cm/s up to the maximum speed 50 cm/s provide a poor map that lead to failure in the navigation process.

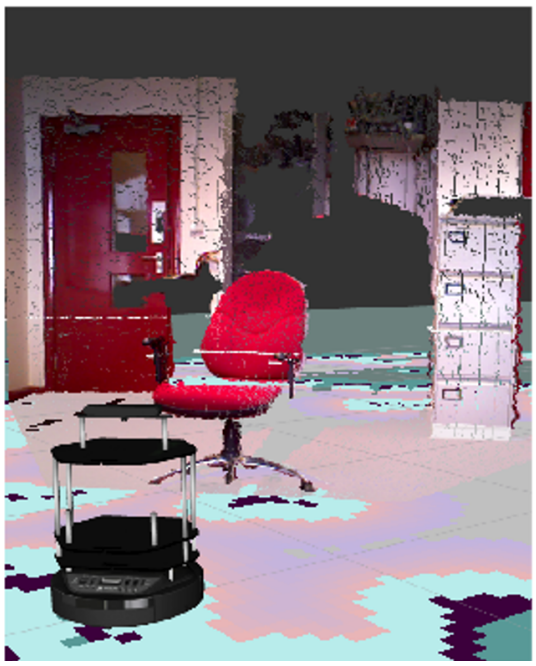


(a) RPi with CCRP
(Aagela, Al-Nesf, & Holmes, 2017)



(b) Laptop with local ROS
(Aagela, Al-Nesf, & Holmes, 2017)

Figure 8.7: Map of the University of Huddersfield Robotic Lab
(Aagela, Al-Nesf, & Holmes, 2017)



(a) RPi with CCRP

(b) Laptop with local ROS

Figure 8.8: 3D model generated by Turtlebot (Aagela, Al-Nesf, & Holmes, 2017)

8.5.3 3D Object Model

One of the processes that uses intensive processing power and requires high bandwidth is to create a 3D model for the environment using a point cloud application, which use the depth image data collected by Asus Xtion pro sensor.

In addition to the response measurements and 2D mapping functionality, the difference between the Raspberry Pi in contrast to the laptop were benchmarked, in terms of the network and processing capability in generating a 3D model of the surrounding environment. Based on a subjective analysis of the results shown in Figure 8.8, the quality of the 3D model that generated by the Raspberry Pi with the CCRP was very close to the one that generated by the laptop with the local ROS.

8.6 Summary

This chapter presents a cloud-based robot teleoperation and mapping application which aims to evaluate the performance capability of the CCRP in executing some sensitive real-time robot tasks for two kind of robots. The teleoperation was an effective method in merging a robot skill and a human operator ability, through remote manual control. The proposed CCRP system was able to resolve the lack of the resources in our available robots and overcome constraints posed by lack of information and communication constraints. The outcome of those experiments displays that the CCRP is capable

of running manual teleoperation and mapping application (with an operator's control) due to a low robot response time of between 250ms and 273ms. In addition, it can support heterogeneous robots within the multi-ROS environment. However, for Simultaneous Localisation and Mapping application, there was a trade-off between the movement speed of the robot and the quality of the created map. The speed of the robot may impact the performance of the mapping and teleoperation tasks with the current network latency level, therefore, the controlled robot need to go through speed test performance before can be used. Moreover, the operator in our experiments can view a video streaming from the robot camera, where the result shows the delay on receiving the video was less than the response time of the robot at around 150ms.

Chapter 9

Collaborative Cloud-based Face Recognition for Multi-robots Environment

9.1 Introduction

This chapter discusses the use of Face Recognition (FR) application with the CCRP. The FR is a key task for humanoid robots, which includes both face detection and recognition (Abbas Shangari, Sadeghnejad, & Baltes, 2016). But, they require extensive computing capability and storage, but developing a robot with high processing and storage characteristics can be expensive. Collaboration between robots can overcome the limitation of tasks processed using on-board resources. The key task for the FR applications is defining unique features of the elements in a human face. There are various FR al-

gorithm available, such as Linear Discriminant Analysis (LDA) (Li & Yuan, 2005), and Principal Component Analysis (PCA) (Ismail et al., 2011). The performance of those algorithms are tested and fairly acceptable amongst research in the field of FR. However, they have issues of being computationally intensive (Bolotnikova et al., 2017). There were other projects that attempt to move the FR task to the cloud, according to (Tian, Saitov, & Lee, 2014), who developed a peer-based cloud robotic system which allows a humanoid robot to perform a FR in real-time by utilising the intensive computational power of the cloud. The project utilised ROS as middleware and programming environment. Nevertheless, a knowledge-sharing mechanism was not available in this system and there was a lack of published information on the performance of the system. Similarly, a project called Cloudlet, which was a mobile fog computing system, allows a robot to send images to the cloud via a wireless network linked the robots with a smartphone to perform FR task(Stojmenovic, 2014). The Cloudlet was limited in terms of the mobility and coverage, where the system can only support the robot in the local network reach. in addition, it also lack the capability of sharing the knowledge.

The objectives of this case study is design and implement a real-time FR cloud robotics application, which can allow NAO humanoid robot to perform the FR task in the cloud with acceptable response time and accuracy rate. Additionally, evaluating the use of CCRP with a real life robot application by analysing the FR application performance of CCRP against the local ROS environment, quantifying the impact of the latency on the respective tasks. The CCRP aims to address the lack of sharing knowledge between robots

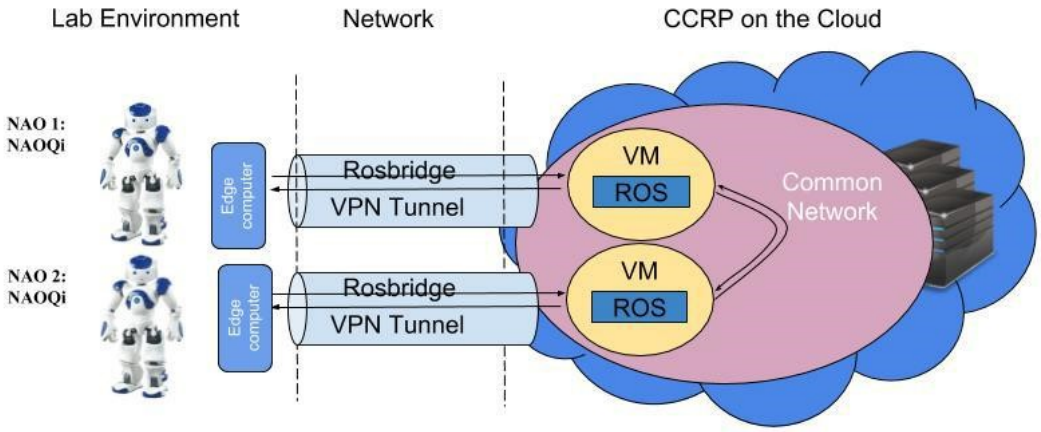


Figure 9.1: CCRP Architecture and Methodology
(Aagela & Holmes, 2019b)

by providing a collaborative learning environment. Also, the system aims to reduce the computational complexity in the robot and offload computation and storage to a cloud as shown in Figure 9.1. The following publications have arisen from my research detailed in this thesis: *Collaborative Cloud-based Face recognition approach for Humanoid robots* by (Aagela & Holmes, 2019b).

9.2 Collaborative Cloud-based FR Algorithm

In this section devises our new cloud-based FR algorithm, which can be explained in Figure 9.2. The algorithm suggest offloading the FR task to the cloud. The robot upload the captures video to the CCRP system. The algorithm has several elements that are used for basic face awareness, face detection, and FR. These elements are responsible identifying the human face. The Faceserver application in ROS will look for a human face in the received images. which is going to be compared against the existing trained images

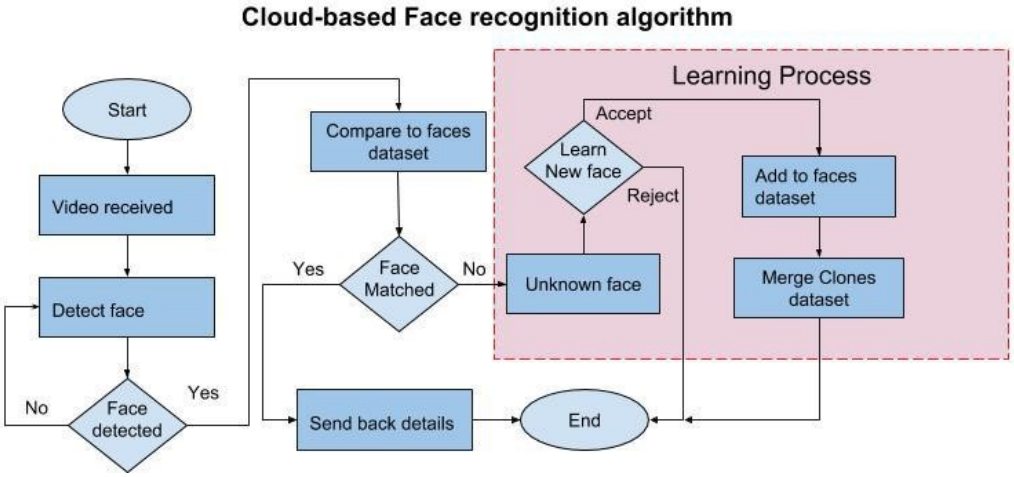


Figure 9.2: The cloud-based face recognition algorithm (Aagela & Holmes, 2019b)

dataset that stored in the cloud. If the face is found, the cloud sends back the person name back to the robot; but, if the face is unknown, the application starts a learning process, which give the the operator chance to add a new record to the face dataset. The learning mechanisms works by merging the newly-learned face to the existing records to be shared between robots every 5 minutes to allow the data to be updated meanwhile not do the update too often. Each robot is capable of storing new faces in a cloud and sharing the trained image data with other robots via the common network. The FR algorithm was executed with 2D coloured video captured by the NAO robot's main camera, with a Video Quality KQVGA, which has resolution of 320x240 and frame rate of 5 fps. The selection of the video quality and frame rate was based on our previous optimisation study as mentioned in chapter 6 (Aagela, Holmes, et al., 2017), (Aagela & Holmes, 2019b).

9.3 Experimental setup

The University's private cloud was deployed using OpenStack and utilised for this experiment. For this experiment two VMs were created on the cloud that runs Ubuntu 16.04 OS. Each VM was created with four cores and eight GByte of RAM. In terms of the edge machine for the NAO robots, two laptops with Ubuntu 16.04 were used. The CCRP system was installed and configured in the cloud VMs as well as the robot edge machines.

All the FR experiments were conducted within the same environment, whereas, the room lighting, image background and the distance between the individuals' faces to the robots attempted to be similar in each test, Hence, those factors can have an impact on the obtained results. The distance between the robot and the individuals' face were between 50 to 70 cm. In the experiments, each robot captured and trained the images for five people. Then, the system synchronised and merge the data between the two robots. The overall collected data by the two robot are over 300 trained images taken from 10 individuals. The experiments were conducted in two scenarios, first performing FR using local ROS environment, and using CCRP. The comparison between these two scenarios will define the impact of offloading the task of FR to the cloud, and will examine the diversity in the algorithm response time and accuracy rate.

Trained image count	Confidence	Failure rate
1	0.21	55
5	0.33	35
10	0.69	15
20	0.76	0
30	0.82	0

Table 9.1: Local FR approach accuracy and failure rate (Aagela & Holmes, 2019b)

Trained image count	Confidence	Failure rate
1	0.18	65
5	0.32	40
10	0.63	15
20	0.77	5
30	0.83	0

Table 9.2: Cloud-based FR approach accuracy and failure rate (Aagela & Holmes, 2019b)

9.4 Experimental results

This section shows the outcome of offloading the FR task to the CCRP system. which proven its capability of acquiring knowledge of new human faces and preforming FR task real-time using created datasets. in addition, it was able to share the knowledge of new faces with other robots. The outcome of the FR tasks in the multi-robot system with local ROS environments were illustrated in Table 9.1. which is evident that the level of the accuracy increases when the number of the trained images increases, whilst the failure rate decreases.

The outcome of the CCRP FR approach are shown in Table 9.2. The accuracy of FR is very close to the one obtained by the local (edge) FR. Therefore, the CCRP approach demonstrated the capability of offloading the FR tasks in the

cloud with a miner affecting on the accuracy rate. Note, each result that shown in previous tables was an average of 20 attempts in FR for a given scenario. The response times performance for FR using local and CCRP are shown in Figure **??**. The result shows the local approach response time were increasing as the number of image increases, from approximately 200 ms with 1 trained image to more that 400ms with 30 trained images for 10 people. However, the CCRP shows a slight increase in the response time, from about 250ms while processing 1 trained image to just above 300 ms, while processing 30 trained images for each individual. The CCRP FR system was able to outperforms the local FR system when there were more than 20 trained images per individual in the dataset. The proposed CCRP has an additional communication delay, due to the distance between the robot and the cloud, Thus, initially, it shows lower performance than the local system. The process of exchange the trained images was done successfully between the robots (Aagela & Holmes, 2019b).

9.5 Summary

This chapter presented part of this research that attempted to reduce the computational complexity in the humanoid robot system by moving the process of FR to the CCRP. A new Collaborative Cloud-based FR Approach Was designed , which was implemented with the NAO humanoid robots, however, it should be applicable for other robots that support by ROS and the ability to capture video data. As shown in the results section, the CCRP proved

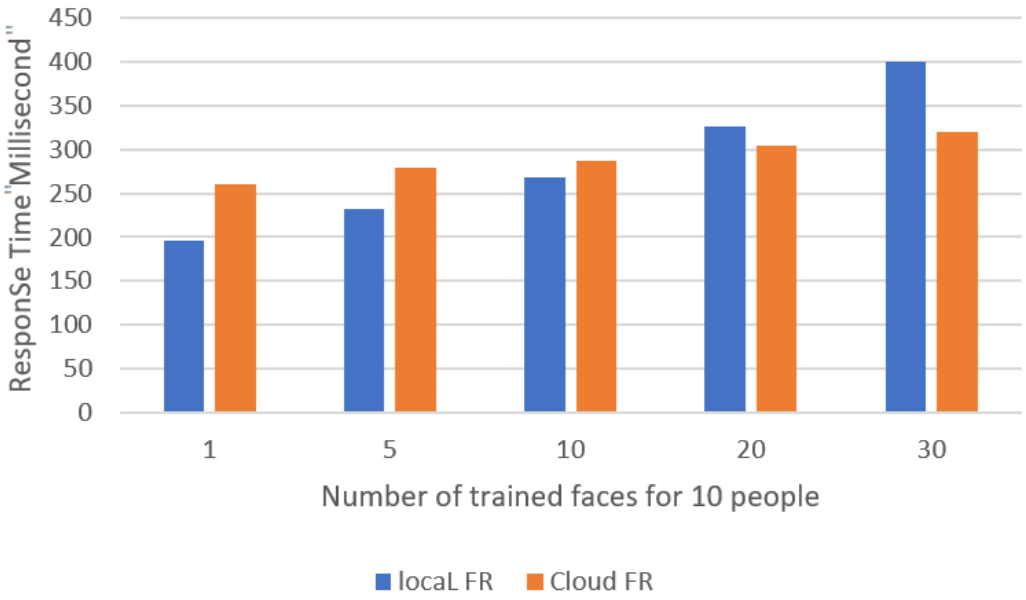


Figure 9.3: Response time of the local FR and the cloud FR (Aagela & Holmes, 2019b)

its capability in running the FR tasks effectively and it managed to outperform the native local system with limited computational power and storage after 20 trained image for each individual. The robot (edge) solution will be less effective as the dataset increases, which lead to longer response times as shown in 9.3. In addition, the CCRP was able to facilitate knowledge sharing between the robots in adequate way. The new learned trained faces images were kept at the dataset of the robots' CRI, then they will be shared with other robots within the same ROS multi-master environment managed by the CCRP framework.

Chapter 10

Conclusion and Future Work

10.1 Conclusion

In conclusion, the cloud robotics concept is an emerging paradigm providing access to computing services as a utility for robot applications remotely. Based on the initial review of the published work in the field, the cloud was already used as a solution to improve the on-board robot performance for both single and multi-robot environments.

ROS was used as a key element in most of the cloud robotics solutions, however, the deployment of multi-robot environments was limited and there were some previous works that vary in their approaches. On one hand, some research suggests the use of the proxy-based model, which is limited in terms of the resource availability. Others have used the clone-based model, such as Rapyuta project, which adopted the use of multi-master ROS approach, in order to

increase the robot robustness in multi robotics framework solution. However, due to the continuous upgrades released by the ROS framework authors, the robot applications usually supported are in limited ROS distribution groups, as Rapyuta only support the early ROS distributions. Other issues, namely, the difficulty of configuration of the multi-master environment and the additional network overhead due to the use of the container.

10.1.1 CCRP Availability and Scalability

As the research developed, the lack of availability of a unified compatible multi-robot' environment becomes visible. Therefore, the need of an alternative solution is important. In this thesis a novel multi-robot cloud robotics platform was demonstrated that addresses the shortcoming of QoS performance, compatibility and scalability in ROS cloud robotics framework, though the clone-based cloud robotics platform CCRP.

CCRP was inspired from the Rapyuta framework, but, the design of the system has distinguishing features and supports more modern versions of ROS and Ubuntu. The clone-based cloud robotics has a generalised architecture that inherited the ROS capability and is compatible with ROS-based robots and ROS distributions. This architecture provides a secure, collaborative and configurable system to allow the robots to offload their on-board computational and memory intensive tasks onto the single/multi-robot cloud computing environments, which can be deployed into public or private cloud infrastructures. Moreover, the Multi-ROS approach allows the CCRP to be a scalable solution that provides robots with dedicated resources, while allowing secure knowl-

edge sharing with other robots. In terms of the network design, the use of a VPN technology provided the system with two main advantages: enhancing the security and allowing the deployment of the multi-master ROS in a variety of network typologies.

10.1.2 API Integration

The CCRP can provide the linked robots with external cloud application access by using the API manager, motivated by the fact that the robots are increasingly required to work in the same environment with other robots and share some of the tasks such as navigation, mapping, and image processing. Therefore, collaborative learning between the robots is becoming more important. It will optimise the robots performance and use of resources by reducing common task repetitions. Collaborative learning will help the robots to inherit the knowledge that have been acquired previously by other robots.

10.1.3 Research Outcomes

The initial evaluation attempts to measure the impact of the physical location of the cloud provider on the network latency as well as analysing the effect of using the VPN. The outcome of this test has proven that it is vital for the selected VM to be as close as possible to the linked robot. Also, the use of the VPN has a minor effect on the overall network delay. In addition, the quality of Service (QoS) and Application Performance took place to evaluate the performance of the Clone Based Model (CBM) against the Proxy Based

Model (PBM), where CBM had no impact when the number of robot increases, while it had a negative impact on the response time for the PBM and this impact rise more as the number of robots increases. The response time can be a major factor that determines the QoS and Application Performance for most cloud robotics tasks. As part of the evaluation process for CCRP, there were two cloud robotics tasks that were designed and implemented. The first case study was the mapping and teleoperation task, which aimed to assist the CCRP capability in handling a hard real-time robot application as well as handling various robots type. The CCRP was able to accomplish the the real-time mapping and teleoperation. where the tasks were fully offloaded to the cloud the CCRP system and were able to deliver response time in a range of 250ms to 273ms depending on the robot used. However, the TurtleBot robot highlights a limitation in the communication network while moving in a speed more than 30 cm/s, which is about two third of the maximum Turtlebot robot speed.

The other case study conducted face recognition, which aimed to examine the capability of the CCRP in running complex tasks and testing the knowledge sharing capability over the multi-master network. Our new FR algorithm was capable of detecting and recognising human faces and send back feedback to the robot.

The result shows the local approach response time were increasing as the number of image increases from 1 to 30 trained image the response time was doubled from 200ms to more than 400ms. On the other hand, the CCRP shows a better performance, where with the same trend the response time increased

from about 250ms to 300ms. Therefore, the cloud approach proves its capability to exceed the local system with bigger dataset, which is more applicable when dealing with a realistic scenario for FR applications. The result showed that the number of the trained images for each individual have a direct impact on the confidence and failure rate of the system. where when the trained image increase the FR confidence rate improved and the failure rate decreased. The new approach was able to facilitate knowledge sharing between the robots without any difficulties. The new learned facial images were stored at the data set of the robots CRI, then they were shared and were able to be accessed by other robots within the same ROS multi-master environment. The case study in Chapter 9 evaluates this capability within the context of CCRP, fulfilling the research objective to perform tasks in an unknown environment based on multi-robot acquired knowledge.

From this evaluation, the outcome of this work addresses the initial research aim, to create a generalised architecture for offloading intensive processes to the cloud, proven by utilising the system to execute several complex tasks remotely. In terms of the communication issues, which can be defined as one of the fundamental problems which limits the adoption of cloud robotics implementations, our results validate the possibility of running robot tasks in a real-time manner by raising the awareness of the impact of network optimisation in enhancing the efficiency of using the available network bandwidth and managing the network latency. This can be controlled by the use of the IaaS cloud services model and can have a major impact on determining the real-time cloud robotics application performance. In addition, the CCRP has

been tested with both public cloud and private cloud deployments. Privacy was a key factor in deciding to develop the private cloud support, a factor that leads organisations to develop their own cloud providers, avoiding the need of the third party provider in case of running a sensitive data or application. Therefore, validating the use of the private cloud with our system was essential.

10.1.4 Research Contributions

There is no need to say that one thesis will not be able to solve all challenges of cloud robotics research field. However, when devising complete multi-robots' cloud-based framework, it will have to deal with these challenges at least marginally. Especially, system integration and compatibility must be addressed whenever a new element is introduced and whenever an existing element is utilized in a way that was not expected by the original developers. The key objectives of the work, which led to the results defined in this thesis, is to permit a various type of robots to offload complex tasks to a unified cloud platform that allow robots to collaborate with each other. In order to accomplish those objectives, numerous well-known tools and robot libraries and software were integrated in a single system. The main contributions of this thesis are attempting mainly to address the Security, Communication, Heterogeneity and Knowledge representation challenges. The contributions are defined as follows:

- The main achievement of this work is that design a unified clone based multi-robots cloud robotics platform, which is support the integration of

the ROS applications with every supported robot that has networking capability. The lack of the performance of the Proxy-based environment discussed in chapter 6 led to increase the need of developing such as robust platform. Although, a number of other frameworks tried to tackle this issue, which have been explained individually in chapter 2. They still have some critical issues such as compatibility, security, availability and so on, which have to be considered while designing our new architectures. The evaluation of the new system will require testing the system performance and applicability for some complex robot's scenarios.

- Current robot perception systems allow robots to operate several vision tasks such as face recognition and object recognition. Due to high requirement of computational and storage for those robot's applications, in addition to challenge in the sharing knowledge procedure between robots. In Chapter 8 it will be established how face recognition vision task can be improved by moving the process to the cloud and developing a collaborative approach that allow robot to share the new learned data with others. This should limit the requirement in the robot side and increase the optimization where the process of learning one individual will not be repeated.

- The network and hardware optimization is a challenge that can determine cloud robotics application performance. Chapter 9 investigates the question of whether there is an significant impact of the transmitted data quality selection on the network and task performance, and how can analysing those factors can led toward increasing efficiency of the

system. In addition, investigating the potential of developing a low cost robot that can work almost as efficient as the high aspect robot with the integration of the proposed solution.

10.1.5 System Limitations

Despite the demonstrated advantages, the CCRP still has some limitations:

- The brainless state issue, which can occur in case of network failure. The robot will not be able to complete the given task on-board if the connection with the master node / clone image is lost.
- The CCRP still limited in terms of automation and self-manageability, where several configurations still need to be done manually by the operator in both the cloud and robot sides.
- In terms of using the raspberry pi as alternative solution to develop a low cost robot, the lack of the available RAM on the model raspberry pi 3 lead to reduce the performance of the map creation.

10.2 Future Work

As the CCRP presents itself as a general cloud robotics solution for ROS supported robots, the features will need to be verified with different kind of robots, as well as more case studies. There are a number of future research path and idea can be accomplished based on this work.

10.2.1 Cooperative Navigation

- The navigation task can be further developed on the cloud in order to allow the generated map to be shared between the robots in the same workspace.
- The teleoperation process can be investigated more by combining an autonomous and operator-based control of remote robots to achieve the best teleoperation result. The installation process can be developed to be more autonomous and simpler, by creating two scripts contain the basic elements for both cloud and robotic sides. In addition, running the robots though more complex tasks and complicated maps, i.e. Using the robot to create a MAP of the University and use it as shared general map by other robot via the CCRP.

10.2.2 Face Recognition Application

- The result of the FR experiments task can be verified further by using larger faces datasets, in order to verify and validate the FR with CCRP in the case of using big data. In addition, introducing the system with some real life application for the FR.
- Improving the knowledge sharing process of the learned faces can be vital, where an autonomous mechanism can be developed that only start the update of the faces database when one of the robot acquiring a new trained data.

10.2.3 Brainless Robots Issue

- A network awareness application that can run in the robot side to evaluate the strength and the state of the connection between the robot and the cloud provider is a potential solution to this problem which can apply machine learning. Basically, the application can work as an autonomous switch that helps in allocating the robotic tasks based on the state of the network or a possibility to trigger a default application. It is necessary to overcome one of the most important cloud robotics challenges, which is the potential for a brainless robot, which might happen when the robot loses the connection to the cloud.

10.2.4 Web-based Application

- Developing a web-based application that provides the authorised users with a direct link to their robots, which can make the process of connection to the cloud VM much easier. The website can contain various function such as robot section, where the same user can have multi-robot direct access to the VM file system. In addition, this would enable an online terminal that will allow the users to write or run their ROS application faster as well as allow user visualise some of sensor data such as video, battery level and so on.

- The web application can include the Wireless wake-on-LAN, which defined in details in Appendix E, adding the function of controlling the power status of listed robots to the main dashboard. By enabling the

user ability to turn ON or OFF the robots remotely, it can add more optimisation to the cloud robotics system, where robot only works upon request. Finally, it would facilitate a notification channel providing alerts to the user in case of any error which occurs on the robot.

References

Aagela, H., Al-Nesf, M., & Holmes, V. (2017). An asus_xtion_probased indoor mapping using a raspberry pi with turtlebot robot turtlebot robot. In *Automation and computing (icac), 2017 23rd international conference on* (pp. 1–5).

Aagela, H., & Holmes, V. (2019a). Cloud robotics-based system for robot teleoperation. In *Emit19, isbn: 978-0-9933426-2-6, 2019 conference on university of huddersfield* (pp. 1–3).

Aagela, H., & Holmes, V. (2019b). Collaborative cloud-based face recognition approach for humanoid robots. In *Emit19, isbn: 978-0-9933426-2-6, 2019 conference on university of huddersfield* (pp. 1–4).

Aagela, H., Holmes, V., Dhimish, M., & Wilson, D. (2017). Impact of video streaming quality on bandwidth in humanoid robot nao connected to the cloud. In *Proceedings of the second international conference on internet of things and cloud computing* (p. 134).

Abbas Shangari, T., Sadeghnejad, S., & Baltes, J. (2016). Importance of humanoid robot detection. *Humanoid Robotics: A Reference*, 1–9.

Agüero, C. E., Koenig, N., Chen, I., Boyer, H., Peters, S., Hsu, J., ... oth-

ers (2015). Inside the virtual robotics challenge: Simulating real-time robotic disaster response. *IEEE Transactions on Automation Science and Engineering*, *12*(2), 494–506.

Al-Aqrabi, H., & Hill, R. (2018). Dynamic multiparty authentication of data analytics services within cloud environments. In *20th ieee international conference on high performance computing and communications (hpcc-2018), ieee computer society.*

Al Aqrabi, H., Liu, L., Hill, R., & Antonopoulos, N. (2014). A multi-layer hierarchical inter-cloud connectivity model for sequential packet inspection of tenant sessions accessing bi as a service. In *High performance computing and communications, 2014 ieee 6th intl symp on cyberspace safety and security, 2014 ieee 11th intl conf on embedded software and syst (hpcc, css, icess), 2014 ieee intl conf on* (pp. 498–505).

Ali, S. S., Hammad, A., & Eldien, A. S. T. (2018). Fastslam 2.0 tracking and mapping as a cloud robotics service. *Computers & Electrical Engineering*, *69*, 412–421.

Anwaar, W., & Shah, M. A. (2015). Energy efficient computing: a comparison of raspberry pi with modern devices. *International Journal of Computer and Information Technology*, *4*(02).

Arumugam, R., Enti, V. R., Bingbing, L., Xiaojun, W., Baskaran, K., Kong, F. F., ... Kit, G. W. (2010). Davinci: A cloud computing framework for service robots. In *Robotics and automation (icra), 2010 ieee international conference on* (pp. 3084–3089).

Austin, G. (2018). Robots writing chinese and fighting underwater. In *The political economy of robots* (pp. 271–290). Springer.

Balasuriya, B., Chathuranga, B., Jayasundara, B., Napagoda, N., Kumarawadu, S., Chandima, D., & Jayasekara, A. (2016). Outdoor robot navigation using gmapping based slam algorithm. In *2016 moratuwa engineering research conference (mercon)* (pp. 403–408).

Bogue, R. (2016). The role of robots in the battlefields of the future. *Industrial Robot: An International Journal*, *43*(4), 354–359.

Bolotnikova, A., Demirel, H., & Anbarjafari, G. (2017, Sep 01). Real-time ensemble based face recognition system for nao humanoids using local binary pattern. *Analog Integrated Circuits and Signal Processing*, *92*(3), 467–475. Retrieved from `https://doi.org/10.1007/s10470-017-1006-3` doi: 10.1007/s10470-017-1006-3

Cheng, A. L., Bier, H., & Mostafavi, S. (2017). Deep learning object-recognition in a design-to-robotic-production and-operation implementation. In *2017 ieee second ecuador technical chapters meeting (etcm)* (pp. 1–6).

da Silva, B. M., Xavier, R. S., do Nascimento, T. P., & Gonsalves, L. M. (2017). Experimental evaluation of ros compatible slam algorithms for rgb-d sensors. In *2017 latin american robotics symposium (lars) and 2017 brazilian symposium on robotics (sbr)* (pp. 1–6).

Doriya, R., Chakraborty, P., & Nandi, G. (2012a). Robotic services in cloud computing paradigm. In *Cloud and services computing (iscos), 2012 international symposium on* (pp. 80–83).

Doriya, R., Chakraborty, P., & Nandi, G. (2012b). 'robot-cloud': A framework to assist heterogeneous low cost robots. In *2012 international conference on communication, information & computing technology (iccict)* (pp. 1–

5).

Fazli, P., Davoodi, A., & Mackworth, A. K. (2013). Multi-robot repeated area coverage. *Autonomous robots*, *34*(4), 251–276.

Furht, B. (2010). Cloud computing fundamentals. In *Handbook of cloud computing* (pp. 3–19). Springer.

Garage, W. (2011). Turtlebot. *Website: http://turtlebot. com/last visited*, 11–25.

Ghani, M. F. A., Sahari, K. S. M., & Kiong, L. C. (2014). Improvement of the 2d slam system using kinect sensor for indoor mapping. In *2014 joint 7th international conference on soft computing and intelligent systems (scis) and 15th international symposium on advanced intelligent systems (isis)* (pp. 776–781).

Gliem, J. A., & Gliem, R. R. (2003). Calculating, interpreting, and reporting cronbach's alpha reliability coefficient for likert-type scales..

Google cloud platform overview. (2018, Dec). Retrieved from `https://cloud.google.com/docs/overview/` (Accessed: 2019-01-11)

Gouveia, B. D., Portugal, D., Silva, D. C., & Marques, L. (2015). Computation sharing in distributed robotic systems: A case study on slam. *IEEE Transactions on Automation Science and Engineering*, *12*(2), 410–422.

Hamzeh, O., & Elnagar, A. (2015). A kinect-based indoor mobile robot localization. In *2015 10th international symposium on mechatronics and its applications (isma)* (pp. 1–6).

Hoffman, G. (2016). Openwoz: A runtime-configurable wizard-of-oz framework for human-robot interaction. In *2016 aaai spring symposium series.*

Hosseini, H., Xiao, B., & Poovendran, R. (2017). Google's cloud vision api

is not robust to noise. In *2017 16th ieee international conference on machine learning and applications (icmla)* (pp. 101–105).

Hu, G., Tay, W. P., & Wen, Y. (2012). Cloud robotics: architecture, challenges and applications. *IEEE network*, *26*(3).

Hunziker, D., Gajamohan, M., Waibel, M., & D'Andrea, R. (2013). Rapyuta: The roboearth cloud engine. In *Icra* (pp. 438–444).

Inaba, M. (1997). Remote-brained robots. In *Ijcai* (pp. 1593–1606).

Ismail, L., Shamsuddin, S., Yussof, H., Hashim, H., Bahari, S., Jaafar, A., & Zahari, I. (2011). Face detection technique of humanoid robot nao for application in robotic assistive therapy. In *2011 ieee international conference on control system, computing and engineering* (pp. 517–521).

ITU, T. S. S. O. (2003). *Itu-t recommendation g. 114-one-way transmission time.* May.

Jamsa, K. (2011). *Cloud computing.* Jones & Bartlett Publishers.

Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *374*(2065), 20150202.

JoSEP, A. D., KAtz, R., KonWinSKi, A., Gunho, L., PAttERSon, D., & RABKin, A. (2010). A view of cloud computing. *Communications of the ACM*, *53*(4).

Juan, S. H., & Cotarelo, F. H. (2015). Multi-master ros systems. *Institut de Robotics and Industrial Informatics*, 1–18.

Jusuf, F. (2016). Auto-navigation for robots. implementation of ros. *Automation Engineering*.

Kamei, K., Nishio, S., Hagita, N., & Sato, M. (2012). Cloud networked

robotics. *IEEE Network*, *26*(3), 28–34.

Kehoe, B., Patil, S., Abbeel, P., & Goldberg, K. (2015). A survey of research on cloud robotics and automation. *IEEE Trans. Automation Science and Engineering*, *12*(2), 398–409.

Keller, E., & Rexford, J. (2010). The" platform as a service" model for networking. *INM/WREN*, *10*, 95–108.

Kuffner, J., & Robots, C.-E. (2010). In proc. of the ieee intl. In *Conf. on humanoid robots, nashville.*

Kumar, R., Pattnaik, P. K., & Pandey, P. (2017). *Detecting and mitigating robotic cyber security risks.* IGI Global.

Li, M., & Yuan, B. (2005). 2d-lda: A statistical linear discriminant analysis for image matrix. *Pattern Recognition Letters*, *26*(5), 527–532.

Liu, H., Li, F., Xu, X., & Sun, F. (2018). Multi-modal local receptive field extreme learning machine for object recognition. *Neurocomputing*, *277*, 4–11.

Liu, X. F., Shahriar, M. R., Al Sunny, S. N., Leu, M. C., & Hu, L. (2017). Cyber-physical manufacturing cloud: Architecture, virtualization, communication, and testbed. *Journal of Manufacturing Systems*, *43*, 352–364.

Magyar, G., Sinčák, P., Magyar, J., Yoshida, K., Manzi, A., & Cavallo, F. (2017). Cowooz—a cloud-based teleoperation platform for social robotics. In *2017 ieee 15th international symposium on applied machine intelligence and informatics (sami)* (pp. 000049–000054).

Mahmood, Z., & Saeed, S. (2013). *Software engineering frameworks for the cloud computing paradigm.* Springer.

Manzi, A., Fiorini, L., Esposito, R., Bonaccorsi, M., Mannari, I., Dario, P., & Cavallo, F. (2017). Design of a cloud robotic system to support senior citizens: The kubo experience. *Autonomous Robots*, *41*(3), 699–709.

Mell, P., Grance, T., et al. (2011). The nist definition of cloud computing.

Mohanarajah, G., Hunziker, D., D'Andrea, R., & Waibel, M. (2014). Rapyuta: A cloud robotics platform. *IEEE Transactions on Automation Science and Engineering*, *12*(2), 481–493.

Navale, V., & Bourne, P. E. (2018). Cloud computing applications for biomedical science: A perspective. *PLoS computational biology*, *14*(6), e1006144.

Paradies, Y. (2006). A systematic review of empirical research on self-reported racism and health. *International journal of epidemiology*, *35*(4), 888–901.

Peter, M., & TIM, G. (2010). The nist definition of cloud computing. association for computing machinery. *Communications of the ACM. New York: Association for Computing Machinery*.

Rehman, U. A. (2013). Using robots and slam for indoor wi-fi mapping in indoor geolocation [revised and extended version].

Ren, K., Wang, C., & Wang, Q. (2012). Security challenges for the public cloud. *IEEE Internet Computing*, *16*(1), 69–73.

Rosado, T., & Bernardino, J. (2014). An overview of openstack architecture. In *Proceedings of the 18th international database engineering & applications symposium* (pp. 366–367).

Ros distributions — ros. (2018, Dec). Retrieved from `http://wiki.ros.org/Distributions` (Accessed: 2019-03-13)

Ros : Intro to the robot operating system — robohub. (2013, Jun). Retrieved from `Robohub.org` (Accessed: 2018-12-22)

Roth, G., Livingston, J., Blair, M., & Kolonay, R. (2010). Create-av davinci: computationally-based engineering for conceptual design. In *48th aiaa aerospace sciences meeting including the new horizons forum and aerospace exposition* (p. 1232).

Saxena, A., Jain, A., Sener, O., Jami, A., Misra, D. K., & Koppula, H. S. (2014). Robobrain: Large-scale knowledge engine for robots. *arXiv preprint arXiv:1412.0691*.

Sefraoui, O., Aissaoui, M., & Eleuldj, M. (2012). Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, *55*(3), 38–42.

Serrano, N., Gallardo, G., & Hernantes, J. (2015). Infrastructure as a service and cloud technologies. *IEEE Software*, *32*(2), 30–36.

Small, N., Lee, K., & Mann, G. (2018). An assigned responsibility system for robotic teleoperation control. *International journal of intelligent robotics and applications*, *2*(1), 81–97.

Song, Y., Guo, S., Yin, X., Zhang, L., Wang, Y., Hirata, H., & Ishihara, H. (2018). Design and performance evaluation of a haptic interface based on mr fluids for endovascular tele-surgery. *Microsystem Technologies*, *24*(2), 909–918.

Sotiriadis, S., Bessis, N., Antonopoulos, N., & Hill, R. (2013). Meta-scheduling algortithms for managing inter-cloud interoperability. *International Journal of High Performance Computing and Networking*, *7*(2), 156–172.

Stojmenovic, I. (2014). Fog computing: A cloud to the ground support for smart things and machine-to-machine networks. In *2014 australasian telecommunication networks and applications conference (atnac)* (pp. 117–122).

Tian, S., Saitov, D., & Lee, S. G. (2014). Cloud robot with real-time face recognition ability. *Adv. Sci. Technol. Lett*, *51*, 77–80.

Tuna, G., Gulez, K., Gungor, V. C., & Mumcu, T. V. (2012). Evaluations of different simultaneous localization and mapping (slam) algorithms. In *Iecon 2012-38th annual conference on ieee industrial electronics society* (pp. 2693–2698).

TurtleBot2. (2014). *Turtlebot2 open-source robot development kit for apps on wheels.* Retrieved from `https://www.turtlebot.com/turtlebot2/`

Tzafestas, S. G. (2013). *Introduction to mobile robot control*. Elsevier.

Vaske, J. J., Beaman, J., & Sponarski, C. C. (2017). Rethinking internal consistency in cronbach's alpha. *Leisure Sciences*, *39*(2), 163–173.

Waibel, M., Beetz, M., Civera, J., d'Andrea, R., Elfring, J., Galvez-Lopez, D., ... others (2011). Roboearth. *IEEE Robotics & Automation Magazine*, *18*(2), 69–82.

Wang, L., Liu, M., & Meng, M. Q.-H. (2012). Towards cloud robotic system: A case study of online co-localization for fair resource competence. In *2012 ieee international conference on robotics and biomimetics (robio)* (pp. 2132–2137).

Zamora, I., Lopez, N. G., Vilches, V. M., & Cordero, A. H. (2016). Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo. *arXiv preprint arXiv:1608.05742*.

Appendices

Appendix A

Engagement with Research Community

Refereed Publications

Aagela, H., Holmes, V.: "Novel clone-based cloud robotic model to overcome limitation of the multi robots QoS.", Robotics and Autonomous Systems, Elsevier Journals.

Abstract: *"Cloud computing technologies have had a critical impact on various research fields. Recently, a new path of research, cloud robotics, has emerged enabling robots to exploit the capabilities of cloud computing. This can enhance the robots' performance dramatically and help to overcome the lack of on-board robot resources. Nevertheless, implementing a cloud robotic platform that is capable of handling multi-robot applications can be challeng-*

ing. This paper presents a new clone-based cloud robotic architecture Platform-as-a-Service (PaaS) for individual client robots via a reserved Virtual Machine(VM). The system uses Robot Operating System (ROS) as a middleware environment for robot systems development. Collaboration between robots is managed with multi-ROS master software, where several ROSs run in the same cloud environment, each being connected via a Virtual Private Network(VPN). The robots' real-time responses define an overall Quality of Service(QoS) that is measured in two environments. First, in a Clone-Based Model(CBM) and second, in a Proxy-Based Model (PBM). The results show that the average response time in PBM changes significantly when the number of robots in a system increases. However, the CBM shows that average response time remains steady regardless of the number of robots used. We conclude that the CCRP demonstrates an efficient and secure way of utilising external resources via the application programming interface (API) manager, to extend the capability of individual robots".

Higgins, J, Al-Jodi, T, Aagela, H., Holmes, V.: "Inspiring the Next Generation of HPC Engineers with Re-configurable, Multi-Tenant Resources for Teaching and Research", Journal of education, SAGE Journals.

Abstract: *"There is a tradition at our university for teaching and research in High Performance Computing (HPC) systems engineering. With exascale computing on the horizon, and a shortage of HPC talent, there is a need for new research computing specialists to secure the future of research computing. Whilst many institutions provide research computing training for users within*

their particular domain, few offer HPC engineering and infrastructure related courses, making it difficult for students to acquire these skills. This paper outlines how and why we are training students in HPC systems engineering skills, including technologies used in delivering this goal. We demonstrate a potential for a multi-tenant system for education and research which can be supported by other institutions, using novel container and cloud based architecture. An evaluation of our activities over the last 2 years is given in terms of recruitment metrics, skills audit feedback from students, and research outputs enabled by the multi-tenant usage of the resource.".

Aagela, H., Holmes, V.: "Collaborative Cloud-based Face recognition approach for Humanoid robots.", EMiT19, ISBN: 978-0-9933426-2-6, University of Huddersfield, Huddersfield, April 2019, UK.

Abstract: *"The ability to recognise human faces in real time is an important requirement for most humanoid robots. One of the challenges in face recognition (FR) applications is the time it takes a robot to search through a large dataset of known faces. As a database of known images is increasing, a robot's ability to store and process the data in real time is decreasing. In this paper we present a new cloud-based FR algorithm which will enable faster processing of data in face detection and recognition by humanoid robots. An improvement in robot's performance will be achieved by offloading storage and processing tasks from limited on-board robot resources to the resources in the cloud. In the case of multi-robot systems, their performance can be further improved through cloud-based collaborative learning and information sharing. We created a new*

dataset containing over 300 trained facial images of 10 people. The result shows that the proposed FR can achieve 83% accuracy rate and exceeds the local FR performance in terms of the response time which is slightly increased in comparison to local system performance which sees significant increase when the dataset size grows. The system proved its capability to share knowledge between robots in the same multi-robot environment."

Aagela, H., Holmes, V.: "Cloud Robotics-Based System for Robot Teleoperation.", EMiT19, ISBN: 978-0-9933426-2-6, University of Huddersfield, Huddersfield, April 2019, UK.

Abstract: *"Robots are now able to assist humans in many demanding tasks. Teleoperation is one way to combine robot skills and human operator abilities, through remote automatic or manual control. However, modern applications require larger processing and memory resources than those currently available in most robotic systems. The main challenges associated with networked robots occur due to resource constraints, information and learning constraints, and communication constraints. In this paper we present our approach in dealing with teleoperation processes in multi-robot environments that attempt to tackle the Heterogeneity robot challenge. We have implemented clone-based cloud robotic platform (CCRP) which is designed to provide platform-as-a-service (PaaS) for the client robots. In this system, a virtual machine (VM) is assigned in a cloud for every robot. The platform uses Robot Operating System (ROS) as a middleware environment for robot development. The result show that the response time in teleoperation was on average 240ms for Turtlebot robot and*

273ms for NAO robot".

Al-Aqrabi, H., Hill, R., Aagela, H., Holmes, V.: "Securing Manufacturing Intelligence for the Industrial Internet of Things.", Springer, ICICT2019, Brunel, London, UK.

Abstract: *"Widespread interest in the emerging area of predictive analytic is driving industries such as manufacturing to explore new approaches to the collection and management of data provided from Industrial Internet of Things (IIoT) devices. Often, analytic processing for Business Intelligence (BI) is an intensive task, and it also presents both an opportunity for competitive advantage as well as a security vulnerability in terms of the potential for losing Intellectual Property (IP). This article explores two approaches to securing BI in the manufacturing domain. Simulation results indicate that a Unified Threat Management (UTM) model is simpler to maintain and has less potential vulnerabilities than a distributed security model. Conversely, a distributed model of security out-performs the UTM model and offers more scope for the use of existing hardware resources. In conclusion, a hybrid security model is proposed where security controls are segregated into a multi-cloud architecture".*

Aagela, H., Al-Jodi, T., Holmes, V.: "Web-based Wireless Wake-on-LAN approach for Robots.", IEEE, ICAC'18, Newcastle University, Newcastle upon Tyne, September 2018, UK.

Abstract: *"The ability to remotely wake-up robots over a wireless LAN can*

improve the performance and the power consumption of remote robots or cloud robotics system. This paper presents a solution for power management of a mobile robot, using web-based wireless Wake-on-LAN (WWoL). The focus in on power management of a mobile robot, but this approach is also suitable for most of the IoT mobile devices, and other systems that are designed to be used remotely. The proposed solution allows the targeted device to be powered ON and OFF remotely via a web-based dashboard. This approach is validated in a case study with an AR. Drone 2.0. and demonstrated substantial power optimisation for the drone. In addition, security issues are explored when WWoL is deployed in remote control of mobile devices. It was established that a power consumption is reduced when the drone is in a standby mode waiting for an operator to send a wake-on request message wirelessly, without compromising security posed by wireless remote access to the devices".

Aagela, H., Holmes, V.: "An Asusxtionpro based indoor MAPPING using a Raspberry Pi with Turtlebot Robot.", IEEE, ICAC'17, University of Huddersfield, Huddersfield, September 2017, UK.

Abstract: "The developers of path planning algorithms and localisation have significantly improved the usability of the robot those days. By using software such as a Gmapping, the robot will be able to create a map of the surrounding area. This research utilises the 3D sensor Asusxtionpro to create an indoor map using SLAM and create 3D models for surrounding objects with a Turtlebot robot. In the first case, we used the Turtlebot to generate an indoor map of the robotic lab room using the Gmapping ROS packet. In the second case, we

used the robot to create 3D models for the surrounding objects in the room. We used the Raspberry Pi 3 as a replacement of the laptop that was used to control the Turtlebot. The same implementation of the first and second tasks have been repeated to compare the performance. The Raspberry Pi accomplishes the given tasks successfully; however, there is some delay due to the different on the CPU power. Finally, the low cost proposed solution is capable of running ROS based SLAM algorithm and using the point on cloud to create 3D models. In addition, the use of Raspberry Pi allows the robot save considerable amount of power in contrast with the use of a normal laptop".

Aagela, H., Holmes, V.: "Impact of Video Streaming Quality on Bandwidth and Face Recognition Accuracy in Humanoid Robot NAO.", ICC '17, March 2017, Cambridge, UK.

Abstract: *"This paper investigates the impact of video streaming quality on bandwidth consumption during the transfer of video data from a humanoid robot 'NAO' to computing devices, used to perform face recognition tasks, and to the cloud. It presents the results of profiling the network performance of connecting NAO with an edge controller, and discusses the effect of using different qualities of video streaming on the consumed up-link bandwidth. This study considers the limitation of the up-link bandwidth in the Wi-Fi network. It compares the performances of Wi-Fi and Ethernet connections between the NAO robot and a computer. In addition, it examines the accuracy of the face recognition tasks using various streaming scenarios, such as coloured video and black & white*

video. It investigates real-time video streaming using a wide range of frame rates, and video qualities, and their impact on the bandwidth, and accuracy of face identification. The results of our investigations are used to determine the acceptable video quality, frame rate, buffering and bandwidth that would give optimal results in face recognition using NAO robot, and enable efficient data transfer to the cloud".

Appendix B

OpenStack deployment

This section explains the process of Openstack deployment methods that have been utilised during this study. The Openstack private cloud was deployed twice, the section of the deployment method was based on compatibility with the available hardware and the university network. The first attempt was in 2016, where the we used the metal as a service '*MaaS*' and Juju. MaaS is allowing user to get full control and access to the provided hardware. Where Juju is an automatic service orchestration tool allow easy deployment for various applications, such as Hadoop cluster, WordPress, Openstack and so on. (OpenStack Docs: Install Juju, 2019). The second Openstack deployment method that has been applied is called Devstack, which is basically a sequence of extensible scripts utilised to rapidly set up a full OpenStack system. Here are some of the essential steps that needed for configuring both OpenStack environments. (OpenStack Docs: DevStack, 2019)

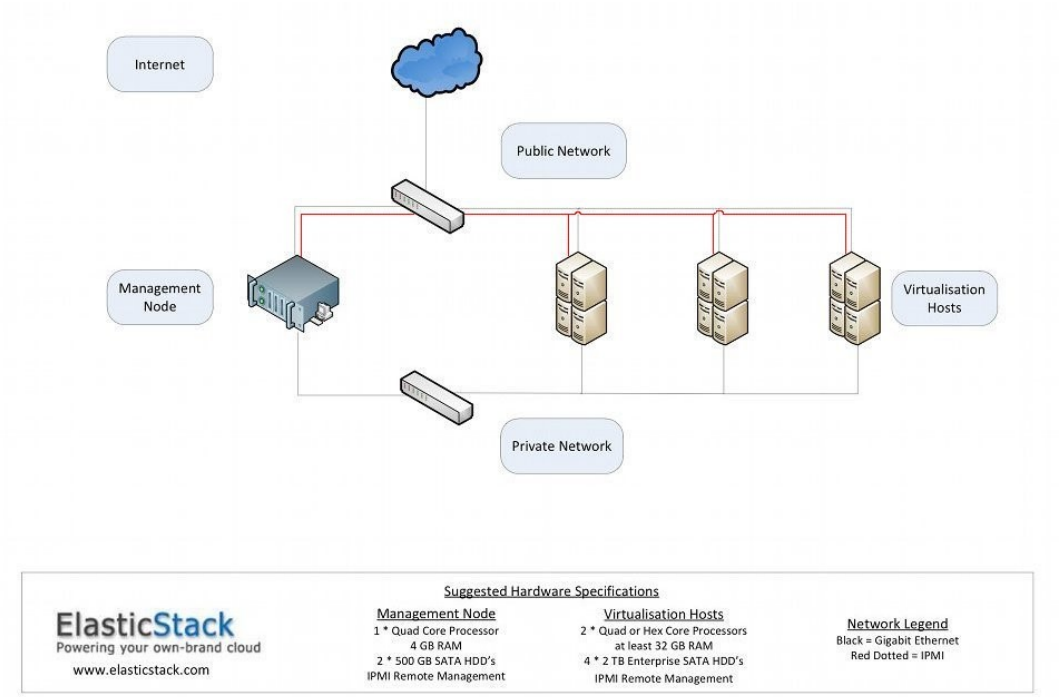


Figure B.1: The network diagram of the hardware and network components.

B.1 MaaS and Juju OpenStack Configuration

The MaaS and Juju system was installed using 7 machines, which was the minimum Openstack requirement at the time of the development. As shown in Figure the network diagram of the hardware and network components, which consist of one management node running the MaaS software and 6 other nodes linked together with two switches, first one connected to the University network and the Internet and the other switch is forming a cloud local private network.

First step for the installation is adding the required repositories for both MaaS and Juju. The we need to imitate the MaaS installation by using the following command line.

```
$ sudo add-apt-repository ppa:juju/stable

$ sudo add-apt-repository ppa:maas/stable

$ sudo add-apt-repository ppa:cloud-installer/stable

$ sudo apt update
```

```
$ sudo apt-get install maas
```

Once the MaaS software is installed the dashboard for MAAS can be accessed via the server IP address:

http://<MaaS_server_IP>/MAAS/

”

The dashboard allow user select and download the OS images that can be used for remote installation to linked nodes. The administrator can edit the network or create new one if required via the dashboard. Once the nodes are added to the MaaS as shown in Figure D.2, user can monitor and control the state of those nodes.

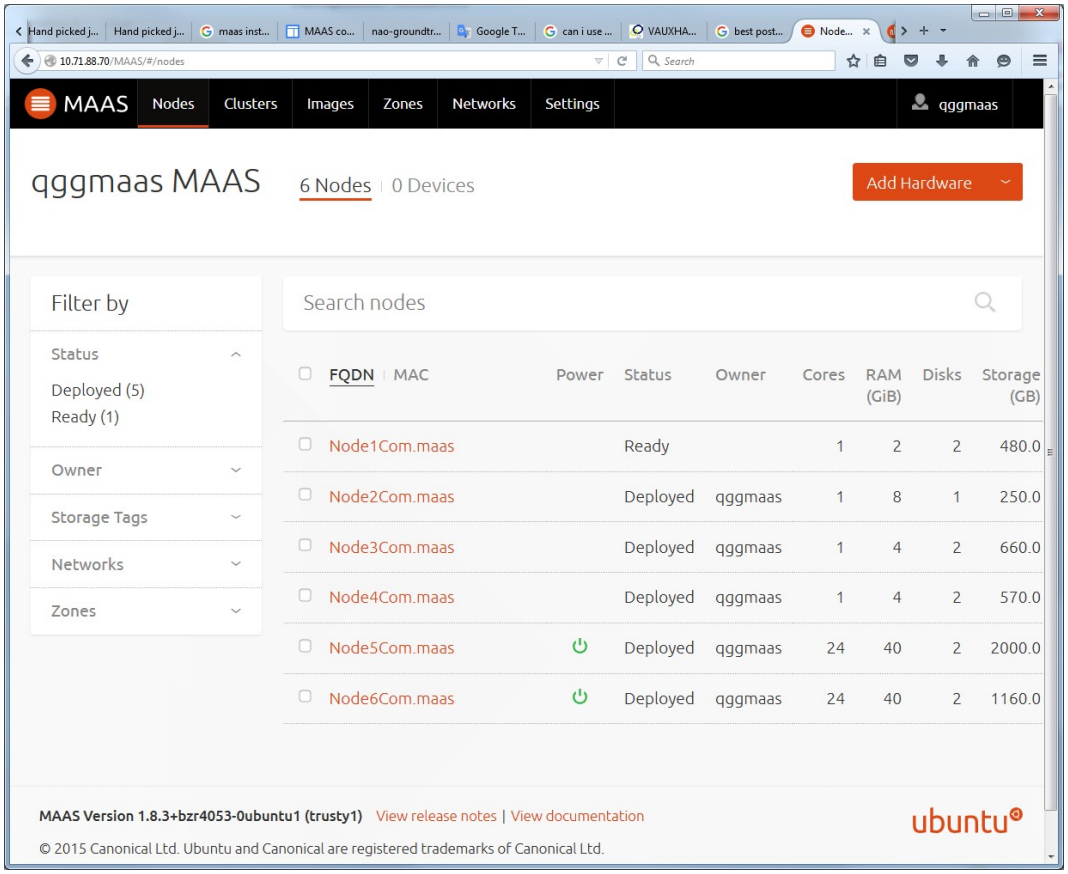


Figure B.2: MaaS dashboard

Install JuJu

Installing Juju require the user to create a configuration script before the installation and name it environments.yaml, which contains several parameters as follows:

defult: maas

environments:

maas:

type: maas

maas-server: 'http://10.71.x.x/MAAS'

maas-oauth: 'b2WFdyvrucffKV7sfR:KsVnrrC9P4v78qMyhy:MCYxHZ7W5JBd9wpURbW36fjt

authorized-keys-path: ∼/.ssh/id_rsa.pub

admin-secret: 'cloud'

default-series: trusty

bootstrap-timeout: 3600

apt-http-proxy: http://10.71.x.x:8000

lxc-clone: true

no-proxy: localhost,127.0.0.1,10.71.x.x

http-proxy: http://uniproxy:3128

https-proxy: http://uniproxy:3128

data-dir: /tmp/juju

To install Juju with the following command.

```
$ sudo apt install juju
```

At this stage juju should be active, however, the juju GUI will need extra step, which can be applied by using juju deployment tool. The juju GUI dashboard shown in Figure D.3.

```
$ juju deploy --to=0 juju-gui
```

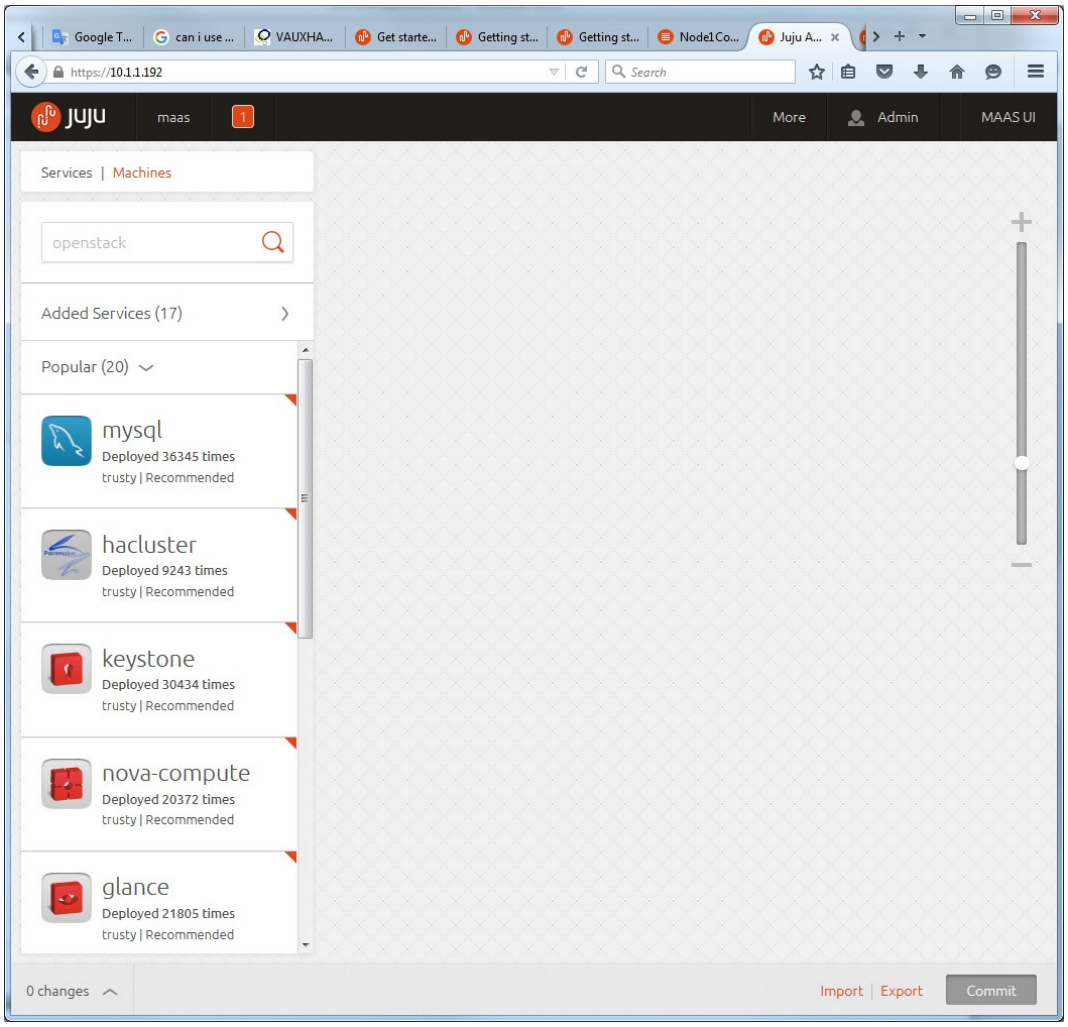


Figure B.3: The Juju GUI dashboard

Install Openstack

The quick start plugin for Juju is allow Juju to deploy all the required nodes and establish the connection amongst them quickly and easily. As shown in

Figure D.4 the Openstack deployment in Juju dashboard.

```
$ juju-quickstart openstack-base [-e <JUJU_ENV>]
```

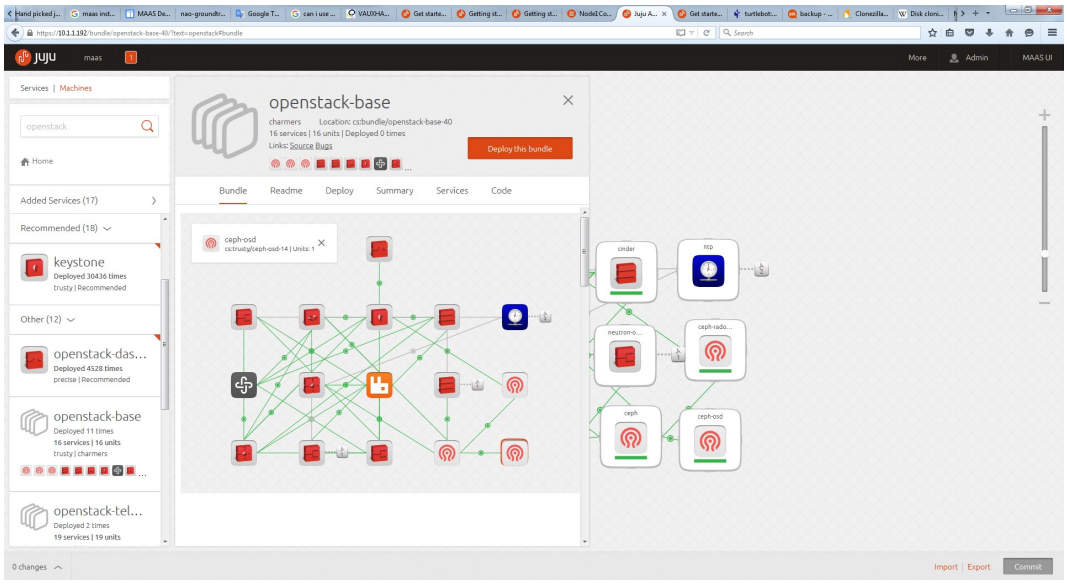


Figure B.4: The Openstack deployment within Juju environment

B.2 DevStack OpenStack Configuration

The second deployment method of Openstack is Devstack, which have used three server nodes as shown in Figure 0-3. The nodes linked together via a switch and the controller node connected to the University network, which used as a public cloud network. The process of creating Openstack using the Devstack method is easier and faster than MaaS and Juju companion, however, the later provide more scalability, redundancy and stability. The

implementation initiated by installing the OS in all the nodes, then following steps should be applied

Step 1: create a new user which must be named ”stack”, this user will be used to run DevStack.

```
$ sudo useradd -s /bin/bash -d /opt/stack -m stack

$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack

$ sudo su - stack
```

Step 2: Download the Devstack file from GitHub, then a configuration file must be edited with the personal information, network configuration and services that should be runs into each node.

```
$ git clone https://git.openstack.org/openstack-dev/devstack

$ cd devstack
```

Create the local scripts configuration for both the controller and the compute nodes. Step 3: Run the Devstack script

```
$ ./stack.sh
```

As shown in Figure D.5, the result of successful installation of the Devstack, the terminal should show the URL of the Openstack Dashboard.

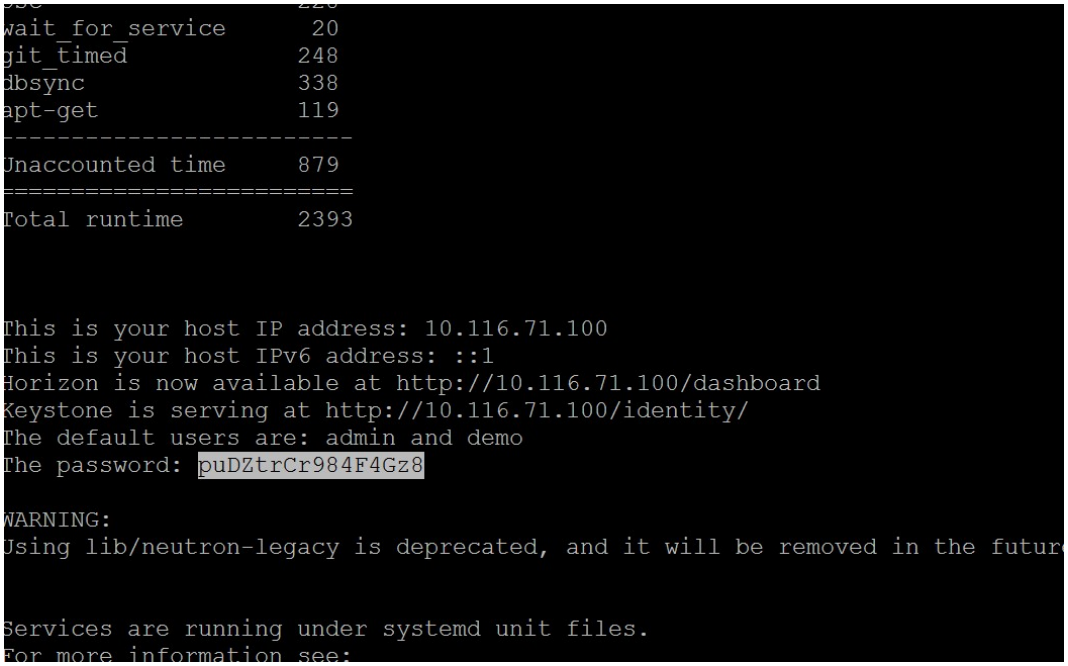


Figure B.5: Successful installation of Devstack output with the URL link of the dashboard

As a result of the both deployment method above, the user should have access to a web-based Horizon dashboard as shown in Figure D.6. Horizon is the official GUI application of OpenStack's Dashboard, which offers a web-based user interface for various OpenStack services such as Nova, Keystone, Swift and so on.

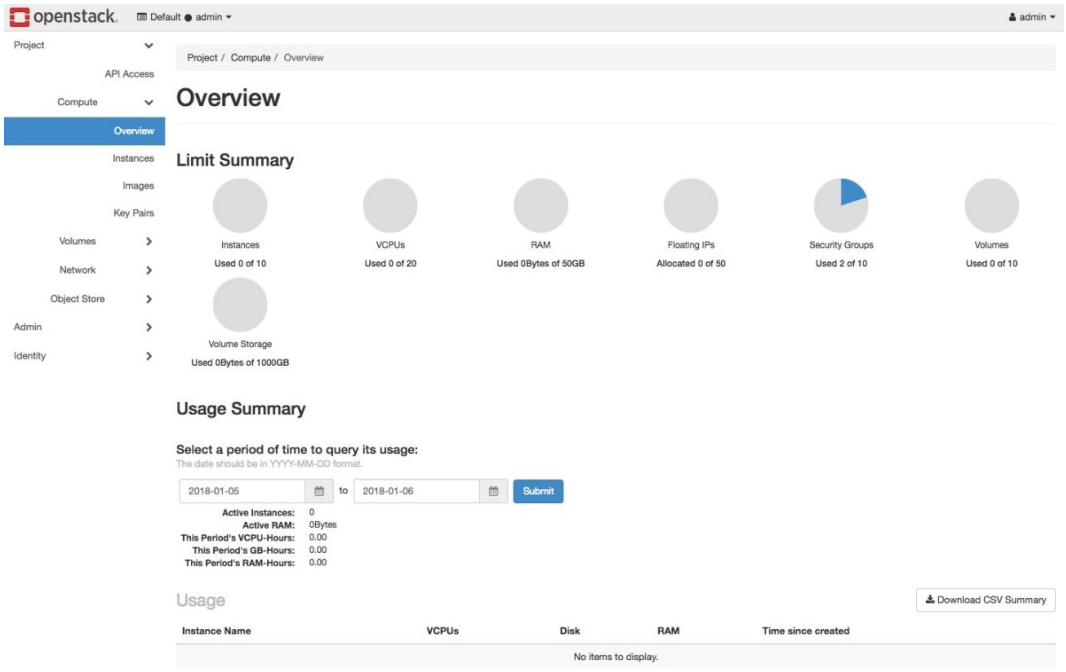


Figure B.6: Openstack web-based dashboard

Appendix C

ROS Installation

The ROS framework is a main element in the body of the CCRP, where the initial CRI will require the user to install the ROS in a cloud VM, the Installation process for ROS into a Ubuntu operating system was accomplished by following the procedure given by the ROS.org official page.which gives a guideline on how to execute the installation in the local machine robot/edge device. The work in this thesis mainly utilises two versions of ROS (Indigo and Kinect), and the OS that were used Ubuntu 14.04 LTS, 16.04 LTS and Ubuntu mate 16.04.

The installation is imitated by configuring the source list to allow downloading the software packages from ROS repositories as well as configuring the security keys.

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

```
$ wget https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -O - | sudo apt-key add -
```

```
$ sudo apt-get update
```

Next step is to install ROS main package.where there are three different installation package bundle. 1) the first one is the a full installation option. which includes all ROS basic packages as well as the ROS GUI software such as Rviz, Gazebo, rqt and robot-generic libraries. 2) The second option is the basic desktop installation which has some of the ROS GUI software included Rqt and Rviz. 3) The third option is the ROS-basic installation, which is limited to main ROS package,with no ROS GUI software.

```
$ sudo apt-get install ros-indigo-desktop-full

or

$ sudo apt-get install ros-indigo-desktop

or

$ sudo apt-get install ros-indigo-basic
```

Once the main ROS package is downloaded, then the rosdep is required to start the process installation and checking up the system dependencies, in addition, running some core elements in ROS.

```
$ sudo rosdep init

$ rosdep update
```

The new ROS environment should be added as a variables to bash session. This was done by initiating the following commands.

```
$ echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc

$ source ~/.bashrc
```

Once the installation is done, the ROS should be ready to run in the Ubuntu OS environment.

Appendix D

Collected Data

This section demonstrates some of the collected data throughout the research, the presented data are mostly the average of a larger set of data, which were demonstrated in a graph or table in previous sections. Followed by the reliability analysis table that show the Cronbach's Alpha value greater than 0.7, therefore, the dataset can be considered as a reliable source of data.

Response time of Clone-based model "CBM" and proxy-based model "PBM" delay

Response time of the Clone-based model "CBM" and proxy-based model "PBM" with 100ms delay, collected averaged results of both CBM and PBM,

Number of Robots	**Response Time CBM (ms)**	**Response Time PBM (ms)**
1	152	147
2	153	162
3	153	178
4	154	180
5	153	204
6	151	223
7	154	241
8	155	264
9	153	284
10	154	301

Reliability Statistics		
Cronbach's Alpha	Cronbach's Alpha Based on Standardised Items	N of Items
.726	.872	3

1. Response time of the Clone-based model "CBM" and proxy-based model "PBM" with 500ms delay, collected averaged results of both CBM and

PBM.

Number of Robots	**Response Time CBM (ms)**	**Response Time PBM (ms)**
1	551	542
2	550	620
3	553	718
4	550	821
5	555	951
6	552	1188
7	558	1281
8	555	1712
9	553	1994
10	554	2211

Reliability Statistics		
Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	N of Items
.723	.731	2

D.1 Experimental results of the delay time in millisecond for the response time delay and the video delay.

	Video		**Response**	
Attempt	NAO	Turtlebot	NAO	Turtlebot
1	142	130	264	241
2	143	131	265	242
3	144	132	266	243
4	145	133	267	244
5	146	134	268	245
6	147	135	269	246
7	148	136	270	247
8	149	137	271	248
9	150	138	272	249
10	151	139	273	250
11	142	140	274	251
12	143	141	275	252
13	144	142	276	253
14	145	143	277	254
15	146	144	278	255
16	147	145	279	256
17	148	146	280	257
18	149	147	281	258
19	150	148	282	259
20	151	149	283	260

Reliability Statistics for Video delay		
Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	N of Items
.857	.856	3

Reliability Statistics for response time delay		
Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	N of Items
1.000	1.000	3

D.2 The response time of the local FR and the cloud FR

Each of the results presented in the tables is an average of 20 attempts in FR for a given scenario.

1. The Result of the local FR approach shows the accuracy rate for 10 people and with various trained images.

No TF/P	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**
1	0.22	0.08	0.3	0.18	0.3	0.11	0.23	0.31	0.2	0.17
5	0.28	0.21	0.34	0.36	0.35	0.33	0.34	0.48	0.38	0.23
10	0.59	0.67	0.73	0.72	0.79	0.69	0.63	0.75	0.64	0.69
20	0.72	0.69	0.81	0.79	0.82	0.75	0.78	0.76	0.78	0.7
30	0.75	0.74	0.85	0.79	0.91	0.81	0.88	0.85	0.78	0.84

Reliability Statistics		
Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	N of Items
0.346	0.997	11

1. The Result of the cloud FR approach shows the accuracy rate for 10 people and with various trained images.

No TF/P	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**
1	0.19	0.11	0.22	0.14	0.23	0.18	0.22	0.2	0.16	0.15
5	0.31	0.3	0.34	0.28	0.36	0.32	0.33	0.32	0.3	0.34
10	0.64	0.63	0.66	0.58	0.68	0.6	0.61	0.65	0.64	0.61
20	0.8	0.76	0.81	0.79	0.77	0.75	0.74	0.77	0.72	0.79
30	0.84	0.84	0.8	0.81	0.86	0.83	0.85	0.83	0.84	0.8

Reliability Statistics		
Cronbach's Alpha	Cronbach's Alpha Based on Standardised Items	N of Items
.364	.998	11

1. The response time of the local FR and the cloud FR

Number Trained Faces for 10 People	**local FR**	**Cloud FR**
1	196	261
5	232	280
10	268	287
20	327	304
30	400	320

Reliability Statistics		
Cronbach's Alpha	Cronbach's Alpha Based on Standardised Items	N of Items
0.775	0.998	3

D.3 Face recognition accuracy rate with different video quality

1. Face recognition accuracy rate with different video quality Colour video streaming.

Video Quality	**1**	**5**	**10**	**15**	**20**	**25**	**30**
kQQVGA	56	64	68	68	68	68	72
kQVGA	72	76	80	88	88	84	84
kVGA	88	96	92	88	84	80	84
k4VGA	92	92	88	88	84	84	80

Reliability Statistics		
Cronbach's Alpha	Cronbach's Alpha Based on Standardised Items	N of Items
0.954	0.975	7

1. Face recognition accuracy rate for different video quality Black and white video streaming

Video Quality	**1**	**5**	**10**	**15**	**20**	**25**	**30**
kQQVGA	48	56	60	60	64	64	64
kQVGA	68	72	72	76	72	72	76
kVGA	84	88	84	84	84	80	80
k4VGA	92	88	84	84	80	76	76

Reliability Statistics		
Cronbach's Alpha	Cronbach's Alpha Based on Standardised Items	N of Items
0.970	0.993	7

Appendix E

Web-based Wireless Wake-on-LAN

This section presents an approach that allow to control the robots over a network remotely can enhance the efficiency of the energy usage performance for robots. This work try to address the lack of power for a mobile robot, by deploying the web-based wireless Wake-on-LAN WWoL. which aimed to permit user to control the power state of a remote robot via a web-based dashboard. The system was tested by using a case study with an AR drone 2.0 and shows substantial power optimisation for the drone.

The main features of the designed system can be listed as follows:

1. Control the Power state (on/off) of the remote robot.
2. Autonomously establish a secure channel and connect to the web-based application.

3. Easy to set-up and configure.

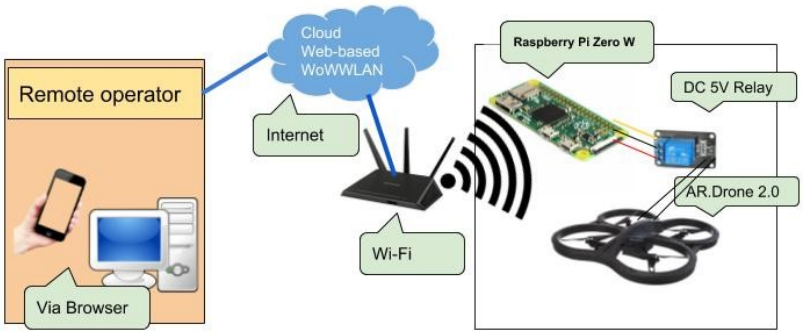


Figure E.1: The proposed System architecture

The WWoL system architecture shown in Figure E.1 the key components of the system. The user will get access to the targeted device via a browser and use the Network to send the signal to the smart switch. which connects the device to its power supply for the robot, in this case is a drone.

The Wi-Fi was used as the connection media; nevertheless, we can use most of the wireless internet enable technologies as well. The Raspberry Pi will be used as a secondary Linux-based OS for the original robot system. The system used a raspberry Pi Zero-W micro-controller as shown in Figure E.2 to work as a receiver for the control signal. There is a 5 V DC relay that and 3.7 V battery Lipo 1500 mAh. The key reason for choosing Raspberry Pi Zero W over other available micro-controllers is its compatibility and support for most of the internet protocols and low price. Also, the model Raspberry Pi Zero W is wireless-enabled with an inbuilt 802.11 b/g/n wireless LAN chip

Figure E.2: Raspberry Pi Zero W

and Bluetooth 4.1 LE.

The Parrot AR drone 2.0 is remotely controlled quadcopter drone that include several of sub-elements. It has on-board two boards mounted on the drone body. First board is a motherboard that uses a 32bits ARM9-core called Parrot P6 processor (468 MHz). It has a Wi-Fi chip. The second board, a navigation board, has a 16bits PIC microcontroller running at only 40 MHz, and used to interact with the drone sensors namely, two cameras, and two Prowave ultrasonic vertically-oriented sensors. The drone has a 11.1 V Lipo battery supplying 1500 mAh.

Our approach design a web-based application on both client and server sides. The client side was implemented using Python and NodeJS. where the Python is used to control the power status of the DC relay (on or off). The NodeJS part is responsible for connecting the Raspberry Pi with the web server, which

feeds the status of the Switch every 3 seconds (Here the time is optional to be defined by the user). The server side running a web page as a dashboard that demonstrate the switch current status and control Raspberry. As shown in Figure E.3 the system start by acquiring and checking the status of the Pi; then if it is online, the switch status will be checked that will be reflected in the state of a switch button on the web site. Therefore, the change the drone power status accordingly.

The Drone power system is managed by a web application that is hosted on a cloud virtual machine as an online server. Figure E.4 shows the main dashboard of the WWoL web application on different power statuses.

The prototype of the web-based WWoL system has been implemented and tested. The result of implementing the system is that it enables remote user to manage the power status of the drone, by using the web application as a smart switch. Figure E.5 shows the system with the done on both statuses (when the button is turned A) OFF and B) ON). It was established that power consumption can be optimised when the drone is in a standby mode waiting for a wake-on request message wirelessly from the operator, without compromising security posed by wireless remote access to the devices. The use of the system proved its value in terms of increasing the Idle-time for the Drone in standby mode. The following publications have arisen from my research detailed in this thesis: *"Web-based Wireless Wake-on-LAN approach for Robots." by* Aagela, Al-Jodi, Holmes (2018).

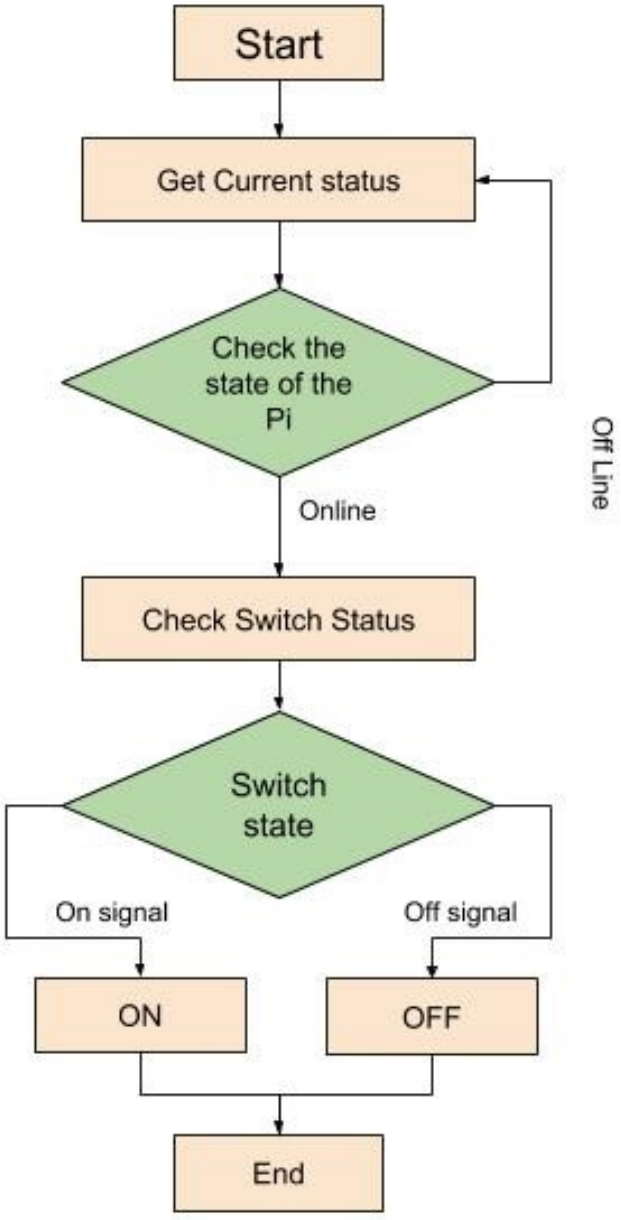


Figure E.3: The Work-flow of the Web-based application WWoL

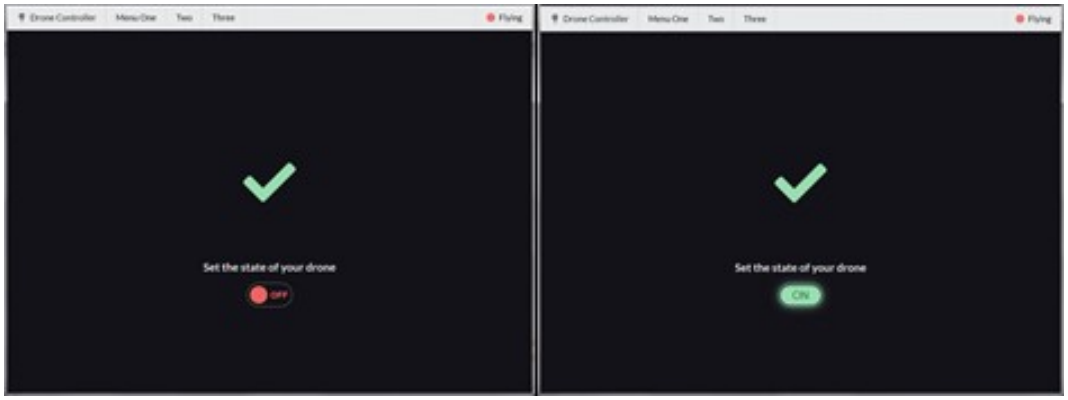


Figure E.4: Dashboard of the Web Application shows states when the button is turned OFF and turned ON.

A

B

Figure E.5: The drone state A) turned OFF B) turned ON