MASTER THESIS

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Mechatronics/Robotics

QUANTUM COMPUTING FOR DESIGNING BEHAVIORAL MODEL AND QUANTUM MACHINE LEARNING ON A HUMANOID ROBOT

By: Jacob Viertel, B. Eng.

Student Number: 1810331024

Supervisors: FH-Prof. Ing. Mag. Dr. Gerd Krizek
Dr.techn. Mohamed Aburaia MSc
Dr. Marek Perkoski

Vienna, October 12, 2020

Declaration

Vienna, October 12, 2020 Signature

Abstract

The robotics sector has been steadily developing ever since the first application of a robot in an industrial context in 1958. According to the Boston Consulting Group, as of 2016 over 29.6 million service robots were in use worldwide. This number is expected to grow over nine times to 264.3 million service robots by the year 2026. When it comes to sectors like entertainment and manufacturing, humans will collaborate with robots more frequently, the sensory stimuli used as inputs for their algorithms multiply, thereby paving the way for further expansion of their field of application. Furthermore, robots will likely interact more closely with human beings, which will lead to increasingly stricter regulations. As a result, the increasing amount of data and the more complex tasks will push the computational capacity of contemporary robots to its limits. Today's robots are controlled with classical computers that make decisions based on algorithms designed for classical computing architecture. These algorithms then make decisions that result in actions which are executed using effectors. In order to compensate for the increasing amount of data that need to be processed in real time, classical computers need more raw computational power. Therefore, it is necessary to downsize the computers' components as per Moore's law and thereby fitting more CPUs in a given space or simply mounting more same sized CPUs onto a service robot. However, the capacity to further downsize without facing other complications is limited and thereby not a viable long-term solution. Similarly, mounting more CPUs will lead to increased cooling system requirements and result in locomotion instabilities due to more weight mounted on the robot's frame. In addition to the aforementioned complications, the subgroup of robots that are subject to the research of this Master Thesis, namely humanoid robots, not only need more computational capacity but also require an algorithmic design based on a behavioral model which avoids causing antipathy when interacting with human beings. In order to solve both the computational capacity as well as the behavioral model design problems of humanoid robots, this Master Thesis researches the application of *quantum robotics* through using simulated and real quantum computing systems. According to Pitchbook, in the U.S. alone the amount of investment into quantum computing start-ups has increased from about $4 million to about $300 million in the last five years. In comparison to classical computing, fundamentally different concepts of physics are the basis of quantum computing and thereby quantum computing has the potential to solve specific computational problems more efficiently. This Master Thesis uses real quantum computer systems to design human-like behavioral models for humanoid robots and researches different quantum computing methodologies to classify datasets and solve problems such as the Shortest Path problem.

Keywords: Quantum Robotics, Human-Robot-Interaction, Quantum-Machine-Learning

Acknowledgements

Quantum cognition is an emerging research area that combines different aspects from the quantum theory with the human brain. This mixture of research areas seemed to be mainly esoteric to me at the beginning of this Master Thesis. With time, I figured more and more connections do exist between those two areas.
As an example, the isolation of a quantum system must completely shield the qubits from interactions with the environment. Without a perfect isolation a qubit gets influenced by external factors like vibration or gravity. In this case I would like to thank my professor FH-Prof. Ing. Mag. Dr. Gerd Krizek for trusting me with the definition of this Master Thesis. As a qubit needs protection from external influences to do its calculation, my creativity needed the freedom to unfold within this new type of research field.
I encountered another parallel between quantum computing and human cognition when reading about ancilla qubits. These qubits are mainly used to help achieving certain goals within a calculation. In this case I want to thank Dr. Perkowski and the entire team of the PSU intelligent robotics lab. In cases I got stuck during my calculations Dr. Perkowski and his team were always there to back up my knowledge regarding quantum computing.
But the biggest correlation between the two research areas of quantum cognition I encountered is called entanglement. Entanglement is one of the strongest connections between two particles, as even when those particles are separated physically, they are still connected. As I wrote this Master Thesis 9000km away from my home university, my family and friends still supported me during my entire stay.
Thank you so much.

Contents

1 Introduction

The Introduction gives a short summary what this Master Thesis is about, how it is structured and what research question it tries to answer. This Master Thesis examines the potential and need of combining quantum computing and robotics.
Classical computers can perfectly optimize smaller systems, but when it comes to larger systems such as transportation routes or product pricing, they only find incremental improvements (Malham, 2016). With increasing problem size, the amount of data often grows exponentially. This faster than polynomial growth of data leads to an accumulation of computational costs that in most cases can be hard to handle by a classical computer.
In robotics, the amount of data also increases continuously - a fact that pushes classical computers that control these robots to their limits. One famous humanoid robot named ASIMO (Sakagami et al., 2002*a*) which reportedly has the most advanced walking capabilities, needs to be able to process data from video cameras, gyroscopes, acceleration sensors and manage its 57 degrees of freedom (DOF) in next to real time. The data complexity and variety tend to deepen when collaborating with humans, because the environment requires higher safety regulation than with industrial robots. The collaboration robot Baxter (Silva et al., 2016) is a role-model when it comes to Human-Robot Interaction (HRI). According to Silva et al. (2016), Baxter sets the highest standards of security in case of collaborating robots. Due to the fact that the robot Baxter consists only of an upper body, it must not process additional data needed for locomotion. Considering a humanoid robot like ASIMO, it would be computational cost expensive to ensure the same high security standards and perform a stable locomotion simultaneously.
In addition, not only the safety standards have to increase when robots collaborate with human beings, but also the quality of the interaction between them. Humanoid robots need to express their *feelings* and *emotions* in order to increase the level of communication. A higher level of communication leads to a better understanding and a higher probability of acceptance regarding a robot working with human beings. The research area of quantum cognition is built on the theory that human behavior and quantum are closely related and focuses to enhance the HRI quality, (Bruza et al., 2015). Areas like quantum machine learning (DeBenedictis, 2018; Rebentrost et al., 2014; Perkowski & Al-Bayaty, 2014) or mood and emotion generation with quantum computing (Bevacqua & Mancini, 2004; Lukac & Perkowski, 2007*a*; Deng, 2020) have been researched more frequently in recent publications. The results of these publications are promising enough to consider quantum computing as a solution to handle the increasing amount of data and the complexity of HRI. A quantum computer can solve certain problems with less computational costs than a classical computer and is built on an entirely different physics than classical

computing. These two facts are used to tackle the problem of the computational capacity and the communication with humans.
This Master Thesis researches the potential of a hybrid computer-controlled robot. A hybrid computer combines classical computer- and quantum computer aspects. The incorporated data a humanoid robot has to process is generally extremely sensitive, therefore, the robot's data needs highest protection against potential eavesdropping and manipulation at any time. The Rivest–Shamir–Adleman algorithm (Rivest et al., 1978) is one of the most popular algorithms to encrypt data with regard to data protection in classical computing. The concept of the Rivest–Shamir–Adleman algorithm comprises the problem classical computers have with factorizing higher numbers. For classical computing it takes a long time to break this encryption, as even in the best classical algorithm the factors are guessed. Instead, a more effective approach is the quantum algorithm of Shor (Shor, 1997) with which it is possible to factorize even larger numbers easily. Shor's algorithm renders encryption techniques such as RSA no longer secure, as it can effectively factorize even bigger numbers and break any classical encryption. A quantum computer compromises the security level of communication in classical computing with eavesdropping and manipulation. On the other side, such a quantum-based communication includes a high protection against eavesdropping and manipulation from classical computing systems (Qi et al., 2010). Just to illustrate one of the many benefits quantum computing offers; if a hybrid computer controls the robot, the security of the data communication on the robot increases. As the industry and research rapidly develops and improves the area of quantum computing, its real potential might not have been discovered yet.
One might now raise the question why a quantum computer should not entirely control a robot, without the classical computing part. A major drawback of quantum computers is their liability to decoherence, which leads to a potential higher rate of errors (Kiefer & Joos, 1999). Also, even if quantum computing reaches a sufficient state of the art to handle real life problems reliably, a hybrid computer will always be superior. Nowadays, quantum computers have not yet been fully developed, lack isolation and are comparatively new to the market. Therefore, their calculations have higher error rates than classical computers. Due to the liability to decoherence, quantum computing algorithms need a certain amount of redundancy to errors in terms of communication and calculations, which means for one logical quantum bit hundreds of error-prone physical quantum bits are required (Roffe, 2019). The additional error-prone physical quantum bits hold additional information either to detect a faulty quantum bit or in order to restore the message if an error occurred. In comparison to the hundreds of error-prone physical quantum bits, the quantum computers from IBM used in this Master Thesis consisted of maximum 16 quantum bits. A technology that has enough quantum bits to make calculations more robust has not yet been developed. Not only the sensibility disqualifies quantum computers as single control systems for robots, but also the hardware for the computer's setup. The physical environment must be around 0 Kelvin to avoid thermal motion of the atoms in the computers structure, furthermore, the isolation of the quantum system must protect the system completely from environmental influences in order to regularize the potential error of a quantum computers‘

calculation under a certain threshold (Naihin, 2018). Considering the isolation and the cooling device, such a quantum computer is too heavy to be mounted onto a humanoid robot, which is why a hybrid version considers a mounted classical computer and a cloud quantum computer which is consulted if needed.
Now, another question could be raised with regard to the development of an existing classical computer: To avoid a hybrid version, could the existing system of a classical computer be developed further into a faster version that can handle increasing amounts of data, similar to a quantum computer? According to Moore´s law, the number of transistors in a dense integrated circuit doubles approximately every two years. The transistors might get smaller to fit this law, but other components like material, devices and circuits are already at their limits regarding downsizing (DeBenedictis, 2018). Nowadays, electronic components like transistors have already been scaled down to the size of but a few tens of nanometers. Riwar et al. (2013) discovered while researching the limits of downsizing, that quantum effects like superposition and entanglement will occur when an electron current is sent through structures with the size of an atom. The concept of downsizing the classical computer further to meet the needs of weight and computational capacity will most likely not be a possibility for future generations. When meeting the downsizing limits, quantum effects will occur at classical computing which influence the results and increase the error rate. These quantum effects must be handled in order to get reliable results from the computers‘ calculation. This Thesis suggests a robot that includes both systems, a classical system and a quantum system collaborating together. Both systems have their strengths and their combination offers higher potential than the standalone versions.
Another solution might be two classical computers that control the robot. One classical computer mounted on the robot and one classical computer cloud connected with higher computational capacity. This solution is also only temporary, regarding the increasing number of robots worldwide. As the number of robots will increase in the future and also the complexity of problems, several robots must share one cloud connected computer. To meet the needs of the increasing problem sizes the hardware of the more powerful cloud connected computer must increase with the problem size. This extension of the hardware lead to more CPUs and a higher loss of energy regarding the cooling and the heat buildup. As the extension of the cloud connected computer has its limitations, this computer cannot process all incoming request in real time. In addition to that, the properties of quantum computing in terms of quantum machine learning, data security and behavioral modelling are not available with this solution.
With respect to robotics, a quantum computer might offer improvements in comparison to classical computing in areas like data security (Shor, 1997), behavioral modelling (Lukac & Perkowski, 2007*a*), and quantum machine learning (Rebentrost et al., 2014). The following example explains a quantum behavior considering the greeting gesture of a biped robot:

> The shoulder joints are effectors M1 for the left shoulder and M2 for right shoulder. With a classical circuit controlling this robot, the output would be repetitive, which means the same input state would generate the same output state. Whereas in a quantum circuit,

> the same input states lead to different output states. In quantum computing, an algorithm refers to the same as a circuit. Due to this probabilistic behavior, the gesture of the robot will differ on every evaluation of the circuit. Such a behavior conveys a more human-like nature in the movement itself when compared to the deterministic behavior of classical computing. For instance, if the input state of a two-quantum-bit system is $|01\rangle$, the robot with classical control would always raise its right arm. If a quantum computer controls the robot, the output gesture of the robot will vary. Sometimes the robot will raise the right hand, other times it will raise both hands, and even the left hand for the greeting gesture. With quantum circuits the programmers can also imitate totally deterministic behaviors as with the classical circuits.

However, the definition of quantum computing would not be complete without an adequate definition of quantum robotics. Two different definitions exist about a quantum robot. One definition comes from Benioff (Benioff, 1997) and it describes the quantum robot as a robot strictly operating in a quantum world. A quantum world is defined by quantum sensors, quantum computing, and quantum effectors. In this case, the motions of these quantum effectors happens on quantum level. The other definition origins from the *Intelligent Robotics Laboratory* at Portland State University (PSU) and considers a normal sized robot controlled by a hybrid computer.
Nowadays, the realization of Benioff's definition is impossible, as the technology has not yet been developed, whereas the PSU definition is realizable due to IBMs accessible quantum computers or realistic quantum computing simulations using libraries like Qiskit (Asfaw et al., 2020). There are two possibilities of a setup of such a hybrid computer, one considers only the classical computer mounted on the robot and the quantum computer located externally somewhere stationary, the other considers both computers mounted on the robot. The first setup is the currently possible setup and the emerging technology of 5G (Paudel & Bhattarai, 2018) is capable to transmit even greater amounts of data between those two computers. The second setup with both computers on the robot is nowadays unlikely because of the high weight of a quantum computer caused by the huge cooling system that is needed to keep the temperature around 0 Kelvin. Nevertheless, it might also be possible to put both computers on humanoid robots in the future when cooling systems like in Ouellette (2017) will become commercially viable. This centimeter-sized silicon chip technology uses a specific method to cool down the quantum bits when passing the chip.
After having considered the definition of a quantum robot, the focus will now lie on a comparison of classical computing and quantum computing in general. As quantum computers are an uprising technology with high potential, Arute et al. (2019) expect it to outperform classical computing in solving certain problems, which is then called quantum supremacy. It is still controversial if such a quantum supremacy has been reached yet. Google claims to have already reached this supremacy with a certain problem (Arute et al., 2019), while IBM raised doubts and asserts that super-computers could solve the problem faster (Cho, 2019). Nevertheless, as the limitations of classical computers are approaching and quantum computing's full potential has not been unfolded, it is just a matter of time. Research in quantum computing will con-

tinuously develop, and with the right technology such as the Noisy Intermediate-Scale Quantum (NISQ)-technology (Preskill, 2018) quantum computers with limited computational power will be commercially available in the near future. Such NISQ-computers are special types of quantum computers with less isolation and ,therefore, a higher error rate. The longer the quantum algorithms are, the higher is the potential error, but making the technology available for the public will most likely accelerate its development, which was exactly what happened when the first classical computers became commercially viable. So far, the biggest quantum computer with the most quantum bits inherent has a 53-quantum bit processor and is currently being researched in papers like Fei (2020). This quantum computer is expected to outperform a classical computer in certain tasks and ,therefore, reaching quantum-supremacy which would be a milestone for quantum computing (Edwin et al., 2019).
Not only quantum computers are currently on the uprising, but also the field of robotics is being rapidly developed. Robotics, and in particular intelligent robotics (Segura Velandia & Moreno, 2019), is one of the most developing sectors nowadays. According to the Boston Consulting Group, robots like industrial- or household-robots will perform up to 25% of overall labor tasks by 2025 (BCG, 2015). To tie in with this statement, Berg Insight Group prognosed a growth of the service robots' instalments by 2026 up to 264.3 million in total. Compared to this number, in the year 2016 only 29.6 million service robots were installed worldwide. Humanoid robots, which serve as the main focus of this Master Thesis, are one of the smallest shares in the robot market today, but it is the sector with the highest potential in the future (BCG, 2015).
Humanoid robots are not only considered to be used for various physical labor tasks in hazardous environments, but it is also likely for them to be installed as robotic toys, household servants, receptionists, museum guides, actors, nurses or helping autistic children like in Wood et al. (2019). From the first robot used in an assembly line in 1958 (Hockstein et al., 2007) until the highly developed robots like ASIMO nowadays (Sakagami et al., 2002*a*), the main capability shifted from relying on the mechanical parts towards the software used. As the software has the biggest leverage regarding the potential or capability of robots, using quantum computers after reaching the quantum supremacy will increase their potential.
This Master Thesis will attempt to bridge the gap between the areas of humanoid robots and quantum computing. Both areas will benefit from this research project, as both are new and have great inherent potential. The combination offers lucrative application potential for various industries; therefore, research is more likely to be subsidized which leads to enhanced development. Both areas combined can set new standards that cannot be met with any classical computing. The area of quantum computing is still at its infancy and quantum computers are still rare. The goal of this paper is not to outperform known classical computing systems, but simply to show the raw potential of quantum humanoid robots.

The following chapter 2 starts with the history of robotics and briefly summarizes the group of humanoid robots. This chapter 2 explains the development of robotics in general in order to make assumptions about its future. Because this Master Thesis uses the humanoid robot

HR-OS5, chapter 2 introduces this subgroup of robotics and current research in this area in order to clarify the possible problems caused by humanoid robots. Furthermore, various examples will highlight the reasons to include a faster processing unit. After the basics of humanoid robots, chapter 3 will explain the fundamentals of quantum computing. These fundamentals of quantum computing consist of basic quantum gates and different quantum phenomena like superposition, entanglement and quantum parallelism. The knowledge of quantum gates is fundamental in order to understand the quantum circuits of chapter 9. This Thesis uses the Python library Qiskit which is an open-source framework to design all quantum circuits of this Master Thesis.

The combination of both chapters 2 and 3 result in the research area of quantum robotics in chapter 4, which explains the importance of machine learning and the expression of mood and emotion for humanoid robots. This chapter 4 includes both definitions of quantum robotics with focus on the interpretation of PSU.

Beyond the basics of quantum computing, chapter 5 provides more in-depth knowledge about specific quantum algorithms and phenomena. The chapters 6, 7, 8, and 9 rely on these algorithms and phenomena and those chapters include the main contribution of this Thesis. These algorithms have high potential in applications on a humanoid robot. With NISQ-technology quantum computers will become commercially viable in the near future. The designer of a quantum circuit must consider the properties of this NISQ-technology as not every theoretical quantum algorithm is realizable on a such a NISQ-computer.

Chapter 6 extends and generalizes the results of a previous project at the PSU on the HR-OS5 robot regarding quantum motion generation. This chapter 6 explains the concept of quantum probabilistic behavior and how to generate motions on the HR-OS5, which is the first attempt to generate quantum motions on such a humanoid robot.

Chapters 7, 8, and 9 include the main contributions of this Master Thesis. These chapters introduce two quantum circuits and the design of a robotic head for smaller humanoid robots that aim to improve the quality of HRI. This Master Thesis defines a smaller humanoid robot as humanoid robot that has not the size of an average full-grown human being, but smaller. The main contribution includes a comparison of the Quantum-Support-Vector Machine (QSVM) algorithm with the Swap Test algorithm. The depth and processing time of the two algorithms are the two factors of this comparison. Moreover, the Swap Test algorithm classifies two different datasets, the OCR dataset (Escrivá, 2012) and the Breast-Cancer dataset (Dr. William H. Wolberg, 1995), using two different approaches on how to classify datasets. The OCR dataset has lower complexity in comparison to the Breast-Cancer dataset, therefore, the data samples from the Breast-Cancer dataset need more variables during the classification process. The last contribution of this Master Thesis is an approach to use an adaptation of the Grover algorithm in order to solve several variations of the Shortest Path problem. The Grover algorithm adaptation can analyze a graph if a path between starting node and target node exists. The Grover oracle can also find a path with lower cost than a given threshold and the Shortest Path in general.

The Master Thesis tries to empathize the potential of the new area of quantum robotics. Again,

it is not the intention to replace the classical computer entirely, as quantum computing is, according to the current state of technology, not a possible standalone solution. Only in some specific cases, quantum computing is superior to classical computing, which suggests a hybrid solution of a classical- and quantum-controlled robot.
The hypotheses of this Master Thesis are that only quantum computing can imitate real human-like behavior and also that quantum computing is superior in comparison to classical computing in solving algorithmic problems related to robotics like finding the shortest path.
The research question of this Master Thesis is:

> How to implement quantum computing on a robot to create a behavioral model and solve algorithmic problems?

A cloud connection should establish the communication between a quantum computer and classical computer in order to set up the first hybrid controlled biped robot. With a reliable connection between a quantum computer and classical computer, quantum circuits try to create a behavioral model. Similar to the Braitenberg Vehicle concept that expresses behaviors like shyness and aggression using a trivial methodology, these quantum circuits should be able to create behavioral models with low computational costs. Additionally, the second part of this Master Thesis researches the performance of quantum algorithms and compares it to classical algorithms.

2 Biped Humanoid Robots and Animatronic Heads

This chapter shortly summarizes the historical development of robotics, because knowing the history and development of robots give a better understanding of and insight into what the future for robotics might look like. Afterwards, the subgroup of robots, the humanoid robots, are introduced, as the robot of this Master Thesis, the HR-OS5, belongs to this subgroup. Specifically, the Thesis will discuss the head of such a humanoid robot and its purpose, which tends to be more complex than in other subgroups of robots.

2.1 History of Robotics

In the Greek-Hellenistic era philosophers defined the primary function of a robot which focused primarily on supporting the human executing tasks or doing it all by itself. Such devices were then known as automata, which refers to a human-like robot (Gasparetto & Scalera, 2019).
Humanoid robots, as they are known and defined today, origin from science-fiction literature and have become one of the most promising areas of industry as mentioned in the introduction. Around 1920, several Czech authors started to write short stories including humanoid robots for the first time like they are understood and seen today. The first author who mentioned humanoid robots was Karel Capek in his drama Rossum´s Universal Robots (R.U.R.) (Christoforou & Mueller, 2017). The word robot derives from the word *robota*, which is Czech and means serf or laborer. Since the first reference of a robot in a short story, the area of robotics continuously developed to help with mechanical labor in industry projects. It was General Motors in 1958 that first used robots in their assembly line. Most of the time, the robots' purpose was to replicate or improve the humans' function (Hockstein et al., 2007).
The launch of industrial robotics occurred around the 1950´s. This was the first-time robots helped with trivial tasks in factories and were no longer only pure science-fiction. Starting with the 1950's, Wallén (2008) divides the industrial robots' development into four generations. The first generation from the 1950´s until the late 1960´s were simple robots. These robots worked in a closed environment and were not able to adapt to task changes, as they lacked input signals. Mechanical logical gates controlled these robots and most actuators were pneumatic (Gasparetto & Scalera, 2019). During the first generation, a robot was able to perform simply one task, like spot-welding or handling workpieces (Wallén, 2008). Only two years after Japan had installed its first robot Versatran from AMF Corporation (Cyberneticzoo, 2013) in 1967, companies in Japan started to invest significantly in robotics, which was the first time Japan got

into robotics. Starting seventeen years after the launch of the first generation, Japan managed to become one of the leading countries in robotics to the present state of scientific knowledge.
The second generation of industrial robots started in 1968 and lasted until the late 1970's. During this generation, robots were able to adjust to task changes, but only in a very limited manner. The two main kinds of robots that were developed during this second generation were mobile robots (Simon, 2017) and robot arms (Moran, 2007). With the usage of servo-controllers, robots could perform point-to-point motions and continuous paths. The robots of the second generation were either controlled with microprocessors or programmable logic controllers. Around the middle of this second generation, the pneumatic actuators had been replaced with electric actuators, to improve the accuracy, lower maintenance cost while increasing speed (Gasparetto & Scalera, 2019).
The third-generation phase lasted until 1999 and focused on extending the environmental interface of the robot. More sensors were added, and more precise feedback was given to the operator. The diagnostic capabilities were improved dramatically during this generation. Until then, the robot could only indicate whether a failure occurred or not. To speed up the error search the human operator could read out the error memory of the robot. The newly introduced parallel kinematic (Zivanovic, 2000) was superior to classical serial kinematic (Yeshmukhametov et al., 2017) in terms of robustness, size of workspace, and speed (Xie et al., 2009).
Within the fourth generation, which is considered to last until today, the focus of robotics development was on machine learning to make the robots more independent of a human as operator (Gasparetto & Scalera, 2019).
With the background of the historical evolution of robotics, the fifth generation might be robots controlled with quantum computers. Today, reaching the limitations of classical computing, further development with greater application lies within the field of quantum computing. Researchers from Japan believe that the amount of data that need to be processed in form of differential or logical equations can be handled in next to real time with quantum computing only. To get a glimpse of the ambition in the robotic sector, Japan recently claimed to have a robot soccer team by 2050 that can beat the soccer world champion of the human team (Murphy, 2015). This goal might appear very ambitious, however, it must be noted here that in the past, it was a basic perception that a computer can never beat a human in strategy games like chess. However, in 1997, IBM´s Deep Blue was able to beat the world champion in chess, Kasparow (Campbell et al., 2002).

2.2 Humanoid Robot

Nowadays, there is a variety of different robots available. As robots differ in shape, size and capabilities, it is sometimes hard to distinguish the affiliation of a certain robot to one specific group. In Rodney & Gill (2015), robots are divided into 15 categories, among others, major ones are the following:

- Industrial
- Exoskeleton
- Drone
- Humanoid

The general definition of humanoid robots was given by Graefe & Bischoff (2003) and defines the humanoid robot as a programmable machine which can imitate the actions and appearance of human beings.
This Master Thesis will consider mainly the group of humanoid robots such as DARwIn-OP (Muecke & Hong, 2007). This humanoid robot is popular among researchers in the area of humanoid robots and is very similar to Quanty. Due to the popularity of DARwIn-OP, many researchers developed a great variety of open-source software specifically for this robot, which can be used as a fundament for any project regarding this robot. The name DARwIn-OP is an acronym for **D**ynamic **A**nthropomorphic **R**obot **w**ith **I**ntelligence-**O**pen **P**latform. It was developed in 2010 by the Korean company Robotis. With its robust structure and reliable motors, and its open platform hardware and software, this robot is best suited for educational purposes (Muecke & Hong, 2007). The price of such a humanoid robot can vary from $12.000 for a robot like DARwIn-OP to $20.000 for a robot-like HR-OS5. Compared to the well-known humanoid robot Nao (Al-Hami & Lakaemper, 2015), DARwIn-OP and HR-OS5 are perfectly suitable for educational purposes due to their open-source architecture.
Exactly like DARwIn-OP, the HR-OS5 robot belongs to the category of humanoid robots and is similar to the smaller version HR-OS1 referred in Sunardi (2020). This Master Thesis uses the knowledge gained on HR-OS1 to work on Quanty. HR-OS5 was created in a collaboration between INTEL and Trossen Robotics and is used mainly for educational purposes. To our knowledge, there is no paper published on the HR-OS5 so far. The high-school project of Deng (2020) in the only known report on HR-OS5. The researchers in Deng (2020) use quantum computing to create motions for HR-OS5 based on music notations.
Considering the category of humanoid robots, communication with human beings serves as a major aspect, however, at the same time, it is also one of the major problems. As a result, new research areas like HRI are being developed to increase the communication quality (Goodrich & Schultz, 2007). This synergy of communication channels like facial expressions mainly focuses on the increase of the level of communication between human and robot. In human-to-human

communication, different channels like facial expression, gestures, speech and other body language, are used to communicate. Within the different categories of robots, humanoid robots are the only ones that are capable of using the same communication channels as humans. The challenges when analyzing the nonverbal information exchange of a robot is the establishment of an appropriate communication channel to increase the quality human-robot-communication. In the area of humanoid robots, one is usually confronted with at least one of the four main problems, namely:

- Robot head, design and programming
- Robot hand, design and programming
- Robot locomotion, methodology
- Robot learning behavior

Robot head, design and programming
In this respect the main problem with the robot head is the effective communication via methods like expressing emotions that can be interpreted correctly by its discussion partner. In order to research more on how different robotic expressions influence humans, robotic head systems like Character Robot Face (Fukuda et al., 2004) have been developed. To increase the depth and complexity of interactive communication further, it is also important to synchronize the mouth movement. It is not only possible for a human to process acoustic signals that are produced and sent by the robot, but the visual input of robotic facial expression should also be taken into consideration.

Robot hand, design and programming
As humanoid robots are designed to look and act like human beings, the robotic hands are most important when it comes to interacting with objects, like grabbing and moving. The motion of a human hand appears to be an unconscious and obvious process for a human. Even the hand´s anatomical structure is implicit for a human being. However, applying these complex structures and processes to different tasks a robot has to execute is highly complex due to the structure and design of making robots. The main issues are the hands' weight, dimensions and the degrees of freedom.

Robot locomotion, methodology
Indeed, the management of the extremities of humanoid robots is not a trivial job, as there are not only the hands that have to execute motion sequences, but also the feet with regard to the ability to walk. A major theoretical approach to handle the walking problem is called the Zero-Moment-Point theory (Vukobratovic & Borovac, 2004). In this concept, invented by Vukobratović, the locomotion is planned with respect of the two points of contacts between the humanoid robot and the ground. Considering a two-legged humanoid robot, the feet represent the two engaging points. The zero-moment point represents all forces acting on the mechanism, in this case the legged locomotion, by a single force. With this assumption, it is possible to calculate the bipeds' gait (Erbatur et al., 2002).

Robot learning behavior
A major reason for the development of humanoid robots is the possibility for them to take over jobs that are too dangerous or too repetitive for human beings. Within this scenario, different problems can arise. For example, the circumstances within a problem can differ and an assigned task is unlikely to be repetitive for a humanoid robot, e.g. while doing a household job like cleaning, the amount of dirt or the location varies. Cleaning and cooking jobs might not appear to be dangerous for a human at the first sight, but according to the Robert-Koch institute in Germany annually 3.500 people die in traffic accidents, while 8.000 people die in household accidents (Institute, 2019). Additionally, processes like cooking and cleaning include repetitive movements, which can cause chronical joint pain (Helliwell & Taylor, 2004). Another example for such a household servant is a cooking robot like Moley (Park et al., 2017). Such a cooking robot must prepare different meals according to the humans' health-condition. In order to handle these problems appropriately, the humanoid robot must be capable of learning (Hester et al., 2010). This learning capability is given through different methods for example reinforcement learning (Hammoudeh, 2018) with decision trees (Rokach & Maimon, 2005).

Popular problems related to robotics that can be solved using learning methods are the satisfiability problem (Gu et al., 2002) and the shortest path problem (Elmasry & Shokry, 2018). Usually, all bigger problems in robotics can be decomposed to smaller problems like satisfiability or shortest path. This decomposition makes it easier to solve the bigger problem as well-known algorithms are used to handle the decomposed sub-problems. Satisfiability has practical applications in different areas of machine vision and robotics like the packing problem, calculating the trajectories and task planning problems (Gu et al., 2002). Developing quantum algorithms that handle such problems like shortest path, can speed up the calculation and process more data.

2.3 Robot Head

As humanoid robots often have a human-like head, more communicational channels can be used during an interaction with human beings.
Humanoid robots that interact with human beings must have a way to express their internal states to enhance verbal and non-verbal communication. The internal state of a robot represents its mood and emotional level. The robot's conversation partner must be able to easily interpret expressions of mood and emotion. Moreover, it is essential to refer to characteristics like being left- or right-handed. Designing a humanoid robot that can express *feelings* and have characteristic features tends to increase the communicational quality, as the human is more likely to accept the robot as an equated communication partner.
Just like an interaction between human beings, an exchange between robot and human being has a higher level of communicative features when all participants share their assumptions of the interactions *mood*. Thus, both sides get the opportunity to act accordingly. To imitate a

human behavior, the robot´s behavior must be consistent with the internal state which means that it should not vary too much with equal input signals. If the behavior occurs too random, it will easily cause aversion. On the other hand, if the motions are too deterministic, the robots´ behavior appears trivial which also leads to human aversion. As an example how to express the internal state of a robot, the battery charging status may be considered. If the human can interpret the battery status by the robot´s expressions and behaviors, a total discharging can be avoided. The robot´s expressions with lower charging level will appear slower than usual and the total range of its motion will get smaller. In this way, the user can easily conclude that the robot must be charged to the next possible moment.

Due to the fact that there will be more frequent interactions between robots and humans in the future, as humans will use robots in sectors like entertainment or help with labor, behavioral models must be designed. Several models inspired by biology and philosophy were developed in Breazeal & Scassellati (2003) and Kwon et al. (2007) with low computational costs. In these models, emotions and moods do not only affect the state of the body, but also on a higher, logical level (Lukac & Perkowski, 2007*b*).

The Uncanny-Valley theory (Mori et al., 2012) relates the human affinity to the human-like appearance and motions of a robot. A human-like robot and a cartoon-like robot will both usually be considered to be pleasant for a human, as it reflects the human being itself and its appearance. But if the robots' appearance is somewhere in-between the human-like-robot and the cartoon-like-robot its appearance will arouse human antipathy. According to the Uncanny-Valley theory, this minimum of empathy must be avoided, or a human tends to feel uncomfortable which results in a lower communication quality (Barciela et al., 2008).

Robots that look partly like humans can provoke uncomfortable feelings in the human-robot interaction. When the human participant realizes that something is unusual, antipathy might arouse. A robot with a human-like face and a robotic body like Albert Hubo (Oh et al., 2006) for example tends to cause antipathy, as this combination breaks usual structures. The brain feels fooled by the human-like head on the robot-like body and tends to be alerted. Humans need to structure things in order to feel comfortable. A design mixture of human-like parts and robotic parts tends to break the commonly-known structure such as only human parts or only robot parts, and is therefore often seen as unpleasant. Additionally, a humanoid robot that realistically looks like a human being might intimidate human beings in general, like the humanoid robot Sophia (Korn et al., 2018). For human beings, the main concern is to be outperformed and replaced in certain areas, especially by a machine that looks alike (Lee, 2018).

As part of this Thesis, a robot head was designed with respect to the Uncanny-Valley assumption. Designing a robot head on the right side of the Uncanny-Valley, the human-like side demands numerous effectors, as it must imitate the human face with its 21 muscles (Marur et al., 2014). Constructing such a head is not effectively possible for a robot like Quanty, as the heads' final weight would be too great due to the servomotors needed. Therefore, the constructing engineer should design a robot head on the left side of the Uncanny-Valley, which displays a less human-like shape. Even if this means that possibly less affinity is aroused, antipathy and

weight are avoided, as not many effectors are needed.
In Prakash & Rogers (2014), robots of different human likeness levels were shown to participants. Independent of the age of the participant, commonly a robot with human-like and robotic parts, were assumed to be more unpleasant. There was a tendency that older people preferred a very human-like robot, referring to the right side of the Uncanny-Valley, and younger people preferred robots with low human likeliness referring to the left side of the Uncanny-Valley (Prakash & Rogers, 2014).
To gain more specific information about the exact location on the human likeness axis of the Uncanny-Valley theory in Strait et al. (2017), tests were conducted. These tests enabled more insights into how to make rough assumptions on designing a robot that avoids the Uncanny-Valley. The reason a robot falls in this valley is not only dependent on the degree of human likeness of the appearance and motions. Two driving factors that determine if the robot falls into the Uncanny-Valley are atypicality and category ambiguity. An atypical robot, like Albert Hubo (Oh et al., 2006), violates the expectations of appearance. The body is robot-like and the head human-like. The atypicality postulate states that a robot is more likely to cause antipathy if shape expectations are violated. In the case of Albert Hubo, the human-like head does not fit to the robot-like body. With Geminoid HI (Kakenhi, 2007), one can study the effects of the second factor, category ambiguity. As the robot looks very similar to a human, it is not trivial to perceive it as a robot. Pictures and simulations of over 60 different robots have been shown to over 72 participants. The simulations of human-like robots were more often prematurely terminated by the participants than the simulations of less human-like robots. As the reason for the premature termination, people claimed to experience higher antipathy watching human-like robots as it shows that human-like appearance and behavior are no longer unique (Strait et al., 2017).
Table [2.1] shows five different types of robots that have different effects on the affinity of a human being.

Table 2.1: This table shows the data points of Fig. [2.1]. The category of Cartoon human characters is the highest point on the left side of the Uncanny-Valley. According to the information from Mori et al. (2012), the design role model of the robotic head for smaller humanoid robot is a cartoon human character like the puppets from the Sesame Street TV-show.

t1	Industrial robot arm
t2	Mechanical humanoid robot
t3	Cartoon human character
t4	Humanoid robot with skin
t5	Human

Fig. [2.1] shows the affinity curve with the Uncanny-Valley of table [2.1].

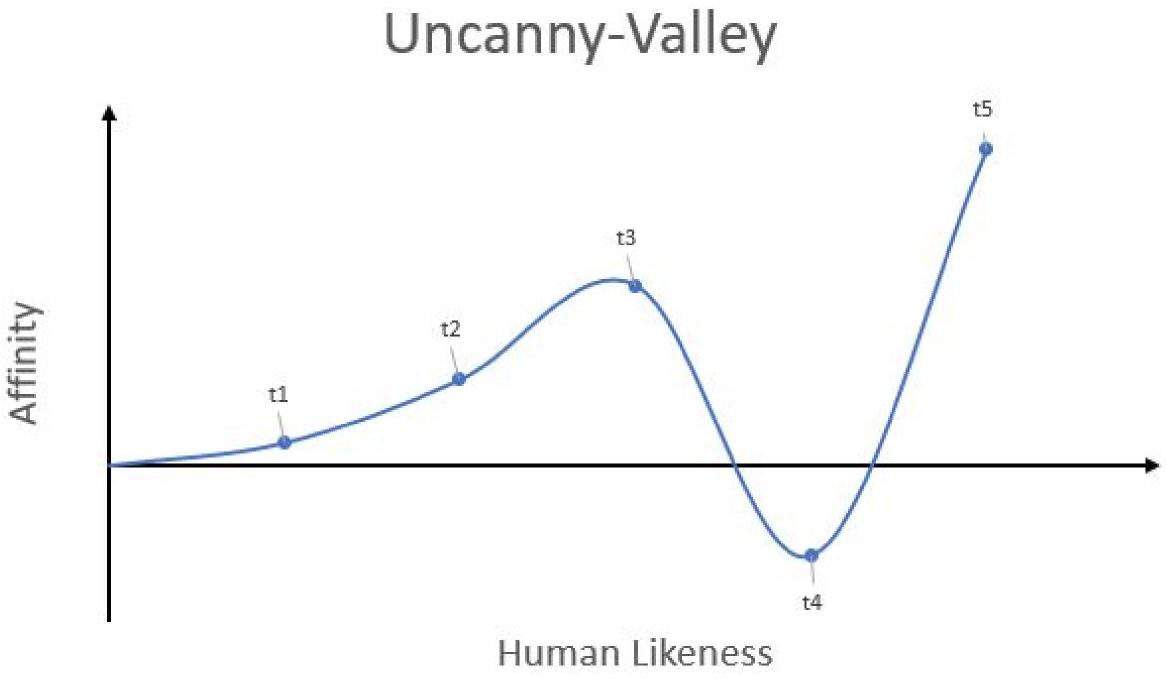


Figure 2.1: The graph illustrates the Uncanny-Valley theory with real-life examples. The marked points relate to a specific type of robots mentioned in Tab. [2.1]. The affinity increases when the robot becomes more human-like until a certain level. At this level the affinity decreases as the robot's appearance arouses uncomfortable feelings in humans (Mori et al., 2012).

3 Fundaments of Quantum Circuits

In this chapter, the fundamentals of quantum computing with respect to quantum circuits will be explained. This chapter does not include all fundamentals of quantum computing but covers all required information to understand the basic concept of this Master Thesis.

3.1 Qiskit

Qiskit (Asfaw et al., 2020) is a library in Python 3.x (Haslwanter, 2016), especially suited for quantum simulations. While writing code with Qiskit, it is recommended to use a Jupyter notebook, which combines programming, text and images. The Jupyter notebook (Cardoso et al., 2019) compiler is suited to execute different registers of the circuit independently. In this Master Thesis every figure of spheres, quantum circuits and the measurement results were designed with Jupyter notebook. The open-source framework Qiskit was developed by IBM and can be used to perform simulations of quantum circuits or execute circuits on real quantum computers from IBM (García-Martín & Sierra, 2017). Qiskit provides all standard quantum gates and offers the possibility for users to customize their own gates. A quantum computer can be either simulated locally on a computer or the Python script is sent to quantum computer and is executed. It is also possible to measure the noise of real quantum computers and apply it to simulations in order to see how robust the quantum circuit is (Asfaw et al., 2020).

3.2 Matrix Types in Quantum Computing

Every operation or gate in quantum computing can be represented as a matrix transformation. It is fundamental to know how these transformations are applied and how they affect the Quantum bit (qubit) state. A qubit is a basic unit to store information in quantum computing and represents the most trivial two-state quantum system (Asfaw et al., 2020).

3.2.1 Unitary Matrices

Quantum gates like Pauli operators I, X, Y, and Z are commonly used in quantum computing and can be represented with unitary matrices. A matrix is unitary if the following eq. [3.1] and eq. [3.2] are true:

$$\mathrm{U}^{-1} = \mathrm{U}^{\dagger} \tag{3.1}$$

$$U * U^{\dagger} = U^{\dagger} * U = I \tag{3.2}$$

In eq. [3.1] and eq. [3.2] the dagger notation is used, which is common in quantum computing to express the complex conjugated form of a matrix (Arfken, 1985). Eq. [3.3] forms the complex conjugated of a matrix by first transposing the matrix, followed by negating the signs of the complex parts of the numbers.

$$\begin{aligned} A &= \begin{bmatrix} 1+2i & 2 & 2-i \\ 2i & 3 & 2+3i \end{bmatrix} \\ A^{T} &= \begin{bmatrix} 1+2i & 2i \\ 2 & 3 \\ 2-i & 2+3i \end{bmatrix} \\ A^{\dagger} &= \begin{bmatrix} 1+2i & -2i \\ 2 & 3 \\ 2+i & 2-3i \end{bmatrix} \end{aligned} \tag{3.3}$$

All Pauli operators can be represented with a permutation matrix. Such a permutation matrix contains in every row and column only a single 1, the other entries are 0 (Golub & Van Loan, 1996). To understand the effects of such unitary operators, two vectors in the same vector-space are considered. If those two arbitrary vectors in the same vector-space are being transformed with the same unitary matrix, let's say a Pauli-Z gate, the transformation does not change the length of the vectors, or angle between them (Allen, 2003*a*). Subsection 3.7.1 will explain the Pauli-Z gate, but for now it is not important to know what it does in particular, only that matrix from eq. [3.4] represents its transformation.

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.4}$$

To illustrate how such a unitary operator will transform two vectors, the two vectors $\vec{u}$ and $\vec{v}$ will be considered in a two-dimensional space. Fig. [3.1] displays how the matrix of the Pauli-Z gate rotates two vectors around the origin, without changing the angle between them or their size.

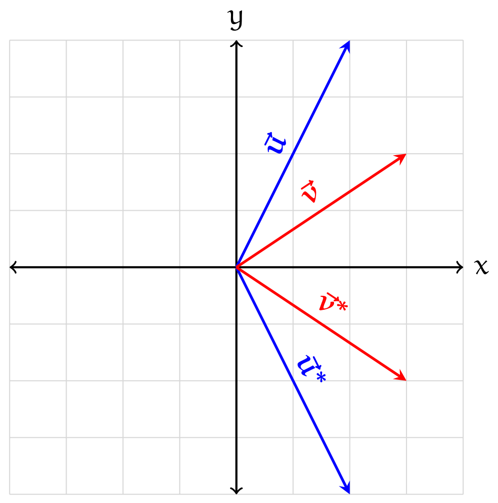


Figure 3.1: The transformation matrix of eq. [3.4] rotates both vectors $\vec{u}$ and $\vec{v}$. The transformation results in vector $\vec{u}$ becoming $\vec{u}^*$ and $\vec{v}$ becoming $\vec{v}^*$, which represents the x-axis mirrored vector of the original input vector. The angle between the input vectors and the output vectors are the same. All figures within this Master Thesis, that illustrate vectors in a two-dimensional vector-space were created within this Thesis using the Latex package *TikZ*.

Fig. [3.1] displays the transformation of eq. [3.4] which represents the Z-gate, $\vec{u}$ becomes $\vec{u}^*$ and $\vec{v}$ becomes $\vec{v}^*$. As the angle between the two vectors of Fig. [3.1] will not change after the transformation, the inner product of the two vectors will also be the same after the transformation. The inner product of two vectors in a Hilbert-space (Buric, 2010), gives information about the similarity of those two vectors. Subsection 3.4 covers the properties of the Hilbert-space. The inner product of the vectors $\vec{u}$ and $\vec{v}$ is the same as the inner product of $\vec{u}^*$ and $\vec{v}^*$, which is true for any transformation *U* in a Hilbert-space, eq. [3.5].

$$|u, v\rangle = |U(u), U(v)\rangle \;\forall\, u, v \;\in \mathbb{R}^n \tag{3.5}$$

Eq. [3.5] shows the Dirac-Notation, which is a special notation for states in quantum computing (Brown, 2006). Eq. [3.6] represents the ground state of a two-dimensional Hilbert-space.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{3.6}$$

The Dirac-Notation has two variations to describe a vector, bra- and ket-notation. Eq. [3.6] uses the ket-notation and eq. [3.7] uses the bra-notation which refers to the transposed version of a ket-notation.

$$\langle 0| = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{3.7}$$

3.2.2 Hermitian Matrices

Quantum computing commonly uses Hermitian matrices which belong to the group of unitary matrices (Arfken, 1985). Those Hermitian matrices have applications in algorithms like polylog algorithm for linear systems of equations (Harrow et al., 2009). Hermitian matrices are always defined in a complex vector-space $\mathbb{C}$, composed of complex numbers. They have a square form and their complex conjugated matrix is the same as their original matrix, eq. [3.8].

$$A = A^{\dagger} \tag{3.8}$$

Hermitian matrices are the complex counterpart to a symmetric matrix which consists of only real numbers. A matrix is Hermitian if the entries on the main diagonal are real numbers, and if the entry at a_{ij} is the complex conjugate of entry a_{ji}. Eq. [3.9] shows a generalized form of the 2*2 Hermitian matrix.

$$A = \begin{bmatrix} r_{11} & r_{12} - c_{12} \\ r_{21} + c_{21} & r_{22} \end{bmatrix} \tag{3.9}$$

In the matrix of eq. [3.9], *r* stands for any real number and *c* for any complex number. To be Hermitian, the diagonal must not have any complex number, entry r_{12} must be equal to r_{21}, and c_{12} equal to c_{21}. Hermitian matrices have three properties as follows (Allen, 2003*b*):

- Their eigenvalues are real
- Their eigenvectors are orthogonal
- They are diagonalizable

Section 3.3 will explain the concepts of eigenvalues and eigenvectors. If *A* is a Hermitian matrix, the special properties of eq. [3.10] and eq. [3.11] are valid (van Dommelen, 2018).

$$\langle g| A |f\rangle = \langle f| A^{\dagger} |g\rangle \tag{3.10}$$

$$\langle f| A |f\rangle \text{ is real} \tag{3.11}$$

Eq. [3.13] shows a practical example of eq. [3.10] and eq. [3.11] with:

$$A = \begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix} , |f\rangle = \begin{bmatrix} 1 \\ 2 \end{bmatrix} , |g\rangle = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

The following example shows that given two arbitrary vectors and an arbitrary matrix, the vectors are interchangeable if the matrix will be transposed.

$$\begin{aligned}
\langle g| A |f\rangle &= \begin{bmatrix} 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \\
\langle g| A |f\rangle &= \begin{bmatrix} 3 & 1 \end{bmatrix} \begin{bmatrix} 1+2i \\ -i+2 \end{bmatrix} \\
\langle g| A |f\rangle &= 3 + 6i - i + 2 \\
\langle g| A |f\rangle &= 5 + 5i \\
\\
\langle f| A^{\dagger} |g\rangle &= \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} \\
\langle f| A^{\dagger} |g\rangle &= \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 3-i \\ 3i+1 \end{bmatrix} \\
\langle f| A^{\dagger} |g\rangle &= 3 - i + 6i + 2 \\
\langle f| A^{\dagger} |g\rangle &= 5 + 5i
\end{aligned} \tag{3.12}$$

If the same vector in bra and ket notation factorizes a Hermitian matrix as in eq. [3.13], the result will always be a real number.

$$\begin{aligned}
\langle f| A |f\rangle &= \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \\
\langle f| A |f\rangle &= \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 1+2i \\ 2-i \end{bmatrix} \\
\langle f| A |f\rangle &= 1 + 2i + 4 - 2i \\
\langle f| A |f\rangle &= 5
\end{aligned} \tag{3.13}$$

3.3 Eigenvalue and Eigenvectors

If a number λ satisfies the condition in eq. [3.14], the number is an eigenvalue and the corresponding vector $\vec{v}$ is an eigenvector of the matrix *A*.

$$A |v\rangle = \lambda |v\rangle \tag{3.14}$$

As the matrix *A* does not affect the orientation of the vector $|v\rangle$, a single scaling value λ can represent the transformation of the matrix. The vector $|v\rangle$, which orientation remains the same after the transformation is called eigenvector.

Fig. [3.2] shows how the corresponding value λ, the eigenvalue, changes the length of the vector $|v\rangle$ (Zwanikken, 2018).

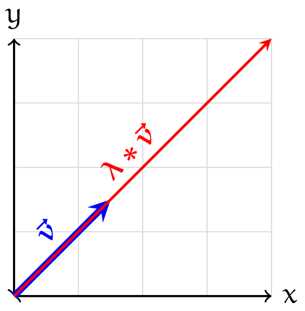


Figure 3.2: The blue colored vector $\vec{v}$ represents the initial vector. After the transformation with matrix *A*, vector $\vec{v}$ keeps its original orientation but scales according to the value of λ.

If only one eigenvector satisfies the eq. [3.14] with a specific eigenvalue, the eigenvalue is non-degenerate. A set of vectors $\vec{v}$ forms an eigenbasis, if all vectors in this set $\vec{v}_i$ are eigenvectors of *A* and the set contains *n* vectors describing the *n*-dimensional space of *A* (Glorioso, 2013). Eigenvectors describe the transformation of an operator with independent vectors. Considering a two-dimensional vector-space, the eigenvectors and eigenvalues of a matrix give precise information about the stretching, squeezing and mirroring on the x-axis and y-axis of a vector. The matrix of eigenvectors describing an orthonormal basis of the Hilbert-space, represents every possible output value of a measurement (Shao, 2019). An orthonormal basis creates a vector-space which basis vectors are unit vectors and oriented orthogonal to each other.
Eq. [3.15] shows the notation of the exponential form of a matrix.

$$U = e^{i\gamma H} \tag{3.15}$$

In order to rewrite the transformation matrix in exponential form, the eigenvalues of the operator must be known. Quantum computing often uses the exponential form to represent the transformation of an operator *U*. In eq. [3.15], *H* represents a Hermitian matrix and γ some real number. Eq. [3.16] is the exponential form of eq. [3.14].

$$e^{H} |v\rangle = e^{\lambda} |v\rangle \tag{3.16}$$

Eq. [3.17] shows the exponential form of *A* if it is a n-square matrix.

$$A^0 = I,\ A^1 = A,\ A^2 = A_0 * A_1, \ldots,\ A^n = A_0 * A_1 * \ldots * A_n \tag{3.17}$$

Eq. [3.18] forms the infinite matrix power series.

$$I + \frac{t}{1!}A + \frac{t^2}{2!}A^2 + \ldots \tag{3.18}$$

Eq. [3.19] shows the infinite matrix power series from eq. [3.17] rewritten in exponential form.

$$e^{tA} = \sum_{n=0}^{\infty} \frac{t^n}{n!} A^n \tag{3.19}$$

3.4 Hilbert-Space in Quantum Computing

The transformations in quantum computing happen in Hilbert-space, which is a specific type of vector-space. A Hilbert-space has any properties that a normal vector-space has plus some additional properties. As an example of these additional properties, the Hilbert-space has the ability of the inner product operation. The inner product of vectors must satisfy certain conditions as follows:

- Conjugate symmetry: $\langle\psi_1|\psi_2\rangle = \langle\psi_2|\psi_1\rangle^T$
- Linear with regard to 2^{nd} vector: $\langle\psi_1|\alpha\psi_2 + \beta\psi_3\rangle = \alpha\langle\psi_1|\psi_2\rangle + \beta\langle\psi_1|\psi_3\rangle$
- Antilinear with regard to 1^{st} vector: $\langle\alpha\psi_1 + \beta\psi_2|\psi_3\rangle = \alpha^T\langle\psi_1|\psi_3\rangle + \beta^T\langle\psi_2|\psi_3\rangle$
- Positive definiteness: $\langle\psi|\psi\rangle = |\psi|^2 \geq 0$
- Distance formula: $|\psi_2 - \psi_1| = \sqrt{\langle\psi_2 - \psi_1|\psi_2 - \psi_1\rangle} = d$

Transformations on a qubit happen in a vector-space that belongs to the finite-dimensional Hilbert-spaces. $\mathbb{C}^n$ expresses a Hilbert-space with *n* dimensions that needs *n* basis vectors to be defined (Loaiza, 2017). A Bloch sphere (Bloch, 1946) represents the Hilbert-space used in quantum computing. Fig. [3.3] shows this Bloch sphere which always has a radius of 1.

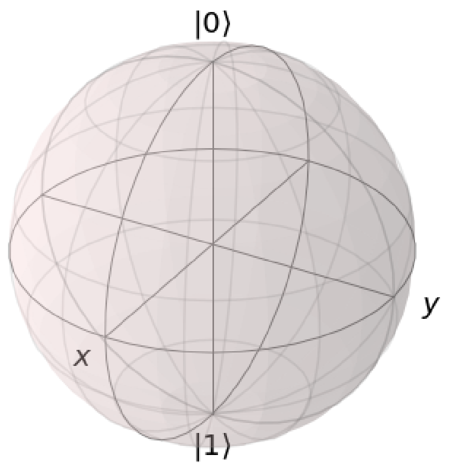


Figure 3.3: This figure represents an empty Bloch sphere in Hilbert-space. A vector originating from the center of this Bloch sphere, pointing anywhere onto the Bloch sphere represents the state of a qubit. All figures that illustrate Bloch spheres were created within this Master Thesis using the Python library *Qiskit*.

3.4.1 Global and Local Phases

In quantum computing, the amplitudes of the qubits state are related to the probability *P* and are an observable quantity of a state. Eq. [3.20] calculates the probability of measuring the state $|1\rangle$ on a qubit in superposition.

$$\begin{aligned}
&|\psi\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}} \\
&|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \\
&P(1) = |\frac{1}{\sqrt{2}}|^2 \\
&P(1) = \frac{1}{2}
\end{aligned} \tag{3.20}$$

The next example considers a state ψ with additional complex global phase θ and also calculates the probability of measuring the state $|1\rangle$ on a qubit in superposition. The complex phase can also be written in its polar coordinates $re^{i\theta}$.

$$\begin{aligned}
&|\psi\rangle = \frac{|0\rangle+i|1\rangle}{\sqrt{2}} \\
&|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}e^{i\theta}|1\rangle \\
&P(1) = |\frac{1}{\sqrt{2}}e^{i\theta}|^2 \\
&P(1) = |\frac{1}{\sqrt{2}}e^{i\theta}||\frac{1}{\sqrt{2}}e^{-i\theta}| \\
&P(1) = \frac{1}{2}
\end{aligned} \tag{3.21}$$

Despite the additional complex global phase in eq. [3.21], the probabilities and, thus, the only observable property in eq. [3.20] and eq. [3.21] of the state ψ, are equal. This means, a global phase is not observable in quantity and is indistinguishable.
In contrast to the global phase, the local phase is an observable quantity. The local phase describes the phase between two states and is further important in terms of density matrix or decoherence. Such a density matrix describes the static state of a system. In eq. [3.22] r_1 and r_2 are any real numbers and θ any arbitrary phase shift.

$$|\psi\rangle = r_1 e^{i\theta_1}|0\rangle + r_2 e^{i\theta_2}|1\rangle \tag{3.22}$$

$$|\psi\rangle = e^{i\theta_1}(r_1|0\rangle + r_2 e^{i(\theta_2-\theta_1)}|1\rangle) \tag{3.23}$$

Calculating the probability of one of the basis states in eq. [3.23], the global phase θ_1 vanishes, but the relative phase between the two states $\theta_2 - \theta_1$ affects the result. Thus, the local phase is an observable quantity.

3.5 Hamiltonian

Hamiltonian and Lagrangian transformations both describe a mechanical system in terms of its coordinates and velocities (Malham, 2016). Both methods describe dynamic systems. The Hamiltonian method solves mechanical problems and it is better suited than the Lagrangian method for handling systems with a larger number of particles like in statistical-mechanics, systems involving gas or systems with no particles at all, like in quantum mechanics. Newtons equations do not apply in quantum mechanics as they do not hold for the size of a quantum system. The momentum operator is entirely different on quantum scale. The Hamiltonian refers to the total energy of a system and considers the momentum when describing the mechanical energy of a system. In quantum mechanics, the Hamiltonian represents the energy of the wave function. Hamiltonian mechanics describe a deterministic and reversible evolution in a phase space (Malham, 2015). A phase space of a system that includes all possible states this system can represent. Eq. [3.24] shows a Hamiltonian function that describes the kinetic and potential energy of a particle.

$$H = \frac{-\hbar^2}{2m}\frac{\delta^2}{\delta x^2} + V(x) \tag{3.24}$$

Eq. [3.24] describes the time independent Schrödinger equation. Extending eq. [3.24] with the wave function the potential energy level will result in eq. [3.25].

$$E_n\psi_n(x) = \frac{-\hbar^2}{2m}\frac{\delta^2\psi_n(x)}{\delta x^2} + V\psi_n(x) \tag{3.25}$$

The first term in eq. [3.25] describes the kinetic energy and the second term describes the potential energy of a particle. In eq. [3.25] the angle ψ_n describes the wave function and gives information about the possible location of the particle. The quantized energy level E_n in eq. [3.25] can only take certain values, as its values depend on the frequency of the wave function (Ward & Volkmer, 2006).

3.6 Fundamentals of Qubits

Quantum computing is a combination of quantum mechanics, information theory, and many aspects of computer science. There is more relation between classical computer science and quantum computing than between classical computer science and classical computing.
The basis of quantum computing is a quantum bit, which is like the normal bit in classical computing. This section explains the nature of a qubit in order to understand the bigger picture of quantum computing.
Light consists of photons which are causing electromagnetic radiation. These photons are responsible for the wave- and particle-properties of light. The double split experiment explained in E. Yousif (2016) visualizes the wave properties of light, where interference occurs as a sign of inherent wave properties. The photoelectrical effect (Greenberger et al., 2016) verifies the particle property, as light must have an impulse to extract the negative electrons. According to Carcassi et al. (2020), quantum mechanics is built on four postulates which are follows:

- The superposition principle
- The measurement principle
- Unitary evolution

- No-Cloning-Theorem

The superposition principle describes the possible states of a given quantum system. In classical computing a bit can be either 0 or 1, but nothing in between. In quantum computing, the state of the qubit can represent any linear superposition of the base states $|0\rangle$ and $|1\rangle$ with complex coefficients.
Considering the base states of eq. [3.26], a qubit describes a two-state quantum system.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{3.26}$$

In this system, the amplitudes of the base states are defined by two complex numbers c_1 and c_2 as members of complex space $\mathbb{C}$. The amplitude of a specific state corelates with the likelihood of measuring this state. With a single qubit, two different pure vectors form the orthogonal basis and the amplitudes of each pure state define the orientation of this state.
Unlike other vector-spaces, Hilbert-space has the property of an inner product. Eq. [3.27] shows the notation for calculating the inner product of the same vector.

$$\langle\psi|\psi\rangle = \begin{bmatrix} a_0, a_1, ..., a_{n-1} \end{bmatrix} * \begin{bmatrix} a_0 \\ a_1 \\ . \\ . \\ . \\ a_{n-1} \end{bmatrix} = 1 \tag{3.27}$$

The inner product of two different vectors gives information on how similar these two vectors are. Therefore, calculating the inner product of a vector with its own transposed vector as in eq. [3.27] always results in number 1. In eq. [3.27] the notation $\langle\psi|\psi\rangle$ refers to the inner product of the state ψ. The notation $\langle\psi|$ is the conjugate transpose or Hermitian transpose of $|\psi\rangle$.
The measurement principle deals with how much information of the state of a qubit is accessible. When measuring a qubit in a superposition once, it is impossible to gain any specific information about the amplitudes of the state. As the superposition state will collapse into one of the orthogonal bases, the result of the measurement will be either 0 or 1 when using the common base of $|0\rangle$ and $|1\rangle$. In order to obtain more information of the amplitude, the measurements must be executed several times.
Within a quantum circuit, phenomena like superposition speed up the calculation. In general aspects, a circuit consists of gates that represent logical operations, wires and registers. One evaluation of a quantum circuit defines a quantum circuits that transforms an input state to an output state once. After every evaluation, the qubits should be reinitialized to the ground state and the ancilla bits must be restored to their initial state for the next iteration (Perkowski, 2020). Therefore, a quantum circuit usually includes a mirrored part of the operations at the end of the circuit in order to initialize the qubits for the next evaluation.
If a state in superposition is measured, the output will always be probabilistic. Considering

state $|\psi\rangle$ in eq. [3.28] with arbitrary amplitudes α and β, eq. [3.29] calculates the probabilities of measuring one of the base states.

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \quad \text{with} \quad \alpha, \beta \in \mathbb{C} \tag{3.28}$$

$$\begin{gathered} P(0) = |\langle 0|\psi\rangle|^2 = |\alpha|^2 \\ \langle 0|\psi\rangle = \begin{bmatrix} 1 & 0 \end{bmatrix} * \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \alpha \end{gathered} \tag{3.29}$$

Eq. [3.29] and eq. [3.30] notate P(0) and P(1) as the probability of measuring the state 0 and 1. Calculating P(0), the eq. [3.29] forms the inner product of the base state $|0\rangle$ and the state ψ. Eq. [3.30] uses also the inner product to calculate P(1).

$$\begin{gathered} P(1) = |\langle 1|\psi\rangle|^2 = \beta^2 \\ \langle 1|\psi\rangle = \begin{bmatrix} 0 & 1 \end{bmatrix} * \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \beta \end{gathered} \tag{3.30}$$

To express the state on the Bloch sphere, the amplitudes of any state in a Hilbert-space must fulfil eq. [3.31].

$$|\alpha|^2 + |\beta|^2 = 1 \tag{3.31}$$

Eq. [3.32] shows that the condition referred in eq. [3.31] holds true, even after the transformation of an arbitrary gate *U*.

$$U\,|\psi\rangle = |\psi'\rangle \longrightarrow \langle\psi'|\psi'\rangle = (U\,|\psi\rangle)^T U\,|\psi\rangle = \langle\psi|\,U^T U\,|\psi\rangle \tag{3.32}$$

Eq. [3.33] shows that if a specific gate like the Pauli-Y gate (Y) transforms a qubits' state, the transposed of this Pauli-Y gate (Y^T) will transform the state of the qubit back to the initial state.

$$Y * Y^T = 1 \tag{3.33}$$

The overall transformation of any gate and its transposed results in the identity matrix. In this case the two gates, the Pauli-Y gate and the transposed of the Pauli-Y gate are aligned serial and a matrix multiplication of both gates express the total transformation. Eq. [3.34] shows the inner product of the initial state ψ with the state ψ after the arbitrary operator *U* and its transposed U^T transformed it.

$$\langle\psi|\,U^T U\,|\psi\rangle = \langle\psi|\psi\rangle = 1 \tag{3.34}$$

Fig. [3.4] shows a qubit-state on a Bloch sphere in a Hilbert-space.

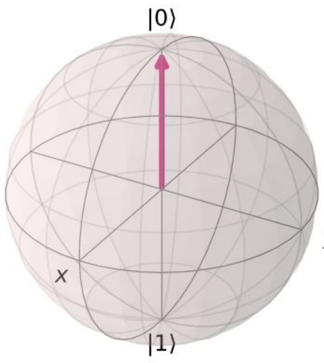


Figure 3.4: This figure shows the current state of a qubit, in this case the state $|0\rangle$ in a Bloch sphere. The classical states 0 and 1 can be compared to the quantum states $|0\rangle$ and $|1\rangle$. Operations on the qubits' state will change its orientation but not its length, as those transformations are unitary.

The z-axis in Fig. [3.4] goes through the base states $|0\rangle$ and $|1\rangle$. Considering the Fig. [3.4], the global phase θ refers to the rotation around the z-axis. In this Bloch sphere the state $|0\rangle$ is represented by a vector pointing upward and $|1\rangle$ by a vector pointing downward.
In order to get any information about the phase θ, the appropriate basis states must be chosen. As mentioned before, with the basis states $|0\rangle$ and $|1\rangle$ no information can be gained about the angle θ, but when changing the basis states to $|+\rangle$ and $|-\rangle$ the angle θ becomes an observable. This basis state change is done by changing the basis from $|0\rangle$ and $|1\rangle$ into $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.
In the Bloch sphere of Fig. [3.4] the state $|+\rangle$ would be a vector pointing towards the *x* aligned with the x-axis and the basis $|-\rangle$ would be a vector aligned with the x-axis pointing away from the letter *x*.
Due to the probabilistic nature of quantum computing, a quantum state usually must be measured various times to conclude anything. In the basis states $|0\rangle$ and $|1\rangle$ the complex part that represents the angular shift of θ has no impact on the measurement as shown in eq. [3.35] and following.

$$|q_0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ i \end{bmatrix} \tag{3.35}$$

In eq. [3.35] the basis state $|0\rangle$ has the amplitude of $\frac{1}{\sqrt{2}}$ and the basis state $|1\rangle$ has the amplitude of $\frac{i}{\sqrt{2}}$. The probability of measuring one of the two basis states is the amplitude of each basis state to the power of two. While squaring the amplitudes, the complex part of $|1\rangle$ in eq. [3.35] will disappear. As mentioned before, the complex part of the number describing the amplitude, represents the global phase θ of the state vector. With the basis states $|+\rangle$ and $|-\rangle$ the global phase θ becomes an observable. When speaking about an observable, it is referred to a quantity that can be measured (Cacciapuoti et al., 2019). For example, with the basis states $|0\rangle$ and $|1\rangle$ the angle θ, which is the rotation around the z-axis in a Bloch sphere, does not belong to the set of observables as no information can be gained about this rotation. Eq. [3.36] calculates the probability of measuring 1 with the state of eq. [3.35] (Asfaw et al., 2020).

$$\begin{aligned} P(1) &= |\langle 1|q_0\rangle|^2 \\ P(1) &= |\begin{bmatrix}0 & 1\end{bmatrix} * \tfrac{1}{\sqrt{2}}\begin{bmatrix}1\\ i\end{bmatrix}|^2 \\ P(1) &= |\tfrac{i}{\sqrt{2}}|^2 \\ P(1) &= 0.5 \end{aligned} \tag{3.36}$$

Basis states and superposition states are pure states. Basis states are usually chosen to be orthogonal, like $|0\rangle$ and $|1\rangle$. Common basis states at a one-qubit-system are $|0\rangle$ and $|1\rangle$, or $|+\rangle$ and $|-\rangle$. If a vector of a qubit can be written as a combination of both basis states like $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and neither α nor β is zero, the qubit is in superposition (Hari Dass, 2013). In general, the basis states do not have to be orthogonal, but this Master Thesis assumes that the basis states are always orthogonal.

The unitary evolution postulate states that the evolution of the quantum system between the measurements is unitary. Such an evolution can be represented by any rotation around the origin (Vazirani, 2012).

Considering a two-qubit system, eq. [3.37] shows that the output can be any combination of the two basis $|0\rangle$ and $|1\rangle$.

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \tag{3.37}$$

Einstein, Podolski and Rosen wrote a paper describing entanglement (Einstein et al., 1935), which is the strongest possible correlation between two qubits. Fig. [3.5] shows the Einstein-Podolski-Rosen (EPR)-circuit described in Garcia-Escartin & Chamorro-Posada (2011) that entangles two qubits.

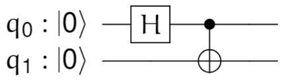


Figure 3.5: The EPR-circuit in the figure results in the strongest possible correlation between qubits. This correlation maintains even if the two qubits are separated. All figures that illustrate quantum circuits were created within this Master Thesis using the Python library *Qiskit*.

The EPR-circuit consists of a Hadamard-gate followed by a controlled-Not (CNOT)-gate, both gates will be discussed in subsection 3.7.1. Any arbitrary basis input state to this EPR-circuit will cause as output state one of the four so-called Bell-states (Bell, 1964). Eq. [3.38] and eq. [3.39] show all of the four Bell-states.

$$\left|\Phi^{+/-}\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle + / - |11\rangle) \tag{3.38}$$

$$\left|\Psi^{+/-}\right\rangle = \frac{1}{\sqrt{2}}(|01\rangle + / - |10\rangle) \tag{3.39}$$

According to the unitary evolution postulate, quantum operations and quantum circuits are always reversible. The only part in quantum circuits that is not reversible is the measurement, as the collapse of the superposition is probabilistic and, therefore, not reversible. Compared to the reversibility in quantum computing, operations in classical computing are usually irreversible. Only operations in classical computing like inverter are reversible due to the one-to-one mapping of input state and output state. This irreversibility in classical computing results in loss of information during the computation, which further results in a loss of the overall process efficiency. The lost information in classical computing will not simply disappear but transforms from informational energy into heat energy according to Landauer's principle (Landauer, 1961). The reversibility of quantum computing retrieves not only advantages but also disadvantages as it requires more memory and computational costs to maintain the information (Homeister, 2015). As reversibility in quantum computing is mandatory, quantum computing needs additional qubits, ancilla qubits, that assist with temporary calculations.

The No-Cloning-Theorem claims that a state of a qubit can only be copied if both qubit states, the original and the template state, are identical or orthogonal to each other (Homeister, 2015). In Fig. [3.6] the Toffoli gate clones the input qubit *a*.

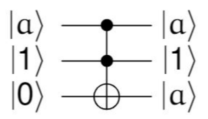


Figure 3.6: In case of *a* being a base state $|0\rangle$ or $|1\rangle$, a Toffoli gate will copy the state. The second qubit is an ancilla qubit in this case as it helps to execute the cloning process.

Section 3.7 explains the Toffoli-gates' function, it is only illustrated to show that it is possible to copy a qubits' state to a limited extend.

There are different types of states in quantum computing like pure, mixed, basic and superposition states. A pure state can be represented as vector in a Bloch sphere and is defined by its amplitude α and β. Eq. [3.40] defines such an example a pure state.

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{3.40}$$

A single source of laser light is an example for a pure state. A filter can divide the polarization of this light into horizontal and vertical polarizations. Eq. [3.41] composes the two different polarizations to describe the pure state ψ with *H* representing the horizontal polarization and *V* the vertical polarization.

$$|\psi\rangle = \alpha H + \beta V \tag{3.41}$$

A polarization beam splitter can split the horizontal and vertical states. A horizontal filter detects the amount α of horizontal polarization *H* and a vertical filter detects the amount β of the vertical polarization *V*. A polarization compensator can change one polarization of the state ψ in order to detect 100% of all photons with one detector. The combination of two laser light

sources is an example for a mixed state. With a combination of two sources it is impossible to change the polarization compensator in order to detect 100% of all photons (Trippe, 2014).
The basis states in this Master Thesis are always $|0\rangle$ and $|1\rangle$, but in general the basis states can be chosen arbitrarily.
In classical computing, a bit can only represent one basis state at once, either 0 or 1. In quantum computing, a qubit can also represent its basis states and, additionally, it can constitute any linear superposition of this basis states. Eq. [3.42] shows such a superposition of the basis states $|0\rangle$ and $|1\rangle$ with equal amplitudes.

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle\rangle \tag{3.42}$$

Eq. [3.43] represents the superposition state ψ of eq. [3.42] in a vector notation.

$$|\psi\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \tag{3.43}$$

This Master Thesis defines circuits in either classical computing or quantum computing as an arrangement of input signals, operators and output signals like in Fig. [3.6]. A circuit can be of different types, like sequential circuits or circuits with feedback. With a sequential circuit, the data go from input to output only. Adding a feedback loop to a sequential circuit, the output states will also affect the calculations of the circuit. Sequential circuits, also called combinational circuits, are more developed than circuits with feedback loop in quantum computing. Thus, when referring to circuits in this Thesis, combinational circuits are meant.
Interactions of the qubit system with the environment cause a higher error rate. If a qubit is exposed to noise, its superposition states tends to collapse. This unintended collapse of superposition makes a quantum computer more prone to noise than a classical computer. Such exposure to noise leads to decoherence, which is an unwanted and irreversible change of the state. In general, quantum systems are highly sensitive to noise. Noise can be caused by internal or external effects. An internal effect would be the scattering of a single photon and an external effect the gravitational interaction between objects in greater distance. An ideal measurement and control function do not cause any decoherence which is only realizable in theory. Furthermore, decoherence violates the superposition principle as certain states can no longer be observed (Kiefer & Joos, 1999).

3.7 Quantum Gates

This section explains the concept of quantum gates starting with a real-life example of the functionality of an AND-gate. Having a door that only opens if both of the two switches are active at the same time, is an example for a AND-gate. If no switch, or only one switch is active the door stays shut. Only in the case that both controlling switches are active, the door opens. This use case refers to the functionality of a classical AND-gate. A classical AND operator with its truth-table [3.1] is irreversible, as several input states will cause the same output states.

Table 3.1: The table represents the truth-table of an AND-gate. Input signals *a* and *b* result in the output signal *X*. Only with output signal X = 1 it is possible to conclude the state of the input signals. If the output signal is X = 0, three different constellation of input signals *a* and *b* are possible which makes the gate irreversible.

b	a	X
0	0	0
0	1	0
1	0	0
1	1	1

With quantum gates only one combination of input qubit state can cause a certain output qubit state, which is called one-to-one-mapping.
In quantum computing, various quantum operations exist that transform the qubits' state reversibly. There are quantum gates that operate like classical gates, as for example the Pauli-X gate. This gate works similarly to the classical NOT-gate. If the input state is 0, the output is 1 and vice versa. Furthermore, quantum gates exist that affect the input state in a way that is impossible in classical computation, like the Hadamard gate. This section introduces and explains the fundamental gates and their properties.

3.7.1 Single Qubit Gates

As the name indicates, a single qubit gate has one input state and one output state, as it only operates on one qubit.
The identity matrix of eq. [3.44] represents the Pauli-I gate and Fig. [3.7] shows its notation with an *id* inside the box.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.44}$$

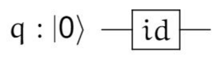


Figure 3.7: The identity gate leaves the qubits´ state unaltered. Its notation usually includes an *id* or *I* inside the box.

Considering an arbitrary Input state $|\psi\rangle$ with its amplitudes α and β, eq. [3.45] shows that the state of the qubit will not change after passing the identity gate.

$$I\,|\psi\rangle = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = |\psi\rangle \tag{3.45}$$

The matrix of eq. [3.46] represents the Pauli-X gate and Fig. [3.8] shows its notation with an *X* inside the box or with a circled cross.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{3.46}$$

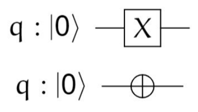


Figure 3.8: The Pauli-X gate mirrors the qubits' state at the x-axis. It is usually represented with *X* inside the square or with cross inside a circle. This figure illustrates both notations of the Pauli-X gate.

Eq. [3.47] considers an arbitrary input state of $|\psi\rangle$ and shows that the amplitudes of the qubit will change its relative phases.

$$X\,|\psi\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \tag{3.47}$$

Fig. [3.9] shows the transformation of a Pauli-X gate with the dots along the meridian of the Hilbert-space.

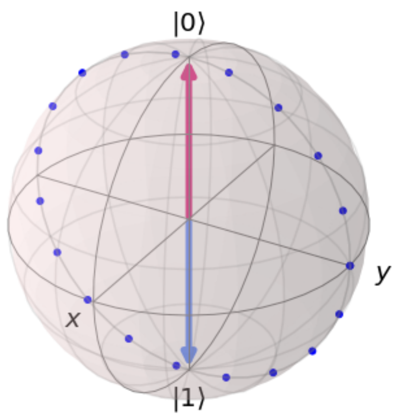


Figure 3.9: If the input state is the purple vector, the output state will be the blue vector and vice versa. The transformation matrix moves the input vector along the meridian, rotating it around the x-axis with 180 degrees.

The matrix in eq. [3.48] represents the Pauli-Y gate and Fig. [3.10] illustrates its notation with the letter *Y* inside a box.

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{3.48}$$

$$q : |0\rangle \; -\boxed{Y}-$$

Figure 3.10: The Pauli-Y gate mirrors the qubits´ state at the y-axis. Its notation has usually a *Y* inside the square.

Eq. [3.49] considers an arbitrary input qubit of state $|\psi\rangle$ and shows how the Pauli-Y gate will change the qubits' state.

$$Y|\psi\rangle = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} -i\beta \\ i\alpha \end{bmatrix} \tag{3.49}$$

The global phase in eq. [3.49] is represented with *i*. Fig. [3.11] shows that the input/output relation is the same as with the X-gate, but the states amplitudes in eq. [3.47] and eq. [3.49] are not equal.

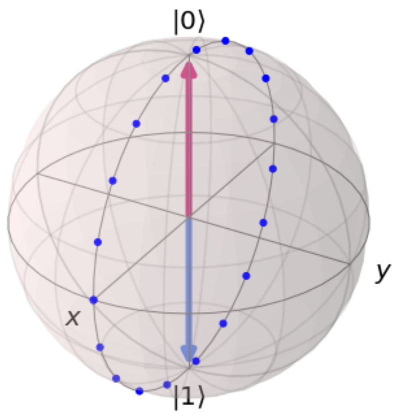


Figure 3.11: If the input state is the vector in the purple, the output state will be the vector in blue and vice versa. The result in this case is the same as applying a Pauli X gate, but the transformation happened along the y-axis.

The matrix in eq. [3.50] represents the Pauli-Z gate and Fig. [3.12] shows its notation with the letter *Z* inside the box.

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.50}$$

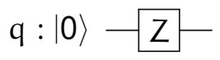


Figure 3.12: The Pauli-Z gate mirrors the qubits´ state at the z-axis. It is usually represented with *Z* inside the square.

The Pauli-Z gate has no observable effect on the states $|0\rangle$ nor $|1\rangle$ as it rotates the input state along the latitude. Having no observable effect means that the transformation does not influence the result of the measurement. It changes only the global phase of $|1\rangle$. As $|0\rangle$ and $|1\rangle$ are eigenvectors of the Z-gate, the operation does not have any effect on the measurement. The Pauli-Z gate rotates the vector around the z-axis, therefore, the transformation is not observable on basis states $|0\rangle$ or $|1\rangle$. Eq. [3.51] considers an superposition input state ψ with $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and a Pauli-Z gate that transforms this state to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$.

$$Z\,|\psi\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{3.51}$$

Fig. [3.13] shows that the transformation from eq. [3.51] is a rotation of the state vector along the latitude.

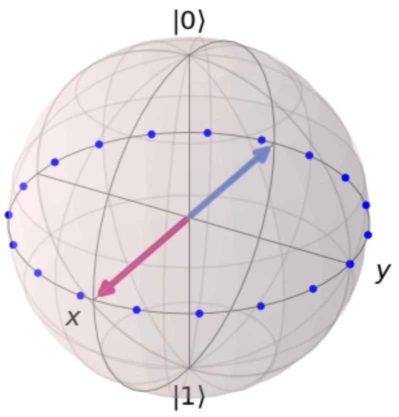


Figure 3.13: If the input state is the purple vector, the output state will be the blue vector and vice versa. This transformation moves the state along the latitude of the Bloch sphere.

The matrix in eq. [3.52] represents the S-gate and Fig. [3.14] shows its common representation with an *S* inside the box.

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \tag{3.52}$$

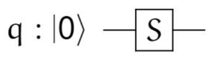


Figure 3.14: The S-gate rotates the qubits' state 90° around the z-axis. Its common representation is the letter *S* inside a box.

Considering an arbitrary input state of $|\psi\rangle$ the S gate will affect the $|1\rangle$ phase and will leave the $|0\rangle$ phase unaltered. Assuming the input state is $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$, eq. [3.53] shows the transformation after passing the S-gate.

$$S|\psi\rangle = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ i \end{bmatrix} \tag{3.53}$$

A different notation of the output state of eq. [3.53]is $\frac{|0\rangle+e^{\frac{i\pi}{2}}|1\rangle}{\sqrt{2}}$. Fig. [3.15] shows that as well as the Z-gate, the S-gate does not affect the input $|0\rangle$ and changes the phase of $|1\rangle$ by $\frac{\pi}{2}$.

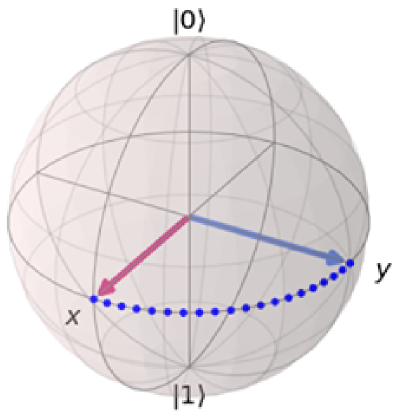


Figure 3.15: The purple colored vector represents the input state and the blue colored vector represents the output state. The S-gate rotates the state half as much as a Z-gate.

The T-gate does a similar transformation as the S-gate. The only difference is in the degree of phase shift. Eq. [3.54] shows the correlation between the S-gate to T-gate.

$$S = T^2 \tag{3.54}$$

There are also other similar relations such as in eq. [3.54]. Eq. [3.55] shows some relations between the identity gate and the Hadamard gate or the Pauli-X gate.

$$\begin{gathered} I = H^2 \\ I = X^2 \end{gathered} \tag{3.55}$$

The matrix of eq. [3.56] represents the T-gates and Fig. [3.16] shows its common representation is with an *T* inside the box.

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix} \tag{3.56}$$

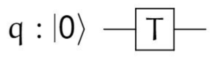


Figure 3.16: The T-gate rotates the qubits' state 45° around the z-axis. It is usually represented with a *T* inside the square.

Fig. [3.17] shows that, considering an input state of $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$, the state of the qubit will change to $\frac{|0\rangle+e^{\frac{i\pi}{4}}|1\rangle}{\sqrt{2}}$, which is a 45 degrees rotation around the z-axis.

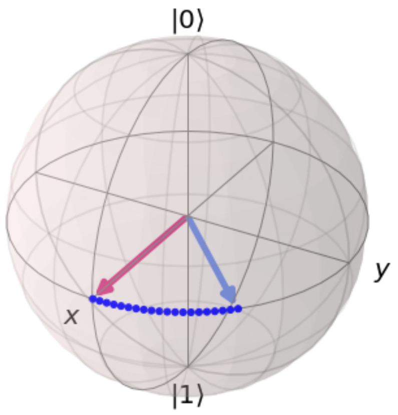


Figure 3.17: The purple colored vector represents the input state and the blue colored vector represents the output state. The T-gate rotates the state half as much as an S-gate.

The S-gate does not have a visual effect to the input $|0\rangle$ but will change the phase of $|1\rangle$.
The matrix of eq. [3.57] represents the V-gate and Fig. [3.18] shows its common representation with a squared letter *V*.

$$V = \begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix} \tag{3.57}$$

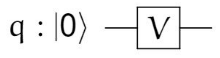


Figure 3.18: The V-gate is a truly quantum gate like the Hadamard gate. The representation of this gate is usually a squared letter *V*.

The V-gate, also called the square-root-of-NOT, and the Hadamard gate are the only gates this Master Thesis represents, that are truly quantum gates. Truly quantum gates cannot be adapted or imitated in other areas than quantum computing, which makes them unique. In inverse logic for example, the square-root-of-NOT, and the Hadamard gate can be replicated but not to their full potential. Considering an input state of $|0\rangle$ the state of the qubit will change according to the matrix of eq. [3.58].

$$V|\psi\rangle = \begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1+i \\ 1-i \end{bmatrix} \tag{3.58}$$

The Hadamard gate can be represented with a Hermitian matrix as described in subsection 3.2.2. A Hermitian matrix has beneficial properties regarding computation. Such a matrix is diagonalizable, as is has complete set of orthogonal eigenvectors inherent. Additionally, its

spectrum only contains real numbers. With these two properties, a Hermitian matrix can be approximated by matrices of lower rank.
The Hadamard gate transforms the basis states $|0\rangle$ and $|1\rangle$ into superposition. Eq. [3.59] shows its common matrix representation and Fig. [3.19] shows usual notation with the squared letter *H*.

$$\mathrm{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3.59}$$

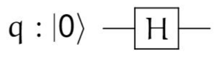


Figure 3.19: The H gate is a truly quantum gate like the square-root-of-not gate. The representation of this gate is the squared letter *H*.

Eq. [3.60] and Fig. [3.20] shows that, considering the input state is $|0\rangle$, the Hadamard gate will change this state to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$.

$$\mathrm{H}\,|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{3.60}$$

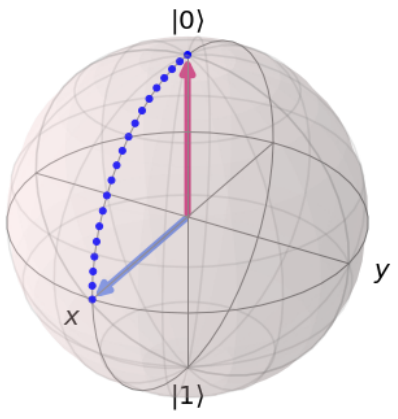


Figure 3.20: The purple colored vector represents the input state and the blue colored vector represents output state. After applying an ideal Hadamard gate to a basis state, the probability of measuring the state $|0\rangle$ or $|1\rangle$ is 50%.

Hadamard gates can generate truly random numbers. Other technologies, like serial autometer or linear feedback shift register, try to generate such random numbers but in every other area than quantum computing those numbers are pseudo random numbers.
Compared with the Hadamard gate, the pseudo Hadamard transforms the state counter wise. The matrix of eq. [3.61] represents the transformation of the pseudo Hadamard gate and Fig. [3.21] shows its common notation with a squared letter *h*.

$$\mathrm{h} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \tag{3.61}$$

$$\mathrm{q} : |0\rangle \; -\boxed{\mathrm{h}}-$$

Figure 3.21: The pseudo Hadamard gate is a truly quantum gate like the square-root-of-not gate. The notation of this gate uses a squared letter *h*.

Eq. [3.62] and Fig. [3.22] show that, considering an input state of $|0\rangle$, the state of the qubit will change to $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$.

$$\mathrm{h}\,|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{3.62}$$

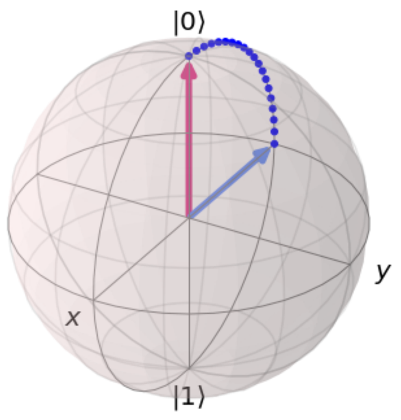


Figure 3.22: The purple colored vector represents the input state and the blue colored vector represents the output state.

3.7.2 Multi Qubit Gates

As mentioned at the beginning of subsection 3.7.1, quantum gates are reversible. Whereas a single qubit gate only transforms one qubit at a time, a multi-qubit gate operates on several qubits. In most cases, the input qubits of a multi-qubit gate has a target register and a control register. The transformation will usually be conducted on the target register only if the control register is active. The control register consists of either one qubit or multiple qubits that control if the transformation will be performed on the target qubit.
This subsection will start off with the CNOT-gate, also called the Feynman gate (Toffoli, 1980). The matrix of eq. [3.63] represents the Feynman gate and Fig. [3.23] shows its common representation.

$$\mathrm{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.63}$$

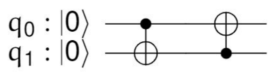


Figure 3.23: The Feynman gate is not necessarily controlled from the upper qubit, the circuit designer can align the Feynman gate also the other way around. The black dot represents the control register and a cross within the circle represents the target register.

The Feynman gate can be either controlled from the first qubit targeting the second qubit, called controlled-up or via versa, called controlled-down. The standard version of Feynman gate is controlled-up, but a controlled Feynman gate from the bottom is also possible. The black dot marks the control qubit and the circle with the cross in the middle marks the target qubit.
Fig. [3.24] shows how the Feynman gate affects a two-qubit system, in case of a controlled-up Feynman gate with control qubit q_0 being active, and qubit q_1 initialized with $|0\rangle$.

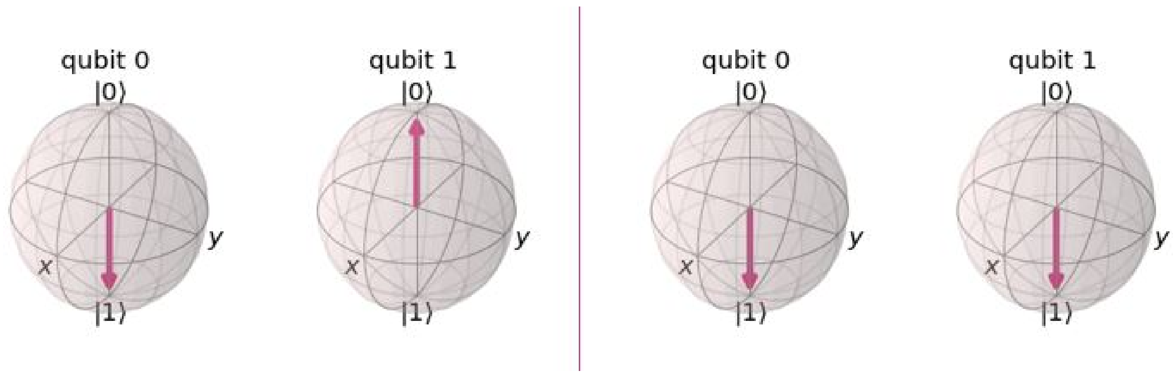


Figure 3.24: The two left qubits represent the state before the CNOT-gate, with the control *qubit 0* being $|1\rangle$. After the transformation, the target *qubit 1* is flipped from the state $|0\rangle$ to the state $|1\rangle$.

The Toffoli gate (Toffoli, 1980), also called controlled-controlled-Not (CCNOT) gate, is an extension of the CNOT gate. The matrix of eq. [3.64] represents this gate and Fig. [3.25] shows its common circuit representation.

$$\mathrm{CCNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.64}$$

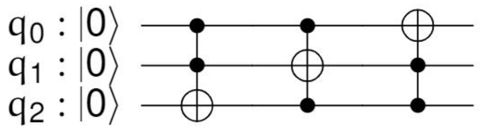


Figure 3.25: The Toffoli gate is not necessarily controlled from the upper two qubits, it can be aligned as needed. The black dots represent the control register and the circle with the cross inside represents the target register.

In a three-qubit system, the Toffoli gate can have any target qubit controlled by the other two qubits. In the common representation of the Toffoli gate, the target qubit is at the bottom, and controlled by the two qubits on top which is called controlled-up. Being controlled by the lowest qubits is called controlled-down and controlled with the outermost qubits is called controlled-outer.
With control qubit 0 and qubit 1 being active $|1\rangle$, and qubit 2 initialized with $|0\rangle$, Fig. [3.26] shows the output state on the right side.

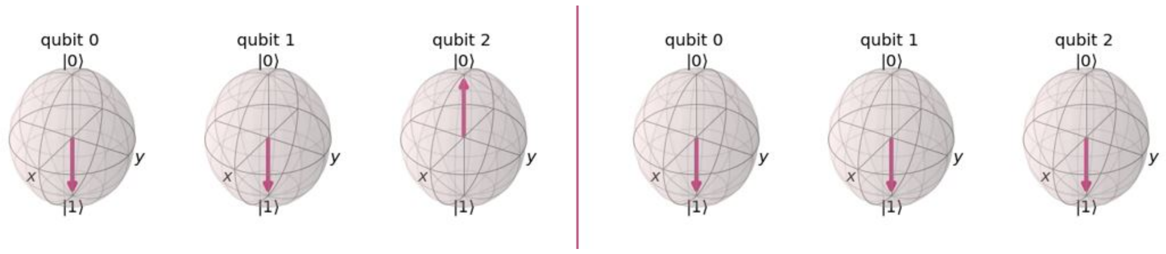


Figure 3.26: The two left most qubits, qubit 0 and qubit 1, are the controlling qubits, and qubit 2 is the target register. The left side represents the state before the transformation with a CCNOT-gate and the right side after the transformation.

The only multi-qubit gate mentioned in this Master Thesis that does not have a target register and a control register is the swap gate. This gate, as the name suggests, swaps the state of two qubits with each other and the matrix of eq. [3.65] represents its transformation.

$$Swap = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.65}$$

Fig. [3.27] is the representation of the swap gate in a circuit and Fig. [3.28] shows a two-qubit system before and after the transformation of a swap gate.

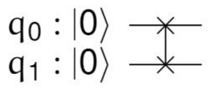


Figure 3.27: The swap gate changes two qubits without collapsing their superposition. After the transformation, the states of q_0 and q_1 change their positions in the register.

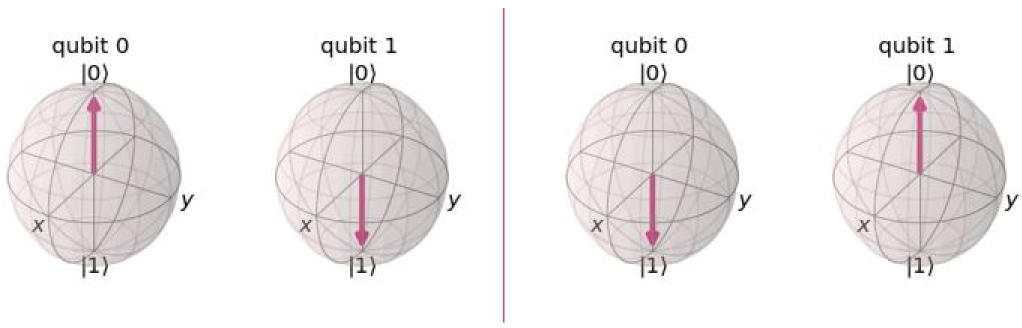


Figure 3.28: On the left side is the two-qubit system before the swap gate, with qubit 0 being $|0\rangle$ and qubit 1 being $|1\rangle$. The swap gate will change the qubits states.

The fourth multi-qubit gate is the Fredkin gate (Toffoli, 1980) also called control-SWAP gate or short CSWAP gate. This three-qubit gate can simulate various behaviors like AND, NOT, CROSSOVER, and FANOUT. The matrix of eq. [3.66] represents the transformation and Fig. [3.29] shows its common notation in a circuit.

$$CSWAP = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.66}$$

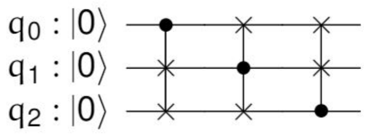


Figure 3.29: The Fredkin gate is not necessarily controlled from the upper qubit, it can be aligned as needed. The black dot represents the control register and the crosses mark the qubits that will be swapped.

This quantum gate can be controlled from any of these three qubits and does not be necessarily controlled-up. Fig. [3.30] shows that the Fredkin gate swaps the qubits states, but only if the control register is active.

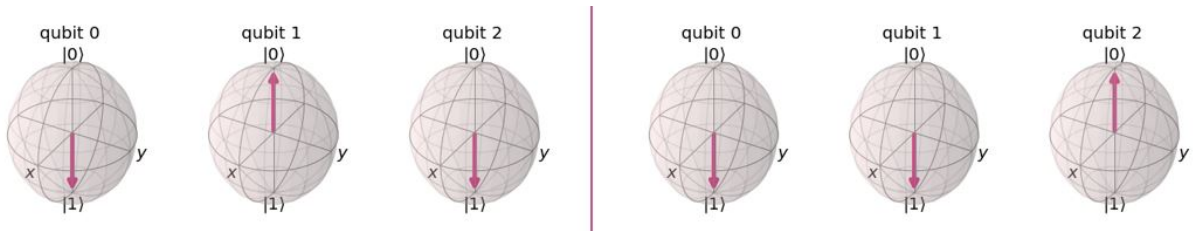


Figure 3.30: In the left three-qubit system, the left most qubit is the control qubit and the two qubits on the right are being swapped. The left side represents the state before the transformation with a controlled-Swap (CSWAP) gate and the right side after the transformation.

Fig. [3.30] shows that with the control qubit 0 in state $|1\rangle$, qubit 1 and qubit 2 are being swapped. Fig. [3.31] shows the functionality of the Fredkin gate in a Blackbox notation.

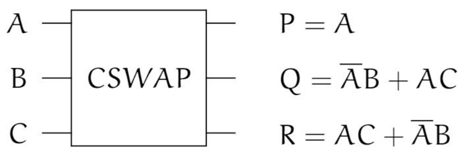


Figure 3.31: In this figure, *A* controls the CSWAP gate and swaps *B* and *C*. The output states of *B* and *C* are defined as *Q* and *R*. On the output side, the line over *A* refers to *A not*. To interpret Q the first term can be red if *A* is inactive the result is *B* and if *A* is active the result is *C*.

If the control input state *A* is active, values *B* and *C* swap and if the control input is inactive the input maps directly onto the output.

Circuit designers use Fredkin gates for example to build quantum multiplexer and quantum demultiplexer circuits. Fig. [3.32] shows that with three Fredkin gates it is possible to build a 4:1 quantum multiplexer.

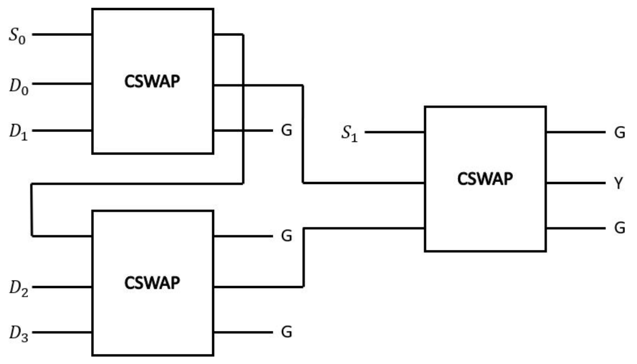


Figure 3.32: This figure shows how to align three Fredkin gates to create a 4:1 multiplexer. A 1:4 demultiplexer uses the inverse structure of this circuit.

Fig. [3.33] represents the same circuit as in Fig. [3.32] using quantum notation.

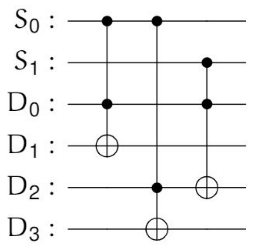


Figure 3.33: This figure realizes the 4:1 quantum multiplexer in quantum notations. This circuit has the same function as the Blackbox notation of Fig. [3.32].

Eq. [3.67] shows the output function of Fig. [3.32] and Fig. [3.33] (Kole et al., 2016).

$$Y = \overline{S}_0\overline{S}_1D_0 \oplus S_0\overline{S}_1D_1 \oplus \overline{S}_0S_1D_2 \oplus S_0S_1D_3 \tag{3.67}$$

3.8 Tensor and Kronecker Product

Tensor and Kronecker product describes a method of constructing a vector-space consisting of various vector subspaces. The Tensor product is related to operators and vectors in the multi-particle system, whereas the Kronecker product is applied to matrices. The Kronecker product yields to a common vector-space with multiple operators and is used in if a quantum system consists of multiple qubits (Fernández, 2016). If combining two vector-spaces, *V* with n-dimensions the *W* with m-dimensions, the resulting vector-space will be (n*m)-dimensional.

Eq. [3.68] represents a vector in the combined vector-space of $V \otimes W$.

$$\vec{v} \otimes \vec{w} = (\sum_{i}^{n} v_i \vec{e_i}) \otimes (\sum_{i}^{m} w_j \vec{f_j}) = \sum_{i}^{n} \sum_{i}^{m} v_i w_j (\vec{e_i} \otimes \vec{f_j}) \tag{3.68}$$

Eq. [3.69] represents the basis of the new vector-space.

$$\vec{v} \otimes \vec{w} = \begin{bmatrix} v_1 w_1 \\ v_1 w_2 \\ .... \\ v_1 w_m \\ v_2 w_1 \\ v_2 w_2 \\ .... \\ v_2 w_m \\ v_n w_1 \\ v_n w_2 \\ .... \\ n_2 w_m \end{bmatrix} \tag{3.69}$$

The result of the Kronecker product is a matrix of higher dimensions in comparison to the input matrices. A multiplication of a 2*2 matrix *A* and a 3*2 matrix *B* will result in a 6*4 matrix. Eq. [3.70] shows that the Kronecker product is not restricted on squared matrices.

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \otimes \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \\ B_{31} & B_{32} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} & A_{11}B_{12} & A_{12}B_{11} & A_{12}B_{12} \\ A_{11}B_{21} & A_{11}B_{22} & A_{12}B_{21} & A_{12}B_{22} \\ A_{11}B_{31} & A_{11}B_{32} & A_{12}B_{31} & A_{12}B_{32} \\ A_{21}B_{11} & A_{21}B_{12} & A_{22}B_{11} & A_{22}B_{12} \\ A_{21}B_{21} & A_{21}B_{22} & A_{22}B_{21} & A_{22}B_{22} \\ A_{21}B_{31} & A_{21}B_{32} & A_{22}B_{31} & A_{22}B_{32} \end{bmatrix} \tag{3.70}$$

Eq. [3.70] shows the specific method to calculate the Kronecker product. Eq. [3.71], and eq. [3.72] show different properties a vector-space has that is composed with the Tensor product.

$$\det(A \otimes B) = (\det A)^m (\det B)^n \tag{3.71}$$

In eq. [3.71] the abbreviation *det* stands for determinant, which is a volume scaling factor of a matrix (Hassani Monfared, 2011).

$$\mathrm{Tr}(A \otimes B) = (\mathrm{Tr}\, A)(\mathrm{Tr}\, B) \tag{3.72}$$

In eq. [3.72] the abbreviation *Tr* stands for the trace of a matrix, which is the sum of all diagonal elements of a matrix.
With classical mechanics, every dimension has two DOF, rotations and translation. In quantum

mechanics there is only one DOF per dimension. In comparison to classical mechanics, this DOF in quantum mechanics consists of a complex number in a complex vector-space. As the Hilbert-space in quantum computing has three dimensions namely x-axis, y-axis and z-axis, it has three DOF, p_x, p_y, and p_z (Buric, 2010). Eq. [3.73] shows that if one qubit in a vector sub-space is in a superposition state, the composed Tensor product space is in superposition.

$$\begin{aligned}(v_1 + v_2) \otimes w = v_1 \otimes w + v_2 \otimes w, \\ v \otimes (w_1 \otimes w_2) = v \otimes w_1 + v \otimes w_2\end{aligned} \tag{3.73}$$

Quantum computing distinguishes between normal matrix multiplication and the Kronecker product. If two gates are connected in sequence, a normal matrix multiplication is performed to get the total transformation of both gates. If the gates are arranged in parallel, the Kronecker product is used to calculate the total transformation. Fig. [3.34] shows a practical example of both operations on the EPR-circuit.

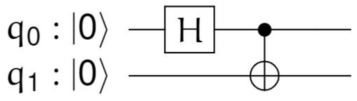


Figure 3.34: This is the famous EPR-circuit that creates entanglement between two input qubits q_0 and q_1, which is the strongest correlation between two qubits.

The calculation splits the circuit of Fig. [3.34] in two parts according to the Hadamard gate and the CNOT-gate. As the qubits q_0 and q_1 are aligned parallel, the Kronecker product will be used to calculate the transformations. As both transformations are aligned sequentially a normal matrix multiplication calculates the overall transformation.
The calculation splits the circuit in two parts according to the gates. Fig. [3.35] shows that the first part contains the Hadamard gate and an identity gate.

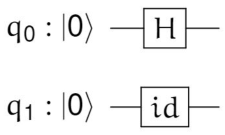


Figure 3.35: The Kronecker product calculates the transformation of two parallel gates. The *id*-gate represents the identity gate and the *H* represents the Hadamard gate.

The identity gate does not affect the qubit q_1 and is mainly included to help visualize the calculation.

Eq. [3.74] and eq. [3.75] show the matrices of the Hadamard and the identity gate.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3.74}$$

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.75}$$

As the identity matrix and the Hadamard matrix are in parallel, the Kronecker product will be used to calculate the total transformation. Eq. [3.76] shows the calculation of the Kronecker product from this first part P_1 in Fig. [3.35].

$$P_1 = H \otimes I = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{3.76}$$

Fig. [3.36] shows that the second part of the circuit P_2 consists of the Feynman gate and eq. [3.77] represents this gate in matrix notation.

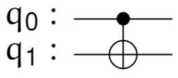


Figure 3.36: The Toffoli gate represents the second part of the EPR-circuit.

$$P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.77}$$

As P_1 and P_2 are sequentially, not the Kronecker product but the normal matrix multiplication of eq. [3.78] calculates the overall matrix P_{tot},

$$P_{tot} = P_2 * P_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix} \tag{3.78}$$

P_1 and P_2 have to be calculated in inverse order to represent the EPR-circuit. To finish the practical example, the input state ψ_0 of the EPR-circuit is assumed to be $|00\rangle$. Eq. [3.79] shows all two-qubit states in a vector notation.

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.79}$$

Eq. [3.80] shows how to calculate the output state ψ_1 with the input state $\psi = |00\rangle$.

$$\psi_1 = P_{tot} * \psi_0 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.80}$$

Eq. [3.81] further decomposes the result of eq. [3.80] basis states from eq. [3.79].

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{3.81}$$

3.9 Quantum Parallelism

Some gates operate only on one qubit and other gates affect multiple qubits. Usually these multi-qubit-gates are controlled gates, which means that one or more qubits control if the operation is executed on the target qubit. This functionality is different if the control register is in superposition. Executing such a controlled gate with its controlled qubits in superposition leads to the so-called quantum parallelism. Quantum parallelism describes the ability of a circuit to work on many individual states simultaneously. Compared to quantum parallelism, in classical computing the circuits can only work on one state at a time. An example considers the functionality of a simple adder with two inputs. In classical computing, the adder can calculate the sum of two input signals. With a quantum adder and input states in superposition, all possible output states will be calculated at once. The benefit of the quantum adder is the fast calculation of any possible output. The downside of the quantum adder is that the solution is not available directly after the calculation. As the output state is also in superposition, the solution must be extracted in the first place in order to use it. Eq. [3.82] describes the operation of a general controlled operator$U$ with $|x\rangle$ being the control register and $|y\rangle$ being the target register.

$$|x, y\rangle \mapsto U_f |x, y\rangle = |x, y \oplus f(x)\rangle \tag{3.82}$$

The following example considers a trivial setup, where both registers have only one qubit each. The state of qubit *x* is in superposition and the target qubit *y* is in state $|0\rangle$. The operation of *U* is formalized as a function *f*. The eq. [3.83] states that the function is being evaluated simultaneously for both possible base states $|0\rangle$ and $|1\rangle$.

$$U_f |x, y\rangle = \frac{1}{\sqrt{2}}(|0, f(0)\rangle + |1, f(1)\rangle) \tag{3.83}$$

Measuring the state of eq. [3.83], one would obtain the result of f(0) and f(1) with the same probability (Kitaev et al., 2002). In order to make use of quantum parallelism, the important information needs to be extracted selectively like in the Deutsch algorithm (Deutsch, 1984).

3.10 Quantum Automata

A quantum automaton is a quantized form of a probabilistic automaton. When designing circuits, one distinguishes between circuits with memory and without memory. The circuits without memory are independent from previous evaluation loops and the circuits with memory rely on the results of the previous evaluation loop. So-called quantum automata belong to the type of circuits with memory. Such a probabilistic automaton is also called a probabilistic state machine. A state machine constitutes a mathematical framework to describe a sequence of actions (Stoelinga, 2004).

A deterministic finite autonoma (DFA) is a trivial introduction to how such a quantum automaton works. The first step is to create a language *L* as in eq. [3.84] which defines the possible solutions of the DFA.

$$L = \{1, 01, 10, 11, 001, ...\} \tag{3.84}$$

In this example, the possible solution is any string containing at least one 1. In its most trivial form, the DFA constitutes of two states, one starting state (S), and one target state (T). The starting state reads the first bit of the string. Whenever the first bit is a 1, the state of the DFA changes to the target state. If, however, the first bit is a 0, the state will loop with itself and tries the next bit. Reaching the target state, the residual bits of the string will create a self-loop with this state. Fig. [3.37] visualizes the state machine and tab. [3.2] represents it in a state-transition-tabular.

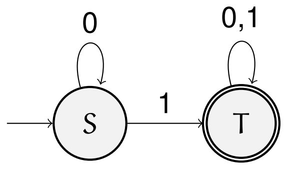


Figure 3.37: Trivial DFA accepting any strings containing at least one 1. When a string reaches *T*, it will self-loop through the entire sting without changing the state. The target state is marked with a double circuit and the start with a single arrow pointing towards the node. All figures of state machines were created within this Master Thesis using the LaTeX package *TikZ*.

Table 3.2: The state-transition-table maps the input state to the output state. The first column indicates the current state with *S* for start and *T* for target. The actions such a node can do are *C* for change the state, and *SL* for self-loop. As seen in the row of *T*, the state machine will not change the current state anymore and only self-loops independently of the read-out.

Input State	1	0
S	C	SL
T	SL	SL

The state machine can be interpreted in two ways, as string interpreter and generator or as string acceptor. A string interpreter and generator reads or creates a string and the string acceptor checks if a string satisfies a specific condition. In this example, the state machine is a string acceptor. The operations such a node is able to execute can be represented with matrices. If the input state is 0, the state will not be changed. Eq. [3.85] represents this action with an identity matrix.

$$M_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.85}$$

Eq. [3.86] represents the transformation matrix if the input state is a 1.

$$M_1 = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{3.86}$$

Considering an input string "101", the matrix multiplication of $M_1 M_0 M_1$ describe the operations of the state machine.

A basic example to visualize the framework of a probabilistic automaton is a coin flip. Fig. [3.38] shows that flipping a fair coin twice will result in one of four states.

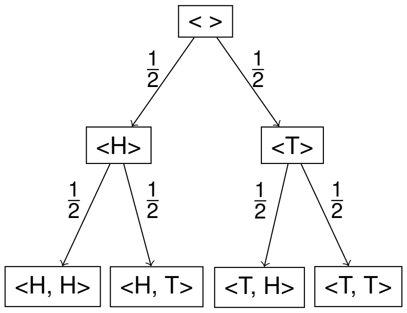


Figure 3.38: A probabilistic automaton of two fair coin flips with *H* for head and *T* for tail. The happened events are inside the box and the probabilistic are along the connecting lines. At the beginning the box is empty as no event did happened yet. After the first flip the event will be written inside the box and the next level starts. After the second flip the event will be written in the box to the previous event. All figures that illustrate decision trees were created within this Master Thesis using the LaTeX package *TikZ*.

The matrix form of eq. [3.87] represents the operations along the decision tree in Fig. [3.38].

$$M = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} \tag{3.87}$$

The matrix of eq. [3.87] show that probabilistic automata also allow fractional values. The most important properties of such a transformation matrix are that the sum of each row adds up to 1 and the numbers must be real.

With quantum finite automata, the entries of the transformation matrix are not restricted to be real as they are in a complex vector-space. One hard condition is that the matrices must be unitary (Jafarpour et al., 2007). Fig. [3.39] visualizes a trivial example of a quantum automaton.

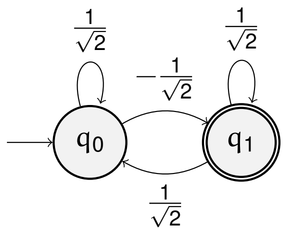


Figure 3.39: In the quantum automaton visualization, q_0 and q_1 define the two different qubits of a register. The amplitudes of the qubits are related to the possibilities of changing the state or self-looping.

4 Quantum Robotics

This chapter discusses the approach for classical learning on humanoid biped robots and leads towards learning in quantum robotics. As mentioned in section 1, there are only a two definitions of quantum robotics so far, one from Benioff and one from PSU. Around the same time as PSU, Dong et al. (2006) also formulated a similar definition of quantum robotics. In his definition a quantum robot consists of multi quantum computing units, quantum controllers, quantum actuators and information acquisition units. This chapter finishes with the definition of mood and emotions and how to design a behavioral model for robots.

4.1 Classical Learning for Humanoid Biped Robot

Machine Learning (ML) uses algorithm-based methods to learn from data. Compared to rule-based programming, ML performs better especially with larger amounts of data. In general, ML has four different types as follows:

- Supervised
- Unsupervised
- Semi-supervised
- Reinforcement learning

These four types train a machine based on a given dataset how to predict labels, cluster data, associate rules, or make decisions (Mohammed et al., 2016). With a proportion of about 7.2%, robotics is one of the major areas which is affected by ML. Examples of ML in robotics can be found in a vast variety of industry and entertainment sectors (Aparanji et al., 2017; Pierson & Gashler, 2017).
For instances, the start-up Skymind applied deep learning methods to their robot SKIL Somatic tool in order to understand and use the control-panel of elevators. With a convolutional neural network-based vision system and a fusion of long- and short-term-memory-based-sensors the robot can understand how to use a control panel in an elevator by learning its general structure (roboticsbusinessreview.com, 2018).
Section 2.3 already mentioned that facial expressions in HRI are important and can communicate feelings or intentions without the need of verbal communication. To avoid miscommunication between user and humanoid robot, human-like communication regarding lip movements

and other facial expression are researched in Bevacqua & Mancini (2004). The robot Robotinho from the University of Bonn, Germany executes actions depending on its emotions, mood, personality and the observation of the communication partner. This robot executes facial expressions with different intensities, in order to imitate a more human-like nature (Nieuwenhuisen et al., 2010).
In order to simulate a characteristic persona of a robot, a lot of projects considered artificial agents controlled by neuronal networks (Azvan & Florian, 2003; Honarvar & Ghasem-Aghaee, 2010). In contrast to normal tools, an artificial agent can handle tasks with a higher level of autonomy. An agent is anything that makes decisions considering past and current precepts, aiming for the best possible output. Further, agents can be classified into three different classes as follows:

- Artificial agents
- Biological agents
- Hybrid agents

For instances, robots belong to the class of artificial agents, humans and animals belong to the class of biological agents and the mixture of both is a hybrid agent (Tweedale et al., 2007). A robotic agent in its trivial form can consist of cameras to gain information about the environment, a computer with prior knowledge and goals that processes the gained information, and effectors to act according to its knowledge and goals (Goy & Torre, 2013).
A main drawback of these neuronal networks in a genetic algorithm is that they are computational expensive to initialize. In Hautop & Miglino (1998) the evolution from the model of the neuronal network is done by simulation to shrink the costs. In order to train such a neuronal network control system in a simulation, the robots' sensory activation is used as input stimuli for the net. After the model is initialized, the artificial agent is tested in a real environment. When the simulated model is transferred onto the real robot, its performance decreases compared to the simulated environment as during the system evolution the noise of a real environment has not been considered. The simulation did not consider noises like uneven surfaces or fluctuating lights during the evolution. To shrink the performance decrease, the simulation adds artificial noise during the evolution process. After transferring the model onto the real robot, a test and validation phase has to be done to improve the artificial agent further (Hautop & Miglino, 1998).
Future robots are supposed to be controlled by intelligent and autonomous methods in order to adapt faster to their tasks. To act intelligently and autonomously, the robot is supposed to have enough sensors and effectors to interact with its environment sufficiently. The intelligent and autonomous automata (IAA)(Holling & DeGaris, 2003) needs to evaluate the performance of the executed functions and adapt the weights that control the functions' output accordingly. In order to design such an IAA, two versions are considered in Holling & DeGaris (2003):

- A mixture of Delayed Pointer Network and electronic Learning Evolutionary Model
- BrainChild Technology

The delayed Pointer Network is based on *Collect and Distribute* Neural Net technology and constructs adaptable self-learning and brain-like structures. Holling & DeGaris (2003) considers the combination of the delayed Pointer Network with the electronic Learning Evolutionary Model to be intelligent and autonomous. The Electronic Learning Evolutionary Model uses fitness chromosomes to evaluate the performance and evolution of complex systems, whereas BrainChild technology focuses on cognitive processes rather than a certain hardware implementation.
All three methods, Delayed Pointer Network, electronic Learning Evolutionary Model, and BrainChild use flexible, variable size networks. Only the BrainChild technology can feature networks with up to one-hundred-thousand nodes that can be considered intelligent according to (Holling & DeGaris, 2003). Yet, the other two methods have not enough nodes to handle complex problems and cannot be considered as intelligent (Holling & DeGaris, 2003).
A behavioral model or a cognitive model defines a character's personality on a high level, without spending too much time defining every possible event. A behavioral model tries to meet short-term goals, whereas a cognitive model aims to fulfil long-time behavioral or planning goals. A short-term goal would be to react to a specific input signal such as recognizing a person. A long-term goal often goes often along with planning tasks, like shortest path planning. According to Dinerstein et al. (2004) a cognitive model is more complex, as more variables have to be considered with long-term decisions. To speed up the execution and decrease the complexity of the cognitive model, a combination of artificial neural network and reinforcement learning is used in Dinerstein et al. (2004).

4.2 Towards Quantum Learning for Humanoid Biped Robot

Quantum robots can benefit from quantum phenomena like entanglement, superposition, quantum parallelism and measurement expressed with EPR-circuits. Robots, which perform action planning or vision based on the Grover algorithm have already been proposed in (Li et al., 2006), (Wang & Perkowski, 2011), and (Lee & Perkowski, 2016).
Degen et al. (2016) extended the quantum robot concept of PSU by adding quantum sensors to the robot. According to the concepts developed in this Master Thesis, the focus will be on quantum robots with classical sensors, a hybrid-computer and classical effectors. A hybrid computer consists of a classical and a quantum computer.
As quantumness is used to control the input/output behavior of the robot, it can be used in several ways like imitating a behavioral model or planning tasks. One robotic application of quantumness is to generate motions such as in Deng (2020). Moreover, Lee & Perkowski (2016) already uses quantumness to solve different decision and planning problems. In this Master Thesis, the main focus is on generating motions, creating a behavioral model and various learning applications using real quantum systems.
One of the goals in combining quantum computing and robotics is to potentially reduce the

complexity of searching algorithms. A classical robot controlled with a Central Processing Unit (CPU) has limited resources to process all environmental data that is needed to make appropriate decisions. In comparison to the classical robot, the concept of a quantum robot includes an additional quantum computing unit which uses quantum parallel processing to reduce the complexity of solving problems and to speed up the processing (Dong et al., 2006).
The probabilistic nature of quantum computing offers the opportunity to create a complex behavioral model with lower computational costs than a classical behavioral model described in subsection 4.1. The internal state affects the robot's behavior and the way it executes motions. The factors mood and emotion decide how the signals are transformed before being sent to the effectors. Fig. [4.1] shows that this transformation is done in Hilbert-space, while the memory is stored on a classical computer.

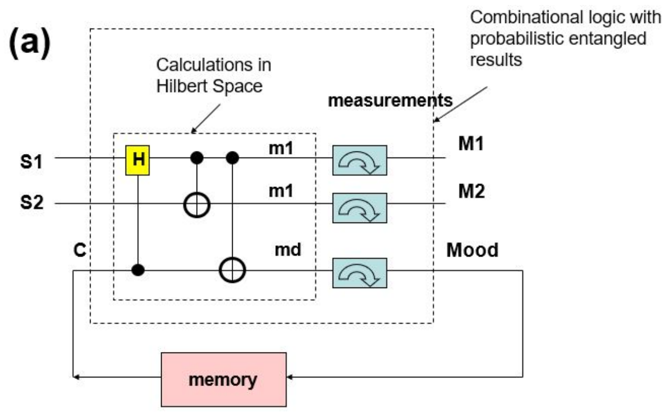

(a)
Combinational logic with probabilistic entangled results
Calculations in Hilbert Space
measurements
S1
S2
C
H
m1
m1
md
M1
M2
Mood
memory

Figure 4.1: Potential setup of a quantum robot with classical sensors and actors. The mood is presented with classical bits and the transformation of sensor to motor data is done in Hilbert-space (Raghuvanshi et al., 2007).

According to Aristidou & Lasenby (2009), a quantum automaton, as seen in Fig. [4.1] that includes combinational and probabilistic functions, bears more potential than homogeny functions. The amount of data and the complexity of problems will increase in the future. A humanoid robot such as Leonardo (Breazeal et al., 2004), with its 65-DOF, is more complex to control than a six-axis robot (Aparnathi, 2014). When robots imitate the human body, the number of DOFs will increase. The human body has up to 244 DOFs, which will lead to intense computational cost in case of handling all these DOF on a robot (Bonnet & Venture, 2015). Therefore, more powerful computers are needed in order to ensure real time behavior and inclusion of all relevant data. With quantum parallelism it might be possible to handle the amount of DOFs faster and more reliable than with classical computing.
Taking an example that relates to humanoid robots, the path planning for walking through a room with obstacles, finding a specific location and simultaneously greeting with both hands can be considered. Classical computers will be pushed to their limits when asked to calculate the inverse kinematic, plan the path, generate motions according to the behavioral model and

simultaneously process the input stream of images.
Inverse kinematic is used in areas such as robotics, computer animation, and ergonomics. This Master Thesis defines a skeletal configuration of a human body as posture. Such a posture includes all degrees of freedom of the skeletal configuration. As humanoid robots will attempt to have as many joints as the human body in the future, this configuration will include more data than today. Comparing Quanty with the human body, the HR-OS5 has only 21 DOF, whereas the human body has up to 244 DOF. The joints of a robot like Quanty can be managed with its Next Unit Device (NUC), but the 244 joints of a human-like robot will require more computational capacity than a NUC can provide. The increasing numbers of joints are driven by the will to make a sequence of postures, also called gestures, more complex and intelligent. This Master Thesis represents such gestures in a matrix form. Highly complex tasks, or more human-like motions demand more joint motors in order to execute those movements. Inverse kinematics enables the calculation of all possible alignments of the joints of a given system when having a defined location in a room, e.g. a humanoid robot can reach this specific defined location. Having a defined location in a room, inverse kinematics calculates every possible alignments of the joints of a given system like a humanoid robot to reach this location (Aristidou & Lasenby, 2009). Perkowski (2020) already started to research the possibility of calculating inverse kinematic using quantum computing.
Nowadays, robots have either highly developed skills inherent or have a great ability of learning. ASIMO (Sakagami et al., 2002*a*; Shigemi, 2017; Sakagami et al., 2002*b*) has highly developed walking skills and conversation abilities due to the given access to powerful environmental information, but its learning abilities are restricted with regard to walking and conversation. ASIMO can perform difficult actions like kicking a ball or walking on uneven surfaces. MIT Cog (Scassellati, 2017) is a different popular humanoid robot, which uses interactions with its environment as a base of learning. As MIT Cog has only an upper body, it focuses on upper body movements, which limits its applications. Introducing quantum computing to robotics might increase the computational power of a robot to enable a robot with in-depth skills like ASIMO embodies and the learning capabilities of MIT Cog.
Robots nowadays use ML-methods like Support Vector Machine (SVM), shortest path or reinforcement learning in order to solve problems next to real time. In order to execute such an algorithm on a quantum robot, either existing algorithms must be converted into quantum or entirely new algorithms must be created. To some extent, quantum counterparts of some classical ML-methods already exist. For example, the Quantum Principle Component Analysis (Lloyd et al., 2014) and the Quantum Support Vector Machine (Rebentrost et al., 2014) provide a $O(\log(n))$ speed-up compared to the classical versions. The most famous quantum algorithms are Shor‘s algorithm to factorize large numbers and Grover algorithm for searching unsorted lists of data. Shor‘s algorithm has its highest impact in encrypting techniques but can only be used for limited applications, and Grover can be used to speed up any search problem with a quantum permutation circuit inside its oracle. Therefore, Grover has more potential applications than Shor nowadays. Another weakness of today’s robots is image processing due to

the large amount of data that needs to be processed. Al-Bayaty & Perkowski (2014) addresses this processing issue with Field Programmable Gate Array (FPGA) architectures and parallel computing.
As quantum computers cannot solve every problem in a better way than a classical computer, it can be assumed that future robots will not only include a quantum computer. This Master Thesis assumes that it will rather be the case that these robots have access to quantum computing cloud services in order to perform transformations in Hilbert-space. Therefore, classical CPUs and quantum co-processers need to work together to handle these tasks. Not only the control unit itself might be affected from quantum computing, but also the sensors and effectors. As for example, vision and image processing are computationally expensive, they often push the classical computers to their capacity limits. To solve the capacity issue, studies on quantum vision processing of two-dimensional images already exists, even though mainly in theory (Srivastava et al., 2015).
A robot must be able to process image data from several sensors and merge them into a three-dimensional point cloud in order to locate the object in its surroundings. The area of expressing three-dimensional image data by quantum representation is so far barely touched. Despite the sensors or the processing part, quantum computing will also affect kinematics and dynamics in the future robots. For instances, papers like Dereli & Köker (2019) and Sun & Ding (2010) solve problems such as inverse kinematic or trajectory planning using quantum computing. Currently, robots have trouble in informational rich environments to process all data simultaneously, therefore, it happens that the difference between the expected and the observed environment increases remarkably. To find such data with larger discrepancy, data mining is used. Petschnigg et al. (2019) uses the Grover search algorithm to find such data with larger discrepancy. Considering the search of an unstructured dataset, the quantum Grover algorithm is superior to any classical algorithm as it gives a quadratic speed up. A Grover oracle is a part of the quantum circuit which finds the wanted Boolean string representation in $\{0,1\}^n$. This Boolean function uses superposition to map the desired Boolean representation from $\{0,1\}^n$ to $\{0,1\}$ (Perkowski, 2020).
According to Dooley (2009), quantum computing has the more powerful language compared to classical binary logic. Additionally, there are three other advantages of a quantum-controlled robot such as:

- Possibility of deterministically controlled probabilistic behaviors
- Entangled behaviors are possible
- The property of *butterfly effect*

Butterfly effect describes a sensitive complex system in which small disturbances tend to have higher non-linear impacts on the overall system (Dooley, 2009).

4.3 Mood and Emotion in Robotics

Mood and emotion are often used interchangeably in literature, which is per definition wrong. In research of McNair et al. (1971) mood was defined with the help of six dimensions like anger, confusion, depression, and fatigue. In contrast to the mood, the emotion is defined in the research of Martens et al. (1990) in context with self-confidence, cognitive and somatic symptoms of state anxiety. According to Terry & Lane (2011) mood and emotion are commonly distinguished according to the following eight subjects:

- Duration
- Intentionality
- Cause
- Consequences
- Function
- Intensity
- Physiology
- Awareness

According to Lischetzke (2014), the two important properties to distinguish mood and emotion are directedness and time pattern. Moods are diffuse and unfocused, where emotions are always linked to specific objects. In Lischetzke (2014), emotions are also considered to be phasic and mood is considered to be more persistent.
Considering the definitions of mood and emotion, it can be assumed that it is best to handle them separately in quantum computing. Emotions are considered to appear instantly and last only short time spans and are more connected to certain events. Due to the relation of emotion and events, they could be represented in quantum memory. In comparison to emotion, the mood lasts longer, but is not so much connected to a specific event, which is the reason why it could be represented by an intermediate quantum state.
In the same manner as mood and emotion are divided in long-term and short-term feelings, the memory can be divided in long-term and short-term memory such as Random Access Memory (RAM) and cache. Fig. [4.1] realizes the memory in classical computing, which can separately store mood in RAM and emotion in cache.

5 Basic Quantum Algorithms for Humanoid Robots

This chapter explains several, algorithms and methods that are used with the humanoid robot in this Master Thesis. The content of this chapter is fundamental to understand the contribution of this Master Thesis in chapters 6, 7, 8, and 9.

5.1 Initialization of Quantum States

Quantum computing is facing many different limitations caused by decoherence and noise. Decoherence describes the tendency to interact with the environment. In quantum computing, any interaction with the environment, no matter if intentional such as measurement, or unintentional like decoherence, can cause the state function to collapse (Ponnath, 2006). A control gate or a measurement are theoretically considered to not cause any decoherence. Nevertheless, on a real quantum computer such operations always cause decoherence due to imperfection of the isolation. To make quantum computing communication fail error correcting codes as in classical computing will be a possibility. In classical computing the code includes additional bits that contain additional information in order to reproduce a message in case of faulty transmitted bits. Because of the Non-cloning theorem, it is not possible to copy all qubits states in order to achieve redundancy (Homeister, 2015). If initialized to the ground state, the quantum state should be exactly state $|0\rangle$, which is the lowest energy state possible. In the Bloch sphere the state $|0\rangle$ can be represented by a straight up pointing vector. In reality, due to the interaction with the environment, the state is more likely something around that perfect initialization value. When talking about environmental noise in this context, the noise stands for vibration, radiation, magnetic fields and likewise (Naihin, 2018). An initial offset from the ground state $|0\rangle$ can increase during further transformations on that qubit, as every gate introduces an additional phase uncertainty (Kak, 1999). In order to initialize a stable state, several approaches were made, one of these approaches is dynamic decoupling. This approach alters the spin of the qubit in certain intervals, in order to prevent the qubit from coupling with its environment. To prevent such a coupling, the qubit is exposed to a continuous and periodic field (Naihin, 2018). To get a better understanding of how unwanted phase shifts affect quantum computation, two one-qubit systems are considered in the following example. The first one-qubit system in the following example is an ideal system and the second one-qubit system is a real system which is exposed to decoherence. On each version of the one-qubit system a Hadamard gate is applied.

Eq. [5.1] considers an ideal system with an initial ground state $|\psi\rangle = |0\rangle$ that is transformed by a Hadamard gate.

$$H|\psi\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}} \tag{5.1}$$

Eq. [5.2] shows that in comparison to the ideal system of eq. [5.1], in a real system the initial state is not exactly the ground state $|0\rangle$, but a superpositional state around this state.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{5.2}$$

In addition to the not ideal initial state, the Hadamard gate has a faulty phase shift of θ, which leads to the final state of eq. [5.2].

$$H|\psi\rangle = \frac{e^{i\theta}}{\sqrt{2}}((\alpha+\beta)|0\rangle + (-\alpha+\beta)|1\rangle) \tag{5.3}$$

The more gates a circuit contains, the higher the depth of the circuit. Circuits with higher depths have longer processing time, which results in longer exposure time to interact with the environment. Interaction with the environment will cause higher probability of decoherence.
The models proposed in Warke et al. (2019) and Mishra et al. (2019) claim that errors result in bit flips, phase flips or in a combination of them. Fig. [5.1] illustrates these potential errors introduced in Naihin (2018).

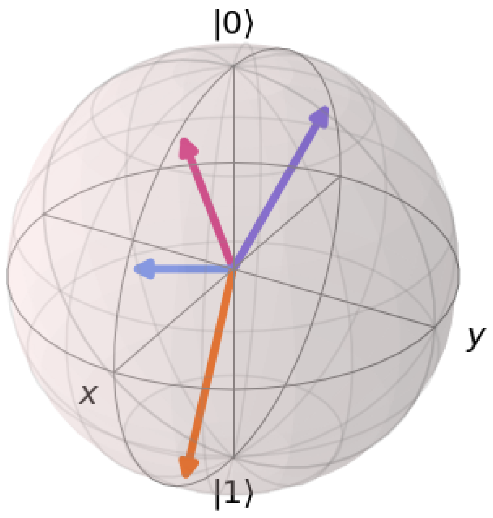


Figure 5.1: The purple vector represents the initialized state. The light blue vector represents a bit flip, whereas the phase flip is represented with the orange vector. The combination of both vectors is shown with the dark blue vector.

The initial state in Fig. [5.1] is $\frac{1}{\sqrt{5}}(2|0\rangle+|1\rangle)$. A bit flip will produce the state $\frac{1}{\sqrt{5}}(|0\rangle+2|1\rangle)$, and a phase flip $\frac{1}{\sqrt{5}}(-2|0\rangle+|1\rangle)$. The combination of both errors results in the state $\frac{1}{\sqrt{5}}(-|0\rangle+2|1\rangle)$. The algorithm of Leonard & Umesh (1999) only uses qubits that are initialized close to $|0\rangle$. With the proposed heat engine algorithm, it is tried to transform the arbitrary initial states into a quantum register, containing only $|0\rangle$ states. This method has analogies to the Carnot-process (Laranjeiras & Portela, 2016). In this analogy the temperature refers to the state of the qubit, that gets separated. At the beginning, the qubits will be sorted according to their expected polarization precomputed by a Monte Carlo simulation (Binder & Heermann, 2010). The following boosting process separates the qubits according to their state's amplitudes. For further computation only the so-called *cold* qubits will be considered (Leonard & Umesh, 1999).
Assuming that all qubits in the register are initialized with $|0\rangle$, the Superposed Quantum State Initialization Using Disjoint Prime Implicants (SQUID)-algorithm (Rosenbaum & Perkowski, 2008) transforms the ground states of the qubits to a specific state. SQUID is a process to design circuits that transform the ground state to a certain target state. There are many methods to design circuits for this initialization, due to the triviality of this SQUID method it is suitable for this explanation. The SQUID algorithm assumes that the ground state is the initial state and for further calculations a different state is needed. To use SQUID, the initial state must be initialized to the lowest energy level which is the ground state $|0\rangle^{n}$, which can be achieved with the molecular scale heat engine and scalable quantum computing algorithm. Generated by the SQUID-algorithm, the set of qubits can be divided in two set of qubits.
The first set is called the target part, and the second set is called the code part. The function of the registers is to store the initialization process data and to call the generator state. The initialization process data contains information about which terms of the target group:

- have been,
- are currently initialized and
- will be used to create more superpositions.

This algorithm uses an initialization operator on the control qubits in order to initialize the target group as needed.
The following is a short example to understand the function of SQUID. The desired state is eq. [5.4] and the initial state is the ground state $|0000\rangle$.

$$|\psi\rangle = \frac{1}{2}|0101\rangle + \frac{1}{2}|0110\rangle - \frac{1}{2}|1001\rangle - \frac{1}{2}|1010\rangle \tag{5.4}$$

Fig. [5.2] represents the state of eq. [5.4] within a phase map.

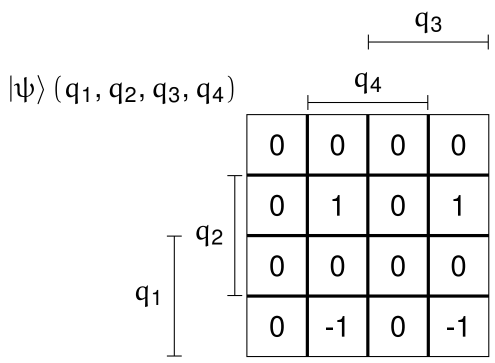


Figure 5.2: Phase map to represent the eq. [5.4]. All phase maps were created within this Master Thesis using the LaTeX packages *TikZ* and *karnaugh*.

Fig. [5.3] simplifies the phase map of Fig. [5.2] with the transformation using two CNOT-gates.

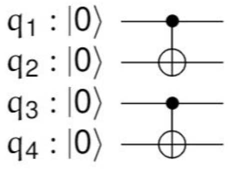


Figure 5.3: This circuit transforms the Karnaugh map (Rushdi, 1997) from Fig. [5.2] to simplify the circuit that initializes the target state.

Fig. [5.4] illustrates the new intermediate phase map after the transformation of Fig. [5.3].

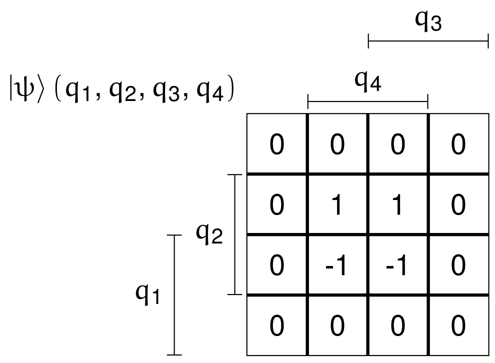


Figure 5.4: The intermediate phase map has the purpose to simplify the circuit to initialize the target state.

The phase group of the intermediate phase map from Fig. [5.4] is represented in eq. [5.5].

$$|\psi\rangle = \frac{1}{2}|0101\rangle + \frac{1}{2}|0111\rangle - \frac{1}{2}|1101\rangle - \frac{1}{2}|1111\rangle \tag{5.5}$$

Eq. [5.6] illustrates the factored form of eq. [5.5].

$$|\psi\rangle = \frac{1}{2}(|0\rangle - |1\rangle)\,|1\rangle\,(|0\rangle + |1\rangle)\,|1\rangle \tag{5.6}$$

Knowing the target state, the single steps of how this algorithm generates a quantum circuit that initializes this state will be explained. As mentioned above, the initial state must be $|0\rangle$. With $|\psi\rangle = |q_1 q_2, q_3 q_4, c_1 c_2\rangle$, *q* indicates the target group and *c* the control group, the initial state will be $|\psi\rangle = |0000, 00\rangle$. At the end of this algorithm the target group will be the target state of eq. [5.5]. Consisting of two qubits, the control group *c* can represent four different actions:

- $|00\rangle$ = no actions
- $|01\rangle$ = indicate that target group has already desired value
- $|10\rangle$ = indicate that target group is currently being initialized
- $|11\rangle$ = indicate generator mode

In the first step, Fig. [5.5] inverts both qubits from the control group to set the algorithm in generator mode $|\psi\rangle = |0000, 11\rangle$.

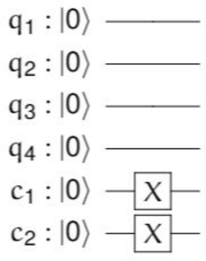


Figure 5.5: The two parallel Pauli-X gates on the control register will set the control state from *no actions* to *generator mode*.

Now the factored form of eq. [5.6] will be realized. As the state of q_2 and q_4 is $|1\rangle$, they will be transformed if the generator mode is active. Such a controlled transformation of these two qubits is done in Fig. [5.6].

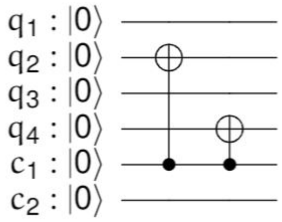


Figure 5.6: To initialize the factored form of eq. [5.6] two CNOT-gates operate on q_2 and q_4, controlled by c_1.

Considering the factored form of eq. [5.6], a CNOT-gate will transform the state of q_1. The state of q_1 in the factorized form is $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ can be generated if a Hadamard-gate operates on the state $|1\rangle$.

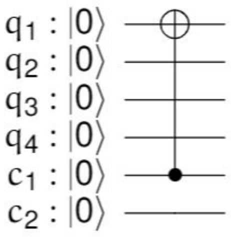


Figure 5.7: In order to generalize the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ with a Hadamard-gate, the state of q_1 has to be transformed to $|1\rangle$.

After these operations, the state is initialized with $|\psi\rangle = |1101, 11\rangle$. Following, the initialization operator from Rosenbaum & Perkowski (2008) is applied. Eq. [5.7] illustrates the matrix of this operator.

$$S_{t,g,p} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{\frac{p-g}{p}} & t\sqrt{\frac{g}{p}} \\ 0 & 0 & -t\sqrt{\frac{g}{p}} & \sqrt{\frac{p-g}{p}} \end{bmatrix} \tag{5.7}$$

With *t* being the phase that is being multiplied to every evaluation, *g* the number of cells in the group on the phase map and *p* the number of minterm that still need to be added, the initialization operator in this case is eq. [5.8].

$$S_{t,g,p} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{5.8}$$

In Boolean algebra a minterm is referred as a product term that includes every variable once. Here, *t* is assigned with 1 as no minus sign is outside the factored superposition in the factored form in eq. [5.5]. As the phase map contains four cells, g=4, further, in this example four minterms need to be added, which result in p=4.
Using the transformation of eq. [5.8] on the six-qubit system only c_1 and c_2 will be affected and result in $|\psi\rangle = |1101, 10\rangle$, which indicates that target group is currently being initialized.
Fig. [5.8] shows that the matrix of eq. [5.8] operates on the control-register, c_1 and c_2 with the S_{144}-gate.

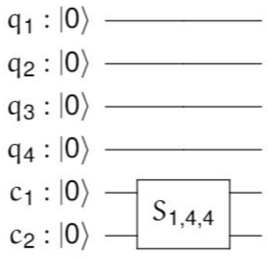


Figure 5.8: The S_{144}-gate is only applied to the control register and does not affect the target register.

Fig. [5.9] shows the decomposition of the S_{144}-gate with the transformation matrix of eq.[5.8].

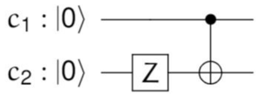


Figure 5.9: The decomposition of the transformation matrix of the S_{144}-gate is a Pauli-Z-gate and a CNOT-gate.

Now, the state has the same state as the in the intermediate phase map of Fig. [5.4]. Fig. [5.10] illustrates the next step, where the target group is set into superposition with a Hadamard-gate controlled by the two control qubits.

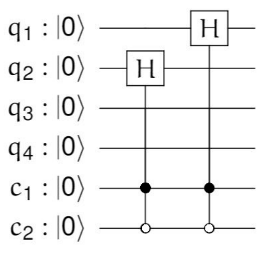


Figure 5.10: The lower control qubits represented with the black thin circle indicates that this is a $|0\rangle$-controlled qubit. With these two controlled-controlled Hadamard gates the target register is set into the target state.

After the two transformations, the state will be $|\psi\rangle = \frac{1}{2}(|0\rangle - |1\rangle)|1\rangle(|0\rangle + |1\rangle)|1, 10\rangle$. In order to initialize the target state, the intermediate phase group must be converted. Fig, [5.11] shows that to generate the target state two multi-controlled-gates are needed on the qubits q_2 and q_4.

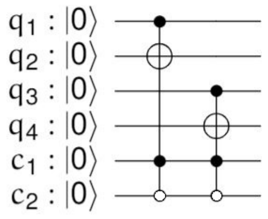


Figure 5.11: The multi-control-gates generate the intermediate phase group to the target state.

After the circuit in Fig. [5.11] the target state has been initialized and the control target has to be changed into the mode *indicate that target group has already desired value*. The swap gate in Fig. [5.12] changes the state of the control register, from $|10\rangle$ to $|01\rangle$ which indicates that the target register has already the target value.

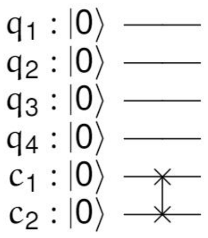


Figure 5.12: The swap-gate changes the control-status in order to indicate that the target state has been reached.

As the final state of the target register is equal to the target state, the algorithm inverts c_2 to the state $|0\rangle$ in Fig. [5.13] to change the state of the algorithm to $|\psi\rangle = \frac{1}{2}|0101, 00\rangle + \frac{1}{2}|0110, 00\rangle - \frac{1}{2}|1001, 00\rangle - \frac{1}{2}|1010, 00\rangle$.

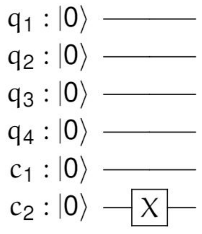


Figure 5.13: The last step of the SQUID-algorithm changes the generator state from *indicate that target group has already desired value* to *no actions* by negating the second control register qubit.

Fig. [5.14] illustrates the entire circuit of the SQUID-algorithm.

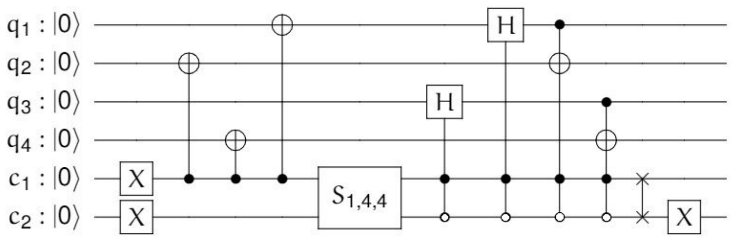


Figure 5.14: The entire circuit of the SQUID algorithm to initialize the target state as needed.

5.2 Noisy Intermediate-Scale Quantum

Handling noise in quantum computing is one of the hardest tasks nowadays. In order to perform quantum error correction, the available quantum computers have far too few qubits to offer redundant coding and it is not very likely that quantum computers in the near future will be able to provide enough qubits for error correction coding. NISQ technology is the first step towards proper quantum computing technologies that accepts the prone nature of quantum computing. These NISQ computers try to deal with noise and the associated increased probability of errors, by limiting the size of quantum circuits. Limiting the circuit size or also called depth of a circuit decreases the time a quantum bit needs to be evaluated. The shorter a qubit is within such a circuit, the less it can be affected by decoherence. As the name already indicates, NISQ computers will not be perfectly isolated from the environment which leads to imperfect control of qubits. The qubit range of such NISQ computers will be between 50 qubits and a few hundred qubits which limits the possible application range (Preskill, 2018). NISQ computers are not expected to solve practical problems better than classical computers due to the limited computational power, but they might outperform classical computers with some abstract problems. As history already showed with classical computers, a lot of useful algorithms were developed experimentally on such a computer before it was able to describe them properly. Experts hope for the same effect by making NISQ technology publicly available (Murali et al., 2019). First machine learning algorithms are already developed for NISQ technology as in Yang et al. (2019*a*).

5.3 Grover Algorithm

Grover algorithm (Grover, 1996) is used for searching unstructured data bases and is with Shor's algorithm (Shor, 1997) one of the most well-known quantum algorithms. Shor´s algorithm has exponential speed-up compared to classical algorithms but only limited possible applications like in quantum encryption (Qi et al., 2010). In comparison to this exponential speed-up, Grover algorithm provides only quadratic speed-up but has a vast amount of possible applications. The Grover algorithm is divided into three steps as follows:

- Superposition
- Negating target state
- Flipping around average

In the first step all input qubits will be set into superposition. If the entire register is in superposition all possible entries of the database are represented. If measurement is taken now, all states have the same amplitude and, thus, the same probability $\frac{1}{2^n}$, with *n* being the number of qubits in that register. In the example database eight states are included that can be represented with a three-qubit-system. Eq. [5.9] shows the probabilistic of a certain state ψ in a three-qubit system.

$$\begin{aligned} P(\psi) &= \frac{1}{2^3} = \frac{1}{8} \\ P(\psi) &= 0.125 \end{aligned} \tag{5.9}$$

Fig. [5.15] illustrates the states of the three-qubit system being in superposition.

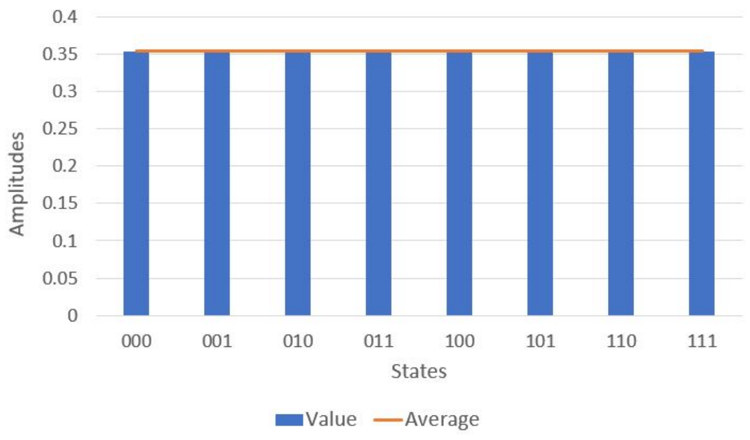


Figure 5.15: The blue bars represent the amplitudes of the particular states and the orange line represents the average value of all amplitudes. In this example the target state is $|011\rangle$. All measurement figures were created within this Master Thesis using either *Qiskit* or *Excel*.

Fig. [5.16] shows that three Hadamard gates in parallel put the three-qubit system into superposition.

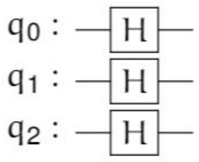


Figure 5.16: The first step *Superposition* needs three Hadamard gates in parallel.

After being set to superposition, Fig. [5.17] locates the target state in the database and negates its amplitude.

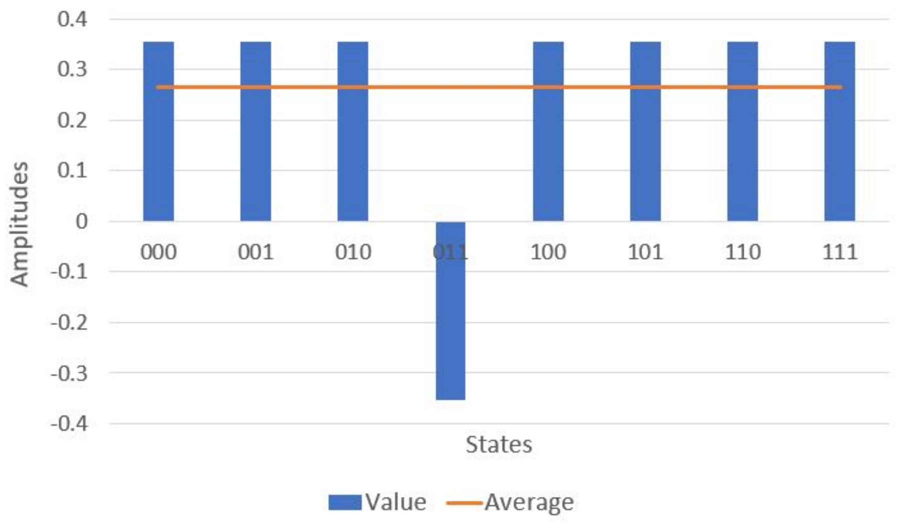


Figure 5.17: Because of the target states' negated amplitude the average value decreases.

A controlled-controlled Z-gate flips the target state $|011\rangle$. Fig. [5.18] shows that a controlled-controlled Z-gate is realized with two Hadamard gates and one Toffoli gate.

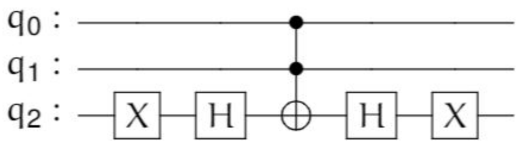


Figure 5.18: The two Hadamard gates and the Toffoli gate realize a controlled-controlled Z-Gate and the Pauli-X gate selects the target state $|011\rangle$.

The average of all amplitudes is still close to the initial amplitudes, as usually more states are in the initial superposition and only one or a few target states exist. The more qubits the circuit has, and the less states are searched, the closer is the average to the initial amplitude.
Fig. [5.17] represents the last step that mirrors all amplitudes around the average value with

the orange line. The amplitudes of the untouched states will shrink a bit, as the average lies only marginally below their values. Fig. [5.19] shows how the amplitude of the negated target state flips around average and increases approximately three times.

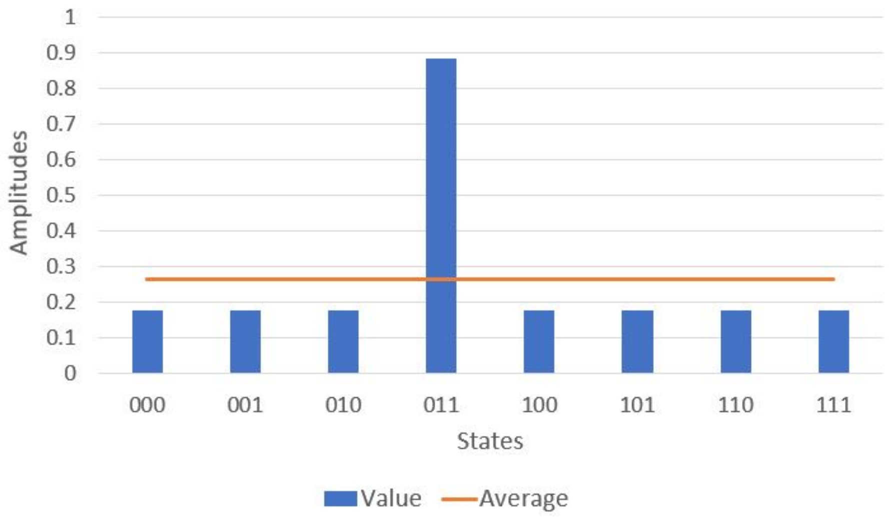


Figure 5.19: The average value in this figure is still the value of Fig. [5.17] and was not renewed calculated for demonstration purposes. The target state got flipped around the average with a high range. The residual states shrink as the average value in Fig. [5.17] was below these states.

Fig. [5.20] represents the flip around average circuit.

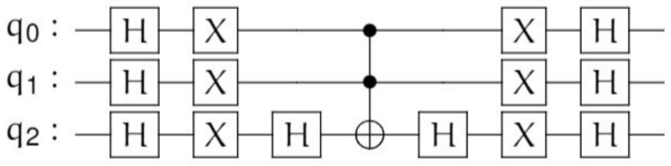


Figure 5.20: The circuits represents the *Flipping around average* steps and is independent from the target state.

Fig. [5.21] illustrates the entire circuit for the Grover search of a three-qubit system with the target state $|011\rangle$.

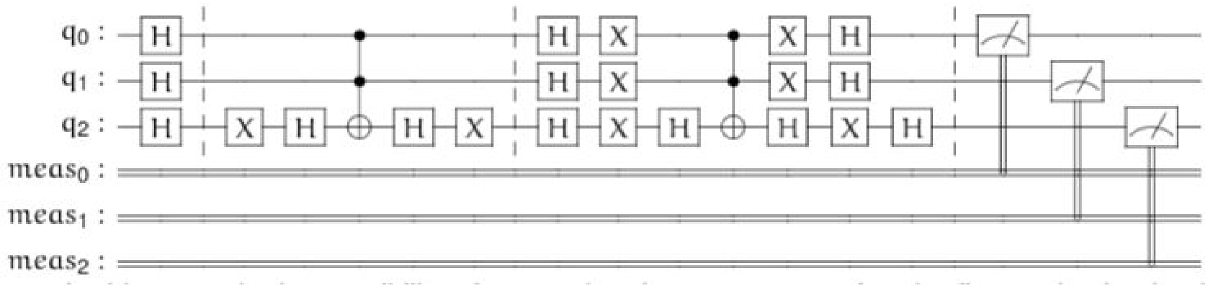


Figure 5.21: The three steps of the Grover algorithm are divided with the dashed line in this circuit. The measurement on the end of this circuit is needed to read out if the result is the target state.

Fig. [5.22] shows that the possibility of measuring the target state after the first evaluation leads to a probability of 77.6%.

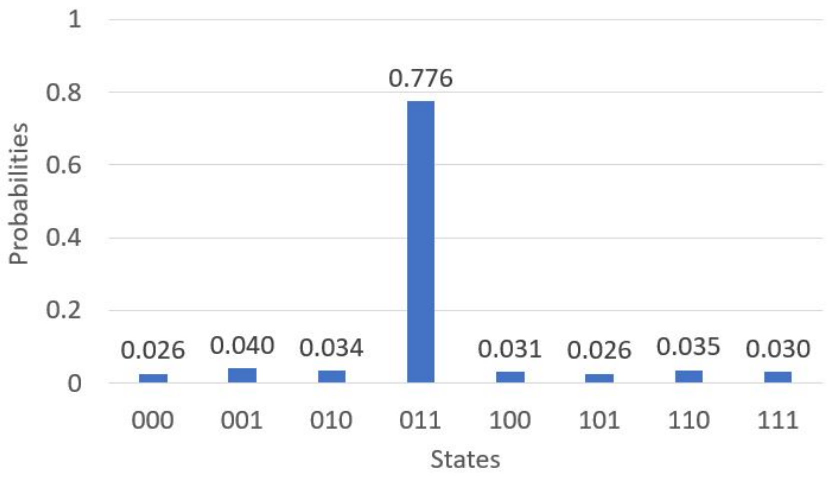


Figure 5.22: The likelihood of measuring the target state after one iteration is up to 77.6%. As the average value is very low now, a second iteration does not boost the target step with the same amount.

Fig. [5.23] shows that if the steps *Negating target state* and *Flipping around average* are repeated again, the probability of measuring the target state will increase to 93.8%.

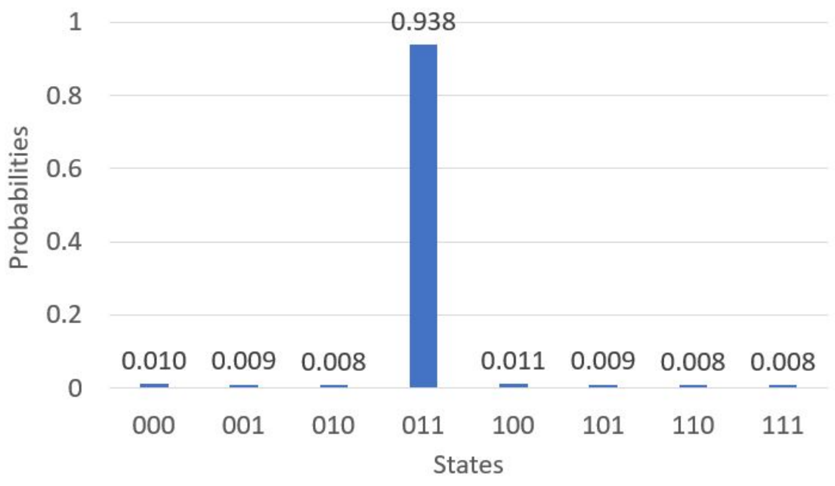


Figure 5.23: The boost of the target state in the second iteration is smaller in comparison to the boost of the first iteration as the average value is very low.

The assumption might be to repeat the steps *Negating target state* and *Flipping around average* again to reach around 100%, however, this assumption is false. Fig. [5.24] shows that after roughly more than $\sqrt{2^n}$ reputation, which is in this case approximately 2.828, of the steps *Negating target state* and *Flipping around average* the amplitudes of the target state will go down to average again and the residual state amplitudes will go up to the average value.

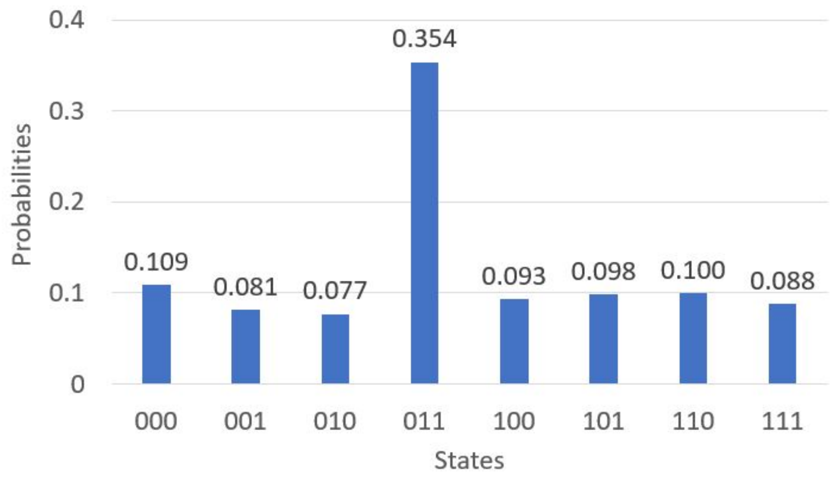


Figure 5.24: A third iteration does not lead to an increase but a decrease of the target state. Doing one more iteration, the states would be back to an equally distributed superposition.

5.4 Phase Kickback and Phase Estimation

Phase estimation uses the property that a controlled-gate transforms the control qubit if this qubit is in superposition. Subsection 3.7.2 introduced that the state of the control qubit on a multi-qubit gate decides if the transformation is executed or not. It was also stated in this subsection 3.7.2 that the control qubits' state will not be affected by that transformation. Nevertheless, in case of a control qubit in superposition, the transformation does have an effect on the control qubit.

Phase estimation has applications when more information about a unitary gate *U* is needed as the transformation matrix of this gate is unknown. In order to get an approximated solution to eq. [5.10], a more detailed description about the eigenphase ϕ is necessary.

$$U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle \tag{5.10}$$

As an example, for phase estimation, the control qubit is initially in the state of $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$. After passing the gate *U*, the state of the control qubit will be changed to $\frac{|0\rangle+e^{2\pi i\phi}|1\rangle}{\sqrt{2}}$. In this process, the phase ϕ of the transformation *U* gets kick-backed on the control qubit.

In order to obtain more information about the phase ϕ of the transformation *U*, three additional resources are needed:

- an extra ancilla qubit
- a Hadamard gate on the ancilla qubit
- a controlled-unitary gate with unknown transformation matrix

Fig. [5.25] illustrates a trivial circuit for the phase estimation.

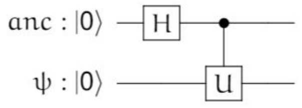


Figure 5.25: A trivial circuit for the phase estimation is a two-qubit system with a Hadamard gate and a controlled operation. This circuit looks very similar to the EPR-circuit.

The state ψ and the ancilla qubit *anc* are both initialized with the ground state $|0\rangle$. The circuit in Fig. [5.25] results in an output state of eq. [5.11].

$$C(U)H \otimes I|0\rangle|\psi\rangle = \frac{|0\rangle + e^{i\phi}|1\rangle}{\sqrt{2}}|\psi\rangle \tag{5.11}$$

Eq. [5.11] shows that the state $|\psi\rangle$ is not changed by the *U* gate. If applying a controlled-phase gate as controlled-unitary gate and the eigenstate of ψ is $|1\rangle$, the eq. [5.11] transforms to eq. [5.12].

$$C_\phi H \otimes I|0\rangle|\psi\rangle = \frac{|0\rangle + e^{i\phi}|1\rangle}{\sqrt{2}}|1\rangle \tag{5.12}$$

The matrix in eq. [5.13] represents the controlled-phase gate.

$$C_\phi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{bmatrix} \tag{5.13}$$

To gain more specific information about ϕ, the observable quantities of the general Pauli-rotations σ_x, σ_y, and σ_z of the ancilla qubit are measured several times.

- $\sigma_x = \cos\phi$
- $\sigma_y = \sin\phi$
- $\sigma_z = 0$

5.5 Classical and Quantum Fourier Transform

This section explains the main concept of the classical Fourier transform first in order to have a solid basis for the Quantum Fourier Transform (QFT).

5.5.1 Fourier Transform

For some problems like data compression, it is easier to find a solution on a different basis. Such a change of basis is also used in spectral transforms by changing the considered domain. The problem is transferred into a different space, solved there and being transferred back into the original space again. There are different methods of spectral transforms like Walsh, Reed-Muller, Haar, or Fourier transform (Thornton et al., 2001). The Fourier transform is used in areas like data compression or signal processing. In order to understand signals like soundwaves better, these signals must be decomposed for the further process steps. A potential application of Fourier transform is to denoise signals. With known frequencies of white noise, the noise can be extracted from the original input signal. In Fourier analysis, these decompositions split the initial periodical signal into a block of sinusoidal functions. This transformation unfolds the original frequency spectrum and the original time dependent signal becomes a frequency dependent signal. When the function is dependent on the frequency, it is no longer possible to distinguish when the noise occurred but with which frequency.
After being already transformed from the original basis to the new basis, only a certain amount of points will be considered during reconstruction of the original function (Müller, 2015) in Discrete Fourier transformation (DFT).

5.5.2 Quantum Fourier Transform

QFT is the quantum implementation of the classical DFT and is fundamental for phase estimation. As discussed in subsection 5.5.1, the DFT maps a function depending on time to a different function depending on frequency. QFT acts in a similar way and maps the input quantum state $\sum_{i=0}^{N-1} x_i |i\rangle$ to the quantum state $\sum_{i=0}^{N-1} y_i |i\rangle$. Eq. [5.14] describe this mapping and eq. [5.15] describes it using a unitary matrix.

$$|x\rangle \longrightarrow \frac{1}{\sqrt{N}\sum_{y=0}^{N-1} w_N^{xy} |y\rangle} \tag{5.14}$$

$$U_{QFT} = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} w_N^{xy} |y\rangle \langle x| \tag{5.15}$$

In Fourier basis, numbers are represented by using different rotations around the Z-axis. The Hadamard-gate can be considered as a single-qubit QFT. With the input state of $|0\rangle$, the Hadamard-gate transforms the input state from the computational Z-basis to the X-basis. The Z-basis is represented with the basis states $|0\rangle$ and $|1\rangle$ and the X-basis is represented with the basis states $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The main goal of the QFT is to map a certain data representation from one basis to a different basis to make certain data observable in this new basis. A practical application of QFT is phase estimation, where the transposed QFT is used to transform the states from the computational X-basis to the computational Z-basis to make it observable. With the switch of the basis from X-basis to Z-basis the data representation will change from $|+\rangle$ and $|-\rangle$ basis into a $|0\rangle$ and $|1\rangle$ basis. As the data representation in $|+\rangle$ and $|-\rangle$ basis cannot be measured, it must be transformed into an observable state. The implementation cost of such a QFT circuit is lower than the representation of a normal DFA but the information that can be used with a QFT are restricted. Fig. [5.26] shows a QFT-circuit implementation.

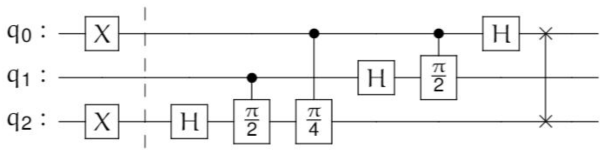


Figure 5.26: The first part divided with the dashed line initializes the circuit. The circuit transforms the state $|101\rangle$ from the computational Z-basis into the computational X-basis.

The Hadamard gate after the initialization part transforms the qubit q_2 in the needed superposition. The output state is in reversed order, therefore, q_0 and q_2 must be swapped to have the same order as the input state. Fig. [5.27] shows the transformation from the Z-basis to the X-basis.

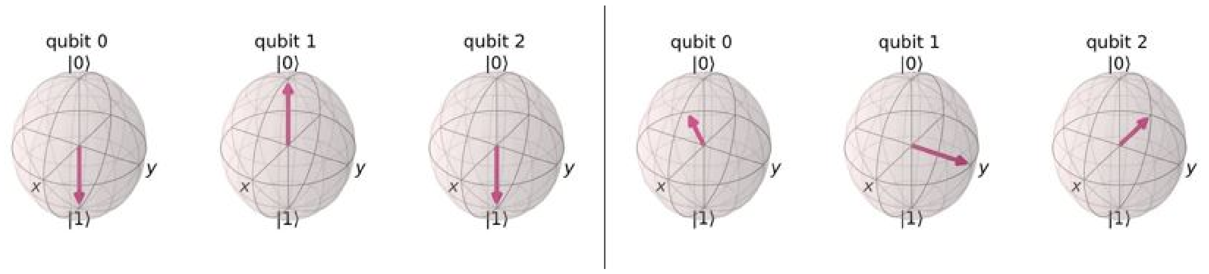


Figure 5.27: The three qubits on the left side are the representation of the three-qubit system of Fig. [5.26] after being initialized. The three qubits on the right represent the mapped qubit state in the computational Z-basis.

5.6 Quantum Algorithm for Linear Systems of Equations

Systems of linear equations affect many real-life applications such as the solution of partial differential equations, the calibration of financial models, or fluids simulations. Least-Squares Support-Vector Machine (LS-SVM) translates the optimization problem into a set of linear equations in order to speed up classical SVMs, and QSVM can speed up the calculation of these linear equations even more (Asfaw et al., 2020). In general, everything a quantum computer does, can be represented with a set of linear equations. The computational cost to solve these linear equations is growing rapidly over the size of the matrix. On a classical computer, the minimal time to solve a set of linear equations is minimum *N*, where *N* represents the size of the dataset. Just to write down the solution takes time of order *N*, whereas the quantum solution can approximate the value of such a linear eq. with a polylogarithmic dependency on *N*. As mentioned before, the whole solution does not always matter. With the method of quantum algorithm for linear systems of equations, just a particular subset of indices will be considered. When it comes to solving linear equations, exact information about the solution is not necessarily needed, as an approximation of the solution or only a part of the solution is good enough. The Harrow-Hassidim-Lloyd (HHL)-algorithm does not return the entire solution but approximates features of the solution vector. Depending on the application, different features like normalization, certain rations, or moments must be extracted and not the entire solution is needed.

The HHL algorithm tries to solve the equation $A\vec{x} = \vec{b}$, with given matrix $A \in \mathbb{C}^{N*N}$ and vector $\vec{b} \in \mathbb{C}^{N}$, and an unknown vector $\vec{x} \in \mathbb{C}^{N}$. To solve this equation, matrix *A* must be inverted to get on the same side as $\vec{b}$. Assume $\vec{b}$ has *N* entries. Eq [5.16] shows the first step of the quantum algorithm that is to map $\vec{b}$ to a quantum state $|b\rangle$ and *A* to a suitable quantum operator.

$$A\vec{x} = \vec{b} \longrightarrow A\,|x\rangle = |b\rangle \tag{5.16}$$

The key requirement is to perform operations in superposition. It is assumed that matrix *A* is s-sparse and Hermitian. A matrix is s-sparse if it has at most *s* non-zero entries per row or column. If matrix *A* is s-sparse the matrix inversion can be realized cost efficient as $e^{iAt}\,|b\rangle$. Starting off with the normalization and mapping of vector $\vec{b}$ to quantum state $|b\rangle$. Vector $\vec{b}$ needs to be decomposed in the eigenvector basis of *A*, using phase estimation. It is assumed that $\vec{b}$ is an Eigenvector of *A* in order to save the basis transformation and additional computation. Eq. [5.17] shows the spectral decomposition of the Hermitian matrix *A*.

$$A = \sum_{j=0}^{N-1} \lambda_j \left|u_j\right\rangle \left\langle u_j\right|, \quad \lambda_j \in \mathbb{R}. \tag{5.17}$$

Eq. [5.18] decomposes the matrix *A* to get the unitary operator e^{iAt}.

$$\begin{aligned} A &= \begin{bmatrix} a+b & c-id \\ c+id & a-b \end{bmatrix} \\ &= \begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix} \begin{bmatrix} 0 & c \\ c & 0 \end{bmatrix} \begin{bmatrix} 0 & -id \\ id & 0 \end{bmatrix} \begin{bmatrix} b & 0 \\ 0 & -b \end{bmatrix} \\ &= aI_2 + c\sigma_x + d\sigma_y + b\sigma_z \end{aligned} \tag{5.18}$$

The matrices σ_x, σ_y and σ_z are Pauli matrices as introduced in subsection 3.7.1. The inverted matrix *A* is on the right-hand side of the eq. [5.16], which leads to eq. [5.19].

$$A^{-1} = \sum_{j=0}^{N-1} \lambda_j^{-1} \left|u_j\right\rangle \left\langle u_j\right|, \quad \lambda_j \in \mathbb{R}. \tag{5.19}$$

As mentioned above $|b\rangle$ is assumed to be an eigenvector of matrix *A*. Therefore, eq. [5.20] shows that $|b\rangle$ can be expressed with the eigenvectors of matrix *A*.

$$|b\rangle = \sum_{j=0}^{N-1} b_j \left|u_j\right\rangle, \quad b_j \in \mathbb{C} \tag{5.20}$$

The solution of the initial problem in eq. [5.16] can be therefore rewritten as in eq. [5.21].

$$|x\rangle = A^{-1}|b\rangle = \sum_{j=0}^{N-1} \lambda_j^{-1} b_j \left|u_j\right\rangle \tag{5.21}$$

Fig. [5.28] shows the quantum circuit implementation of the phase estimation.

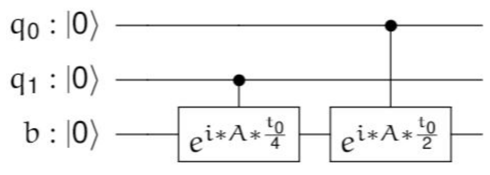


Figure 5.28: The matrix inversion is implemented in Qiskit using two controlled operators on qubit q_0 and q_1. The target register is $|b\rangle$ and the unitary operator has a form of e^{iAt}.

This phase estimation is the dominant source of errors in this algorithm. The algorithm maps the *N* entries of $\vec{b}$ onto $\log_2 N$ qubits. Fig. [5.28] shows that the Hermitian matrix *A* will be transformed into a unitary operator e^{iAt} in order to execute the phase estimation, which is possible for an s-sparse matrix *A* in time *t*, represented in eq. [5.22].

$$\tilde{O}(\log(N)s^2t). \tag{5.22}$$

With the HHL-algorithm, a quantum-mechanical representation $|x\rangle$ of the desired $\vec{x}$ is obtained. As mentioned above, to read out all entries requires at least *N* steps. As one is often only interested in some expectation value, $\vec{x}$ can be further decomposed into $\vec{x}^T M \vec{x}$, with *M* being some linear operator. Mapping also the matrix *M* to a quantum-mechanical operator the classical representation of $\vec{x}^T M \vec{x}$ is turned into $\langle x|^T M |x\rangle$. With this step, a wide variety of vector $\vec{x}$ features can be extracted, including normalization, weights in different parts of the state moments. Important numbers of the HHL-algorithm are κ and ϵ, as eq. [5.23] shows κ describes the ratio between the largest and the smallest eigenvalues of matrix *A* and ϵ is the allowed additive error in the output state $|x\rangle$.

$$\kappa = \frac{max_j|\lambda_j|}{min_j|\lambda_j|} \tag{5.23}$$

With bigger κ the matrix *A* is more unlikely to be inverted, which leads to a more unstable solution. If κ and $\frac{1}{\epsilon}$ are both poly log(N), the runtime will be poly log(N) (Harrow et al., 2009). Fig. [5.29] displays a trivial circuit example of the HHL algorithm, which can solve a 2 by 2 linear system.

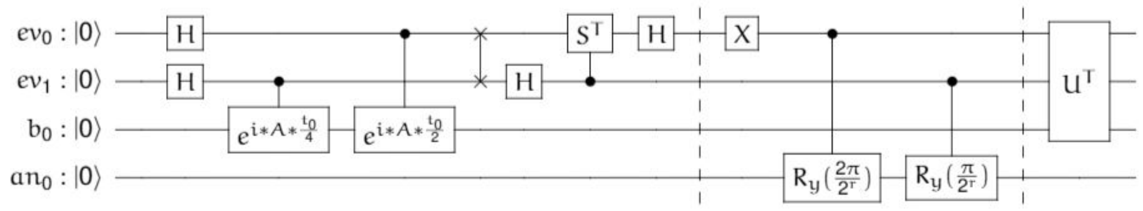


Figure 5.29: The gate U^T represents all mirrored operations from the first part of the circuit.

In Fig. [5.29] the dashed lines divide the circuit into three parts. First part does the phase estimation and the third part is its inverse operation. The second part, also called the controlled rotation, saves the eigenvalues λ_j to the probability amplitudes.

5.7 Classical and Quantum Support Vector Machine

This section introduces the QSVM. In order to understand the main concept, the classical SVM will be introduced beforehand in order to see the methodology of this algorithm.

5.7.1 Classical Support Vector Machine

SVM (Cortes & Vapnik, 1995) has application in text categorization, bioinformatics or image recognition (Ccoicca, 2013). As SVM belongs to the supervised learning category, data and

the respecting binary label are together during the learning process. During the training, the SVM tries to separate two classes of data with a hyperplane of maximum margin. This happens either with trivial separable data in the original feature space or in more complex cases in a higher dimensional kernel space. The nearest vectors to this hyperplane on both sides are called support vectors.
If the dataset is not separable linearly, the data will be mapped with a non-linear kernel from the initial space into a feature space to be separated. A soft margin allows misclassifications upon a certain value. When the misclassification values exceed the threshold value the model gets recalculated (Rebentrost et al., 2014). A main disadvantage of classical SVM is the scalability with increasing training data, which is referred to as quadratic programming problem.
To enhance the classical SVM and solve this quadratic programming problem cost efficient, Suykens & Vandewalle (1999) focused on finding the classifier by solving sets of linear equations, instead of quadratic programming. In order to reduce the computational complexity, the equality constraint method is used. In LS-SVM the support vectors with smaller values are not considered in the kernel computation. In the process of finding the support vectors, the closest vectors to separating hyperplane get weighted the most. To improve computation, vectors with small weights will separated and not considered (Yongsheng Sang et al., 2008).

5.7.2 Quantum Support Vector Machine

It takes a classical SVM up to $O(\log(\epsilon^{-1}poly(N, M))$ time to solve a problem, with ϵ representing the accuracy, *N* the dimensionality of the feature-space and *M* the number of training samples. In comparison to that, QSVM has in the best case a speed-up to $O((N, M))$ Rebentrost et al. (2014). In Havlíček et al. (2019) two methods are represented how quantum computing can attribute to classical SVM. The first method uses a variational circuit to generate a hyperplane in a quantum feature map. The second method estimates the kernel function of the quantum feature space directly and implements a conventional SVM. So far, both methods were only used on fabricated data (Havlíček et al., 2019).
SVM is a ML-method which produces a nonlinear decision boundary in a feature space by constructing linear boundaries in a transformed Hilbert-space. Executed on a classical computer, these algorithms do not scale well with the size of feature space in terms of data points and dimensionality. One of the most significant limitations of classical algorithms using non-linear kernels is that the kernel function must be evaluated for all pairs of input feature vectors. When these feature vectors are of higher dimensions, the computational power of a classical computer tend to be exceeded. The hyperplane is separating observations of a *n*-dimensional space with *(n-1)*-dimensions, for instances, data in a two-dimensional space will be separated with a line. Eq. [5.24] defines the hyperplane by a *n*-dimensional weight vector $W = (W_1, W_2, ..., W_P)$ and the bias parameter *b*.

$$WX + b = 0 \tag{5.24}$$

Moreover, the hyperplane is used to separate data $X^i = (X^i_1, X^i_2, ..., X^i_p)$. Eq. [5.25] classifies the label of each datapoint y^i by either being above or below zero.

$$y^i = \text{SIGN}[WX^i + b] \tag{5.25}$$

The sign-function results in one of the following:

$$SIGN[WX^i + b] := \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

Despite the assigned label, the datapoints of both classes must satisfy eq. [5.26].

$$y^i[WX^i + b] \geq 0 \tag{5.26}$$

The goal of SVM and QSVM is to choose the parameter *W* and *b* to obtain the best separation of both classes, in other words, to get the biggest margin between the support vectors and the hyperplane. If new data is added after the hyperplane is defined, so-called test data, it should be classified correctly. Eq. [5.27] defines the perpendicular distance of a support vector to the constructed hyperplane.

$$y^i[WX^i + b] = M^i \tag{5.27}$$

It is not always possible to separate two classes, only searching for the maximum margin between the data points. Therefore, for some data it is allowed to fall into the space between the margin and the hyperplane or even being misclassified. The error terms ϵ_i is assigned to these misclassified vectors. In order to control the error rate of the hyperplane, eq. [5.28] defines that the sum of these error terms ϵ_i has a limit *K*.

$$\sum_i \epsilon_i \leq K \tag{5.28}$$

The vector *W* is normal to the constructed hyperplane. Eq. [5.29] shows the relation of margin *M* and *W*.

$$M = |W|^{-1} \tag{5.29}$$

Eq. [5.30] increases *M* with minimizing *W*.

$$Min_{W,b,\epsilon}|W \begin{cases} y^i[WX^i + b] \geq (1 - \epsilon_i), & \forall i \\ \sum_{i=0}^{N} \epsilon_i \leq K, & \text{if } \epsilon_i \geq 0 \end{cases} \tag{5.30}$$

With *C* having a control effect of the error terms coming from the support vectors, function eq. [5.30] can be rewritten as eq. [5.31] and eq. [5.32].

$$Min_{W,b,\epsilon}[\frac{1}{2}|W|^2 + C\sum_{i=1}^{N} \epsilon_i] \tag{5.31}$$

$$y^i[WX^i + b] \geq (1 - \epsilon_i), \quad \epsilon_i \geq 0, \quad \forall i \tag{5.32}$$

Using the Lagrange multiplier (Bertsekas, 1976) method with eq. [5.32] being the constraint and eq. [5.31] the function to be optimized, eq. [5.33] maximizes the margin by setting the derivative of the three input variables *W*, *b*, and ϵ_i to zero.

$$\begin{aligned} \frac{\partial \mathcal{L}}{W} &= 0, \\ \frac{\partial \mathcal{L}}{b} &= 0, \\ \frac{\partial \mathcal{L}}{\epsilon_i} &= 0. \end{aligned} \tag{5.33}$$

A kernel is a type of similarity measure between two observations and can be represented in the simple linear case with eq. [5.34].

$$K(X^i, X^j) = X^*, X^j \tag{5.34}$$

The classical dual problem takes $O(log(1/\epsilon)N^2(P+N))$ computational time. This can be improved by a quantum approach, with *n* being the number of qubits, the observations can be represented with $P = 2^n$. By using the HHL-algorithm, the runtime for the quantum support vector classifier method becomes O(log(PN)).
QSVM simplifies the calculation of the hyperplanes parameter by converting the quadratic programming problem in a problem of solving linear equations. To condense, the methodology of QSVM can be divided into four general steps.

1. Preparing the input state $|b\rangle$
2. Executing the quantum phase estimation which results in entanglement
3. Saving the eigenvalues λ of matrix *W* with a controlled rotation process
4. Read out the results

The QSVM will be explained in more detail in the subsection 9.1.2 using quantum circuits on a specific example.

6 Quantum Motions Generator

This chapter extends the results of Deng (2020) to increase their application and generalizing its approach. The chapter starts with a summary of Deng (2020) achievements and continues with the research of this Master Thesis.

6.1 Pre-Project Quantum Motion Generation

Deng (2020) designed a quantum circuit based on EPR-method, consisting of several Hadamard and Toffoli gates. Three types of circuits where compared to each other:

- Deterministic logic circuit
- Quantum circuit with classical probability
- Quantum circuit with quantum probability

A deterministic circuit always maps one input state to one output state. For example, Fig. [6.1] shows a classical inverter gate that is deterministic as it transforms one specific input state to one specific output state.

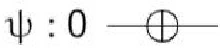


Figure 6.1: The same input state results in the same output state and this property makes the inverter gate deterministic.

Measuring quantum circuits with n-qubits and classical probability, the measurement will be one of all possible output states with the probability of $\frac{1}{2^n}$. Fig. [6.2] illustrates a two-qubit system with parallel Hadamard gates.

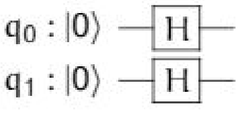


Figure 6.2: The result of two Hadamard in parallel has classical probability behavior.

Fig. [6.3] shows a circuit with quantum probability behavior which will result in two of the four possible Bell states (Bell, 1964).

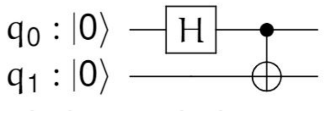


Figure 6.3: The EPR-circuit results in quantum probabilistic, which always includes entanglement.

With the circuit of Fig. [6.3], the movements executed by the robot are neither deterministic nor classical probabilistic but quantum probabilistic. This quantum probabilistic behavior includes entanglement and results in one of the four Bell states.
The motions of the robot HR-OS5 are stored in motion matrices. In these motion matrices, columns correspond to the different time steps and each row defines a certain motor. Table [6.1] is a small example how such a motion matrix can be designed.

Table 6.1: The table shows the angle of both motors for the left and right shoulder over two timesteps. Reaching the angles in column *t_2* the gesture is finished.

	Time steps	
Motor	t1	t2
Left shoulder	90°	120°
Right shoulder	90°	60°

This table represents two motors controlling the right and left shoulder of a robot. In the example motion matrix of Tab. [6.1] the gesture consists of two postures in two-time steps, t_1 and t_2. A posture starts at a time t_x and ends with the next timestep t_{x+1}. In the example motion matrix of Tab. [6.1], both motors start at 90° degrees at t_1 and rotate in opposite directions in t_2. Considering only column t_1, the two motors are arranged in a so-called posture. The first posture ends when the motors start moving from 90° degrees oppositional. The overall sequence of such postures, starting from the initial posture, in this case t_1 and ending with a final posture, in this case t_2 is called a gesture. In (Deng, 2020) a method was proposed of having one qubit for each of the seven basic emotions and one qubit for each DOF. An analogy between music sheet notation and quantum circuits was used to map music to motion. The methodology of the music to motion mapping starts with classifying musical tones in three sections regarding the pitch. Three sections were chosen because each arm consists of three different servomotors as follows:

- Shoulder,
- Elbow,
- Wrist.

Each section belongs to one motor. Depending on the pitch, a Hadamard or Toffoli gate is applied to the qubit of the corresponding motor. After the tones were mapped and the gates were applied to the corresponding qubit, the circuit is measured. In quantum probabilistic circuits, Toffoli gates were applied in addition to the Hadamard gates to create entanglement, which results in a quantum probabilistic measurement. The measured states were decoded to the servo position and sent to the corresponding motor in order to execute the motion (Deng, 2020).

6.2 Extension of Pre-Project Quantum Motion Generation

This Master Thesis extended the results and achievements of Deng (2020), in order to generalize the quantum motion generation. As a result quantum circuit were designed that uses quantum effects like entanglement and superposition, to generate deterministic, classical probabilistic, quantum probabilistic and mixed probabilistic behavior.
Two examples are discussed that show how to use quantum effects like superposition or entanglement to generate these four different motion behaviors. Fig. [6.4] shows the EPR-circuit that represents the first circuit, in which qubit q_0 will control the left shoulder joint and qubit q_1 the right shoulder.

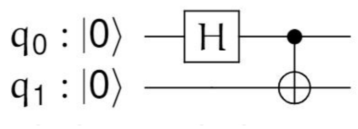


Figure 6.4: In this circuit, q_0 controls the left shoulder motor and qubit q_1 controls the right shoulder motor. The output state for this setup will be either $|00\rangle$ or $|11\rangle$, which means both shoulders are active or none.

Quanty interprets the different output states of the circuit from Fig. [6.4] as different greeting motions such as:

- $|00\rangle$ -> both arms are down
- $|01\rangle$ -> only left arm is up
- $|10\rangle$ -> only right arm is up
- $|11\rangle$ -> both arms are down

The Python library Qiskit executes the circuit either as a simulation or on a real quantum computer from IBM. Fig. [6.5] and Fig. [6.6] show the results of the evaluation in the simulation and on the real quantum computer.

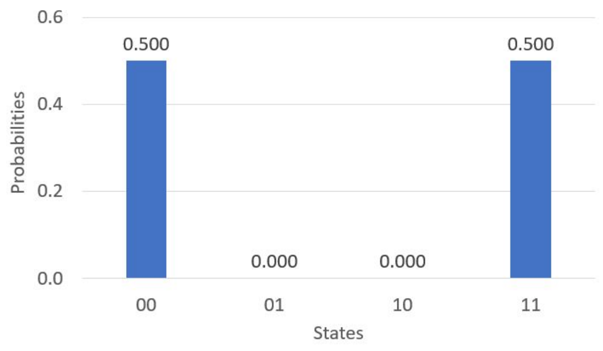


Figure 6.5: Results of the circuit Fig. [6.4] in the simulation. The simulation does not consider noise, and the only two equally distributed solutions are $|00\rangle$ or $|11\rangle$.

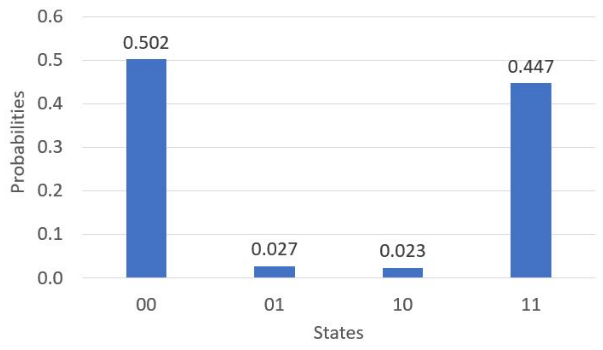


Figure 6.6: This figure shows the results of the circuit from Fig. [6.4] of a real quantum computer. Theoretically, the only possible solutions should be either $|00\rangle$ or $|11\rangle$. As the noise of a quantum computer is not considered in the calculation, the results on the real machine differ slightly.

In Fig. [6.5] and Fig. [6.6] the results of a simulation and a real quantum computer are illustrated. The differences in the results are due to the noise on the real quantum computer which was not included in the simulation. Fig. [6.6] shows that a real quantum computer as backend is exposed to noise, therefore, the results are not only $|00\rangle$ or $|11\rangle$ but also $|01\rangle$ and $|10\rangle$.
Theoretically, the circuit in Fig. [6.4] should only have two output states $|00\rangle$ and $|11\rangle$, which would result in the greeting motions *both arms are down* and *both arms are up*. However, Fig. [6.6] shows that executed on real quantum computer, there is also a possibility that the final state results in $|01\rangle$ or $|10\rangle$, which will result in the greeting behavior *only left arm is up* and *only right arm is up*.

Fig. [6.7] illustrates a two-qubit system resulting in classical probability behavior.

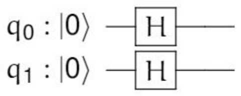


Figure 6.7: In theory two Hadamard gates in parallel result in any possible output state with the probability of $\frac{1}{2^n}$, which in this case is $\frac{1}{4}$.

This circuit will not result in entanglement, but in superposition and its output is equally distributed over every possible output state $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$.
All four possible output states of the circuit from Fig. [6.7] are equally distributed in the simulation with the probability of 0.25. However, Fig. [6.8] shows that if a real quantum computer evaluates this circuit, the probabilities are not equivalent distributed.

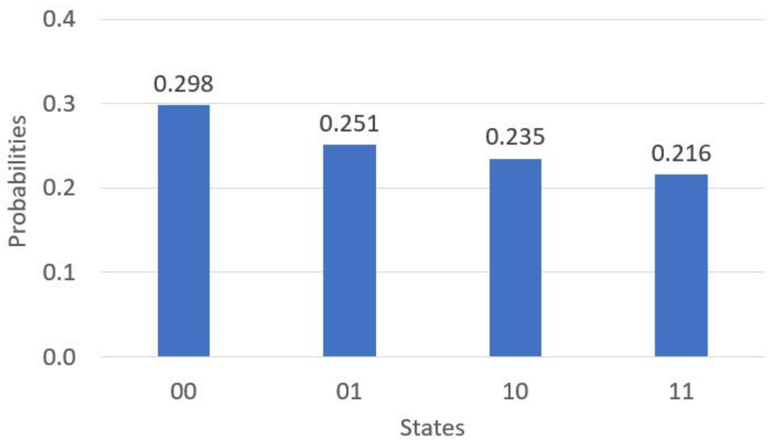


Figure 6.8: This figure illustrates the results of the circuit from Fig. [6.7] of a real quantum computer. The probabilities are not equally distributed as the qubits are exposed to noise.

Fig. [6.8] shows that only in theory, every result is distributed equally. When executed on a quantum computer, the results differ slightly. With the classical probabilistic circuit the results can be interpreted as before with the circuit with quantum probability behavior of Fig. [6.4].

- $|00\rangle$ -> both arms are down
- $|01\rangle$ -> only left arm is up
- $|10\rangle$ -> only right arm is up
- $|11\rangle$ -> both arms are down

If every joint of Quanty is controlled with either one of this circuits, the quantum circuits with quantum probabilistic and classical probabilistic can generate a huge amount of different gestures. The mood and emotion state is the control register of the Hadamard gate, which results in different variety of gestures. Therefore, the same input can imitate different motion behaviors. The greeting behavior can be realized with lifting one arm, both arms and none arms. In the circuit with classical probabilistic behavior, each output state has the same probability and so does each greeting gesture. The circuit of Fig. [6.6] can be adjusted in order to shift the probability towards one particular arm. If the robot is left-handed, the probabilities can be shifted to the output state of $|01\rangle$ by adapting the circuit, see section 7.3.

6.3 Quantum Motion Controlled by Internal State

In Raghuvanshi et al. (2007) a quantum version of the classical Braitenberg vehicle (Braitenberg, 1986) was defined. Fig. [6.9] illustrates the original version of the Braitenberg vehicle is a mobile robot consisting of two sensors, two motors driving a wheel each and a flying wheel in front.

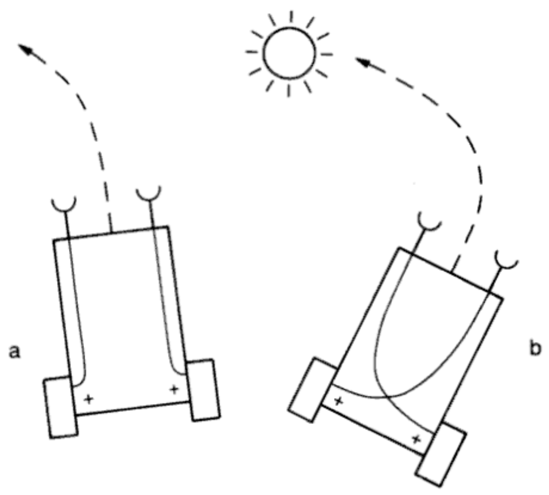


Figure 6.9: Changing the wiring designs two behavioral models. The vehicle on the left marked with *a* expresses a shy behavior as it drives away from the light source. Having the same mobile robot but different wiring results in an aggressive behavior as the right mobile robot will drive towards the light source (Braitenberg, 1986).

The main idea of the Braitenberg vehicle is to mimic complex behavior like fear or shyness with simple methods. Programming a robot that expresses emotions does not necessarily result in greater computational cost. Using genetic algorithms or convolutional nets to create a behavioral model demands higher computational cost. The concept of the Braitenberg vehicle is to imitate feelings with trivial approaches like changing the sensor-servo-mapping.
On the left vehicle in Fig. [6.9] the left motor is connected to the left sensor, likewise on the right side the sensor is connected with the servomotor on the same side. Depending on how much light a sensor detects, it sends a signal to the connected motor to start moving the wheel forward. If a sensor measures more light, the connected motor accelerates. If the sensors

are connected to the servomotors on the same side, the vehicle expresses shy behavior, as it drives away from the light sensor. The version of the Braitenberg vehicle expressing aggressive behavior is wired crisscrossed.
When sensor and motor are connected on the same side, the vehicle represents a shy behavior as the wheel closer to the light source rotates faster and the Braitenberg vehicle will turn away from the source. When connected crisscrossed, the vehicle will turn towards the source which expresses aggressive behavior. An extension of the classical Braitenberg vehicle would be a fuzzy controlled Braitenberg vehicle (Raghuvanshi et al., 2007), in which the sensors will send signals to the motors commensurate to the amount of light that is shining onto the sensor.
Extending the idea of the fuzzy controlled Braitenberg vehicle further, the sensors are assumed to be quantum, with input $|00\rangle$. The first qubit is related to the left sensor and the second qubit to the right sensor. When using quantum phenomena like entanglement and superposition, it is possible to express even more complicated behavior. Sensing no light at all, the robot will either stand still $|00\rangle$ or move forward $|11\rangle$. The quantum mobile robot can be described with two qubits for the motors and one control bit for the mood as can be seen in Fig. [6.10].

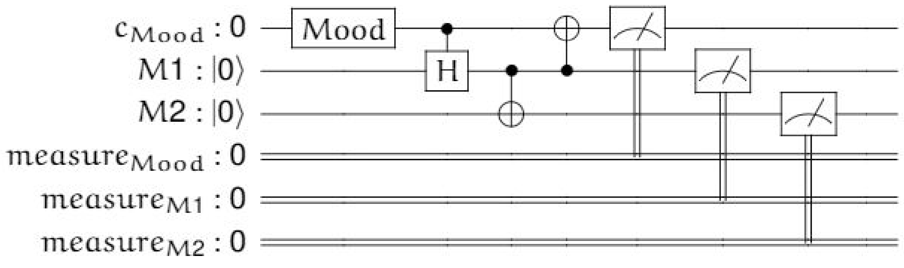


Figure 6.10: In this circuit the *Mood* gate represents an arbitrary operation that affects the expressed behavior of the robot by changing the state of the motors input signals *M1* and *M2*. The rightmost three gates represent a measurement operation. For this measurement, the state of the qubit is mapped onto the classical register represented with the notation *measure*$_{Mx}$.

In example circuit of Fig. [6.10] the servomotors are classical and not quantum. In order to measure the results and interpret them to control the servomotors, the quantum register must be measured with a classical register. Fig. [6.10] illustrates the classical bits with double lines.

6.4 Motion Creation by Hand

At the beginning of this Master Thesis, the motions were created by hand to get a better feeling for the kinematics of the robot. On the NUC mounted on Quanty, Robot Operating System (ROS) kinetic 16.04 Long-Term Support (LTS) was used to manage the internal communication. A NUC is a small computer board manufactured by Intel, which provides with its quad core processor enough computational capacity to run different scripts at the same time. ROS

is an adaptive framework for writing robot software (Kerr & Nickels, 2012).
In order to create and save the generated motions, a new csv-file is generated with information about the current angle of each servomotor, as in a motion-matrix. As mentioned in chapter 4, a gesture consists of many postures. To align these postures, the robot is placed in a certain posture. After being placed in the target position, the current servomotor positions are read out and saved in a csv-file. In order to save the current positions, a node was written to get the current angle of each servomotor and save it in a csv-file. In ROS, the communication is established with so-called nodes. To enable a communication between two nodes, they have to be connected with each other. Defining this connection, certain roles are given to these nodes. One node has to be in the role of a publisher and the other node needs to be defined as the subscriber (Kerr & Nickels, 2012). The publisher, in this case the *motion-controller*-node, publishes the current angle of all servomotors while the *motion-saver*-node saves these angles in defined order in a csv-file.
The connection of two nodes are established with so-called topics. A topic can be seen as a name on a box. The publisher puts its information into this box either on new events or periodically. The subscribers receives a notification that something was put into this box and gets the information. While creating new postures a keyboard event indicated a newly defined posture. The Dynamixel servos, permanently publish their joint state onto the topic *sensor_msgs/JointState*. Subscribing to this topic, the current position is red and saved into a csv-file. This process is repeated until all postures have been saved into the csv-file and the gesture is done. When all postures are saved and the gesture is finished, the gesture can be played. Therefore, the *motion_saver*-node publishes the name of the executable motion file onto the *play_motion*-topic. As the *motion_player*-node subscribes to this topic, it gets the name of the executable motion file and executes it. According to the order of the entries, the *motion_player*-node publishes the *<joint-name>/command* topic to the individual servos. Getting this command, the servos will move to the first position and waits for the next line to be red. If every servomotor sends the status *in_moving = false*, which indicates that they reached the target position of the current posture, the next line will be red.

7 Contributions to Mechanical Robot

This chapter summarizes all achievements regarding the motion execution of Quanty. As generalization of Deng (2020), this Master Thesis generated a motion generator circuit to generate behaviors such as:

- deterministic,
- classical probabilistic,
- quantum probabilistic,
- mixed probabilistic.

This chapter starts with the more trivial circuits to express motion regarding the internal state of Quanty. Further, the design approach with respect of the Uncanny-Valley theory for smaller humanoid robots will be explained. At the end of this chapter, the section explains the realization of the quantum random walk and how it was possible to implement this algorithm on a humanoid biped robot.

7.1 Overview of HR-OS-5 Hardware

The HR-OS5 is a small biped humanoid robot with a height of 27 inches. Having several Dynamixel servos, the robot can express various movements. In total, the HR-OS5 comes with 20 degrees of freedom. The main CPU - Intel NUC D54250WYB SBC with Ubuntu 16.04 LTS distribution offers enough capacity for the framework ROS which manages the robots' communication. ROS provides an own library for controlling Dynamixel servos. The servomotors provide interfaces for TTL and RS-485 serial communication and, therefore, the daisy-chain alignment is possible. TTL stands for Transistor-Transistor-Logic and is a method of serial communication. RS-485 is an acronym for Recommended Standard 485 and is a telecommunication standard. TTL and RS-485 differ only at a hardware level (Eren, 2004). The daisy chained servos are arranged in this case as a bus topology. Here, the power and analogue signals pass through every servo in the chain (Tsitsiklis et al., 2012). As the servos have a unique software Identity (ID) and the analogue signals have matching labels, the signals pass through every servo in this bus-topology and only if ID and label match, the servo will process this message. Given a unique distribution of servo IDs for each servo, the messages from and to the servos are mapped properly. At the beginning of this Master Thesis Quanty had 18 servos as the right shoulder joint servo and the head with the servo for the neck motions were broken. In order to stabilize the walk of HR-OS5, the feet-plates were adjusted. In addition to the setup configuration, a webcam was mounted on the chest of HR-OS5 to gain more information about the environment and to enable vision.

7.2 Mood Quantum Circuit

As mentioned in the introduction to this chapter, at the beginning of this Master Thesis the focus was on trivial circuits that represent the internal state of the robot.
The *mood*-circuit influences Quanty motion execution according to its internal state with mapping two moods onto the circumference of a qubit. Fig. [7.1] shows how the three main internal states are depicted with three different points along the circumference.

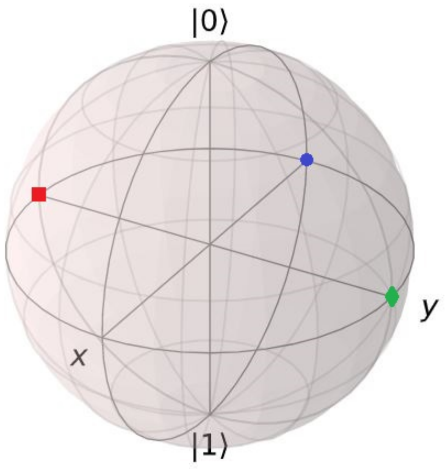


Figure 7.1: The three points along the circumference depict the three different internal states. The blue circle represents the neutral phase. At this point the robot is neither happy nor exhausted. Opposite of the blue circle is another neutral phase. If the internal state transforms towards the red square, the robot becomes *happier*. If a rotation gate transforms the state vector towards the green diamond, the robot becomes more *exhausted*.

The internal state is y-symmetrically, which means that the qubit in Fig. [7.1] has a second neutral phase on the opposite side of the neutral phase represented with the blue circle in Fig. [7.1]. The states in Fig. [7.1] can be represented with phase shits of θ:

- $\theta = 0 \longrightarrow$ opposite of blue circle
- $\theta = \frac{\pi}{2} \longrightarrow$ green diamond
- $\theta = \pi \longrightarrow$ blue circle
- $\theta = \frac{3\pi}{2} \longrightarrow$ red square

Within this Master Thesis, the user sets the internal state of Quanty manually. In further research this concept offers extensions such as the robot's vision defines the internal state. If Quanty sees someone it knows, the internal state transforms towards the *happy* phase and if the person is unknown to Quanty the internal state changes towards the *exhausted* phase.
With this version of the *mood*-circuit, the user's inputs are the *mood* and its *intensity*. While initializing the robot's mood, the user can decide between *happy* which is encoded with a 1 and exhausted which can be encoded with 0. The *mood* input decides on which side of the neutral phase the internal state is. If the input is 1 the state transforms to the blue circle in Fig. [7.1]. The intensity refers to the degree of the mood.
After Quanty receives the users' input, it maps the internal state onto the phase shift of θ to transform the state vector according to the internal state. The initial angle is:

- $\theta = 0$

As the qubit only represents two mood maxima, the user can set the mood of Quanty into a *happy*-state with input 1 or into a *exhausted*-state with input 0. A *happy* mood results in a phase shift of $\theta = \pi$ and a *exhausted* mood in a phase shift of $\theta = 0$. The intensity of this internal state has three different degrees which result in additional a phase shift:

- Intensity 1: $\theta = 0$
- Intensity 2: $\theta = \frac{\pi}{4}$
- Intensity 3: $\theta = \frac{\pi}{2}$

The input of the user constructs the transformation matrix U_1, representing the internal state. A practical example assumes that the input was:

- mood = happy = π
- intensity = 2 = $\frac{\pi}{4}$

Eq. [7.1] calculates the phase shift θ for the transformation matrix U_1 according to the input.

$$\begin{aligned} &\theta = mood + intensity \\ &\theta = \pi + \tfrac{\pi}{4} \\ &\theta = \tfrac{5}{4}\pi \end{aligned} \tag{7.1}$$

The controlled unitary operation *U* represents the total rotation of eq. [7.1]. With the operators U_x the internal state gets transformed in the computational x-basis using phase estimation.
The transformation matrix of U_1 in the example of eq. [7.1] results in eq. [7.2] with $\lambda = \frac{3}{2}\pi$.

$$U_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\lambda} \end{bmatrix} \tag{7.2}$$

The controlled operators U_2 and U_3 from Fig. [7.2] transform the phase information into the observable basis.

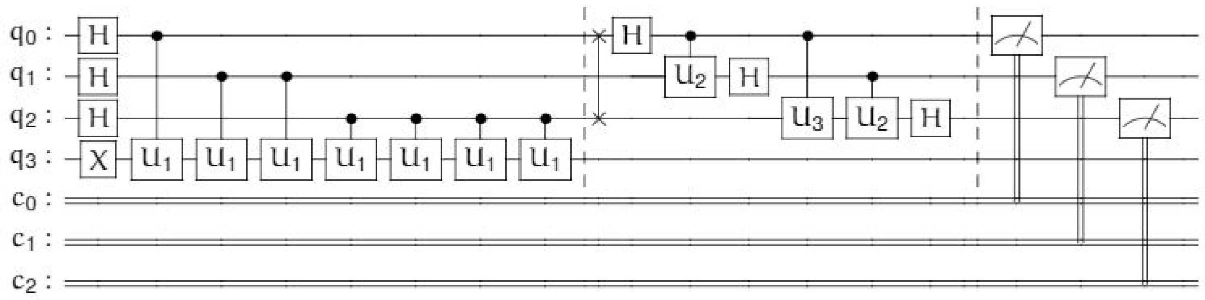


Figure 7.2: The dashed lines divide the circuit into three parts. The first part of the circuit kicks-back the phase of operator U_1 onto the controlled qubit. The second part of the circuit maps the qubits with the kick-backed phase into a different basis to make the information of the phase kickback observable. The last part of the circuit measures the qubits.

To map the internal state described above onto the computational x-basis, a circuit with four qubits and three classical bits is necessary. In the circuit from Fig. [7.2] qubits q_0, q_1, and q_2 kickback the phase of the controlled operator U_1.
Eq. [7.3] and eq. [7.4] illustrate the transformation matrices of U_2 and U_3.

$$U_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{-i\frac{\pi}{2}} \end{bmatrix} \tag{7.3}$$

$$U_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{-i\frac{\pi}{4}} \end{bmatrix} \tag{7.4}$$

Tab. [7.1] decodes the measurement of this circuit.

Table 7.1: The results of the circuit in Fig. [7.2] are further interpreted in ROS and affect the motions of the robot. The *happier* the internal state, the faster and more range has the motion. Some motions, that are considered to express a *happier* internal state are only able if the first qubit is measured one, $|1xx\rangle$.

Mood	**Intensity**	**Output**	**Results**
0	1	$\|000\rangle$	Low speed and range of motions. Duration of generated motions is medium.
0	2	$\|001\rangle$	Low speed and range of motions. Duration of generated motions is short.
0	3	$\|010\rangle$	Very low speed and range of motions. Duration of generated motions is shortest.
1	1	$\|100\rangle$	Medium speed and medium range of motions. Duration of generated motions is normal.
1	2	$\|101\rangle$	High speed and high range of motions. Generated motions are long
1	3	$\|110\rangle$	Full speed and range of motions. Generated motions are longest

With the inputs for *mood* = 1 and *intensity* = 2, Fig. [7.3] illustrates the results of 2048 evaluations of the circuit.

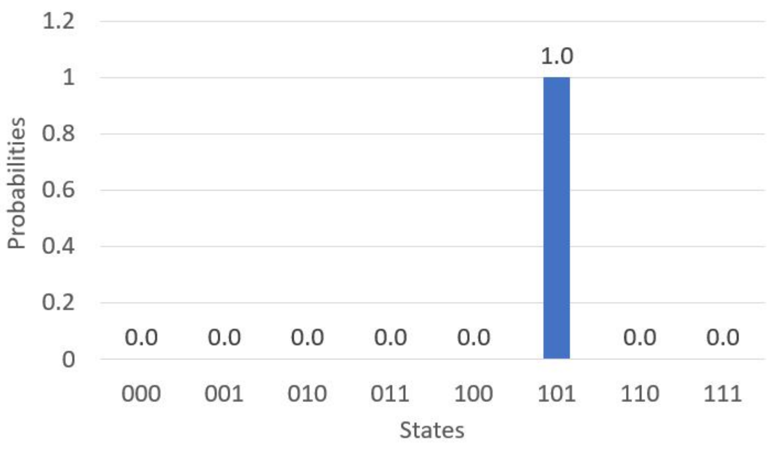


Figure 7.3: When the circuit is performed on a simulation without any noise, the input state always results in the output state according to Tab. [7.1]. This means that after one evaluation, the internal state can be concluded with a high certainty.

In comparison to the accurate results on the simulation, the phase estimation does not work on a real quantum computer. Fig. [7.4] shows that the results on the quantum computer named *ibmq_ourense* are not comprehensive.

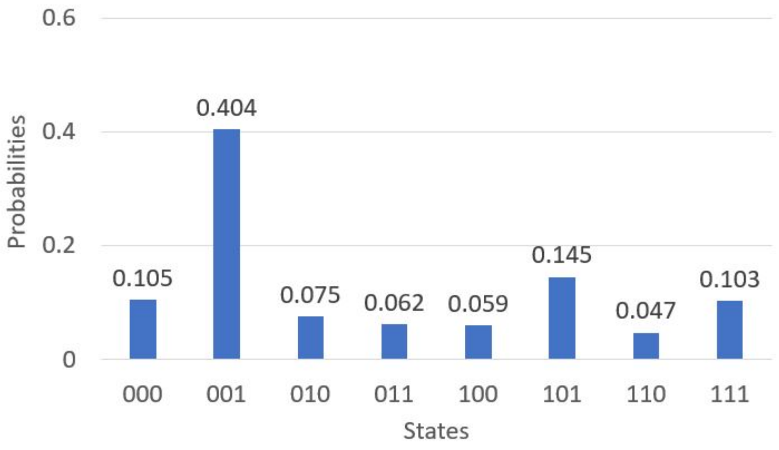


Figure 7.4: The results on a real quantum computer are not comprehensive with the results of the simulation. The distribution shifted on every evaluation remarkably.

The normal QFT does not work on a quantum computer, as all qubits are in equal superposition which leads to a random measurement. The inverse QFT converts the computational x-basis to the computational z-basis and, therefore, solves the problem of the random measurement. Fig. [7.5] shows the inverse QFT-circuit for this example.

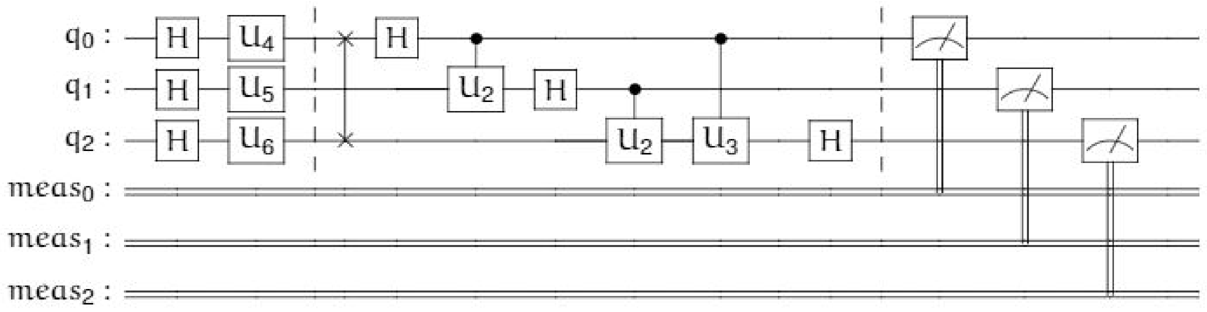


Figure 7.5: The dashed lines split the circuit into three parts. The second and third part of the circuit are independent from the internal mood and remain always the same. The operators U_2 and U_3 are the same as in Fig. [7.2]. Changes on the internal state will adjust only the gates in the first part of the circuit the unitary operators U_4, U_5, and U_6.

The circuit in Fig. [7.5] shows the implementation of the inverse QFT with a circuit of low depth, which makes it very robust against decoherence.

The unitary matrix of eq. [7.5] represent the gates U_4, U_5, and U_6.

$$U_{4/5/6} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\lambda} \end{bmatrix} \tag{7.5}$$

To represent the state $|101\rangle$ in the x-basis, lambda has to be set as follows:

- $\lambda_4 = \frac{5}{4}\pi$
- $\lambda_5 = \frac{5}{2}\pi$
- $\lambda_6 = 5\pi$

Fig. [7.6] illustrates that the first part of the circuit in Fig. [7.5] maps the state $|101\rangle$ onto the x-basis.

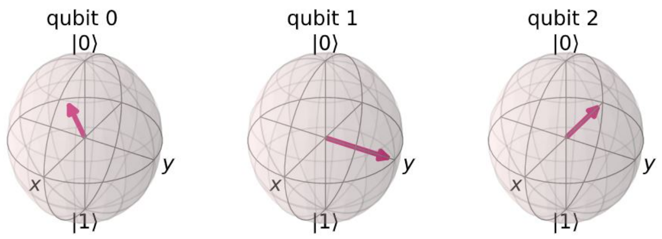


Figure 7.6: The state $|101\rangle$ mapped onto three qubits. The leftmost qubit represents the internal state with the z-rotation of $\frac{5}{4}\pi$.

Fig. [7.7] shows that the results of a real quantum computer executing the inverse QFT align more with the results of the inverse QFT simulation in comparison to the normal QFT.

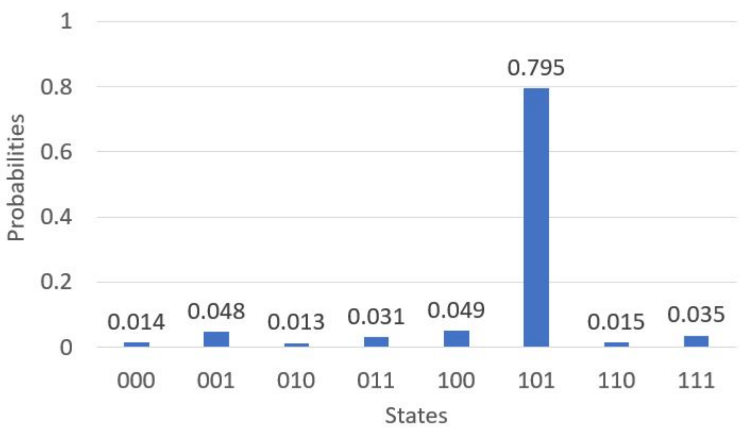


Figure 7.7: The results of the inverse QFT are not as randomly as the results of Fig. [7.2]. As the results are encoded in the Z-basis, they are observable. The circuit has low depth and is, therefore, not prone to decoherence and can be executed on a real quantum computer.

This application aims to increase the HRI quality. For the scope of this Master Thesis it is sufficient to express an internal state between *happy* and *exhausted* with three different intensities each. If an application demands all seven moods, a different method is necessary.
Within this Master Thesis, Quanty learned the identity vectors of two people. The identity vector is a representation of a person's significant features in a n-dimensional vector space. Quanty associated one person with a positive mood, the other with a negative mood. Recognizing the person that was associated with the positive mood, the robot executed a *happier* greeting behavior. With longer gestures, more intense waving and probabilistically using one or both hands Quanty expressed this *happier* greeting behavior. Recognizing the person associated with negative mood, the robots greeting behavior had less variety, the gestures were shorter, and the robot used one hand or no hands.
Fig. [7.8] illustrates the entire ROS-graph.

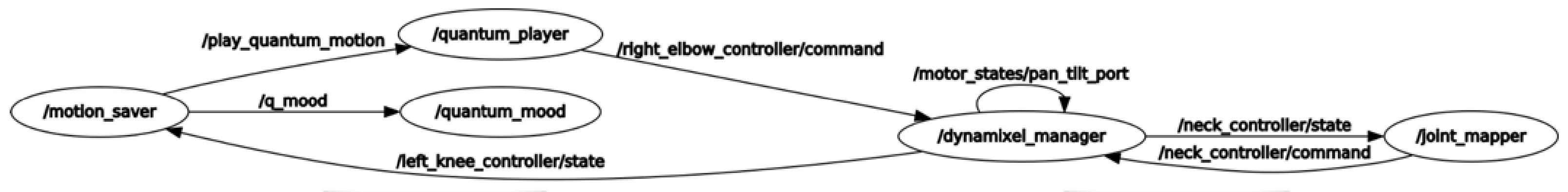


Figure 7.8: The nodes are represented inside the ovals and the topics are displayed above the arrows. The */motion_saver* publishes the executed motion to the */quantum_player*, which executes the motion. Additionally, the */motion_saver* publishes the mood to the */quantum_mood* node. Here, the quantum circuits will be executed with Qiskit inside a subroutine. After this subroutine, the */quantum_mood*-node sets the range and the speed of the servos according to the measured mood. The */quantum_player* executes motions with respect of the servos range and speed that is set by the */quantum_mood*. The servos are controlled with the */dynamixel_manager* and the */joint_mapper*. If the servos are in motion, no further commands will be accepted.

7.3 Characteristic Quantum Circuit

This section extends the concept of a robot's characteristic and personality. Even before a human is born, he/she/it already has the predisposition to be either left- or right-handed. The handed-circuit of this section tries to define the robots' strong hand to form a more in-depth persona with little computational cost. The feature of defining the handedness tries to improve the HRI by giving the robot a more human-like nature. When being left-handed, the robot performs tasks like greeting or picking up an object more often with its left arm or vice versa. Although the handedness also affects the thinking process of human beings, the characteristic quantum circuit only affects the motion execution.
To realize the handedness, the quantum circuit empathizes left- or right-handed motions. Considering the greeting behavior, a two-qubit-system can represent four different greeting behaviors such as:

- $|00\rangle \mapsto$ Greeting with left hand
- $|01\rangle \mapsto$ Greeting with both hands
- $|10\rangle \mapsto$ Greeting with no hands / nodding
- $|11\rangle \mapsto$ Greeting with right hand

The section 6.4 generated the motion matrices in advanced by hand and encoded theses matrices according to the states above. The circuit to realize the behavior is an adaptation of a Grover circuit.
The Grover circuit for left- and right-handed will be both initialized with two Hadamard gates at the beginning of the circuit. The second part of the circuit, the so-called Grover oracle, is the only part that is different for left- or right-handed persona. The last part of the circuit, the so-called reflection around average is the same for both circuits. Fig. [7.9] shows the complete circuit for a left-handed robot.

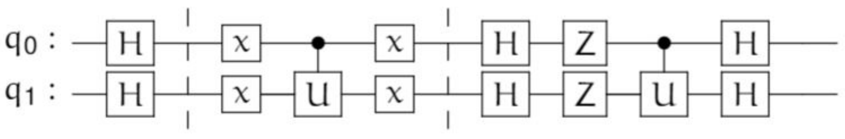


Figure 7.9: The dashed barriers divide the three steps of this circuit. The first part and the third are equal for both personas, left- and right-handed.

The controlled unitary operator *U* in Fig. [7.9] rotates the states with $\theta = 1.3\pi$ and has the transformation matrix of eq. [7.6].

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i1.3\pi} \end{bmatrix} \tag{7.6}$$

As mentioned in section 5.3 the second part flips the amplitude or the target state. The Grover algorithm for a two-qubit system only needs one iteration to find the target state with 100% accuracy. After two iterations the accuracy of finding the target state will decrease again. The evaluation includes only one iteration, but instead of mirroring the state with a rotation of π as described in section 5.3, the rotation gate U rotates the states with 1.3π. Fig. [7.10] shows the probability distribution of the left-handed-circuit.

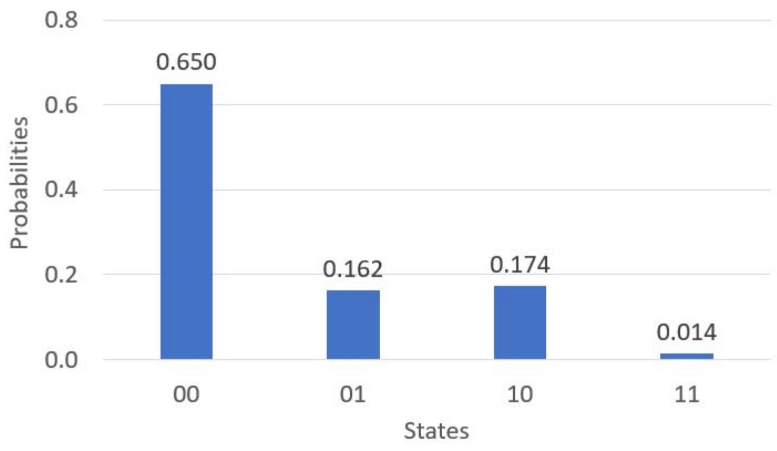


Figure 7.10: In this figure the results of circuit from Fig. [7.9], executed on a simulation are illustrated. As $|00\rangle$ is decoded as greeting with the left hand, the robot will more often execute this motion.

Fig. [7.11] shows that the right-handed-circuit differs in comparison to the left-handed-circuit only in the second part of the circuit, as it does not have the Pauli-X gates.

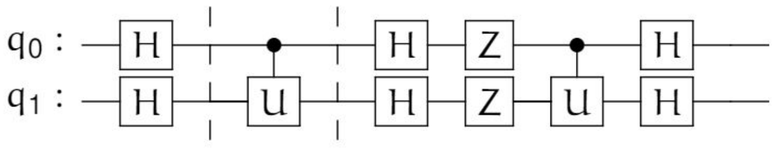


Figure 7.11: The only difference to the circuit of Fig. [7.9] lies in the second part. As the circuit has no Pauli-X gates Fig. [7.12] shows that the probabilities will shift towards the execution of greeting with the right hand.

Fig. [7.12] illustrate the results of the right-handed-circuit. In comparison to the results of Fig. [7.10] the probabilities shift towards the motions that empathizes right hand motions.

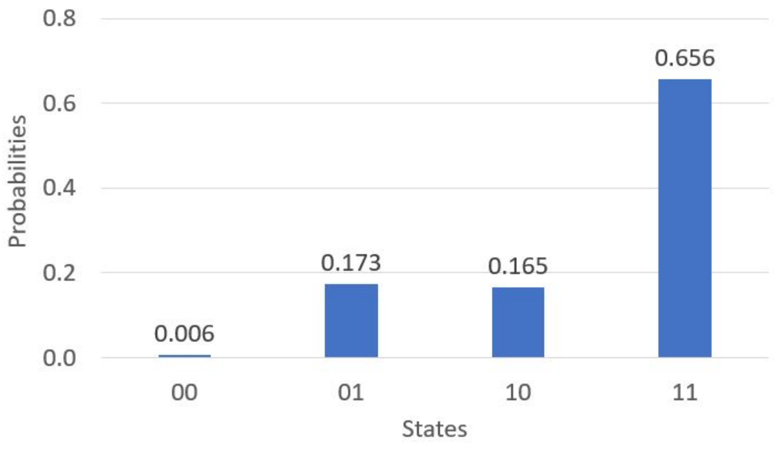


Figure 7.12: The only difference to the circuit of Fig. [7.9] lies in the second part. As the circuit has no Pauli-X gates, Fig. [7.12] shows that the probabilities will shift towards the execution of greetings with the right hand.

Fig. [7.13] shows the result of the right-handed circuit executed on a real quantum computer from IBM called *ibmq_ourense*. The execution on a real quantum computer was to validate that this circuit is robust if executed under real circumstances.

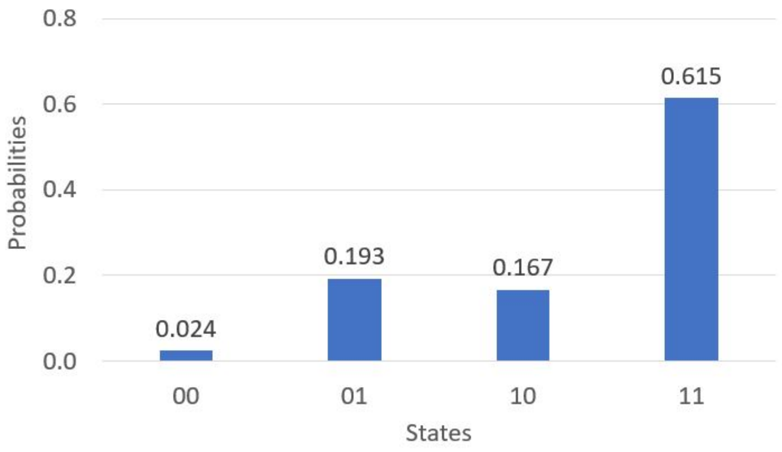


Figure 7.13: Evaluated on the backend *ibmq_ourense* the results of the circuit from Fig. [7.11] differ slightly form the results of the simulation Fig. [7.12].

The results from the simulation and the quantum computer differ slightly, but both variations offer a more human-like behavior as they include random numbers. Using these circuits, with low computational cost, the robot is able to express its personality. Expanding this concept on other movements like grabbing or walking, it is possible to simulate a robot with a more human-like behavior. Fig. [7.14] illustrates the entire ROS-graph of this circuit.

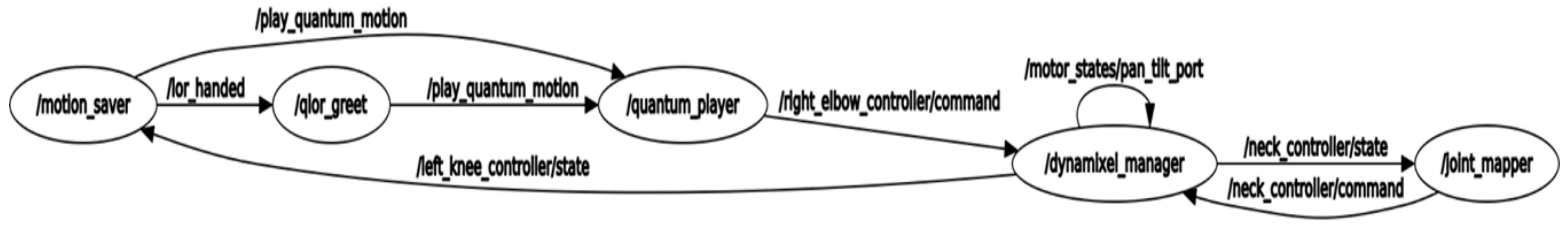


Figure 7.14: The quantum circuit defining the handedness of the robot was executed in a subroutine of the *qlor_greet*-node. Inside this quantum circuit it was decided which of the four greeting behaviors will be executed. The chosen motion file will be published to the */quantum_player*-node which executes the motion.

7.4 Design of Robotic Head

This section designs a robot head for Quanty to express more facial emotions. To our knowledge, smaller humanoid robots like HR-OS5 always lack facial expression. In this Master Thesis, an animatronic head was designed to express a minimum amount of facial expression in order to increase the quality of HRI.
On a standing humanoid biped robot the head is on the highest point, which gives it the biggest lever. To keep the resulting momentum at a minimum the head must not weigh too much in proportion to the body. Therefore, this head realizes only the most important facial with servos to decrease the total weight.
The designed head has in total four DOF and is controlled with Dynamixel MX-28-servos. These servos are very compact and flexible in usage. As only four servos are necessary to enable the four DOF, it was possible to arrange the servos in a small installation space. The total dimensions of the head in its final version are (110*134*138)mm (Width*Height*Depth). The Dynamixel are superior to commonly used servos in robotics because of the straightforward integration to the current system and because of the robustness of the servos' housing. Therefore, the servos housing stabilize the heads construction, which spares the need of additional construction components and material. With less components and material also the assemblage itself is very intuitive. Except for the screws and the servos all additional components are printable in order to produce the head in every laboratory that is equipped with a 3D-printer.
HR-OS5 will inevitable topple often and requires, therefore, robust servos. One reason the robot might fall is due to its educational purpose. Thus, students will try new motions or lack experience in handling the robot. The other reason for the robot's fall might be the robot's structure itself. Walking on two legs is a dynamic process with high momentum as the robot is partially balancing only on one leg, the so-called one-support-phase. Being on two legs, the robots stand forms a parallel kinematic, also called the double-support-phase. During the walking gait, the stable double-support-phase will frequently change to a more instable single-support-phase (Alamdari & Krovi, 2016). The heads' construction considers that in case of falling forward the robot will always land on its upper jaw and falling backward the robot will always land on the neck tilt servo. Both parts are very robust and tend not to break easily. As the upper jaw might break when falling heavily, its replacement is straightforward without unmounting any other parts of the head.
The four DOF are:

- Neck rotation
- Neck tilt
- Eyes rotation
- Lower jaw rotation

As Dynamixel servos offer a daisy chained control, the integration of the additional servos to the already existing system does not result in a lot of adjustments. The order of the connection of the four servos and the connection to the existing daisy-chains is considered in the design of the head. As the wiring is respected in this design the cables can be aligned within the heads' construction and the wires are secure during motion. The order of the servos in the daisy chain is determined as follows:

existing daisy chain $\longrightarrow$ neck rotation $\longrightarrow$ neck tilt $\longrightarrow$ eyes rotation $\longrightarrow$ lower jaw rotation

With only four degrees of freedom the robot's head appearance is less human-like and it is on the left side of the Uncanny-Valley. Aiming for the right side of the Uncanny-Valley will result in more servos, more weight and more complexity of the construction itself. In general, smaller humanoid bipeds should be on the left side of the Uncanny-Valley in order to save weight and complexity. Because of the findings from Mori et al. (2012), the robot‘s head design itself is very similar to the design of Sesame-Street characters. The Sesame-Street characters design tries to reach the highest possible point *t3* in Fig. [2.1] on the left side from the Uncanny-Valley. Therefore, the proportions of head and body on Quanty is different than the proportions from head and body of a human. If the robots' head design aims for the right side of the Uncanny-Valley, it must be next to human-like optically and behaviorally. An example of a human-like robotic head is the robot Albert-Hubo (Oh et al., 2006). The robot head of Albert-Hubo is very realistic and it can be considered to be on the right side of the Uncanny-Valley. To enable all the DOF needed to act human-like, the head of Albert-Hubo itself has 32 DC gearmotors which would be too much for smaller humanoid robots like Quanty.
The four chosen DOF have the highest importance in human-robot communication. With the neck-tilt, the neck-rotation, and the eye-rotation the robot can better express what is currently in its point of interest. What is currently in the region of interest is more obvious when the robot's eyes are looking in the direction of interest and thus, it is easier to grasp the context. The Lower jaw rotation should improve the perception of the robot's speech as the human brain processes both, sound and lip movements (Nicholls et al., 2004). The jaw movement is synchronic to the speech. In this way it appears more human-like when the robot is speaking, and its lower jaw is moving according to the spoken words.
Fig. [7.15] illustrates the upper jaw as the bigger bent gray part in the front.

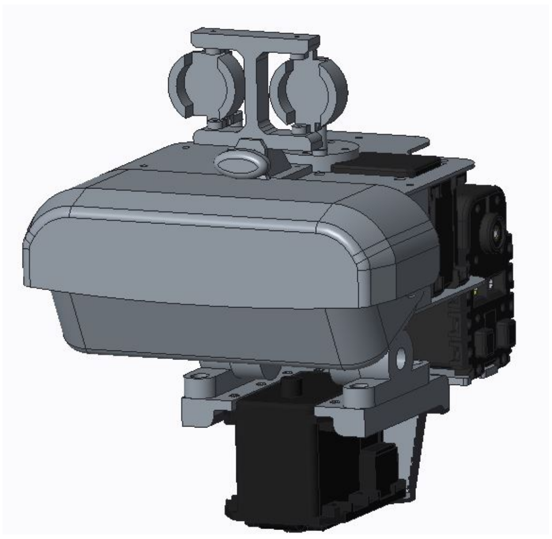


Figure 7.15: Isometric view of the robot head. All grey parts are 3D-printable. The black servo at the bottom was still on Quanty despite the missing head.

Only two screws mount the upper jaw which makes the upper jaw easily replaceable if broken. The neck-rotation servo was on the initial setup of Quanty. The neck-tilt-servo which is the leftmost black servo in Fig. [7.16] is an extension to the neck-rotation-servo which is the black servo on the bottom.

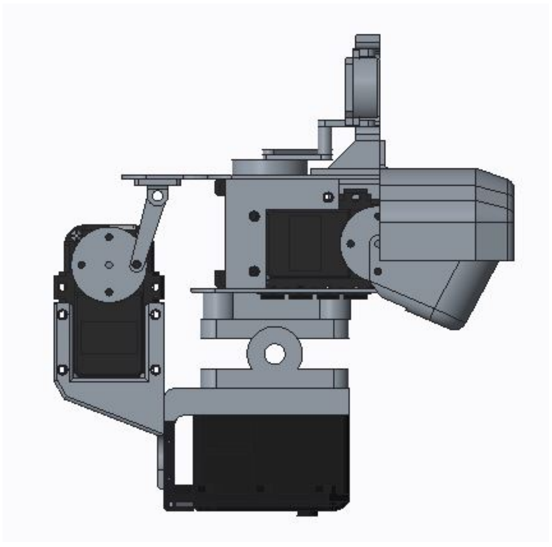


Figure 7.16: Right side of the robot head. The neck-tilt servo in black on the leftmost side is attached to the neck-rotation servo at the bottom. The bar connecting the neck-tilt servo with the upper plate transmits the force from the servomotor.

Mounting the additional three servos for the neck-tilt, the eyes-rotation and the lower-jaw rotation two methods are possible. The first method mounts the three servos were on the robots' upper body and Fig. [7.16] shows the second method, where the servos are on the heads'

structure. Mounted on the upper body, the weight of the head is lower but the motion from the servos must be transferred onto the rotating head. If the servos are on the head, they will move along with the neck's rotation and, therefore, no additional transfer of movements is necessary. The first option would increase the complexity of the construction which is the reason the second method is superior.
Three screws attach the lower jaw is attached on the right side to the last servo in the daisy chain. Fig. [7.17a] and Fig. [7.17b] shows that on the left side, the lower jaw is loosely hooked onto a mounting point.

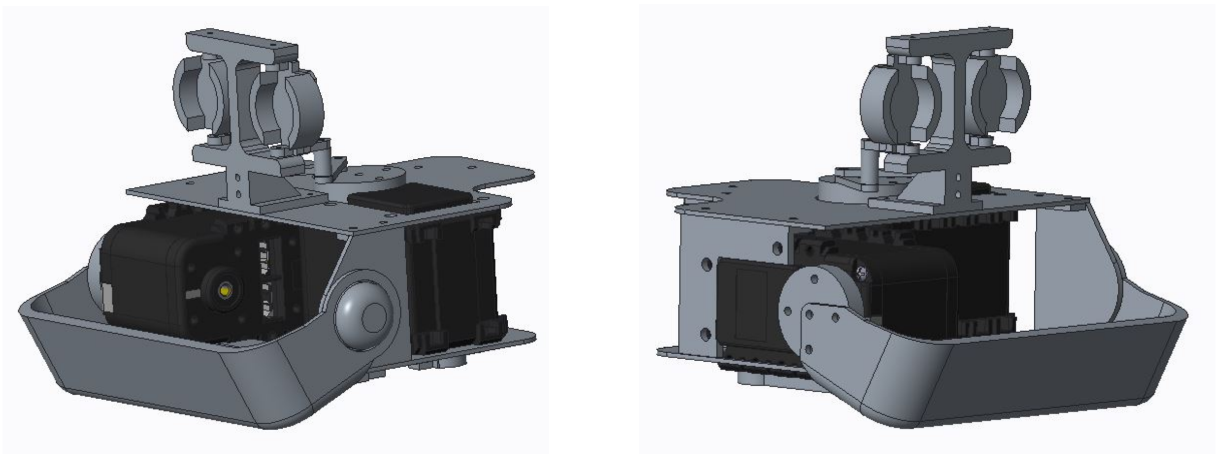

(a) Left side.
(b) Right side.

Figure 7.17: As seen in 7.17a the jaw is hocked onto a support structure, which still enables the jaw to rotate along with its servo. In 7.17b the jaw is attached to its servo using three screws.

The solution for the lower jaw makes the mechanic and the lower jaw can easily be replaced if it is broken. The eye-mechanism is adapted from the French humanoid robot inMoove (Bazzano & Lamberti, 2018). The eyes are mounted rotatable onto the construction on the upper plate. When the servo is moving, both eyes will rotate symmetrically around their mounting point of their mounting construction.

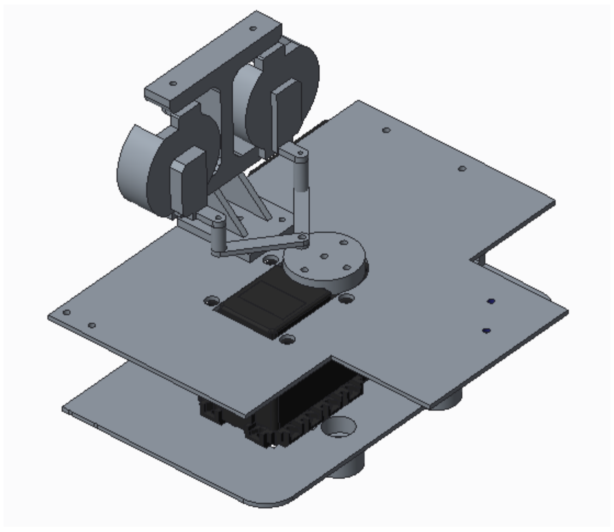


Figure 7.18: The rotation of the servo is transmitted via distance bars to the eyes. The distance bars of the left and right eye are fixed on the same spot on the servo and will, therefore, follow the same rotational movement which makes the both eyes move symmetrically.

In scope of this Master Thesis, is was not possible to print this robot head and mount it onto the Quanty. Therefore, the head was mounted virtually onto the body of the DARwIn robot. As no assembling of the HR-OS5 exists and the capacity to rebuild it would exceed the projects' time frame, Fig. [7.19] shows the open-source available DARwIn body and Quantys' head to see the proportions of the robotic head to the body.

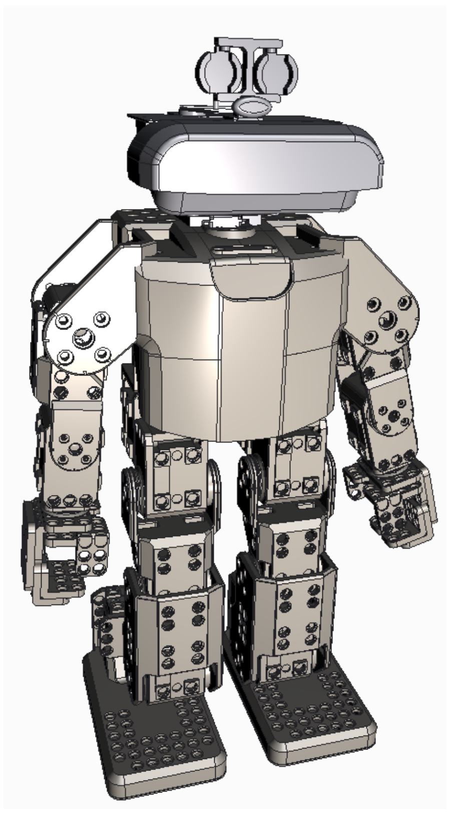


Figure 7.19: The head is bigger proportionally to the body in comparison to a human. The character-design of the Sesame-Street was the role-model for this design. The body of the DARwIn robot was used to illustrate the entire robot with the head as no model of the HR-OS5 body exists. The purpose of the assembling is to show how the head looks mounted on a smaller humanoid robot.

7.5 Generalized Quantum Motion Generation

This section applies the result of section 6.2 to Quanty and extends the concept with a mixed behavior. With one circuit, four different behaviors of motions are possible. The four different behaviors motions are:

- Deterministic
- Classical Probabilistic
- Quantum Probabilistic
- Mixed Probabilistic

Section 6.1 already described the first three probabilistic behaviors. Mixed probabilistic behavior is a combination of classical probabilistic behavior and quantum probabilistic behavior. The difference lies in the initial state of the target qubit. Regarding the quantum probabilistic behavior, the initial state of the target qubit is a base state of either $|0\rangle$ or $|1\rangle$. However, the

target qubit's state in the mixed probabilistic circuit is not a base state, which causes a different entanglement as the Bell states.
A four-qubit-system represents such a quantum motion generation circuit. One circuit controls one servo of Quanty. The circuit has two registers, one control register with two control-qubits and one target register with two target-qubits. The internal state of the control qubits c_0 and c_1 chooses the motion behavior. The motion behaviors are encoded onto the control-register as follows:

- $|00\rangle \mapsto$ Deterministic
- $|10\rangle \mapsto$ Quantum Probabilistic
- $|11\rangle \mapsto$ Classical Probabilistic

Adding a single transformation operator *U* with the matrix eq. [7.7] generates the fourth probabilistic behavior, mixed probabilistic.

$$U = \begin{bmatrix} \cos\frac{\theta}{2} & -e^{i\omega}\sin\frac{\theta}{2} \\ e^{i\omega}\sin\frac{\theta}{2} & e^{i\lambda+i\omega}\cos\frac{\theta}{2} \end{bmatrix} \tag{7.7}$$

The target register consisting of q_0 and q_1 decodes the position of the controlled servo. A control register and target register can represent the state of the four-qubit system as $|q_1q_0c_1c_0\rangle$. Fig. [7.20] shows the circuit to realize all four behavioral motions.

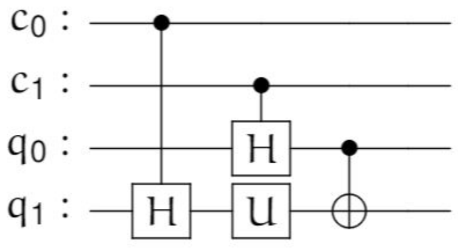


Figure 7.20: The circuit can be divided in two quantum registers, target- and control-register. According to c_0 and c_1 the first three probabilistic are chosen. With the angle θ of *U* the mixed state is defined.

The following examples demonstrates the four different probabilistic behaviors on this circuit.
Deterministic
To represent deterministic behavior, the circuit initializes both control-qubits with $|0\rangle$ and the unitary operator *U* has the angle $\theta = 0$. Fig. [7.21] shows the result for the deterministic setup with the input state $\psi = |0000\rangle$.

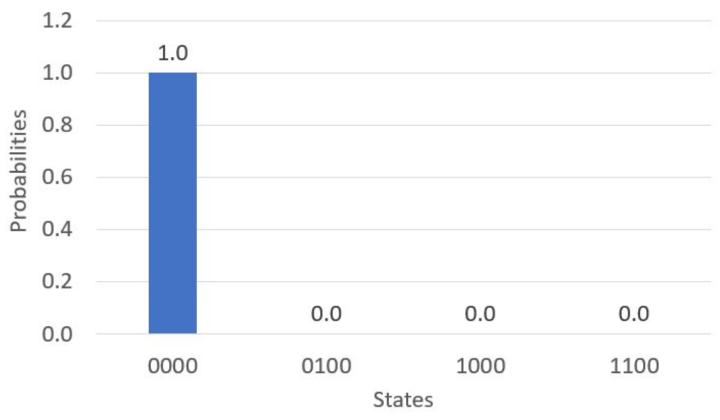


Figure 7.21: For any input state of q_0 and q_1 the output state is deterministic which refers to a one-to-one mapping. After 2048 evaluations, the same input state resulted always in the same output state.

Classical Probabilistic

If the target register of the circuit must result in all four possible states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$ with similar probability, the input state of the control register must be $|11\rangle$. Fig. [7.22] shows the results of the input state $|0011\rangle$.

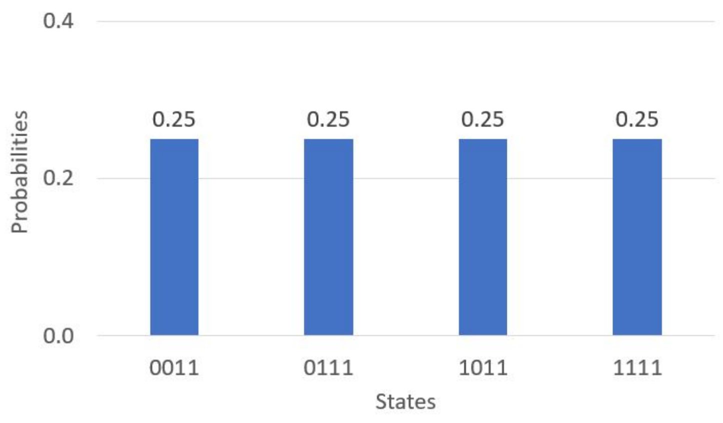


Figure 7.22: Qubits q_0 and q_1 are represented with the first and the second character of the output string below the probability bar. After 2048 evaluations, the same input state resulted equally distributed in each possible output state.

Quantum Probabilistic

For quantum probabilistic behavior, the quantum circuit initializes only the first of the control-register qubits with $|1\rangle$. Quantum probabilistic behavior includes entanglement as defined in section 6.1. Fig. [7.23] illustrates the results of the input state $|0001\rangle$.

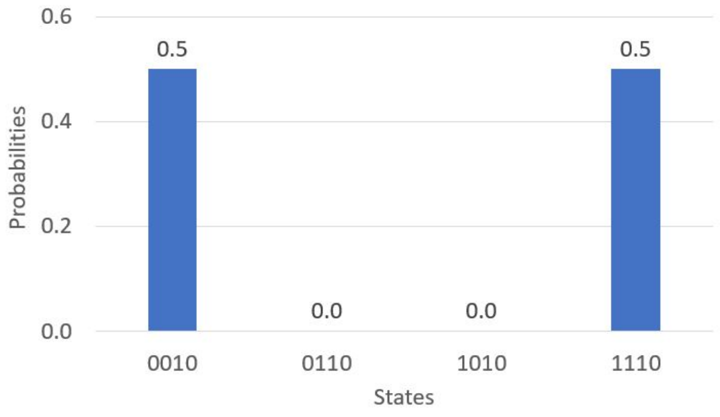


Figure 7.23: The first and the second character in the output string below the probability bar represent the qubits q_0 and q_1. The input state $\psi = |0001\rangle$ will result in the Bell state $|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

Mixed Probabilistic

For mixed probabilistic behavior, the initialization of θ is unequal zero. With both qubits, control- and target-qubit, in superposition, the mixed probabilistic circuit results in entanglement. Normal entanglement as defined in Einstein et al. (1935) has a superposition in the control-qubit and a basis state in the target-qubit. The results of the mixed probabilistic circuit are a combination of Fig. [7.22] and Fig. [7.23]. The amount of the quantum probabilistic behavior and the classical probabilistic behavior is dependent of θ which in this case is $\theta = -0.6\pi$. Fig. [7.24] illustrates the results of this mixed circuit with the input state $\psi = |0010\rangle$ and $\theta = -0.6\pi$.

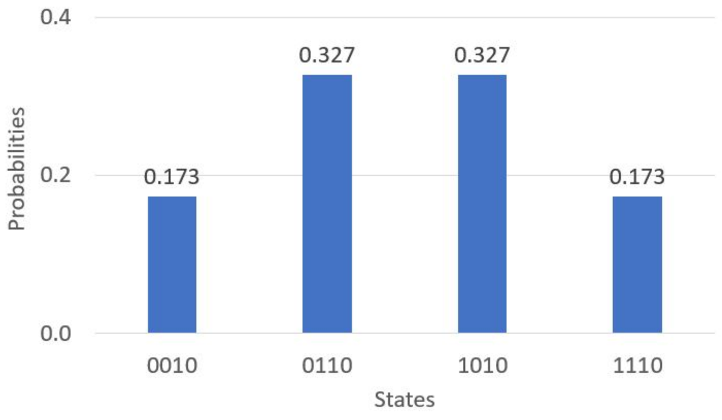


Figure 7.24: The distribution of the probabilities shows the combination of classical probabilistic behavior and quantum probabilistic behavior. Decreasing θ further to $-\pi$ will result in the Bell state $|\psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$.

A practical example created motions for the greeting behavior of the left arm. This greeting behavior includes three servo joints:

- Shoulder
- Elbow
- Wrist

To generate a natural waving behavior the shoulder joint imitated deterministic behavior, the elbow motions imitated quantum probabilistic behavior and the wrist imitated classical probabilistic behavior. The chosen setup in the example created various greeting motions with human-like character. In this example, the elbow rose until a certain position and stayed there for the rest of the greeting gesture. As the elbow had a quantum probabilistic behavior, it randomly chose one of the two possible output states. The wrist had a classical probabilistic behavior and shifted, therefore, randomly between all four possible position.
The length of the generated posture depended also on quantum computing as three Hadamard gates in parallel determined how many postures the created gesture had. A three qubit-system in superposition has 2^3 output states. Each state is encoded as a number from 1 to 8.
In context of the robotic theatre of PSU, the robot actors will perform different motions in every play, despite the same input variables, which appears more interesting for the audience. If this circuit is expanded to other motions like walking or dancing the motions of each play would be unique and the audience would not get bored as fast as with deterministic motions. The quantum motion generation was implemented in ROS as seen in Fig. [7.25].

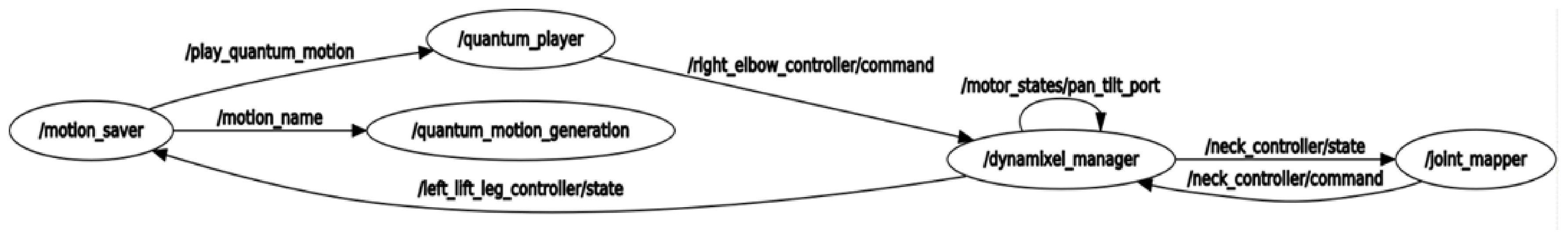


Figure 7.25: The quantum circuit will be executed in */quantum_motion_generation*. Within this circuit the motion matrix will be generated and saved. After generating a new motion matrix, the */quantum_player* can execute this motion file.

7.6 Quantum Random Walk with Humanoid Biped Robot

This application uses the quantum random walk algorithm (Aharonov et al., 1993) to train the robot a method to find its target. The robot tries to find a person in a room by randomly changing its direction and generating a random walking or turning behavior. When Quanty finds the target person, it will approach the person and greet him/her with a quantum generated waving motion. First the identity vector of a person was trained. With this vector, the robot can distinguish if the target person is in front or not. After every sequence of motions, the robot takes a picture of its surrounding and searches for a person with a similar identity vector. In scope of this project a webcam was mounted to the chest of the robot in order to take these pictures. If the target person is in this picture, the robot will walk a randomly chosen number of steps forward, stops and greets the person.
In order to make the robot walk and turn, five different motion-matrices have been generated by hand as they are:

- From stand into walk
- From walk into walk
- From walk into stand
- Turning left
- Turning right

At the beginning the robot stands with both legs next to each other. When it decides to walk forward, it uses the *From stand into walk*-motion file. At the end of this motion file the robot has taken one step with the right foot and one step with the left foot and the left foot will be in front and the right foot in the back. When the robot decides to go another step forward, it will choose the *From walk into walk*-motion file. This motion starts with the left foot in front and the right foot in the back. During the execution of this motion file, again, one step with the right and one with the left foot will be made. This motion file has the same start and end posture. When Quanty is done moving forward, it executes the *From walk into stand*-motion file in which the robot will adjust the right foot in the back next to the left foot in the front into a stable stand.
In the two-support phase the robot has both feet on the ground and while being in the single-support phase the robot has only one leg on the ground. The change from one phase into the other phase is decisive for a stable walking pattern. Fig. [7.26] shows that in order to realize a stable walking pattern, the five motion files follow the same change-method.

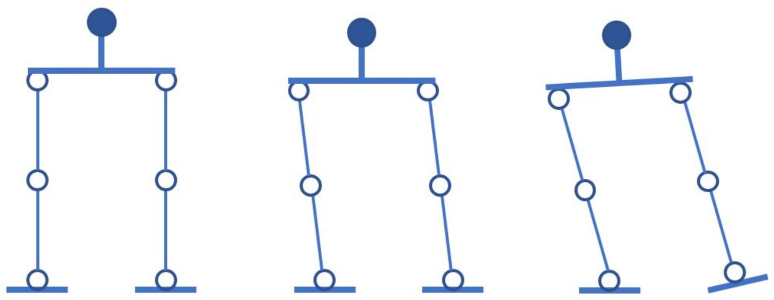


Figure 7.26: The support-phase-change constitutes of three postures. Considering this figure going from left to right the robot changes from the double-support phase in to the single-support phase. Going from right to left, Quanty changes from single-support phase to double-support phase.

The walking pattern from Fig. [7.26] generates less friction while lifting a foot and less swinging during the support-phase-change motion. Starting with the left picture, the first posture is a solid stand. Moving to the next posture in the middle, the weight shifts sideways with both feet flat on the ground. Quanty shifts its weight on the foot that will be on the ground during the single-support phase. To reduce friction and further shift the weight on the foot that will stay on the ground in this step sequence, the lifted ankle joint stops turning while the standing foot keeps rotating. At the end of this rotation, the weight is mainly distributed on the standing leg and the lifted foot touches the ground only with the edge of its foot. In the next step, Quanty lifts the supportive foot with a minimum friction from the ground.
Lifting one foot parallel from the ground is only possible with greater effort. As Quanty had no additional sensors, it needs a static motion matrix. While trying to lift one foot parallel from the ground the speed of the effectors must be managed dynamically. This dynamical regulation would exceed the scope of this Master Thesis which is the reason why Quanty does the method of Fig. [7.26].
This application requires two different circuits, one to decide between walk or turn and left or right and the other circuit to distinguish how many steps forward or how many steps turning. Fig. [7.27] shows the circuit for the decision if walking or turning and if left or right is a one-qubit-system with a Hadamard-gate.

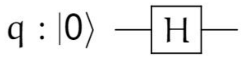


Figure 7.27: After the Hadamard-gate the measurement of the state results in either state $|0\rangle$ or $|1\rangle$ with the same probability. The state $|0\rangle$ is decoded as *Walking* or *Left*, depending on the current level of the decision tree. The state $|1\rangle$ is decoded as *Turning* or *Right*, depending on the current level of the decision tree.

In order to distinguish how many steps forward or steps turning Quanty executes, Fig. [7.28] shows a two-qubit-system with two Hadamard-gates in parallel.

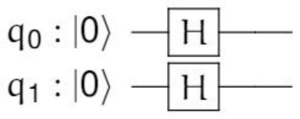


Figure 7.28: Depending on the current level of the decision tree, the output of this circuit decides how many steps forward or steps turning Quanty executes.

The result of the circuit from Fig. [7.28] can be any state of $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ with the same probability. The output states are decoded as followed:

- $|00\rangle \mapsto$ one step
- $|01\rangle \mapsto$ two steps
- $|10\rangle \mapsto$ three steps
- $|11\rangle \mapsto$ four steps

The decision tree in Fig. [7.29] shows the methodology of the entire quantum random walk algorithm.

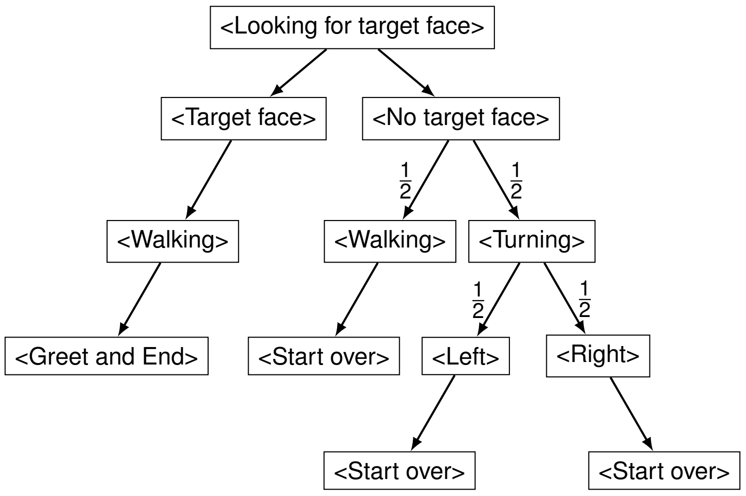


Figure 7.29: Decision tree of the random walk application. All <Walking>-nodes follow the same methodology, as a random number of steps between 1 and 4 will be taken. The <Turning>-node works similar as the <Walking> node. A random amount of turning steps between 1 and 4 will be taken within this node. The numbers next to the arrows indicate the probabilities of the action.

Depending on the chosen action, the script will combine and execute some of the five motion matrices. Therefore, depending on the circuits' result, a node will read out and store the motion matrices in a temporarily motion file. As an example, the robot decided to move forward three steps and search again for the target person. As the robot always finishes the motion with the stand posture, the script reads and stores the *From stand into walk*-motion file always at the beginning of the temporarily motion file. Afterwards, the script duplicates and appends the *From walk into walk*-motion file two times to the temporarily motion file. Quanty must always finish the walk with the same standing posture from the beginning. Therefore, the temporarily motion file always ends with the *From walk into stand*-motion file. When the script finished the temporarily motion file, the *motion_player*-node receives the name of the temporarily motion file, executes it and erases the motion file afterwards. Fig. [7.30] illustrates the entire ROS-graph.

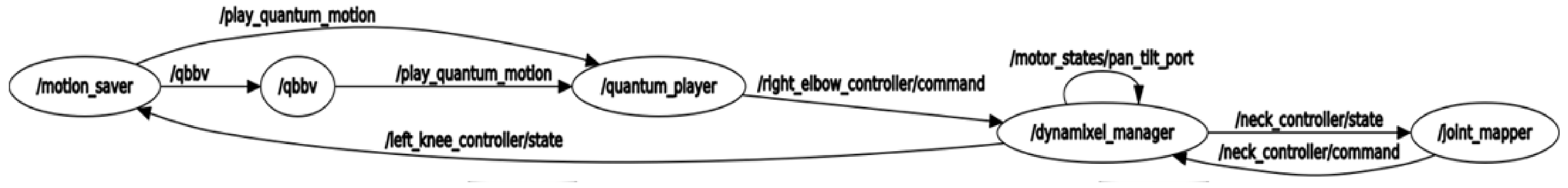


Figure 7.30: The name of the */qbbv* stands for **q**uantum **b**iped **B**raitenberg **v**ehicle. Within this node, the quantum circuits are executed, and the motion matrix is generated. The temporarily motion matrix will be sent to the quantum player, which executes the random generated walk.

8 Grover Oracle for Shortest Path Problem

This chapter introduces the approach to generate a Grover oracle that solves the shortest path problem and also other problems with similar application. The following sections solves problems like path existence, path location, and Path thresholding using quantum computing.
The shortest path problem is about finding a link between two vertices in a graph with the lowest cost. Different classical algorithms try to solve this problem. With *V* being the numbers of vertices and *E* being the number of edges, the time complexity of such classical algorithms are as follows (Demetrescu & Italiano, 2004):

- Bellman Ford O(V*E),
- Dijkstra O((V + E) log(V)),
- Floyd-Warshall O(V^3).

With the Grover algorithm explained in subsection 5.3 the author tries to find a path in the first place in $O(\frac{\pi}{4}\sqrt{\frac{2^E}{M}})$ with *M* being the valid solutions.
The shortest path problem is a well-known problem in robotics and is used for example to plan the path of a tethered robot as in Brass et al. (2015). This section introduces a new approach to solve the shortest path problem, using the Grover algorithm. The start of this new approach uses total symmetric Boolean functions described in Gao et al. (2020) to create a generalized method of designing a Grover oracle that solves the shortest path problem. Section 5.3 describes the Grover oracle and its functionality of flipping the amplitude of the target states. This Grover oracle, together with the Grover diffusor will be executed several iterations in order to increase the amplitude of the target states. Eq. [8.1] shows that if a function is totally symmetric, the output values are invariant to any permutations of input values.

$$f(x_1, x_2, ..., x_n) = f(x_{\sigma(1)}, x_{\sigma(2)}, ..., x_{\sigma(n)}) \tag{8.1}$$

S^k denotes a single index symmetric function *S*, with *k* being the symmetric order of the function. When a single Boolean index function is S^1, every valid minterm only has one positive literal while the other literals are negative. Literals are single variables or their negations of the function. In eq. [8.2] *a*, *b*, and *c* are literals.

$$F = a\overline{b}\overline{c} + \overline{a}b\overline{c} + \overline{a}\overline{b}c \tag{8.2}$$

Minterms are every valid composition of literals for a function. In eq. [8.2] the first minterm is $a\overline{b}\overline{c}$. An example of a S^1 function can be seen in eq.[8.2]. To understand the concept of the Grover algorithm for the shortest path better, Fig. [8.1] shows an example graph.

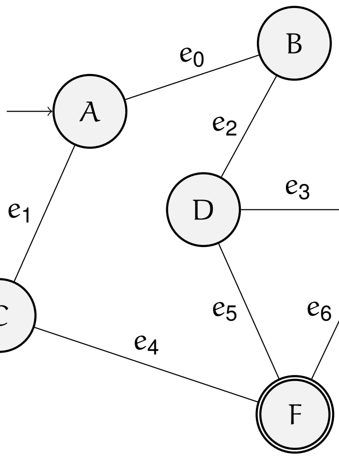


Figure 8.1: This graph is considered in some of the calculations and referred as *graph_1*. As some calculation exceed the computational capacity, additionally, a smaller graph will be used to show the functionality of the Grover algorithm. The arrow pointing on node *A* indicates the start and the double circuits at node *F* the goal. Edges are encoded as qubits and vertices are realized as symmetric function with additional negation.

The edges of the graph are encoded with qubits, which means that every edge is represented by a qubit. For the example graph in Fig. [8.1] the running time for different algorithm are as follows:

- Bellman Ford O(V*E) = 42
- Dijkstra O((V + E) log(V)) = 13
- Floyd-Warshall O(V^3) = 216
- Grover oracle O($\frac{\pi}{4}\sqrt{\frac{2^E}{M}} + \frac{\pi}{4}\sqrt{\frac{2^E}{M-1}} + \frac{\pi}{4}\sqrt{\frac{2^E}{M-2}}) \approx 20$

In order to show all applications of the shortest path problem, the smaller graph of Fig. [8.2] will be considered in some examples and referred to as *graph_2*.

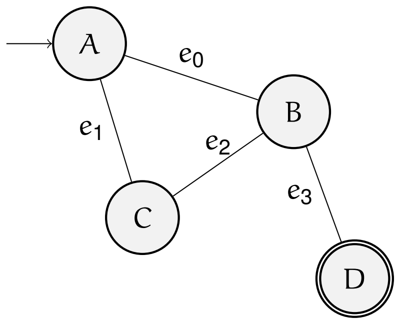


Figure 8.2: This graph has two valid paths from start node *A* to target node *D*. In this application a threshold will be set in order to only find the path with lower path cost.

The first step designs the start and target nodes as symmetric functions. S^1 functions realize start and target nodes as in a valid path, only one edge from the start node and one edge to the target node is active. Referring to the example *graph_1* of Fig. [8.1], there is no path that either has edges e_0 and e_1 or e_4, e_5, and e_6 active at the same time and is a potential path. In the example graph from Fig. [8.1] the start node has two edges and the target node has three edges. Both nodes are realized as S^1 functions with corresponding quantum circuits. For the setup the *graph_1* will be considered as example. Eq. [8.3] shows that the start node has only two edges, therefore, it can be represented as a totally symmetric function with two literals as in .

$$S_A^1 = e_0\overline{e}_1 + \overline{e}_0 e_1 \tag{8.3}$$

Fig. [8.3] shows the eq. [8.3] with quantum circuit notation.

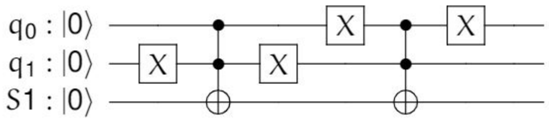


Figure 8.3: This figure shows the quantum circuit of eq. [8.3]. The Pauli-X gate on a qubit is necessary before and after the CCNOT-gate as the qubits input state must be restored. This process is also called mirroring.

Eq. [8.4] represents the target node *F* with an S^1 function, but in comparison to the start node it has three edges which results in a different symmetric Boolean function.

$$S_F^1 = e_4\overline{e}_5\overline{e}_6 + \overline{e}_4 e_5\overline{e}_6 + \overline{e}_4\overline{e}_5 e_6 \tag{8.4}$$

Fig. [8.4] transforms the Boolean function of eq. [8.4] into a quantum circuit.

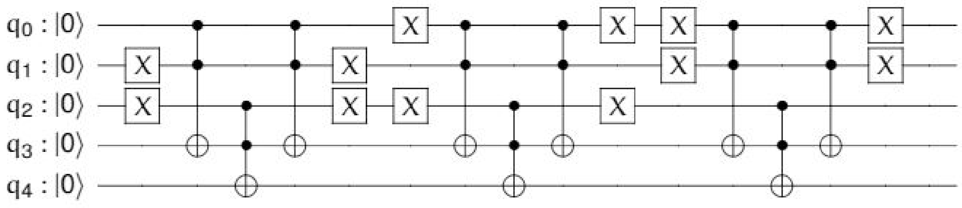


Figure 8.4: A symmetric function as expressed in eq. [8.4] can be realized as quantum circuit as seen in this figure. Some gates in this circuit are redundant, but they are included in order to make the functionality more obvious.

As the start and the end nodes are now realized as quantum circuits, all other nodes in-between must be transformed into quantum circuits. The nodes in-between start and target *B*, *C*, *D*, and *E* are implemented as symmetric functions S^2 with additional negation of all literals. It must be a S^2 function with additional negations as exact two edges or no edges are valid for such an in-between node. When the node is included in a valid path, one edge must go to the node and one edge must move away from the node. Selecting one, three or more edges on an in-between node can never be considered as a path. The negation is included, as it is also possible that the valid path does not include the node at all. Considering node *D* of the example *graph_1*, eq.[8.5] represents the function of this node.

$$I_D^2 = S_D^2(e_2, e_3, e_5) + \overline{e}_2\overline{e}_3\overline{e}_5 \tag{8.5}$$

Eq. [8.6] further decomposes the eq. [8.5].

$$I_D^2 = e_2e_3\overline{e}_5 + e_2\overline{e}_3e_5 + \overline{e}_2e_3e_5 + \overline{e}_2\overline{e}_3\overline{e}_5 \tag{8.6}$$

A S^2-function for the node *D* without an additional negation would only handle the case that the node has been visited which is useful concerning problems like travelling salesman as in Gao et al. (2020) but not with the shortest path problem. Fig. [8.5] shows the oracle of the eq.[8.6] transformed into a quantum circuit.

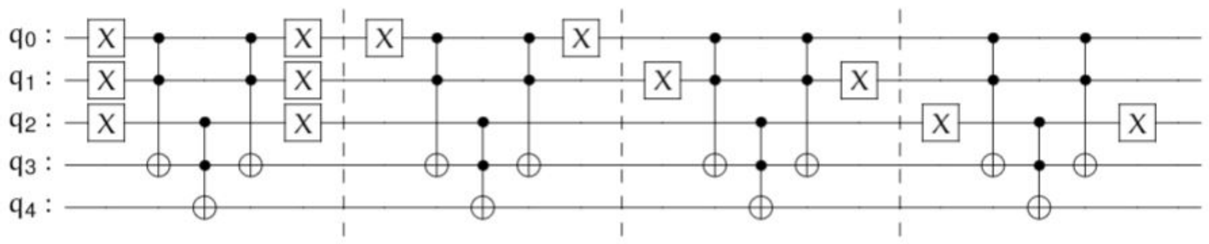


Figure 8.5: The nodes in-between start and target node need an additional negation part to their circuits which is seen in the first part. In this circuit a $I_D{}^2$ with additional negation is realized.

Following, an enumeration of the nodes and their corresponding functions:

- $A \mapsto S_A{}^1$
- $B \mapsto I_B{}^2$
- $C \mapsto I_C{}^2$
- $D \mapsto I_D{}^2$
- $E \mapsto I_E{}^2$
- $F \mapsto S_F{}^1$

A path is considered to be valid if all symmetric functions are satisfied. For this purpose, an AND-circuit will be designed to check if the found path satisfies all symmetric functions. In Fig. [8.6] the gates illustrate all six symmetric functions representing a node each.

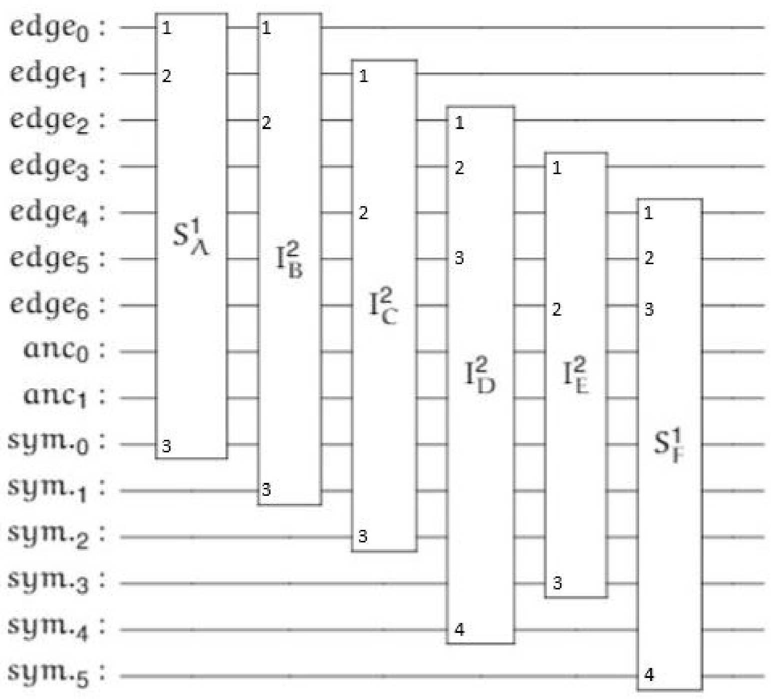


Figure 8.6: The input qubits of each gate are its edges and a symmetric qubit. Gate S^1_A has as input qubits *edge$_0$*, *edge$_1$*, and *smy.$_0$* and gate I^2_C has as input qubits *edge$_1$*, *edge$_4$*, and *smy.$_2$*.

Only if the entire *sym*-register has the value $|1\rangle$ the path is valid. An AND-circuit checks if all conditions are satisfied. Fig. [8.7] illustrates the designed AND-circuit.

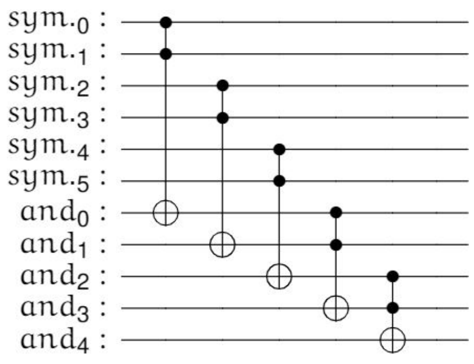


Figure 8.7: The information, if the founded path is valid or not, is encoded in qubit *and$_4$*. For a valid path the qubit *and$_4$* is $|1\rangle$, for every other path it is $|0\rangle$.

It must be mentioned that the circuits represented here are not constructed with respect of low quantum computational costs. The entire first half of the Grover oracle for *graph_1* can be seen in Fig. [8.8].

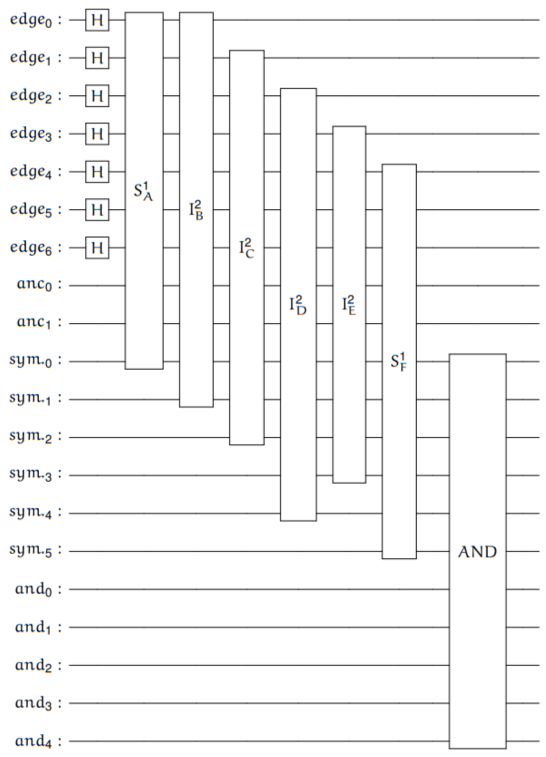


Figure 8.8: The entire Grover oracle will be applied in the Grover algorithm twice, once as in this figure and a second time mirrored. The mirrored version of this oracle is necessary to reinitialize all qubits.

As Fig. [8.8] defines the Grover oracle, the next step designs the diffusor which will is needed for the third step of the Grover algorithm, the flip around average. The diffusor is independent of the path itself and its design is only defined by the number of qubits that the circuits has in superposition. As only the edges are measured, the diffusor will look as in Fig. [8.9].

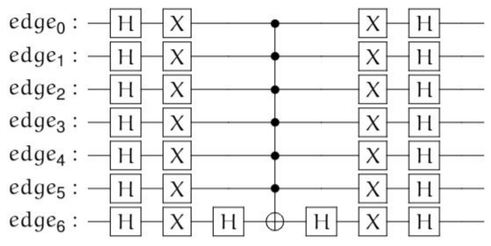


Figure 8.9: The diffusors' construction is independent of the path and will only be applied onto the edge-register. The diffusor function is to flip every state around the average value as described in section 5.3. The gate in the middle with the six control-qubit was customized in Qiskit. To realize this gate, it must be decomposed in standard gates, such as in section 3.7.

Fig. [8.10] shows the entire Grover algorithm which finds a valid path.

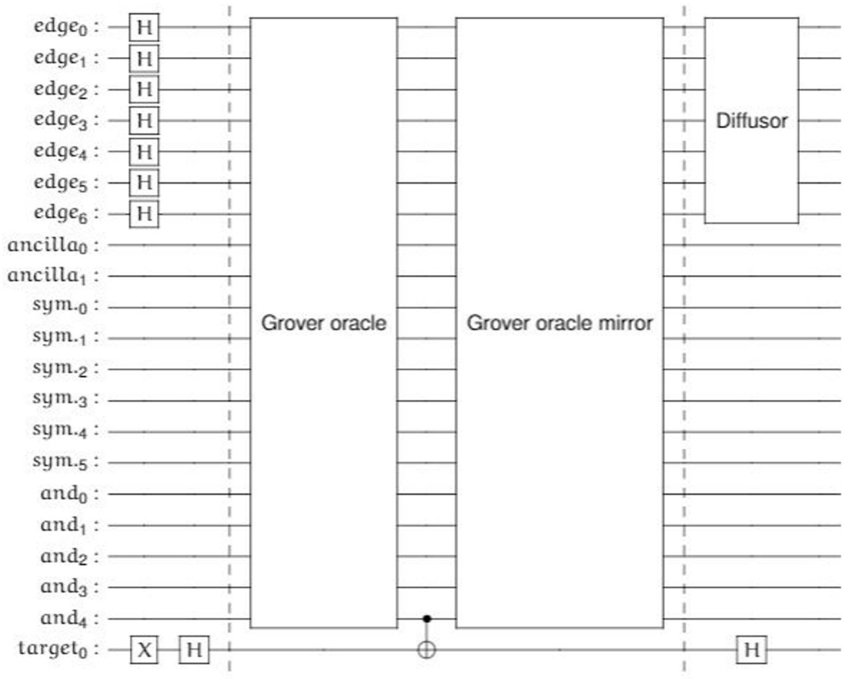


Figure 8.10: The dashed lines split the entire Grover algorithm into three parts. The first part is the initialization and sets the edge-register and the target qubit in superposition. The second part is the Grover oracle and its mirrored version. Between the two oracles the target qubit and the and_4 qubit will be connected with an Fredkin gate. The last part includes the diffusor a Hadamard gate on the target gate. The second and the third part will be repeated in order to increase the target states amplitudes.

As the main Grover circuit is now built, it will be used to solve four different problems such as:

- Does a path exist,
- What is the path,
- How many edges does the path have,
- What is the shortest path.

8.1 Path Existence

The existence of a valid solution can be controlled with the quantum counting algorithm (Brassard et al., 1998). This algorithm uses the inverse QFT on the Grover algorithm. At this part of the problem it is not interesting what path the start and target node connects but if in general a path exists that validates all conditions. For the application of quantum counting *graph_1* is too big, therefore, *graph_2* will be used to demonstrate its functionality. The Grover

circuit for *graph_2* will be designed like the Grover circuit of *graph_1*.
Fig. [8.11] illustrates the Grover circuit for *graph_2* without the initialized superposition.

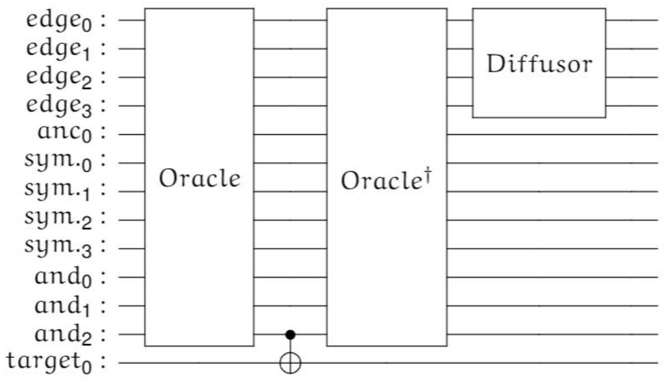


Figure 8.11: This Grover circuit has no initialization, but only the Grover oracle and the diffusor. It was created with the same methodology as the oracle and diffusor in Fig. [8.10].

Fig. [8.12] shows the entire circuit for the quantum counting.

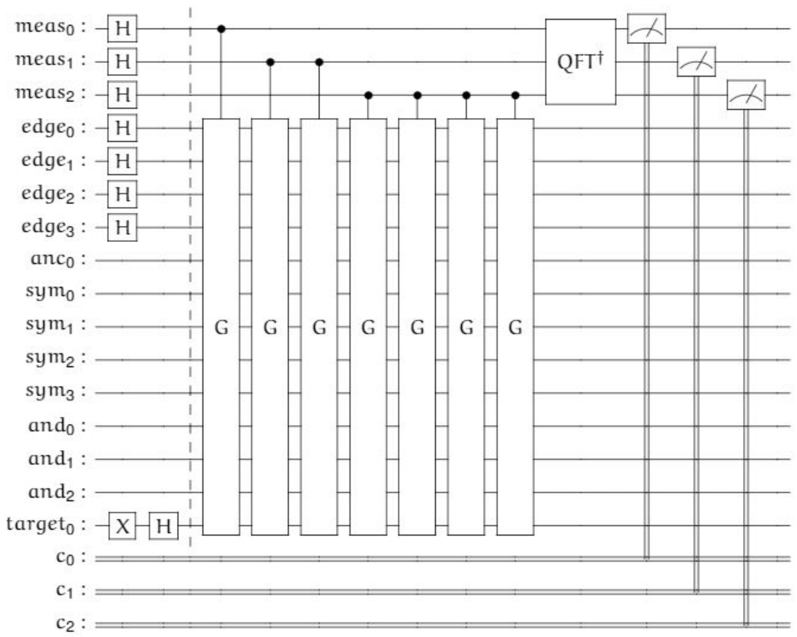


Figure 8.12: The gates with *G* inside represent the Grover oracle and the diffusor. The first three qubits are called the counting qubits as the solution will be stored on those qubits. After the phase kick-back of the *G* gates, the inverse QSVM maps the solution in order to make it observable.

Fig.[8.13] shows the solution of the quantum counting circuit from Fig. [8.12].

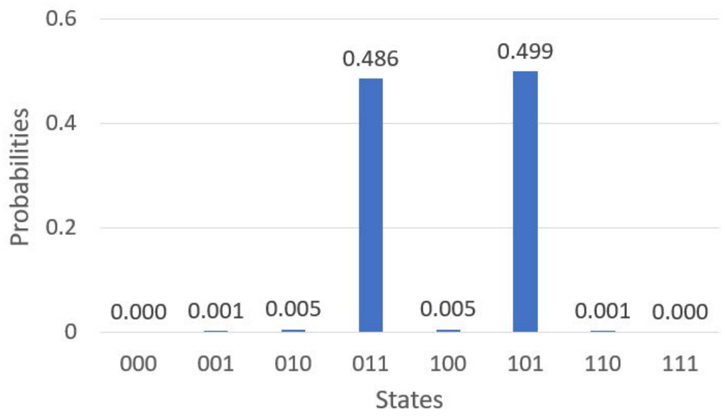


Figure 8.13: From the result of the quantum counting circuit in this figure, it is not possible to conclude how many solutions the graph has. The numbers correspond to the eigenvalues $e^{i\theta}$ and $e^{-i\theta}$.

In order to extract the information about the number of solutions *x* eq. [8.7] is used with *n* being the number of edges and θ the angle of the eigenvalues.

$$x = 2^n - \sin(\frac{\theta}{2})^2 \tag{8.7}$$

In this case, *n* was 4 and θ was 2.36, therefore, eq.[8.8] calculates the number of solutions.

$$x = 16 - \sin(\frac{2.36}{2})^2 = 2.3 \tag{8.8}$$

A weak point of this method is that it will find loops. A path will be considered valid if a path between start and target node is found even though a loop of in-between nodes is included. As an example the *graph_1* from the beginning of this section will be modified in Fig. [8.14] with an additional edge e_7 between nodes *B* and *E*.

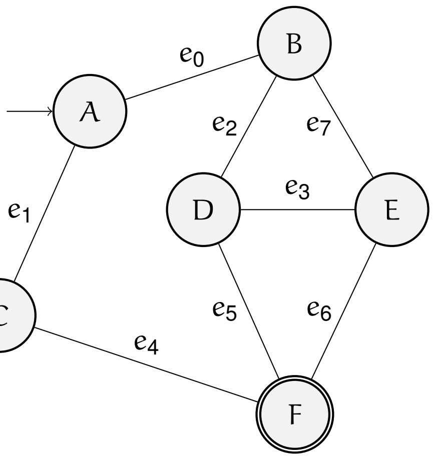


Figure 8.14: The edge e_7 between node *B* and *E* was included in comparison to the example graph in Fig. [8.14]. A valid solution of this method that satisfies all condition would be $e_1 e_4 \overline{e}_2 \overline{e}_3 \overline{e}_7$ which is not a path from start to target node.

8.2 Path Location

After validating that the Grover algorithm has a solution, it is important to measure the exact solution. In this application *graph_1* is the example graph. After calculating the amount of solutions it is possible to calculate the amount of evaluations of the Grover oracle and the diffusor with eq. [8.9].

$$O(\frac{\pi}{4}\sqrt{\frac{2^E}{M}}) \begin{cases} O(9) \text{ if M = 1} \\ O(6) \text{ if M = 2} \\ O(5) \text{ if M = 3} \end{cases} \tag{8.9}$$

In the Grover oracle from Fig.[8.10] the *edges*-register will be measured to get a valid path when measuring. Fig. [8.15] shows the distributed probabilities after one iteration.

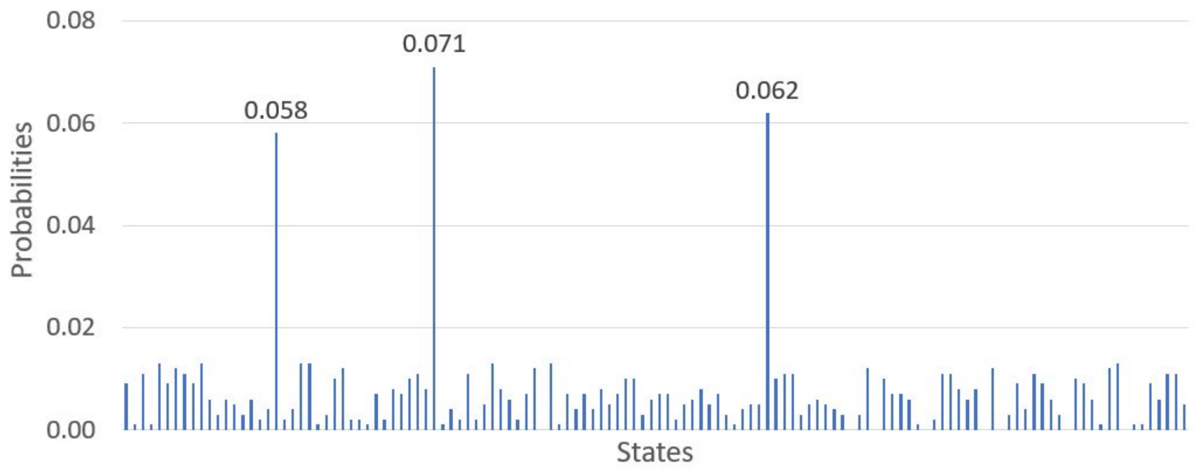


Figure 8.15: After one iteration three possible paths have higher probabilities than the other path constellations. These probabilities belong to the three valid paths and the results are from a real quantum computer.

After three iterations of the Grover oracle and diffusor the probabilities are distributed as in Fig. [8.16].

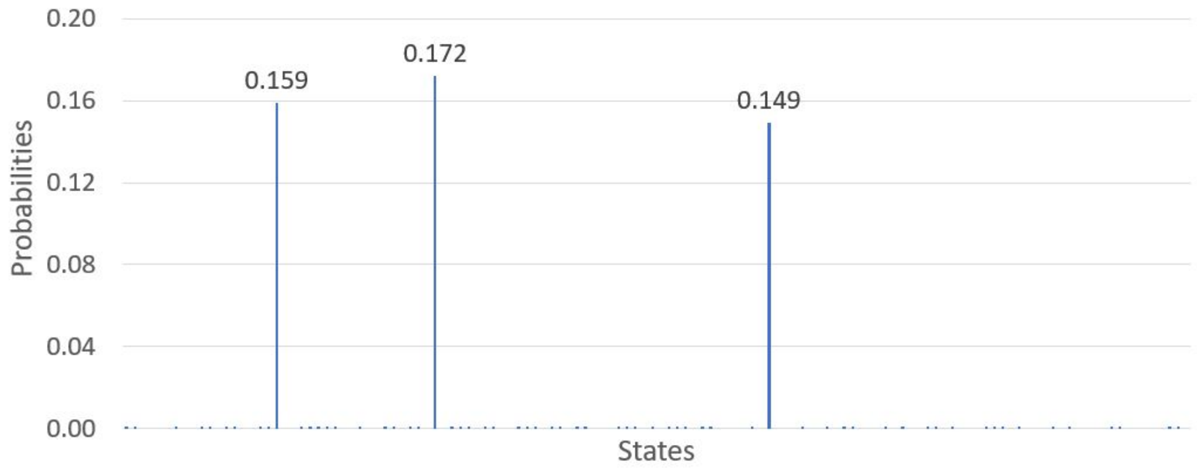


Figure 8.16: After the third iteration the possibility of measuring a path that is not valid is very low. The amplitudes of the valid paths did further increase.

After the fifth iteration, only the three solution paths are left, as seen in Fig. [8.17].

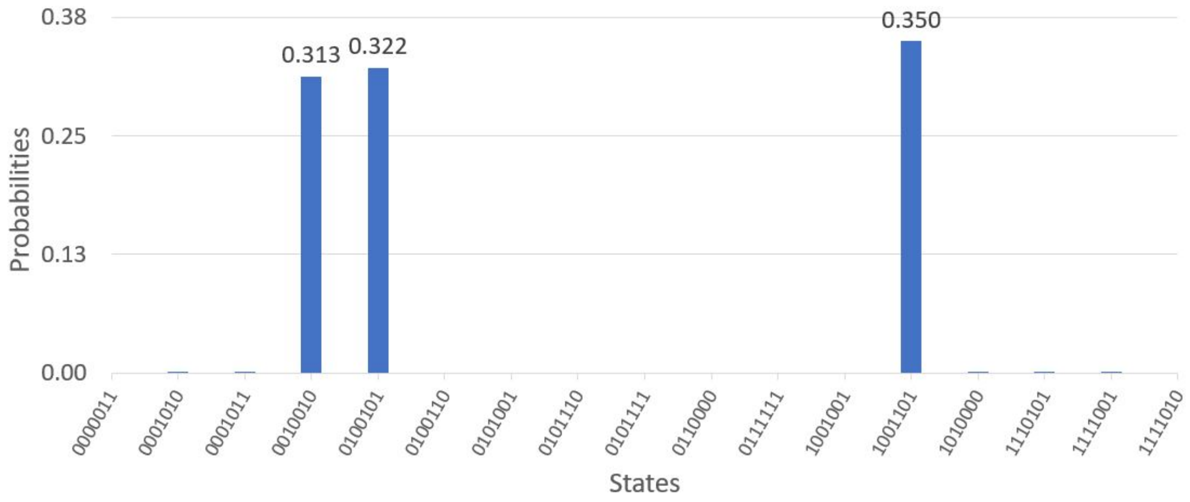


Figure 8.17: The binary strings on the x-axis represent the edges in order like $|e_6e_5e_4e_3e_2e_1e_0\rangle$. Measuring after the fifth iteration will definitely result in one of the three possible paths.

In addition to this method, the oracle can be adapted after a path was found. If a valid path was found, the oracle can be adjusted in order to not consider this path as a valid solution again. This condition will be included to the circuit of Fig. [8.7]. The path condition and all node conditions must be satisfied to be considered as valid path. As example the path from node *A* to *B*, further to *F* will be considered to be already known. The valid path over nodes *A*, *B*, and *F* is realized with qubits e_1 and e_4 being active $|1\rangle$ while the other qubits are ground state $|0\rangle$. Fig. [8.18] expresses the additional condition in the quantum circuit notation.

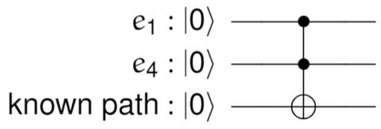


Figure 8.18: This condition will be added to the Grover oracle in order to no longer consider the path as valid. Therefore, the other two solutions will be found with a higher probability.

The additional condition-circuit will be added after the AND-circuit of Fig. [8.7]. Fig. [8.19] shows that extending the Grover oracle with this condition the *and*$_4$ qubit must be active $|1\rangle$ and the *known_path* qubit in Fig. [8.18] must be negative $|0\rangle$ to consider the found path as valid solution.

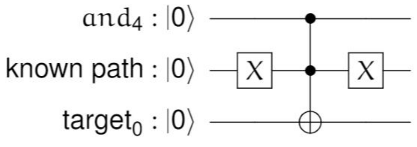


Figure 8.19: Adding this condition to the oracle, only two paths are considered valid. When another path is found, a second condition will be added to this circuit.

According to eq. [8.9] the circuit with two solutions must be iterated six times to get one of the remaining paths. Fig. [8.20] shows the results after six iterations of this Grover algorithm.

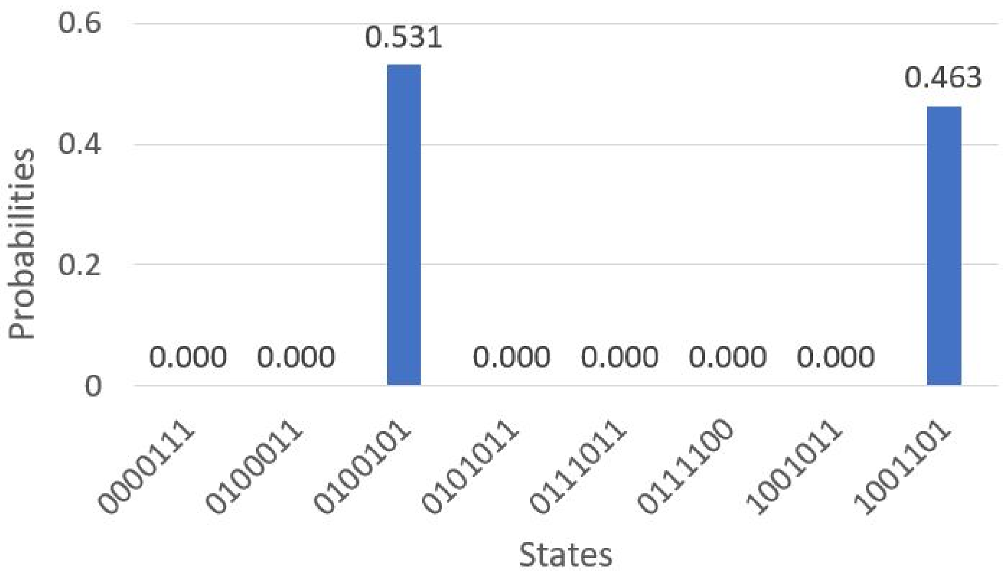


Figure 8.20: Only two paths are now considered to be valid. The Grover oracle and diffusor have to be iterated six times as only two possible solutions are left as seen in eq. [8.9].

8.3 Path Thresholding

Sometimes it is not important to find the shortest path, but a path under a certain threshold. As path thresholding for the graph of Fig. [8.14] would be beyond Qiskit's computational capacity, the smaller graph, *graph_2*, will be considered in this example.
A valid path can be found with the oracle of the previous subsection. The Grover oracle will be now extended to also consider the path cost while searching for a target state. For now, it will be assumed that all edges have the same cost 1. To realize a graph where the edges have different cost would exceed Qiskit's capabilities. The quantum adder-circuit for the example graph consists of three half-adder and one full-adder. Fig. [8.21] and Fig. [8.22] illustrate the

half-adder and the full-adder and Figgatt et al. (2019) describes it in more detail.

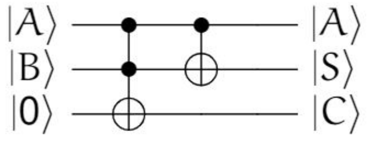


Figure 8.21: A quantum half-adder takes two variables *A* and *B* and calculates the sum of the current order $|S\rangle$ and carries out potential input for the next higher order $|C\rangle$.

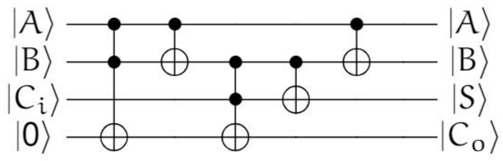


Figure 8.22: A quantum full-adder takes three variables *A* and *B* and a *carry-in* variable from the lower order $|C_i\rangle$. Such a full-adder calculates the sum of the current order $|S\rangle$ and carries out potential input for the next higher order $|C_o\rangle$.

Fig. [8.23] shows the entire cost circuit that calculates the path cost.

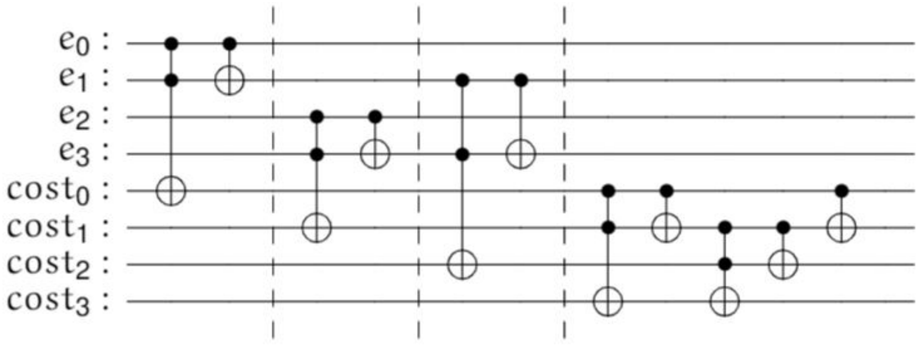


Figure 8.23: The three quantum half-adder and the quantum full-adder are divided with dashed lines. The costs will be encoded on the qubits e_3, $cost_2$, and $cost_3$.

The edges are counted and will be compared to a threshold in the next step. The comparator circuit compares two binary strings to each other. Beginning from the bit with the highest order, the digits will be compared. The first time the qubits differ, a solution is found. The string having a 1 at this position is higher than the other string. Fig. [8.24] shows the result of the comparison encoded on the qubits anc_0 and anc_1.

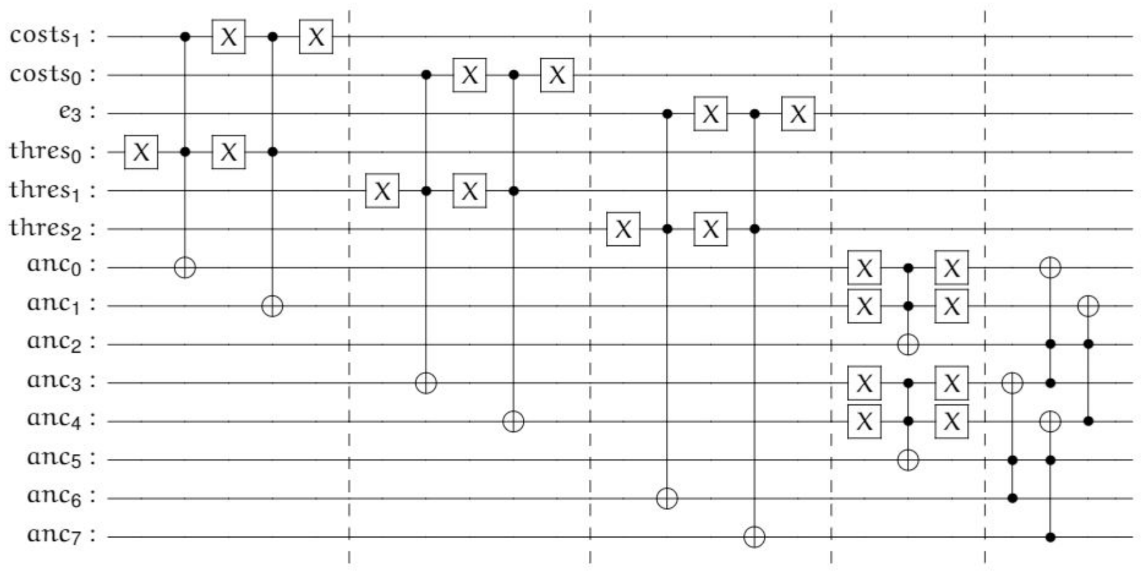


Figure 8.24: The output from the cost circuit will be compared with the threshold in the *thres*-register. The result of this comparison will be stored on qubits *anc₀* and *anc₁*.

With *a* representing the measured path and *b* representing the threshold, the results of this compare-circuit can be interpreted as followed:

$$|anc_0, anc_1\rangle \begin{cases} \text{if } |00\rangle \longrightarrow a == b \\ \text{if } |01\rangle \longrightarrow a > b \\ \text{if } |10\rangle \longrightarrow a < b \end{cases}$$

The new Grover oracle will be assembled with following circuits:

- symmetric
- cost
- compare

Fig. [8.25] shows the entire Grover oracle and diffusor.

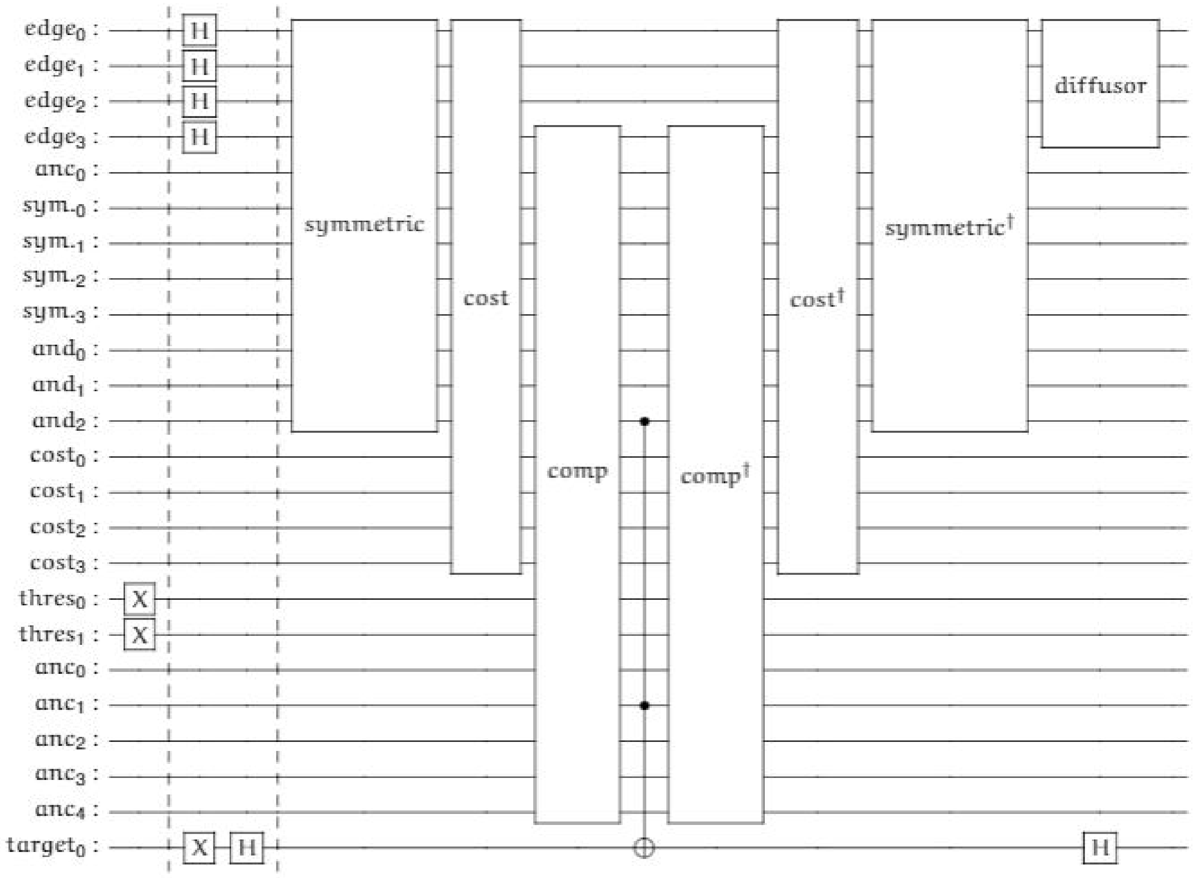


Figure 8.25: The comparator-circuit does only compare the first two digits of the cost-string. Comparing all three digits would exceed the limitations of Qiskit. In the first part of this circuit the threshold is set to only consider the path with two edges as valid. In the second part the superpositions will be initialized. The Toffoli-gate in the middle of the oracle flips only the target state that satisfies all symmetric-conditions and it below the threshold.

Fig. [8.26] shows the result after one iteration of the Grover oracle and diffusor.

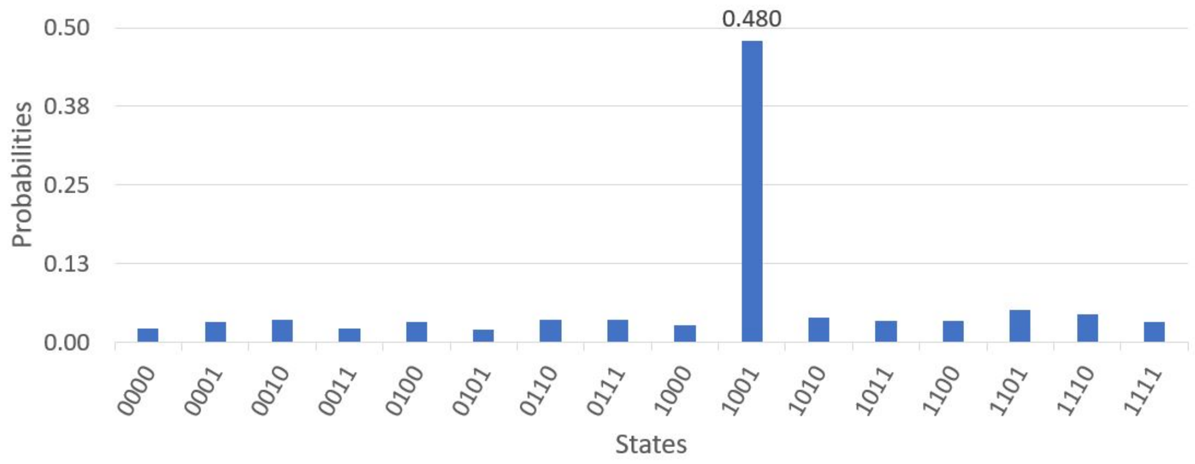


Figure 8.26: The only path from the example graph of Fig. [8.2] that satisfies the threshold condition is the path with active edges e_0 and e_3, which will be measured after the first iteration with the probability of 0.48.

The path threshold methodology can be extended with edges of different weights. It is more realistic to weight such an edge individually as some edges are longer or steeper and have therefore higher cost. As this Grover oracle would be beyond Qiskit's capacities, the extension will be discussed as standalone solution. For the weight extension the edges of the graph in Fig. [8.2] have following weights:

- $e_0 = 3$
- $e_1 = 1$
- $e_2 = 1$
- $e_3 = 2$

The Toffoli gate is used in order to activate the weight of the edge, if it is included in the chosen path as in Fig. [8.27].

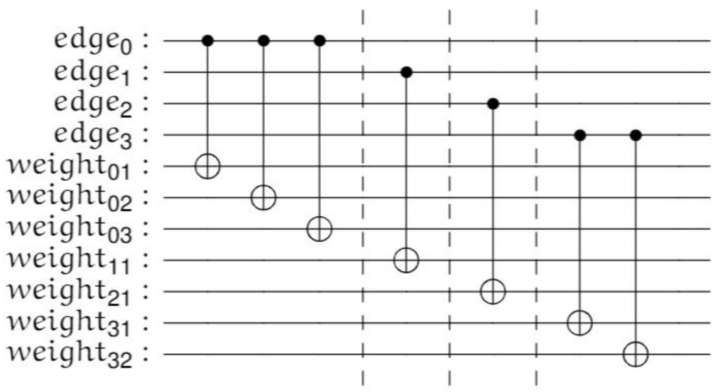


Figure 8.27: The four edges are separated with the dashed lines. According to the current path, the weight are either activated or not. After this circuit the weight will be calculated in an adder circuit such as in Fig. [8.23].

8.4 Shortest Path Methodology

The shortest path methodology will be composed of the previous developed solution in subsection 8.1, 8.2, and 8.3.
In the first instance, a threshold will be set. For the *graph_1*, the thresholds will be stored on a classical computer in a list. Eq. [8.10] shows the list of thresholds *thresh* beginning from the lowest possible path-cost.

$$\text{thresh} = [001, 010, 011, ...] \tag{8.10}$$

The list in eq. [8.10] will be looped beginning from the lowest threshold until the quantum counting finds a solution. The total path cost must be below the current threshold to be considered as valid. The numbers of solutions will be calculated with the quantum counting approach of subsection 8.1. If the threshold is found, which includes for the first time one or more possible solutions, eq. [8.11] determines the iterations *i* of the Grover oracle and diffusor.

$$O(\frac{\pi}{4}\sqrt{\frac{2^E}{M}}) \tag{8.11}$$

In eq. [8.11] *E* is the number of edges and *M* the number of possible solutions that satisfies the threshold and symmetry condition.
In the final step, the Grover oracle and diffusor will be iterated $O(\frac{\pi}{4}\sqrt{\frac{2^E}{M}})$ times with a circuit such as in Fig. [8.25]. This methodology requires a bigger number of qubits and is therefore not executable on a Qiskit simulation. In general this method offers a fast search algorithm if a path exist between start and target node. With the additional *cost*- and *comp*-circuit it can be also verified if a path exists, that does not exceed certain costs.

9 Quantum Algorithm Design for NISQ Technology

This chapter differs from the two previous as it handles not only problems in robotics. At the beginning the author improves the QSVM-algorithm from Yang et al. (2019*b*) and compares this approach with the quantum Swap-Test. As the quantum Swap-Test is superior with respect to the NISQ-technology, the last two section try to expand its applicational potential.

9.1 QSVM and Inner-Product

This section compares two approaches of ML for quantum computing. Subsection 5.7.2 already described the first method QSVM in more detail. Yang et al. (2019*b*) introduced a QSVM-circuit that can distinguish between the numbers 6 and 9. The second approach was created within this Master Thesis and uses the Inner-Product property of the Hilbert space to distinguish between the numbers 6 and 9.

9.1.1 Preprocessing of Data

In comparison to Yang et al. (2019*b*) this Master Thesis adjusted the preprocessing to increase the accuracy of the QSVM-circuit from original 96% to 100%. Fig. [9.1] shows the training samples for both methods, QSVM and Inner-Product, the *Times New Roman* font of the numbers 6 and 9.

(a) Times new roman 6. (b) Times new roman 9.

Figure 9.1: These pictures are the training examples in both methods. With only one train example of each group the methods separate 200 testing samples.

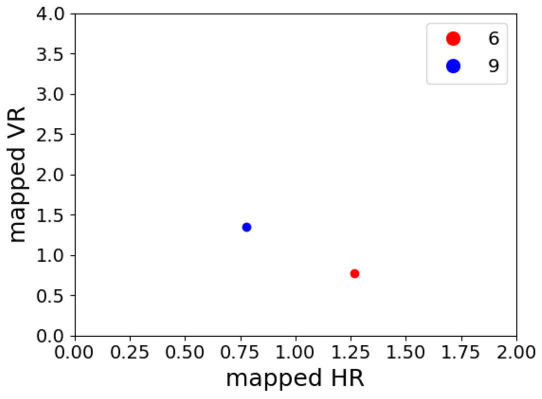
With only one training sample for each group the methods distinguish between 200 different handwritten numbers of 6 and 9 from the dataset Escrivá (2012).
To make the method robust, it must be independent of the numbers' location in the figure. Consider two examples of the number 6 written on a paper. The performance of this method must be independent from the location of the number on this paper. To distinguish between number 6 and 9 the horizontal and vertical pixel ratio will be compared. Only the black pixels will be counted that belong to the actual number, not the white pixel. When considering horizontal- and vertical-ratio of the entire paper, counting all pixels above the middle comparing them with the pixels below the middle, the ratios are dependent on the numbers horizontal and vertical location. To avoid the location dependency, in the first step the horizontal- and vertical minimum and maximum black pixels are found in the figure and the new region of interest will be right around the number.
After the number is focused, the horizontal and vertical ratios are calculated. In order to calculate the horizontal ratio, the black pixels left of the middle and right of the middle are counted. Eq. [9.1] calculates the horizontal ratio *HR*.

$$HR = \frac{\text{left pixels}}{\text{right pixels}} \tag{9.1}$$

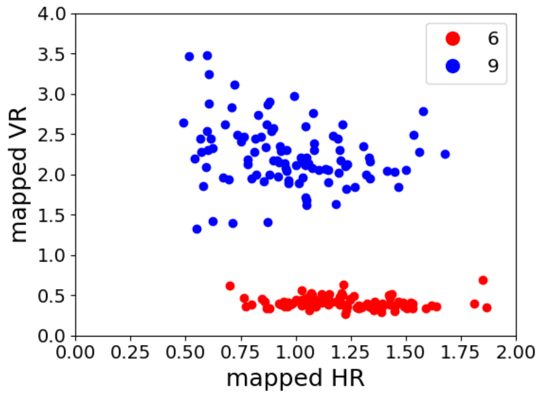
Eq. [9.2] calculates the vertical ratio *VR* in a similar way, counting all black pixels above the middle and below the middle.

$$VR = \frac{\text{upper pixels}}{\text{lower pixels}} \tag{9.2}$$

Fig. [9.2] shows the *HR* and *VR* distribution of the train examples and the testing examples.



(a) Distribution of training examples



(b) Distribution of testing examples

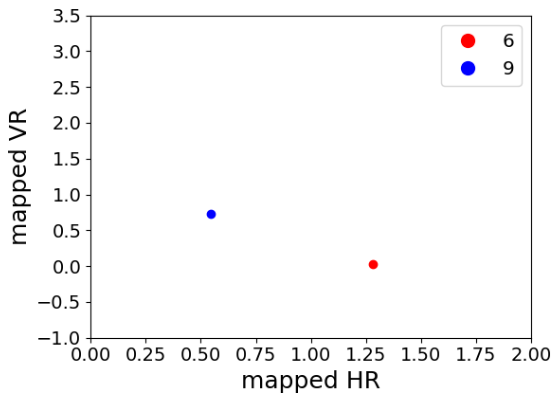
Figure 9.2: In Fig. [9.2a] both training examples are distributed according to their horizontal and vertical ratios. In Fig. [9.2a] 100 examples of each number 6 and 9 of the OCR datasets are distributed according to their horizontal and vertical ratios.

Regarding the distribution of the 200 samples in Fig. [9.2b], the data will be mapped linear in the next step. This step increases the possibility of labeling the data correctly. In the linear mapping step it is tried to shift the data of one group into a different quadrant. Eq. [9.3] linearly maps the training data and the testing data horizontal and eq. [9.4] linearly maps the data according to the vertical ratios.

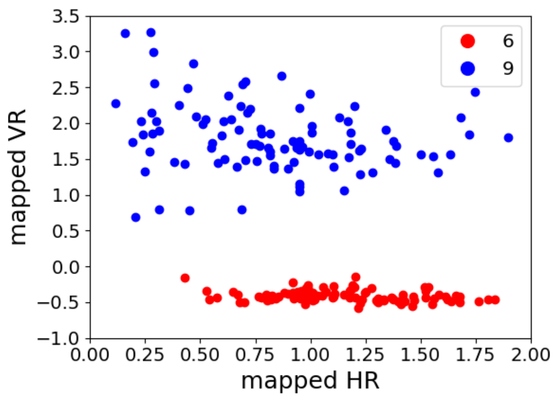
$$HR_{mapped} = HR * 1.5 - 0.62 \tag{9.3}$$

$$VR_{mapped} = VR * 1.2 - 0.9 \tag{9.4}$$

Fig. [9.3] shows how the distribution of Fig. [9.2] changes after the linear mapping.



(a) Linear mapped distribution of training examples

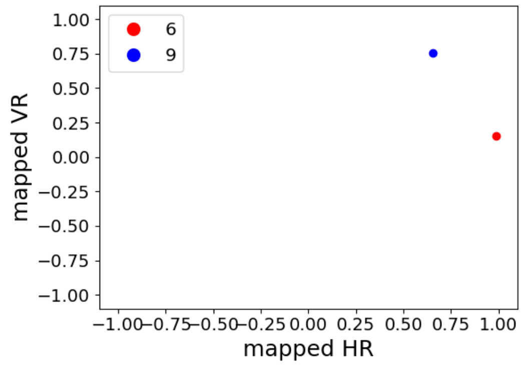


(b) Linear mapped distribution of testing examples

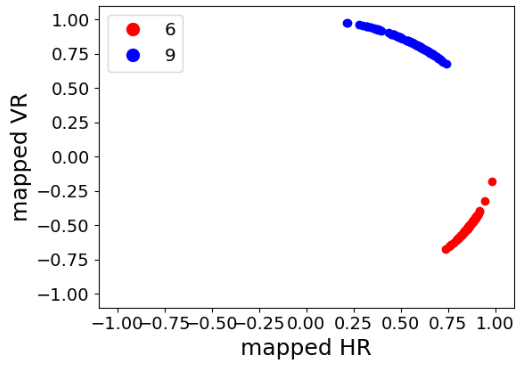
Figure 9.3: The data are mapped in order to separate each group into a different quadrant. With this separation, the gap between the two groups will increase, which makes the labeling easier.

After the linear mapping of training data and the testing data, the next step normalizes them. Fig. [9.4] shows that the data points have the distance 1 to the origin after eq. [9.5].

$$\begin{aligned} \mathrm{norm}_{HR} &= \sqrt{\frac{HR_{mapped}}{|HR_{mapped}|+|VR_{mapped}|}} \\ \mathrm{norm}_{VR} &= \sqrt{\frac{VR_{mapped}}{|HR_{mapped}|+|VR_{mapped}|}} \end{aligned} \tag{9.5}$$



(a) Normalized training examples



(b) Normalized testing examples

Figure 9.4: Due to the linear mapping, the data points of 6 and 9 are separated with a bigger gap in-between. The training sample of 9 is a better representation of its testing data than the training sample of 6 and its testing data.

As the sphere of a qubit also has the radius 1, the normalized data points can be mapped onto the sphere of a qubit. In order to map the normalized data points onto the sphere, the angle θ between the x-axis and the data point has to be calculated as follows:

- if $\text{norm}_{HR} > 0$ and $\text{norm}_{VR} > 0$:

  $\theta = \arctan(\frac{norm_{VR}}{norm_{HR}})$

- if $\text{norm}_{HR} < 0$ and $\text{norm}_{VR} > 0$:

  $\theta = \arctan(\frac{norm_{VR}}{norm_{HR}}) + 3.14$

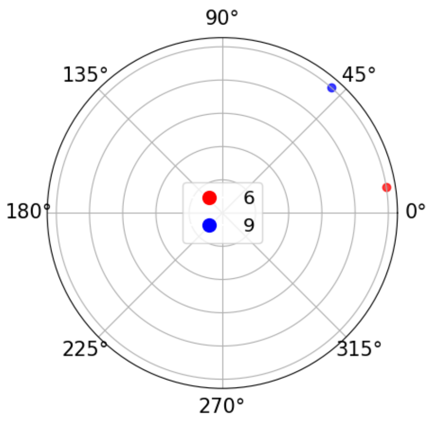
- if $\text{norm}_{HR} < 0$ and $\text{norm}_{VR} < 0$:

  $\theta = \arctan(\frac{norm_{VR}}{norm_{HR}}) + 3.14$

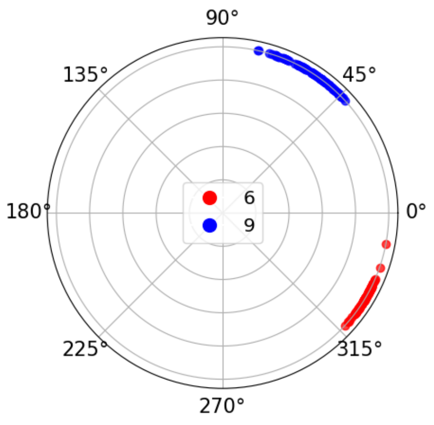
- if $\text{norm}_{HR} > 0$ and $\text{norm}_{VR} < 0$:

  $\theta = \arctan(\frac{norm_{VR}}{norm_{HR}}) + 3.14$

With the calculated angles θ, the rotation-gate maps the qubits state onto the data point. Fig. [9.5] shows the mapping of the data points onto the circumference.



(a) Angles of training examples



(b) Angles of testing examples

Figure 9.5: The orientation of the initial state vector is towards 0° and a unitary operator transforms it towards the data point. The angles θ decide how much the gate rotates the initial vector.

9.1.2 Machine Learning using Quantum Support Vector Machine

The input signals for both methods QSVM and Inner-Product are the same angles calculated in subsection 9.1.1. The general method of QSVM was introduced in section 5.7.2, therefore, this subsection only explains the adaption of the kernel matrix in more detail. QSVM uses the phenomena phase-kickback described in section 5.4 and the HHL-algorithm described in 5.6. The QSVM-algorithm of this section is an adaptation of the LS-SVM-method described in section 5.7.1. Yang et al. (2019*b*) designed the circuits general structure was and the contribution of this Master Thesis regarding QSVM is the improved preprocessing and the adjusted the kernel matrix in order to achieve 100% accuracy with labeling the two numbers.
The complete circuit for the QSVM-circuit is illustrated at the end of this section and the dashed lines split this circuit in three parts. Eq. [9.6] shows what the first part of this circuit does.

$$A\vec{x} = \vec{b} \longrightarrow \vec{x} = A^{-1}\vec{b} \tag{9.6}$$

This matrix inversion is done using the HHL-algorithm introduced in section 5.6. Fig. [9.6] shows the circuit that realizes this matrix inversion.

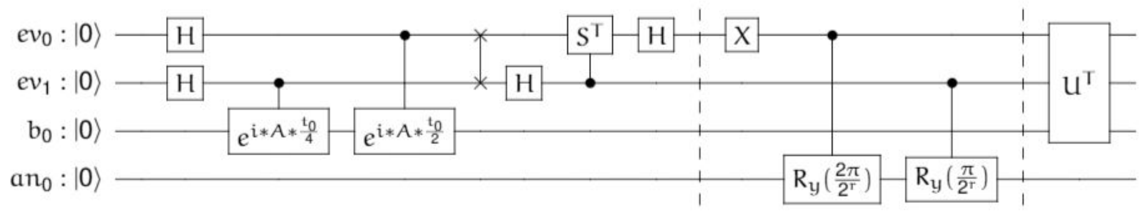


Figure 9.6: Subsection 5.7.2 already introduced this circuit. In order to distinguish the numbers 6 and 9 the kernel matrix *A* must consider the preprocessed ratios calculated in the previous subsection.

With t_0 equals to 4, the expression in the first two controlled-unitary operators in Fig. [9.6] changes to e^{2iA} and e^{iA}. According to Yang et al. (2019*b*), t_0 equals to 4 provides the most stable evaluation. Fig. [9.7] illustrates the second part of the circuit that is responsible for the training process.

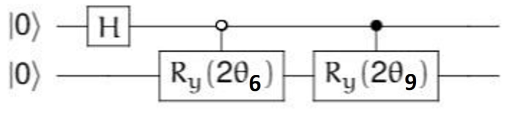


Figure 9.7: This circuit is not realizable in Qiskit as it is not possible to design a $|0\rangle$-controlled gate. This notation is for $|0\rangle$-controlled gates that will execute the transformation only if the control state is $|0\rangle$. The angle θ_6 represents the training angle of the number 6 and the angle θ_9 represents the number 9.

The circuit of Fig. [9.7] only loads the training data if the previous step obtained the matrix

inversion. In this part the ratios of the *Times New Roman* numbers will be used to calculate the kernel. Yang et al. (2019*b*) introduces three different methods to obtain the matrix *A*. Following example explains the manual method to show the general methodology of calculating the kernel matrix. Fig. [9.7] is the basis to calculate the kernel by hand.
This calculation start at the leftmost gate and continues towards the right side of the circuit in Fig. [9.7]. Eq. [9.7] shows that in the first step the Hadamard gate will affect the first qubit.

$$\mathrm{H}\,|00\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |0\rangle) \tag{9.7}$$

The $|0\rangle$-controlled gate transforms $|0\rangle \otimes |0\rangle$ to $|0\rangle \otimes |norm_6\rangle$ and $|1\rangle \otimes |0\rangle$ to $|1\rangle \otimes |norm_9\rangle$ with norm$_6$ and norm$_9$ being the normalized horizontal and vertical ratios of the training numbers calculated in subsection 9.1.1. The vectors norm$_6$ and norm$_9$ can be also written as:

- $norm_6 = \begin{bmatrix} norm_{6HR} \\ norm_{6VR} \end{bmatrix}$
- $norm_9 = \begin{bmatrix} norm_{9HR} \\ norm_{9VR} \end{bmatrix}$

Eq. [9.8] shows the vector form of eq. [9.7].

$$\psi = \frac{1}{\sqrt{2}}(\begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} norm_{6HR} \\ norm_{6VR} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} norm_{9HR} \\ norm_{9VR} \end{bmatrix}) \tag{9.8}$$

Eq. [9.9] shows how to calculate the tensor product described in subsection 3.8.

$$\psi = \frac{1}{\sqrt{2}}(\begin{bmatrix} norm_{6HR} \\ norm_{6VR} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ norm_{9HR} \\ norm_{9VR} \end{bmatrix}) = \frac{1}{\sqrt{2}}\begin{bmatrix} norm_{6HR} \\ norm_{6VR} \\ norm_{9HR} \\ norm_{9VR} \end{bmatrix} \tag{9.9}$$

The results for the horizontal and vertical ratios of the training numbers obtained in subsection 9.1.1 are:

- $norm_{6HR} = 0.988$
- $norm_{6VR} = 0.154$
- $norm_{9HR} = 0.656$
- $norm_{9VR} = 0.755$

In the next step eq. [9.10] calculates the Inner-Product of the state ψ from eq. [9.9], also known as the density matrix.

$$|\psi\rangle\,\langle\psi| = \begin{bmatrix} 0.488 & 0.076 & 0.324 & 0.372 \\ 0.076 & 0.011 & 0.050 & 0.058 \\ 0.324 & 0.050 & 0.215 & 0.247 \\ 0.372 & 0.058 & 0.247 & 0.285 \end{bmatrix} \tag{9.10}$$

Eq. [9.11] calculates from the density matrix of eq. [9.10] the partial trace of the first qubit which equals to K/*tr*(K).

$$\frac{K}{tr(K)} = \begin{bmatrix} 0.488+0.011 & 0.324+0.058 \\ 0.324+0.058 & 0.215+0.285 \end{bmatrix} = \begin{bmatrix} 0.499 & 0.382 \\ 0.382 & 0.5 \end{bmatrix} \tag{9.11}$$

For the normalized training data the trace of K must be 2, as the two main diagonal values will sum up to 2. Eq. [9.12] bring the *tr*(K) of eq. [9.11] to the right side of the equation in order to get the kernel matrix K.

$$K \approx \begin{bmatrix} 1 & 0.764 \\ 0.764 & 1 \end{bmatrix} \tag{9.12}$$

The matrix inversion oracle and the training oracle together define the hyperplane that separates the data points. As the matrix inversion is only obtained when the *an₀* qubit in Fig. [9.7] is $|1\rangle$, the training oracle of Fig. [9.7] is extended with the control-qubit *an₀*. Fig. [9.8] shows the last part of this circuit, the so-called testing process will label the testing vector either to 6 or 9.

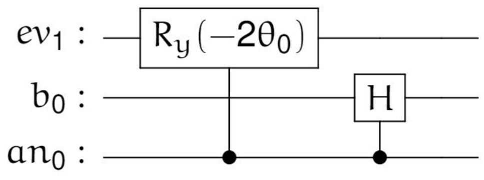


Figure 9.8: The last part of the QSVM-circuit is the testing oracle. In this figure θ_0 is the angle of the currently tested data point. The result will be stored on the qubit an_0.

The entire circuit of Fig. [9.9] evaluates each testing data point of the dataset.

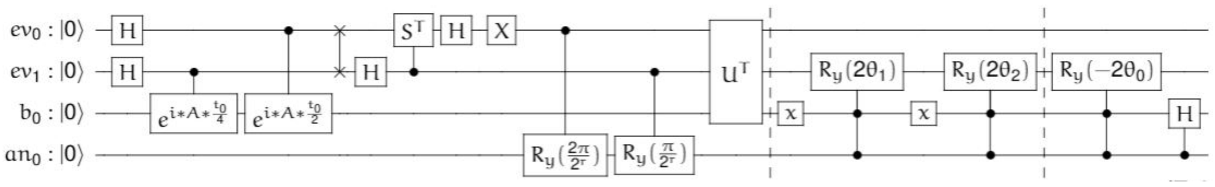


Figure 9.9: The dashed lines split the entire QSVM-circuit in three parts. The first part is the matrix inversion using the HHL-algorithm. The gate U^T mirrors the circuit from the first parallel Hadamard gates until the first Pauli-X gate. The second part is the training part where the training data is loaded. In the last part the testing data will be labeled, and the result is stored in on the ancilla qubit *an₀*.

The evaluation of all 200 data points in the simulation result in a 100% accuracy due to the improved data preparation.

9.1.3 Machine Learning using Swap-Test

This Master Thesis introduces a more trivial approach in comparison to the QSVM using the property of Inner-Product of vectors in Hilbert space. Section 3.4 mentioned that in a Hilbert-space the Inner-Product of two vectors gives information about the similarity of two vectors. The Inner-Product of the same vectors will result in 1 and the Inner-Product of two opposite orientated vectors will result in 0. If two vectors are orthogonal, the result will be $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.
The following example considers two vectors having the state $|0\rangle$. The Inner-Product of these two vectors will be the scalar 1. If one of the two vector transforms towards the state $|1\rangle$ and the other vector remains in state $|0\rangle$, the output of the Inner-Product will transform from 1 to the output of 0 if the two vectors are oriented in opposite directions. The further the vector transforms towards $|1\rangle$ the higher is the likelihood of measuring 0.
The Swap-Test (Wittek, 2014) uses the Inner-Product in calculation and Fig. [9.10] illustrates its circuit.

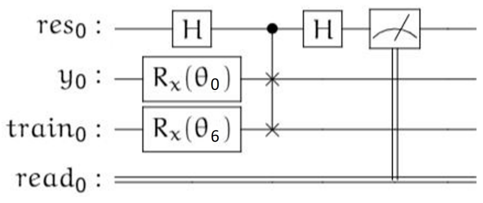


Figure 9.10: The rotation gate transforms the two qubits y_0 and $train_0$ according to the angles calculated in the prepossessing subsection 9.1.1. The Fredkin gate with its control qubit in superposition uses phase-kickback to gain more information about the local phase of the qubits y_0 and $train_0$.

The angle θ_0 is the actual testing data angle and θ_6 is the training data angle of digit 6 (digit 9 uses a different angle) calculated in subsection 9.1.1. When the two vectors y_0 and $train_0$ are equal the measurement of *res_0* qubit results in 0. In case y_0 and $train_0$ are orthogonal the measurement of *res_0* qubit results in 0 in 50% of the time. The following example explains the functionality of the Swap-Test. The initial state of y_0 is $|\phi\rangle$ and of $train_0$ is $|\psi\rangle$, which results in the starting state $|0\phi\psi\rangle$. Eq. [9.13] shows the transformation after the first Hadamard gate.

$$H\,|0\phi\psi\rangle = \frac{1}{\sqrt{2}}(|0\phi\psi\rangle + |1\phi\psi\rangle) \tag{9.13}$$

Eq. [9.14] changes the states $|\phi\rangle$ and $|\psi\rangle$ with the Fredkin gate if the control qubit is $|1\rangle$.

$$\frac{1}{\sqrt{2}}(|0\phi\psi\rangle + |1\psi\phi\rangle) \tag{9.14}$$

With the second Hadamard gate, eq. [9.15] sets the two terms in eq. [9.14] into superposition.

$$\frac{1}{\sqrt{2}}(|0\phi\psi\rangle + |1\phi\psi\rangle + |0\psi\phi\rangle - |1\psi\phi\rangle) = \frac{1}{2} + \frac{1}{2}|\langle\psi|\phi\rangle|^2 \tag{9.15}$$

The evaluation of the Swap-Test of Fig. [9.10] includes 1000 repetitions for both training vectors with each testing data. After the evaluation of the training angles 6 and 9 the similarity is compared. According to the results of the Swap-Tests, the testing data is classified to the training data with the higher similarity. A Qiskit simulation and a real quantum computer named *ibmq_16_melbourne*, consisting of 16 qubits (jaygambetta, 2019), evaluated this method. In both evaluation the Swap-Test archived a 100% accuracy. The Swap-Test on a real quantum computer resulted in an accuracy of 100%, which is an indicator that the method is robust against noise.

9.1.4 Comparison of QSVM and Swap-Test

It was not possible to evaluate the QSVM-circuit on a real quantum computer in order to compare the performance of QSVM and Swap-Test. Therefore, the comparison of both methods includes the decoherence probabilities in order to conclude which method is more robust. The probability of decoherence is dependent on different aspects such as:

- Input state
- Gate type
- Amplitude and Phase damping
- The depth of the quantum circuit

According to the depth of the circuits for the QSVM-method and the Swap-Test-method the calculation time will be compared to conclude which circuit is more exposed to noise and has more decoherence. Tab. [9.1] shows the processing gate times according to Zhang et al. (2019).

Table 9.1: Average time a qubit needs to be processed by different operations using NISQ technology.

	Gate time [μs]	**Variables**
Single Gate	20	τ_S
2-qubit Gate	40	τ_2
Measurement	300 - 1000	τ_M

According to Kim & Choi (2018) a controlled-Swap gate can be effectively decomposed into five single gates and eight 2-qubit gates. The circuit for the Swap-Test in total consists of nine single gates, eight 2-qubit gates and one measurement gate. Eq.[9.16] calculates the theoretical worst-case time.

$$\tau_{tot} = 9\tau_S \mu s + 8\tau_2 \mu s + 1\tau_M \mu s \tag{9.16}$$

$$\tau_{tot} = 9 * 20\mu s + 8 * 40\mu s + 1 * 1000\mu s$$

$$\tau_{tot} = \underline{1.500\mu s}$$

Chien et al. (2013) states that a Toffoli gate can be effectively decomposed into nine single gates and six 2-bit gates. The circuit for the QSVM consist in total of 64 single gates, 47 2-qubit gates and one measurement gate and the worst-case theoretical process time is calculated in eq.[9.17].

$$\tau_{tot} = 64\tau_S \mu s + 47\tau_2 \mu s + 1\tau_M \mu s \tag{9.17}$$

$$\tau_{tot} = 64 * 20\mu s + 47 * 40\mu s + 1 * 1000\mu s$$

$$\tau_{tot} = \underline{4.160\mu s}$$

Fig. [9.11] displays the decomposed circuit of the QSVM-method that was considered while counting the gates. According to eq.[9.16] and eq.[9.17] the calculation time of the QSVM-circuit is more than twice as big as the calculation time of the Swap-Test-circuit. As both circuits perform equally good in classifying the digits, and the calculation time of the Swap-Test-circuit is shorter, the Inner-Product-circuit is superior in this application.

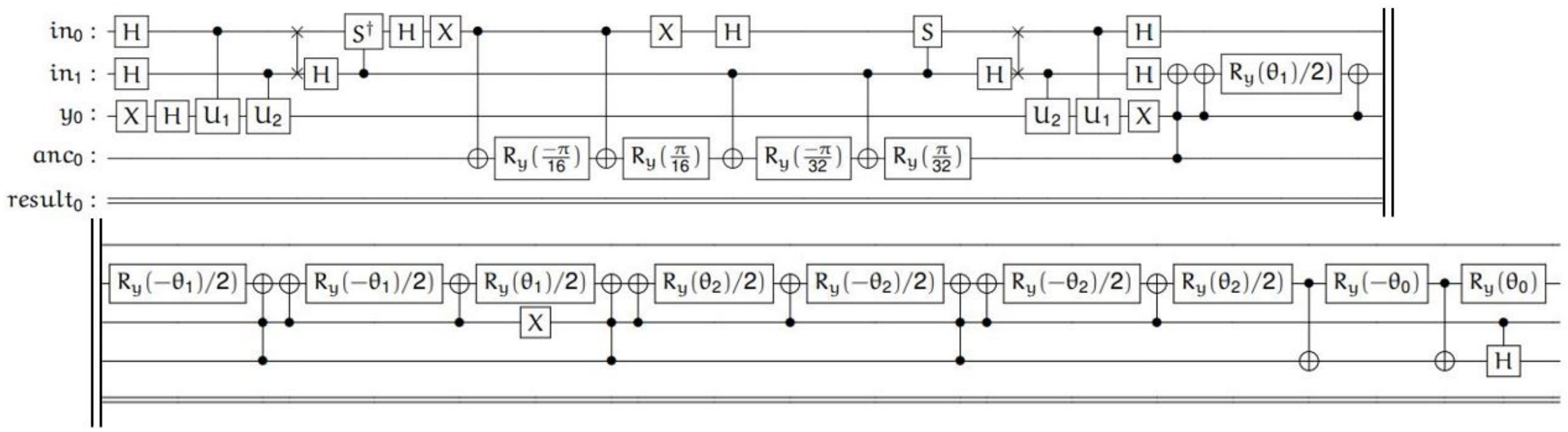
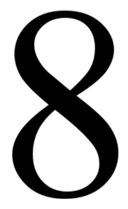


Figure 9.11: The decomposed QSVM-circuit that was used to count all gates and calculate the processing time.

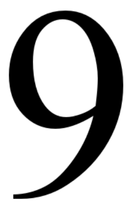
9.2 Inner-Product Methodology for multiple Object Recognition

This section introduces a new approach to distinguish between the three digits 6, 8 and 9. Fig. [9.12] shows the training data of the three digits are the *Times New Roman* font version of each digit.

(a) Training sample for 6



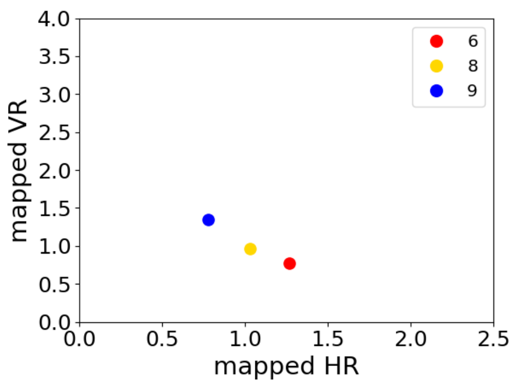
(b) Training sample for 8



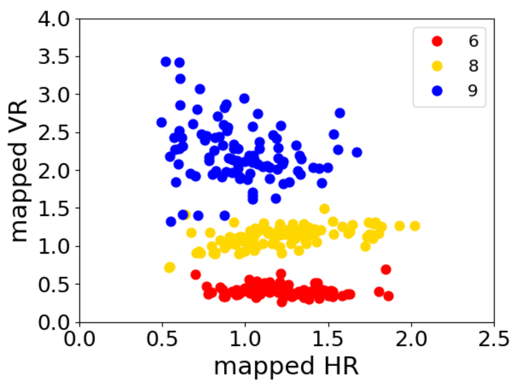
(c) Training sample for 9

Figure 9.12: Additionally to the ML-task from section 9.1 the Swap-Test must classify the digit 8. The fonts of all numbers are *Times New Roman*.

With the distribution of the data points from section 9.1 some data points of the digit 8 have the same attributes as 6 or 9. The three groups are impossible to separate linearly, which would lead to misclassification. Fig. [9.13] shows the distribution of all three numbers.



(a) Distribution of Times New Roman training data



(b) Distribution of OCR dataset testing data

Figure 9.13: The data distribution of testing data and training data in this initial setup is impossible to classify without misclassification as the testing data is not linearly separable.

Fig. [9.14] explains the methodology of the approach to classify three digits.

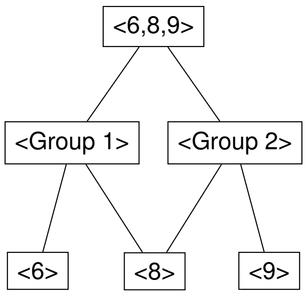


Figure 9.14: At the start all 300 data points of the numbers 6, 8 and 9 will be sorted in Group 1 and Group 2 with the same methodology as in subsection 9.1.3. After being grouped, the data will be mapped linearly again. The second mapping is more specific according to the numbers in this group. All testing samples of 6 are classified in Group 1 and all testing samples of 9 are classified in Group 2. The testing samples of 8 are classified in both groups.

At the beginning, the three datasets cannot be classified linearly without misclassification.
In the first instance all numbers are classified with the same method as in subsection 9.1.3 with the only difference that the numbers are not classified as 6 or 9 but as Group 1 or Group 2. After being classified, the numbers will be mapped again within their groups in order to label all numbers without any errors.
In the first classification of this method the data will be sorted in two different groups. To sort the testing data into two different groups the maximum *VR* value and minimum *VR* value of the training data will be considered. Fig. [9.15] shows the group classification of all three digits.

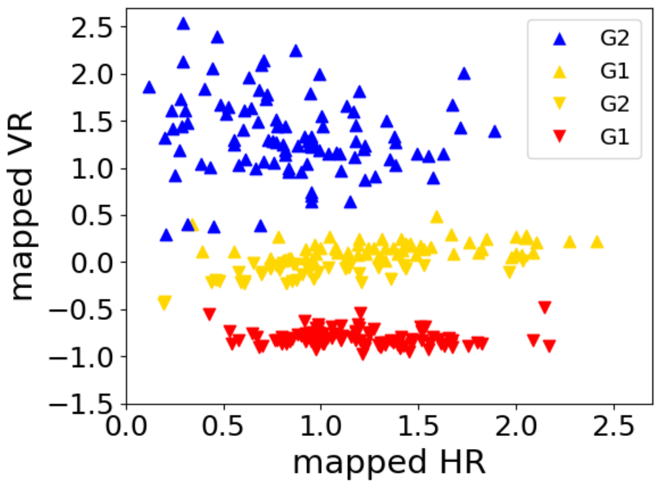


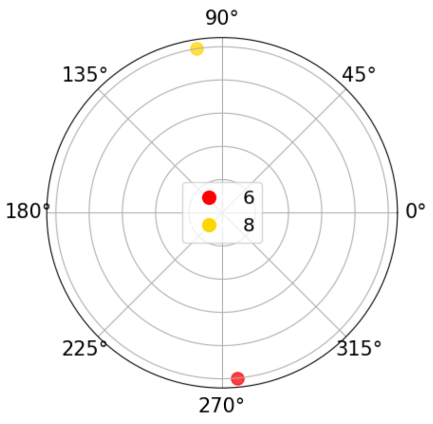
Figure 9.15: All data points from the OCR dataset of the number 9 get sorted to the Group 2 and all data points from the OCR dataset of the number 6 get sorted to the Group 1. The data point of 8 will be divided to Group 1 and Group 2.

In Fig. [9.15] the upward pointing triangles indicates Group 2 and the downwards pointing triangle indicates Group 1. The Swap-Test of subsection 9.1.3 labels every testing data point that is more similar to the training vector of number 6 to Group 1 and labels every testing vector that is more similar to 9 to Group 2.
After the division of the data points into two groups, there is Group 1 with all 6 data points and some 8 data points and Group 2 with all 9 data points and the other half of the 8 data points. The next step linearly maps Group 1 again with another ratios. This time the ratios will have more focus on the difference between the numbers 6 and 8. The numbers 6 and 8 differ always in the upper right corner, where the number 8 has usually more black pixels as the number 6.
The new linear mapping function amplifies the difference between the digits as in eq. [9.18] for the horizontal mapping and eq. [9.19] for the vertical mapping.

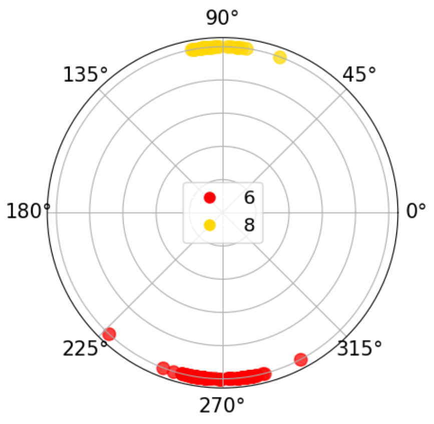
$$\mathrm{HR}_{mapped} = \frac{\text{left pixels}}{\text{right pixels}} - 1.2 \tag{9.18}$$

$$\mathrm{VR}_{mapped} = \frac{\text{upper pixels}}{\text{lower pixels}} * \frac{\text{upper right black pixels}}{\text{upper right white pixels}} * 20 - 10 \tag{9.19}$$

After the mapping of the data points from Group 1 according to eq. [9.18] and eq. [9.19] the data points will be normalized and the angles will be calculated as in subsection 9.1.1. Fig. [9.16] shows the mapped data points of Group 1.



(a) Training data dirstribution



(b) Test data dirstribution

Figure 9.16: Fig. 9.16a and Fig. 9.16b shows that the mapping of the training data points similar to the mapping of the testing data points.

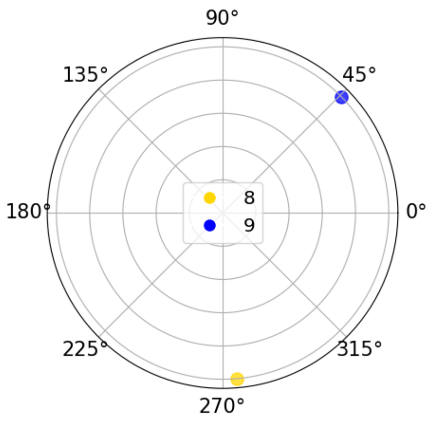
The distributions of Fig. [9.16] results in a 100% accuracy in classifying the numbers 6 and 8.
The same methodology classifies Group 2 with the digits 8 and 9. The difference between the digits 8 and 9 is in the lower left corner of the number. In the lower left corner, the digit 8 usually has more black pixels than the number 9. This approach does not work out for the OCR-dataset as the difference is not big enough to separate the testing samples according to

their numbers. In order to divide the numbers of Group 2, the thickness of the lower part of the digit is measured. The testing samples of 9 have always a small line going down, whereas the testing samples of 8 are in the lower part of the digit at least twice as thick as the digit 9. Eq. [9.20] and eq. [9.21] does the linear mapping of the data points.

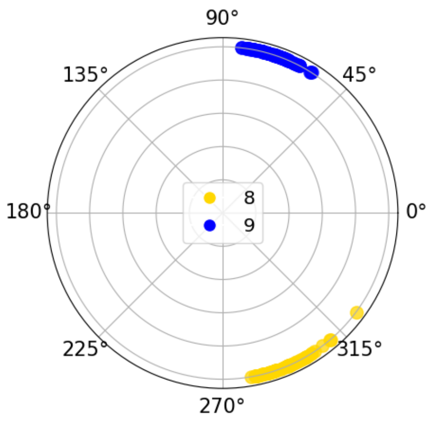
$$\mathrm{HR}_{mapped} = \frac{\text{left pixels}}{\text{right pixels}} - 1.2 \tag{9.20}$$

$$\mathrm{VR} = \frac{\text{upper pixels}}{\text{lower pixels}} + \delta(\text{lower thickness}) * 1.2 + 14 \tag{9.21}$$

After the mapping with eq. [9.20] and eq. [9.21], the data points of Group 2 are normalized, and the angles are calculated as in subsection 9.1.1. Fig. [9.16] shows the results of the preprocessing.



(a) Training data dirstribution



(b) Testing data dirstribution

Figure 9.17: The training data points align with the testing data points which makes the Swap-Test method robust. The labeling process of the Group 2 had no misclassification.

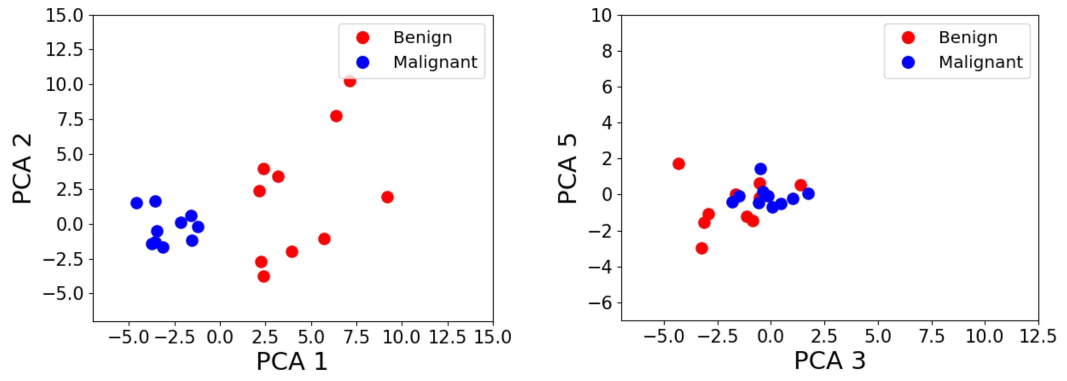
9.3 Swap-Test using entire Bloch Sphere

In this method a more complicated dataset was used to classify than the OCR-dataset. In this example, the Breast Cancer Dataset (Dr. William H. Wolberg, 1995) is tried to be labeled with the Swap-Test method. This dataset consists of 32 attributes and has over 569 instances that are either diagnosed malignant or benign. Each instance is described with a digitized image of a fine needle aspirate of a breast mass. These 32 attributes are too many to be mapped onto the circumference. In this method the data points will be mapped onto the entire sphere of a qubit. As the entire sphere is still not enough for the 32 attributes the Breast Cancer Dataset will be summarized with a Principal Component Analysis (PCA) (Jolliffe, 2016) and then two angles will be calculated that map the data onto the whole sphere and not only the circumference as in subsection 9.1.3.

In datasets like Breast Cancer Dataset each instance is described with a huge number of attributes. To lower the computational cost and only consider the important variables also called PCA-components, a PCA will obtain the most important components from this dataset. The PCA-components are sorted due to their informative value. The higher the variability of a certain attribute, the more informative value the variable obtains. A PCA-component is not necessarily one of the 32 initial attributes. With the PCA-method new variables can be formed as a combination of the existing ones (Jolliffe, 2016).

In this method the PCA-components of the first, the second, the third and the fifth order will be used to calculate the angles as described in subsection 9.1.1. As the fourth PCA-component did not differ too much from the third PCA-component, the fifth PCA-component was chosen in order to get more variety at the mapping process. In comparison to the subsection 9.1.3 and section 9.2 in this case, there is no training data given in advance. With the OCR-dataset the training data was the *Times New Roman* font of the numbers as they are considered to be an accurate representation of this number. The Breast Cancer Dataset does not have a training sample that can be considered as a good representation of all 32 attributes of one class. Due to the missing role model, ten arbitrary data points of both groups, Benign and Malignant are taken and the average is calculated. This average representation was used as training sample.

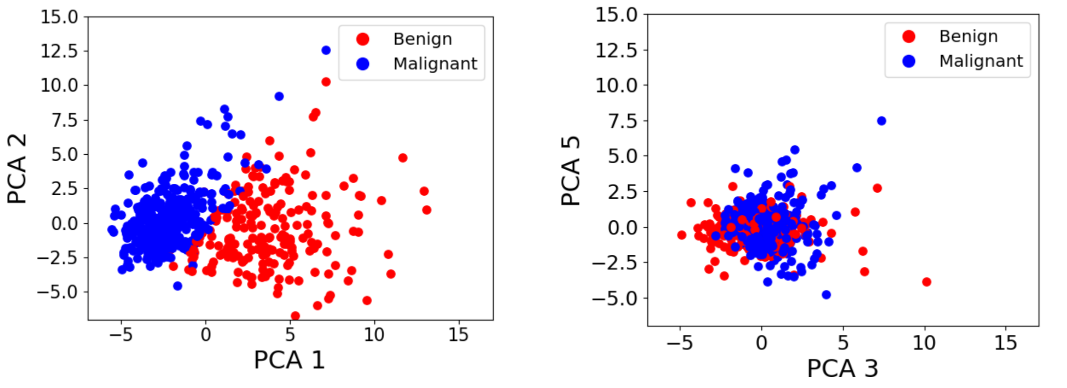
Fig. [9.18a] shows the PCA-components of ten randomly chosen training data of the first and second order and Fig. [9.18b] illustrates the third and fifth order PCA-components.



(a) PCA1 and PCA2 distribution (b) PCA3 and PCA5 distribution

Figure 9.18: The calculation of the average vectors use ten randomly chosen instances of each group. The average vectors represent the training samples in this method. Choosing ten vectors will lower the impact of outliners and is a good representation of the average instance.

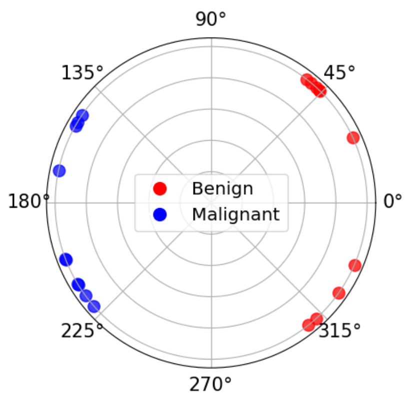
The PCA-components does not need a linear mapping of the data as in the previous subsection 9.1.3 and section 9.2. Fig. [9.19] shows the data distribution of the whole dataset for the four PCA-components.



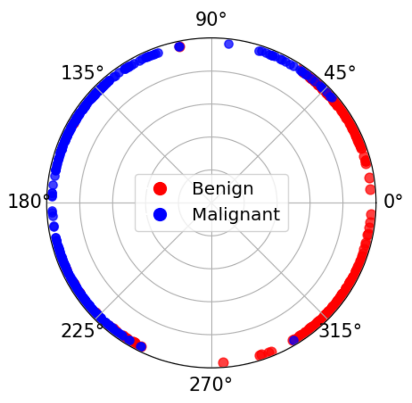
(a) PCA1 and PCA2 distribution (b) PCA3 and PCA5 distribution

Figure 9.19: Fig. [9.19a] and Fig. [9.19b] show that the data points cannot be separated linearly. With these two figures the initial 32 attributes of an instance have been projected onto four PCA-components.

For each testing sample the angle θ will be calculated with its PCA1-component and PCA2-component. The angle θ of a testing sample will map the state along the x-axis circumference. Calculating the angle θ is done by normalizing the data points of Fig. [9.19a] and Fig. [9.20] shows the result using the trigonometry functions of subsection 9.1.1.



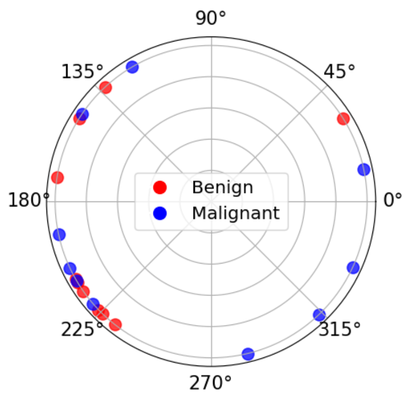
(a) θ training data points



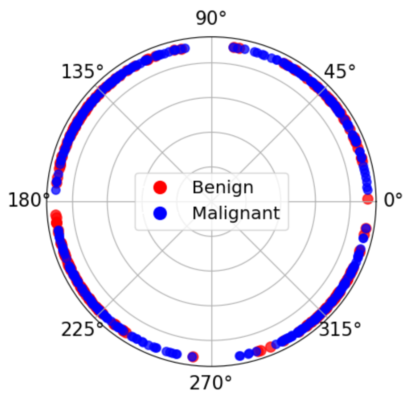
(b) θ testing data points

Figure 9.20: With the angle θ the qubits state will be transformed along the x-circumference with a rotation operator. With θ alone, the accuracy of the Swap-Test method will be lower than combined with the second angle α.

With the PCA3-component and the PCA5-component of a testing sample the angle α will be calculated to map the data points along the z-axis circumference as in Fig. [9.21].



(a) α training data points



(b) α testing data points

Figure 9.21: The distribution for the second angle α seems to have no main groups as all data points are equally distributed over the entire circumference. The combination of both angles θ and α will increase the accuracy in comparison to the single angle θ.

Calculating the angle α is done by normalizing the testing samples of Fig. [9.19b] and using the trigonometry functions of subsection 9.1.1. Fig. [9.22] shows the mapping of the testing samples along the entire sphere with the angle θ for the x-circumference and α for the z-circumference.

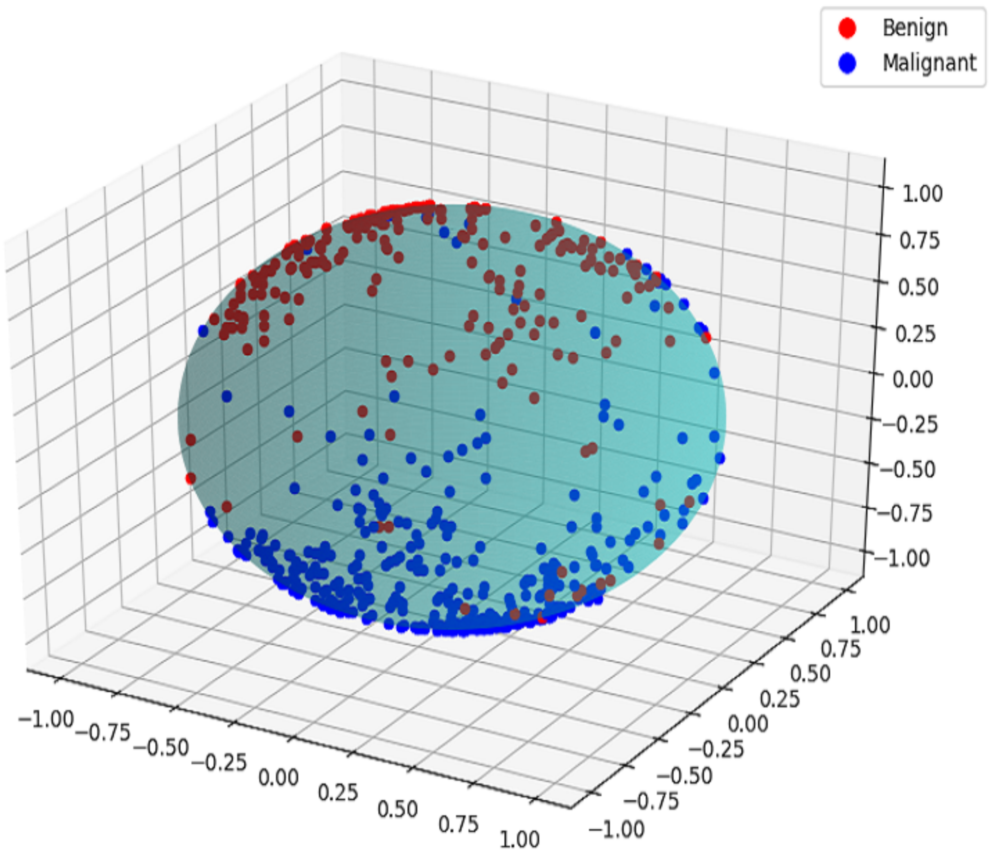


Figure 9.22: All data points of the Breast Cancer Dataset mapped onto a sphere. To transform the state of a qubit to any data point a rotation x-gate with rotation θ and a rotation z-gate with rotation α are needed.

Fig. [9.23] illustrates the expanded Swap-Test circuit of subsection 9.1.3 that achieved the distribution of Fig. [9.22].

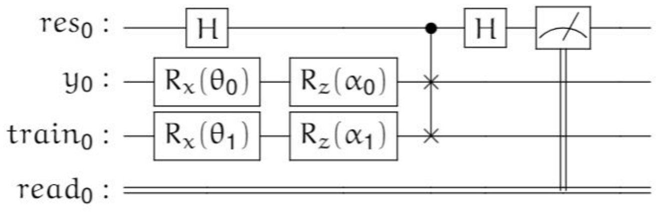


Figure 9.23: The Swap-Test circuit has two rotation gates on the qubits y_0 and $train_0$ in order to map the vector to the location of the testing sample on the sphere. The functionality is equivalent to the Swap-Test explained in section 9.2.

The circuit from Fig. [9.23] with the testing and training dataset result in a total accuracy of 92%. Through the entire dataset, 46 errors occurred with 20 false predicted benign data points and 26 false predicted malignant data points.

10 Conclusion

Regarding the research question, it was possible to connect a quantum computer with the humanoid robot and execute several circuits to express different behaviors. The cloud connection using the Python library Qiskit established the communication between the robot's system and IBMs quantum computers. The Grover oracle of chapter 8 was able to outperform several classical algorithms for smaller graphs, but not all. Therefore, this Master Thesis could not validate the hypothesis that quantum computer is superior in comparison to classical computing solving algorithmic problems like shortest path. Also the other hypothesis that quantum computer are superior in designing human-like behavioral models in comparison to classical computing was not validated. Nevertheless, this Master Thesis showed the potential of quantum circuit in designing behavioral models including random numbers. With more powerful and available quantum computers in the future, quantum computing offers a serious solution in handling bigger problems like the Shortest Path problem or in designing behavioral models using the probabilistic nature of quantum computing.

With Quantys' setup it was possible to start working on the essence of this Thesis which is the first ever humanoid robot controlled by a hybrid quantum/classical computer. With the cloud connection to a quantum computer the robot was capable of executing certain tasks on a quantum computer but was mainly controlled with ROS on a classical computer. Obviously at this moment quantum computers are still an experimental device and are limited to their number of qubits and speed. Therefore, the results of this Thesis do not represent the existing advantage of quantum computing, however, they clearly demonstrate the future advantage when higher numbers of qubits will become available to control similar robots. The disadvantage of the cloud-connected computer was its accessibility. As the quantum computers from IBM are open accessible the sent scripts are queued before being executed. Therefore, the scripts had to wait in such a queue for few minutes which reduced the ability to demonstrate practical experimental advantages of the quantum approaches of this Thesis. Researchers at PSU created a model of a quantum-controlled robot with which was the basis of this research. This way it was possible to create an updated model which was first time implemented on an advanced humanoid robot and implemented on a hybrid-controlled computer rather than a simple mobile robot controlled by a simulated quantum computer.

As ROS uses Python 2.x and Qiskit uses Python 3.x the two Python versions had to be coded separately and combined in subroutines. Quanty's communication was managed with ROS written in Python 2.x, but the scripts that included quantum circuits were only executable written in Python 3.x. In order to enable the communication between the different language revisions of ROS and Qiskit, the Qiskit scripts have the be executed within the ROS script as subroutines. These subroutines made the code convoluted and difficult to organize. As Qiskit has an advanced library and a big user community it was superior to other Python-based quantum libraries like pyQuil which could also have been executed with Python 2.x. Qiskit allowed to execute simulated and real quantum circuits on Quanty. With more in-depth knowledge about quantum coding, it will be better to swap the Qiskit library with the pyQuil library to have a better coding structure.

This Master Thesis demonstrates the first quantum circuit that can imitate behavioral motions as follows:

- Deterministic,
- Classical Probabilistic,
- Quantum Probabilistic,
- Mixed Probabilistic.

With the future availability of larger quantum computer‘s with more powerful quantum simulators and more advanced quantum language the developed methods can be extended to advance motions and emotional behaviors of humanoid robots with higher number of Degree of freedom.

Two quantum circuits enhance the HRI quality by defining a more human-like behavioral model. These circuits uses the probabilistic nature of quantum computing to form a non-deterministic behavioral model using random numbers. The behavioral models do not require higher computational costs as behavioral models using convolutional nets and are they are robust even on a real quantum computer. The implementation of the concepts happened on high level and only on the upper body of Quanty. The concepts set the fundament for more motions being adjusted to the robots’ behavioral model. The circuits can influence to other motions like the locomotion. The circuits for the internal state and the handedness have low depth and are not prone to decoherence, therefore, the quantum circuits are suited for evaluations on a real quantum computer. The designed concept offers a trivial approach to implement behavioral models of any humanoid robot, which is adaptable to different motions.

The only sensory stimuli for the quantum random walk came from the webcam. Therefore, the quantum random walk methodology was quite trivial. Quanty could walk randomly in a room searching for the target person, without recognizing anything else but the target person. To improve this methodology in the future, two possibilities are given as follows:

- Improve the recognition ability of Quanty to recognize different objects and persons with one webcam. This avoids implementing more sensors smaller humanoid robots only offer limited space to mount those sensors.

- Add more sensors like gyroscope or laser sensor to gather more information about Quantys' environment.

A well-known algorithmic problem, with many applications in the area of robotics, is to find the solution to various path problems. These problems include the shortest path, the longest path, fastest path, several traveling salesman variants, the robot team coordination problem, and the two-arm robot cooperation problem, as well as several others. An approach was designed to solve the Shortest Path problem using quantum computing. With the available computational capacity and quantum computing it was possible to solve the Shortest Path problem for smaller graphs only. As the algorithmic problem needs to be solved in practical applications for bigger graphs, the available computational capacity is not yet sufficient enough to provide a comparable alternative to the classical algorithms such as Dijkstra. To find a valid path, the running time of this concept is $O(\frac{\pi}{4}\sqrt{\frac{2^E}{M}})$ with E being the number of edges of the path and M being the number of all valid solutions. The Grover algorithm for the Shortest Path is superior for smaller graphs in comparison to most of the classical algorithms. As the number grows exponentially to the graph's edges, it is not practical for bigger graphs and must be redesigned to use the limited number of qubits more efficiently. The validation of the Shortest Path algorithm was only able for small graphs, as even the simulation on the available computers had trouble setting up the environment for more than 30 qubits. Even with the emerging technology of NISQ, quantum computers do not offer enough qubits to handle bigger graphs yet.

As the Grover algorithm is limited to the number of qubits, other machine learning algorithms like Quantum Support Vector Machine and quantum Swap-Test were used to classify datasets. The QSVM algorithm for classifying digits of Yang et al. (2019*b*) was improved in this Thesis and compared to the quantum Swap-Test algorithm. Both algorithms classified two digits with 100% accuracy. With respect to the NISQ technology, the Swap-Test was superior in terms of decoherence, thus, further research was conducted with the Swap Test. The digits were taken from the OCR-dataset (Escrivá, 2012), which only has 100 samples per digit and is considered less complex than the MNIST-dataset (Y et al., 1998). In another application, the Swap Test was used to distinguish between three different digits with multiple data mappings. Within the classification process of three different digits one additional layer of classification was included and two different data were mappings conducted. This methodology was able to classify the digits of 6, 8 and 9 with a 100% accuracy. Due to the power of the quantum Swap-Test algorithm it was applied to a practical database. Because of the lack of such database in the area of robotics the Master Thesis turned its interest to the famous Breast-Cancer dataset for which about 300 papers were written with different machine learning approaches. The current accuracy of the top algorithm is about 97%. With four PCA-components the datapoints were mapped onto the entire sphere of the Bloch sphere and classified with the quantum Swap-Test algorithm. The Breast Cancer Dataset was classified using only one qubit to map the data points with an accuracy of 92%. Although this is an achievement not related to robotics, it demonstrates that the quantum Swap-Test algorithm can be applied to real size databases and will be useful when such database becomes available for vision or motion of a humanoid robot. For instance, Sunardi (2020) developed a database of human-like humanoid robot motions from human dace acquisition. Feature work may include applications of the quantum Swap-Test algorithm to this database.

11 Future work

In future research the potential of the HRI circuits and the ML circuits of Quanty will be enhanced. The achievements of this Master Thesis can be divided into two different aspects as follows:

- Quantum behavioral modelling
- Quantum machine learning

The quantum behavioral modelling has various further applications that impact quantum robotics either on local or on global scale. Local impacts result in further projects on Quanty and global impacts effect the entire area of quantum robotics in general. Further steps will be:

Global

As discussed in chapter 10, the internal state of only two extrema were mapped onto the Bloch sphere. Mapping the internal state according to the wheel of emotions of Plutchik (1982) achieves a better scalability. With this concept all emotions can be mapped onto the Bloch sphere. A robot does not always need to imitate all emotions, therefore, not all of the seven emotions need to be implemented. In future applications a bigger Hilbert-space can be used to imitate a behavioral model.

Global

A questionnaire as in Sunardi (2020) has to be carried out in order to establish the importance of behavioral aspects on the robot, like being left- or right-handed. Does it increase the HRI quality if the robot is left- or right-footed? Is it enough to imitate the motion execution of a left-handed human like in Gillesen et al. (2011), or must the behavioral model also consider that the handedness of a person is related to different thinking-processes? As left-handed humans think more with the right half of their brain, does a robot also need to change its computational process according to its handedness or is the bare motion execution enough? Those are only some suggestions for this questionnaire, which can help to implement a generalized improved model of humanoid quantum robot behaviors, which is important for assisting and entertaining robot.

Local

When presenting a quantum humanoid robot in a theatre play, the quantum circuits can only be simulated because a real quantum computer would take too long as requests will be queued. As the access to a real quantum computer is limited, a different approach must be found for the PSU robot theatre to imitate more quantum-like behaviors. In order to execute a more realistic quantum behavior, the noise of any publicly accessible quantum computer can be measured and applied in the simulation. This will avoid the waiting time on a real quantum computer and imitate a more quantum behavioral model.

Local

To further increase the DOF of the robotic head an additional controller must be included in order to use smaller and lighter servos and not the Dynamixel servos that were used in this Master Thesis. If no smaller Dynamixel servo will be available in the future, additional DOFs can be only realized using smaller servos. The designed head concept can be extended with the DOF of the eyebrows to execute a bigger variety of facial expression as in Fukuda et al. (2002).

Local

In Sunardi (2020) a huge motion library for the smaller version of Quanty, the HR-OS1, was created. This library can be mapped onto the dimensions of Quanty. If all motions are mapped, quantum behaviors can be applied to these motions in order to be validated by future students working on this project.

Turning towards the future aspects of machine learning of this Master Thesis, they can be distinguished into local and global projects as follows:

Global

The Qiskit simulation has problems with more than 30 qubits. In Bravyi & Gosset (2016) a new idea for simulating quantum circuits was introduced that can simulate even bigger circuits with a few hundreds of qubits. With more qubits a standalone Grover circuit can be built to solve the Shortest Path problem autonomously.

Global

Even with a simulator of better performance, the concept of the Shortest Path must be changed in order to get rid of the polynomial running time and be as efficient as the best classical algorithms. The first step to improve this concept will be to represent the nodes as qubits and the edges as functions in contrast to the current concept which uses as many qubits as edges. As a graph usually has more edges than nodes, the running time would decrease as less qubits will be needed to represent this graph as a quantum circuit. If the qubits represent the nodes, the running time would change to $O(\frac{\pi}{4}\sqrt{\frac{2^N}{M}})$, with *N* representing the number of nodes in a graph. Therefore, the node-adjusted Grover algorithm for the Shortest Path will be still polynomial, but it can represent larger graphs as it will use the qubits more efficiently.

Global

Quantum computing is superior in comparison to classical computing in terms of calculating the dot-product. The dot-product is used when calculating the inner product. Therefore, quantum computing has greater potential doing the Swap Test for classifying data points in comparison to classical computing. A methodology will be created to generalize the Swap-Test on the whole Bloch sphere. For complicated datasets like the Breast Cancer Dataset the concept will be expanded onto multiple qubits to improve its accuracy. In the first step it will be tried to distinguish between the numbers 6 and 9 of the MNIST dataset. Instead of 100 samples per number as in the OCR-dataset, the MNIST dataset has 6.000 samples per number. Despite the quantity of the testing samples also the quality of them will be increased. With other datasets like the Breast-Cancer dataset that have a higher complexity than the MNIST dataset, a methodology will be researched to classify even more complex testing samples. In future research more robotic related datasets will be classified like the facial image dataset. This classification can be included within the quantum random walk algorithm to enhance its potential.

Bibliography

Aharonov, Y., Davidovich, L. & Zagury, N., 1993. Quantum random walks. *Physical review. A*, 48, pp.1687–1690.

Al-Bayaty, A. & Perkowski, M. P., 2014. Project on quantum robotics and quantum machine learning. *Portland State University*, T.12, pp.15–34.

Al-Hami, M. & Lakaemper, R., 2015. Sitting pose generation using genetic algorithm for nao humanoid robots. *Proceedings of IEEE Workshop on Advanced Robotics and its Social Impacts, ARSO*, 2015, pp.137–142.

Alamdari, A. & Krovi, V., 2016. Review of human modeling. , .

Allen, G. D., 2003*a*. *Linear Algebra for Applications*: Texas A&M University.

Allen, G. D., 2003*b*. Math 640 - 600/700 — linear algebra for applications. , . Available at: <https://www.math.tamu.edu/~dallen/m640_03c/lectures/chapter9.pdf> [Accessed !no date!].

Aparanji, V., Wali, U. & Aparna, R., 2017. Robotic motion control using machine learning techniques. , pp.1241–1245.

Aparnathi, R., 2014. The novel of six axes robotic arm for industrial applications. *IAES International Journal of Robotics and Automation (IJRA)*, 3.

Arfken, G., 1985. *Mathematical Methods for Physicists*. third edition. San Diego: Academic Press, Inc.

Aristidou, A. & Lasenby, J., 2009. Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver. , .

Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J., Barends, R., Biswas, R., Boixo, S., Brandao, F., Buell, D., Burkett, B., Chen, Y., Chen, J., Chiaro, B., Collins, R., Courtney, W., Dunsworth, A., Farhi, E., Foxen, B., Fowler, A., Gidney, C. M., Giustina, M., Graff, R., Guerin, K., Habegger, S., Harrigan, M., Hartmann, M., Ho, A., Hoffmann, M. R., Huang, T., Humble, T., Isakov, S., Jeffrey, E., Jiang, Z., Kafri, D., Kechedzhi, K., Kelly, J., Klimov, P., Knysh, S., Korotkov, A., Kostritsa, F., Landhuis, D., Lindmark, M., Lucero, E., Lyakh, D., Mandrà, S., McClean, J. R., McEwen, M., Megrant, A., Mi, X., Michielsen, K., Mohseni, M., Mutus, J., Naaman, O., Neeley, M., Neill, C., Niu, M. Y., Ostby, E., Petukhov, A., Platt, J., Quintana, C., Rieffel, E. G., Roushan, P., Rubin, N., Sank, D., Satzinger, K. J., Smelyanskiy, V., Sung, K. J., Trevithick, M., Vainsencher, A., Villalonga, B., White, T., Yao, Z. J., Yeh, P., Zalcman, A., Neven, H. & Martinis, J., 2019. Quantum supremacy using a programmable superconducting processor. *Nature*, 574, p.505–510. Available at: <https://www.nature.com/articles/s41586-019-1666-5> [Accessed !no date!].

Asfaw, A., Bello, L., Ben-Haim, Y., Bravyi, S., Capelluto, L., Vazquez, A. C., Ceroni, J., Harkins, F., Gambetta, J., Garion, S., Gil, L., Gonzalez, S. D. L. P., McKay, D., Minev, Z., Nation, P., Phan, A., Rattew, A., Schaefer, J., Shabani, J., Smolin, J., Temme, K., Tod, M. & Wootton., J., 2020. Learn quantum computation using qiskit. , . Available at: <http://community.qiskit.org/textbook> [Accessed !no date!].

Azvan, R. & Florian, R., 2003. Biologically inspired neural networks for the control of embodied agents. , .

Barciela, G., Paz, E., López, J., Sanz, R. & Pérez Losada, D., 2008. Building a robot head: Desing and control issues. In: , pp.33–39.

Bazzano, F. & Lamberti, F., 2018. Human-robot interfaces for interactive receptionist systems and wayfinding applications. *Robotics*, 7, p.56.

BCG, 2015. *Takeoff in Robotics Will Power the Next Productivity Surge in Manufacturing*. Computational Intelligence: BCG. Available at: <https://www.bcg.com/d/press> [Accessed !no date!].

Bell, J. S., 1964. On the Einstein Podolsky Rosen paradox. *Physics*, 1(3), pp.195–200. Available at: <https://cds.cern.ch/record/111654> [Accessed !no date!].

Benioff, P., 1997. Quantum robots and quantum computers. , .

Bertsekas, D. P., 1976. Multiplier methods: A survey. *Automatica*, 12(2), pp.133 – 145. Available at: <http://www.sciencedirect.com/science/article/pii/0005109876900777> [Accessed !no date!].

Bevacqua, E. & Mancini, M., 2004. Speaking with emotions. In:

Binder, K. & Heermann, D. W., 2010. Monte Carlo Simulation in Statistical Physics: An Introduction; 5th ed.. , . Available at: <https://cds.cern.ch/record/1339249> [Accessed !no date!].

Bloch, F., 1946. Nuclear induction. *Phys. Rev.*, 70, pp.460–474. Available at: <https://link.aps.org/doi/10.1103/PhysRev.70.460> [Accessed !no date!].

Bonnet, V. & Venture, G., 2015. Fast determination of the planar body segment inertial parameters using affordable sensors. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 23.

Braitenberg, V., 1986. *Vehicles: Experiments in Synthetic Psychology*. Bradford Books: MIT Press. Available at: <https://books.google.com/books?id=7KkUAT_q_sQC> [Accessed !no date!].

Brass, P., Vigan, I. & Xu, N., 2015. Shortest path planning for a tethered robot. *Computational Geometry*, 48(9), pp.732 – 742. Available at: <http://www.sciencedirect.com/science/article/pii/S0925772115000577> [Accessed !no date!].

Brassard, G., HØyer, P. & Tapp, A., 1998. Quantum counting. *Lecture Notes in Computer Science*, p.820–831. Available at: <http://dx.doi.org/10.1007/BFb0055105> [Accessed !no date!].

Bravyi, S. & Gosset, D., 2016. Improved classical simulation of quantum circuits dominated by clifford gates. *Physical Review Letters*, 116(25). Available at: <http://dx.doi.org/10.1103/PhysRevLett.116.250501> [Accessed !no date!].

Breazeal, C. & Scassellati, B., 2003. How to build robots that make friends and influence people. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2.

Breazeal, C., Brooks, A., Chilongo, D., Gray, J., Hoffman, G., Kidd, C., Lee, H., Lieberman, J. & Lockerd, A., 2004. Working collaboratively with humanoid robots. , pp.253 – 272 Vol. 1.

Brown, L. M., 2006. Dirac ' s the principles of quantum mechanics. In:

Bruza, P., Wang, Z. & Busemeyer, J. R., 2015. Quantum cognition: a new theoretical approach to psychology. *Trends in Cognitive Sciences*, 19, pp.383–393.

Buric, N., 2010. Quantum degrees of freedom, quantum integrability and entanglment generators. , .

Cacciapuoti, A. S., Van Meter, R., Hanzo, L. & Van, R., 2019. When entanglement meets classical communications: Quantum teleportation for the quantum internet. *IEEE Transactions on Communications*, .

Campbell, M., Hoane, A. J. & Hsu, F.-h., 2002. Deep blue. *Artif. Intell.*, 134(1–2), p.57–83. Available at: <https://doi.org/10.1016/S0004-3702(01)00129-1> [Accessed !no date!].

Carcassi, G., Maccone, L. & Aidala, C. A., 2020. The four postulates of quantum mechanics are three. , .

Cardoso, A., Leitão, J. & Teixeira, C., 2019. Using the jupyter notebook as a tool to support the teaching and learning processes in engineering courses. , pp.227–236.

Ccoicca, Y., 2013. Applications of support vector machines in the exploratory phase of petroleum and natural gas: a survey. *International Journal of Engineering & Technology*, Volumen 2, pp.113–125.

Chien, C.-H., Van Meter, R. & Kuo, S.-Y., 2013. Fault-tolerant operations for universal blind quantum computation. *ACM Journal on Emerging Technologies in Computing Systems*, 12.

Cho, A., 2019. Ibm casts doubt on google's claims of quantum supremacy. *Science Magazine*, .

Christoforou, E. & Mueller, A., 2017. Robot and robotics: The origin and beyond. In: , pp.613–621.

Cortes, C. & Vapnik, V., 1995. Support-vector networks. *Mach. Learn.*, 20(3), p.273–297. Available at: <https://doi.org/10.1023/A:1022627411411> [Accessed !no date!].

Cyberneticzoo, 2013. 1958-62 – "versatran" industrial robot – harry johnson & veljko milenkovic. *cyberneticzoo.coms*, .

DeBenedictis, E., 2018. A future with quantum machine learning. *Computer*, 51, pp.68–71.

Degen, C., Reinhard, F. & Cappellaro, P., 2016. Quantum sensing. *Reviews of Modern Physics*, 89.

Demetrescu, C. & Italiano, G., 2004. Engineering shortest path algorithms. , pp.191–198.

Deng, R., 2020. *Quantum Motions and Emotions for a Humanoid Robot Actor*. Portland State University.

Dereli, S. & Köker, R., 2019. A meta-heuristic proposal for inverse kinematics solution of 7-dof serial robotic manipulator: quantum behaved particle swarm algorithm. *Artificial Intelligence Review*, .

Deutsch, D., 1984. *Quantum theory, the Church-Turning principle and the universal quantum computer*. London: Dep. of Astrophysics, Oxford.

Dinerstein, J., Egbert, P., de Garis, D. H. & Dinerstein, N., 2004. Fast and learnable behavioral and cognitive modeling for virtual character animation. *Journal of Visualization and Computer Animation*, 15, pp.95–108.

Dong, D., Chen, C., Zhang, C. & Chen, Z., 2006. Quantum robot: structure, algorithms and applications. *Robotica*, 24, pp.513–521.

Dooley, K., 2009. The butterfly effect of the "butterfly effect". *Nonlinear dynamics, psychology, and life sciences*, 13, pp.279–88.

Dr. William H. Wolberg, W. Nick Street, O. L. M., 1995. Breast cancer wisconsin (diagnostic) data set. *UCI*, .

E. Yousif, M., 2016. The double slit experiment re-explained. *IOSR Journal of Applied Physics*, Volume 8, pp.PP86–98.

Edwin, P., John, G., Dmitri, M. & Jay, G., 2019. On “quantum supremacy”. , . Available at: <https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/> [Accessed !no date!].

Einstein, A., Podolsky, B. & Rosen, N., 1935. Can quantum-mechanical description of physical reality be considered complete?. *Phys. Rev.*, 47, pp.777–780. Available at: <https://link.aps.org/doi/10.1103/PhysRev.47.777> [Accessed !no date!].

Elmasry, A. & Shokry, A., 2018. A new algorithm for the shortest-path problem. *Networks*, 74.

Erbatur, K., Okazaki, A., Obiya, K., Takahashi, T. & Kawamura, A., 2002. A study on the zero moment point measurement for biped walking robots. In: , pp.431 – 436.

Eren, H., 2004. *Electronic Portable Instruments-Design and Applications*.

Escrivá, D. M., 2012. basicocr. *GitHub repository*, .

Fei, S.-M., 2020. Entanglement in ibmq superconducting quantum computer with 53 qubits. *Quantum Engineering*, .

Fernández, F., 2016. The kronecker product and some of its physical applications. *European Journal of Physics*, 37, p.065403.

Figgatt, C., Ostrander, A., Linke, N., Landsman, K., Zhu, D., Maslov, D. & Monroe, C., 2019. Parallel entangling operations on a universal ion-trap quantum computer. *Nature*, 572, p.1.

Fukuda, T., Jung, M.-J., Nakashima, M., Arai, F. & Hasegawa, Y., 2004. Facial expressive robotic head system for human-robot communication and its application in home environment. *Proceedings of the IEEE*, 92, pp.1851 – 1865.

Fukuda, T., Taguri, J., Arai, F., Nakashima, M., Tachibana, D. & Hasegawa, Y., 2002. Facial expression of robot face for human-robot mutual communication. , 1, pp.46 – 51 vol.1.

Gao, P., Li, Y., Perkowski, M. A. & Song, X., 2020. Realization of quantum oracles using symmetries of boolean functions. *Quantum Inf. Comput.*, 20, pp.418–448.

Garcia-Escartin, J. C. & Chamorro-Posada, P., 2011. Equivalent quantum circuits. , .

García-Martín, D. & Sierra, G., 2017. Five experimental tests on the 5-qubit ibm quantum computer. *Journal of Applied Mathematics and Physics*, 06.

Gasparetto, A. & Scalera, L., 2019. A brief history of industrial robotics in the 20th century. *Advances in Historical Studies*, 08, pp.24–35.

Gillesen, J., Barakova, E., Huskens, B. & Feijs, L., 2011. From training to robot behavior: Towards custom scenarios for robotics in training programs for asd. *IEEE ... International Conference on Rehabilitation Robotics : [proceedings]*, 2011, p.5975381.

Glorioso, P., 2013. On common eigenbases of commuting operators. , .

Golub, G. H. & Van Loan, C. F., 1996. *Matrix Computations*. third edition: The Johns Hopkins University Press.

Goodrich, M. & Schultz, A., 2007. Human-robot interaction: A survey. *Foundations and Trends in Human-Computer Interaction*, 1, pp.203–275.

Goy, A. & Torre, I., 2013. Artificial agents and intelligent agents: Paradigms, applications and future trends. *Lexia*, 3, pp.299–315.

Graefe, V. & Bischoff, R., 2003. Past, present and future of intelligent robots. In: , pp.801 – 810 vol.2.

Greenberger, D., Weinert, F. & of Section for the History of Science, H., 2016. Compendium of quantum physics: Concepts, experiments, history and philosophy. , .

Grover, L. K., 1996. *A fast quantum mechanical algorithm for database search*.

Gu, P., Purdom, J., Franco, J. & Wah, B., 2002. Satisfiability problem: Theory and applications, chapter algorithms for the satisfiability sat problem: A survey. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp.19–152.

Hammoudeh, A., 2018. A concise introduction to reinforcement learning. , .

Hari Dass, N. D., 2013. The superposition principle in quantum mechanics - did the rock enter the foundation surreptitiously?. , .

Harrow, A. W., Hassidim, A. & Lloyd, S., 2009. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103, p.150502. Available at: <https://link.aps.org/doi/10.1103/PhysRevLett.103.150502> [Accessed !no date!].

Haslwanter, T., 2016. Python. , pp.5–42.

Hassani Monfared, K., 2011. On the permanent rank of matrices. , .

Hautop, H. & Miglino, O., 1998. Evolving and breeding robots. In:

Havlíček, V., Córcoles, A., Temme, K., Harrow, A., Kandala, A., Chow, J. & Gambetta, J., 2019. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567, pp.209–212.

Helliwell, P. & Taylor, W., 2004. Repetitive strain injury. *Postgraduate medical journal*, 80, pp.438–43.

Hester, T., Quinlan, M. & Stone, P., 2010. Generalized model learning for reinforcement learning on a humanoid robot. In: , pp.2369–2374.

Hockstein, N., Gourin, C., Faust, R. & Terris, D., 2007. A history of robots: From science fiction to surgical robotics. *Journal of Robotic Surgery*, 1, pp.113–118.

Holling, G. & DeGaris, H., 2003. On the application of advances in artificial intelligence technologies for the design of autonomous intelligent robots. , .

Homeister, M., 2015. *Quantum Computing verstehen: Grundlagen - Anwendungen - Perspektiven*. Computational Intelligence: Springer Fachmedien Wiesbaden. Available at: <https://books.google.com/books?id=MgVcCgAAQBAJ> [Accessed !no date!].

Honarvar, A. & Ghasem-Aghaee, N., 2010. An artificial neural network approach for creating an ethical artificial agent. *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA*, pp.290 – 295.

Institute, R. K., 2019. Das unfallgeschehen bei erwachsenen in deutschland. *Gesundheitsberichterstattung des Bundes*, .

Jafarpour, S., Ghodsi, M., Sadri, K. & Montazeri, Z., 2007. Computational power of the quantum turing automata. , pp.1077–1082.

jaygambetta, 2019. ibmq-device-information. *GitHub repository*, .

Jolliffe, I. T., 2016. Principal component analysis: a review and recent developments. *Philos Trans A Math Phys Eng Sci*, 374.

Kak, S., 1999. The initialization problem in quantum computing. *Foundations of Physics*, pp.pp. 267–279. [eJournal] .

Kakenhi, 2007. Geminoid. , . Available at: <http://www.geminoid.jp/projects/kibans/resources.html> [Accessed !no date!].

Kerr, J. & Nickels, K., 2012. Robot operating systems: Bridging the gap between human and robot. , pp.99–104.

Kiefer, C. & Joos, E., 1999. Decoherence: Concepts and examples. In:

Kim, T. & Choi, B.-S., 2018. Efficient decomposition methods for controlled-rn using a single ancillary qubit. *Scientific Reports*, 8.

Kitaev, A. Y., Shen, A. H. & Vyalyi, M. N., 2002. *Classical and Quantum Computation*. USA: American Mathematical Society.

Kole, D. K., Dutta, J., Kundu, A., Chatterjee, S., Agarwal, S. & Kisku, T., 2016. Generalized construction of quantum multiplexers and de-multiplexers using a proposed novel algorithm based on universal fredkin gate. In: *2016 Sixth International Symposium on Embedded Computing and System Design (ISED)*, pp.82–86.

Korn, O., Bieber, G. & Fron, C., 2018. Perspectives on social robots: From the historic background to an experts' view on future developments. , .

Kwon, D.-S., Kwak, Y., Park, J., Chung, M., Jee, E.-S., Park, K.-S., Kim, H.-R., Kim, Y.-M., Park, J.-C., Kim, E., Hyun, K., Min, H.-J., Lee, H., Park, J. W., Jo, S., Park, S.-Y. & Lee, K.-W., 2007. Emotion interaction system for a service robot. In: , pp.351 – 356.

Landauer, R., 1961. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3), pp.183–191.

Laranjeiras, C. C. & Portela, S. I. C., 2016. The carnot cycle and the teaching of thermodynamics: a historical approach. *Physics Education*, 51(5), p.055013. Available at: <https://doi.org/10.1088> [Accessed !no date!].

Lee, B. & Perkowski, M., 2016. Quantum machine learning based on minimizing kronecker-reed-muller forms and grover search algorithm with hybrid oracles. , pp.413–422.

Lee, E., 2018. Why robots that look too human make some people uneasy. *Silicon Valley & Technology*, . Available at: <https://www.voanews.com/silicon-valley-technology/why-robots-look-too-human-make-some-people-uneasy> [Accessed !no date!].

Leonard, J. S. & Umesh, V. V., 1999. Difficulties in the implementation of quantum computers. , p.322–329. [ACM symposium] .

Li, L., Thornton, M. & Perkowski, M., 2006. A quantum cad accelerator based on grover´s algorithm for finding the minimum fixed polarity reed-muller form. , pp.33 – 33.

Lischetzke, T., 2014. Mood. , pp.4115–4120.

Lloyd, S., Mohseni, M. & Rebentrost, P., 2014. Quantum principal component analysis. *Nature Physics*, 10(9), p.631–633. Available at: <http://dx.doi.org/10.1038/nphys3029> [Accessed !no date!].

Loaiza, M., 2017. A short introduction to hilbert space theory. *Journal of Physics: Conference Series*, 839, p.012002.

Lukac, M. & Perkowski, M., 2007*a*. Quantum mechanical model of emotional robot behaviors. , pp.19–19.

Lukac, M. & Perkowski, M., 2007*b*. Quantum mechanical model of emotional robot behaviors. In: , pp.19–19.

Malham, S., 2015. An introduction to lagrangian and hamiltonian mechanics. , .

Malham, S. J., 2016. An introduction to lagrangian and hamiltonian mechanics. , . Available at: <http://www.macs.hw.ac.uk/~simonm/mechanics.pdf> [Accessed !no date!].

Martens, R., Burton, D., Vealey, R., Bump, L. & D.A., S., 1990. Champaign, il: Human kinetics. , pp.pp. 117–213.

Marur, T., Tuna, Y. & demirci, S., 2014. Facial anatomy. *Clinics in dermatology*, 32, pp.14–23.

McNair, D., Droppleman, L., Lorr, M., Educational & Service, I. T., 1971. *Profile of Mood States (POMS)*: Educational and Industrial Testing Service. Available at: <https://books.google.com/books?id=MLhVygAACAAJ> [Accessed !no date!].

Mishra, N., Ashutosh, S., Behera, B. & Panigrahi, P., 2019. Implementation of quantum reed-solomon error detection code on ibm quantum computer and improvement by quantum singular value decomposition. , .

Mohammed, M., Khan, M. & Bashier, E., 2016. *Machine Learning: Algorithms and Applications*.

Moran, M., 2007. Evolution of robotic arms. *Journal of Robotic Surgery*, 1, pp.103–111.

Mori, M., MacDorman, K. & Kageki, N., 2012. The uncanny valley [from the field]. *IEEE Robotics & Automation Magazine*, 19, pp.98–100.

Muecke, K. & Hong, D., 2007. Darwin's evolution: development of a humanoid robot. , pp.2574 – 2575.

Murali, P., Baker, J. M., Javadi-Abhari, A., Chong, F. T. & Martonosi, M., 2019. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In: *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. Providence, RI, USA: Association for Computing Machinery, p.1015–1029.

Murphy, M., 2015. Japan has just won the world cup—of robot soccer. , . Available at: <https://qz.com/461061/the-japanese-have-just-won-the-world-cup-of-robot-soccer/> [Accessed !no date!].

Müller, M., 2015. The fourier transform in a nutshell. , pp.39–57.

Naihin, S., 2018. Quantum computing — the basics, the bad, and the solution. *Noteworthy - The Journal Blog*, . [eJournal] . Available at: <https://blog.usejournal.com/quantum-computing-the-basics-the-bad-and-the-solution-59af357fb52> [Accessed 27.03.2020].

Nicholls, M. E., Searle, D. A. & Bradshaw, J. L., 2004. Read my lips: Asymmetries in the visual expression and perception of speech revealed through the mcgurk effect. *Psychological Science*, 15(2), pp.138–141. PMID: 14738522. Available at: <https://doi.org/10.1111/j.0963-7214.2004.01502011.x> [Accessed !no date!].

Nieuwenhuisen, M., Stückler, J. & Behnke, S., 2010. Intuitive multimodal interaction for domestic service robots.. , 1, pp.1–8.

Oh, J.-H., Hanson, D., Kim, W.-S., Han, Y., Kim, J.-Y. & Park, I.-W., 2006. Design of android type humanoid robot albert hubo. In: , pp.1428 – 1433.

Ouellette, J., 2017. Nanofridge could keep quantum computers cool enough to calculate. , .

Park, S. Y., Kim, S. & Leifer, L., 2017. “human chef” to “computer chef”: Culinary interactions framework for understanding hci in the food industry. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp.214–233.

Paudel, P. & Bhattarai, A., 2018. 5g telecommunication technology: History, overview, requirements and use case scenario in context of nepal. , .

Perkowski, M., 2020. Inverse problems, constraint satisfaction, reversible logic, invertible logic and grover quantum oracles for practical problems. , pp.3–32.

Perkowski, M. & Al-Bayaty, A., 2014. *Project on Quantum Robotics and Quantum Machine Learning*: Portland State University.

Petschnigg, C., Brandstötter, M., Pichler, H., Hofbaur, M. & Dieber, B., 2019. Quantum computation in robotic science and applications. In:

Pierson, H. & Gashler, M., 2017. Deep learning in robotics: A review of recent research. *Advanced Robotics*, .

Plutchik, R., 1982. A psychoevolutionary theory of emotions. *Social Science Information*, 21(4-5), pp.529–553. Available at: <https://doi.org/10.1177/053901882021004003> [Accessed !no date!].

Ponnath, A., 2006. Difficulties in the implementation of quantum computers. *arXiv preprint cs/0602096*, .

Prakash, A. & Rogers, W., 2014. Why some humanoid faces are perceived more positively than others: Effects of human-likeness and task. *International Journal of Social Robotics*, 7.

Preskill, J., 2018. Quantum computing in the nisq era and beyond. *Quantum*, 2, p.79. Available at: <https://doi.org/10.22331/q-2018-08-06-79> [Accessed !no date!].

Qi, B., Qian, L. & Lo, H.-K., 2010. A brief introduction of quantum cryptography for engineers. , .

Raghuvanshi, A., Fan, Y., Woyke, M. & Perkowski, M., 2007. Quantum robots for teenagers. In: , pp.18–18.

Rebentrost, P., Mohseni, M. & Lloyd, S., 2014. Quantum support vector machine for big data classification. *Phys. Rev. Lett.*, 113, p.130503. Available at: <https://link.aps.org/doi/10.1103/PhysRevLett.113.130503> [Accessed !no date!].

Rivest, R. L., Shamir, A. & Adleman, L., 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2), p.120–126. Available at: <https://doi.org/10.1145/359340.359342> [Accessed !no date!].

Riwar, R.-P., Splettstoesser, J. & König, J., 2013. Zero-frequency noise in adiabatically driven interacting quantum systems. *Physical Review B*, 87(19). Available at: <http://dx.doi.org/10.1103/PhysRevB.87.195407> [Accessed !no date!].

roboticsbusinessreview.com, 2018. Applying machine learning to robotics. , .

Rodney, B. & Gill, P., 2015. Types of robots. , . Available at: <https://robots.ieee.org/learn/types-of-robots/> [Accessed !no date!].

Roffe, J., 2019. Quantum error correction: an introductory guide. *Contemporary Physics*, 60, pp.1–20.

Rokach, L. & Maimon, O., 2005. Decision trees. *The Data Mining and Knowledge Discovery Handbook*, 6, pp.165–192.

Rosenbaum, D. J. & Perkowski, M. A., 2008. Superposed quantum state initialization using disjoint prime implicants (squid). , pp.144–149.

Rushdi, A., 1997. Karnaugh map. *Encyclopaedia of Mathematics, Supplement Volume I, M. Hazewinkel (editor), Boston, Kluwer Academic Publishers*, 1.

Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N. & FujiMura, K., 2002*a*. The intelligent asimo: System overview and integration. *IEEE International Conference on Intelligent Robots and Systems*, 3, pp.2478 – 2483 vol.3.

Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N. & FujiMura, K., 2002*b*. The intelligent asimo: System overview and integration. *IEEE International Conference on Intelligent Robots and Systems*, 3, pp.2478 – 2483 vol.3.

Scassellati, B., 2017. Mit cog. , pp.1–10.

Segura Velandia, N. & Moreno, R., 2019. Review of intelligent robotics algorithms for assistive agents. *Research Journal of Applied Sciences*, 13, pp.663–674.

Shao, C., 2019. Computing eigenvalues of matrices in a quantum computer. , .

Shigemi, S., 2017. Asimo and humanoid robot research at honda. , pp.1–36.

Shor, P. W., 1997. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5), p.1484–1509. Available at: <http://dx.doi.org/10.1137/S0097539795293172> [Accessed !no date!].

Silva, L., Tennakoon, T., Marques, M. & Djuric, A., 2016. Baxter kinematic modeling, validation and reconfigurable representation. , .

Simon, J., 2017. Autonomous wheeled mobile robot control. *Interdisciplinary Description of Complex Systems*, 15, pp.222–227.

Srivastava, M., Moulick, R. & Panigrahi, P., 2015. Quantum image representation through two-dimensional quantum states and normalized amplitude. , .

Stoelinga, M., 2004. An introduction to probabilistic automata. *Bulletin of the EATCS*, 78.

Strait, M. K., Floerke, V. A., Ju, W., Maddox, K., Remedios, J. D., Jung, M. F. & Urry, H. L., 2017. Understanding the uncanny: Both atypical features and category ambiguity provoke aversion toward humanlike robots. *Frontiers in Psychology*, 8, p.1366. Available at: <https://www.frontiersin.org/article/10.3389/fpsyg.2017.01366> [Accessed !no date!].

Sun, Y. & Ding, M., 2010. Quantum genetic algorithm for mobile robot path planning. , pp.206–209.

Sunardi, M. I., 2020. Synthesizing expressive behaviors for humanoid robots. , .

Suykens, J. & Vandewalle, J., 1999. Least squares support vector machine classifiers. *Neural Processing Letters*, 9, pp.293–300.

Terry, P. & Lane, A., 2011. Mood and emotions. *The New Sport and Exercise Psychology Companion*, .

Thornton, M., Miller, D. & Drechsler, R., 2001. Transformations amongst the walsh, haar, arithmetic and reed-muller spectral domains. , .

Toffoli, T., 1980. Reversible computing. , p.632–644.

Trippe, S., 2014. Polarization and polarimetry: A review. *Journal of The Korean Astronomical Society*, 47.

Tsitsiklis, J., Zoumpoulis, S. & Kreidl, O., 2012. Decentralized detection in sensor network architectures with feedback. , .

Tweedale, J., Ichalkaranje, N., Sioutis, C., Jarvis, B., Consoli, A. & Phillips-Wren, G., 2007. Innovations in multi-agent systems. *Journal of Network and Computer Applications*, 30, pp.1089–1115.

van Dommelen, L., 2018. Quantum mechanics for engineers. , . Available at: <http://www.eng.fsu.edu/~dommelen/quantum/style_a/> [Accessed !no date!].

Vazirani, P. U., 2012. Qubits, quantum mechanics, and computers. *EECS Instructional and Electronics Support*, .

Vukobratovic, M. & Borovac, B., 2004. Zero-moment point - thirty five years of its life.. *I. J. Humanoid Robotics*, 1, pp.157–173.

Wallén, J., 2008. The history of the industrial robot. *Technical report from Automatic Control at Linköpings universitet*, .

Wang, Y. & Perkowski, M., 2011. Improved complexity of quantum oracles for ternary grover algorithm for graph coloring. *Proceedings of The International Symposium on Multiple-Valued Logic*, pp.294–301.

Ward, D. & Volkmer, S., 2006. How to derive the schrodinger equation. , .

Warke, A., Behera, B. & Panigrahi, P., 2019. The first three-qubit and six-qubit full quantum multiple error-correcting codes with low quantum costs. , .

Wittek, P., 2014. Quantum machine learning: What quantum computing means to data mining. , .

Wood, L., Zaraki, A., Robins, B. & Dautenhahn, K., 2019. Developing kaspar: A humanoid robot for children with autism. *International Journal of Social Robotics*, .

Xie, F., Liu, X., Wang, J. & Wang, L., 2009. Kinematic analysis of the spkm165, a 5-axis serial-parallel kinematic milling machine. , pp.592–602.

Y, L., L, B., Y, B. & P, H., 1998. Gradient-based learning applied to document recognition.. , .

Yang, J., Awan, A. J. & Vall-Llosera, G., 2019*a*. Support Vector Machines on Noisy Intermediate Scale Quantum Computers. , . arXiv:1909.11988v1.

Yang, J., Awan, A. J. & Vall-Llosera, G., 2019*b*. Support vector machines on noisy intermediate scale quantum computers. , .

Yeshmukhametov, A., Kalimoldayev, M., Orken, M. & Amirgaliev, Y., 2017. Design and kinematics of serial/parallel hybrid robot. In: , pp.162–165.

Yongsheng Sang, Haixian Zhang & Lin Zuo, 2008. Least squares support vector machine classifiers using pcnns. In: *2008 IEEE Conference on Cybernetics and Intelligent Systems*, pp.290–295.

Zhang, Y., Deng, H., Li, Q., Song, H. & Nie, L., 2019. Optimizing quantum programs against decoherence: Delaying qubits into quantum superposition. *2019 International Symposium on Theoretical Aspects of Software Engineering (TASE)*, . Available at: <http://dx.doi.org/10.1109/TASE.2019.000-2> [Accessed !no date!].

Zivanovic, S., 2000. Parallel kinematic machines. *International Journal of Production Engineering and Computers*, 3, pp.49–54.

Zwanikken, J., 2018. Introduction to quantum mechanics i. , .

List of Figures

List of Tables

List of Abbreviations

CCNOT	controlled-controlled-Not
CNOT	controlled-Not
CPU	Central Processing Unit
CSWAP	controlled-Swap
DFA	deterministic finite autonoma
DFT	Discrete Fourier transformation
DOF	degrees of freedom
EPR	Einstein-Podolski-Rosen
FPGA	Field Programmable Gate Array
HHL	Harrow-Hassidim-Lloyd
HRI	Human-Robot Interaction
IAA	intelligent and autonomous automata
ID	Identity
LS-SVM	Least-Squares Support-Vector Machine
LTS	Long-Term Support
ML	Machine Learning
NUC	Next Unit Device
NISQ	Noisy Intermediate-Scale Quantum
PCA	Principal Component Analysis
PSU	Portland State University
QFT	Quantum Fourier Transform
QSVM	Quantum-Support-Vector Machine

qubit	Quantum bit
RAM	Random Access Memory
ROS	Robot Operating System
SQUID	Superposed Quantum State Initialization Using Disjoint Prime Implicants
SVM	Support Vector Machine