Nao humanoid for physical therapy rehabilitation

Prashani Jayasingha Arachchige

Master's thesis in computer engineering

Supervisor: Professor Johan Lilius

Faculty of Science and Engineering

Information Technologies

Åbo Akademi University

18th November 2021

Abstract

Rehabilitation of patients in the medical field is an extensive area of study where patients are assisted in recovery of their body functions. Furthermore, this enables regaining daily routine functions, speech impairment or physical movement in the body. Depending on the inability, the process is designed for each patient by planning specific goals. This thesis will focus on physical rehabilitation and a system implemented to act as a physical therapist humanoid using Artificial Intelligence and Robotics. Physical rehabilitation deals with all movement injury in the body. This section of rehabilitation has been growing with the addition of computer technology in the past decade. Currently, robotics is used in physical therapy where it assists a part of the human body through direct body-part attachment and assistance. Thanks to the development of artificial intelligence, human interaction with robots is now possible and the best example of study that is being done is creating a social robot as a companion for the elderly to help them with loneliness and emotional well-being. However, this thesis will use deep learning methods to program a humanoid that can act as a customized physical therapist which will be implemented on a programmable humanoid robot, the Nao. This companion will show the movements associated with the recovery goals and provide direct feedback on the performance of the patient. Therefore, the design of this system will contain natural language processing and computer vision for human-based interaction and communication. Both these processes are neural networks that provide analysis and feedback. It will also outline and evaluate the methods that are used for this system to work efficiently and what kinds of changes can be made to improve the accuracy of the humanoid. In addition, a brief discussion will show the scalability and the future developments necessary to complete the system. The benefits of having a humanoid is that the patient will be able to perform his recovery at home at any time without travelling and will be able to customize the robot according to his or her process of recovery.

Acknowledgment

I would like to wholeheartedly thank my supervisor Johan Lilius for supporting and guiding me through this whole project as well as motivating me from the beginning to the end of my thesis. In addition, I would like to express my sincere gratitude to Sebastien Lafond, Sepinoud Azmi and the teachers from the academic writing section of the university who advised and provided technical knowledge in order to complete my thesis. Furthermore, I thank all my friends, colleagues and family in encouraging me towards the success of my work.

Contents

List of Abbreviations

AT	*Assistive Technology*
AI	*Artificial Intelligence*
SDK	*Software Development Kit*
API	*Application Programming Kit*
OCR	*Optical Character Recognition*
DL	*Deep Learning*
ML	*Machine learning*
NN	*Neural Network*
CNN	*Convolutional Neural Network*
NLP	*Natural Language Processing*
POS	*Parts Of Speech*
ReLU	*Rectified Linear Activation Unit*
PAF	*Part Affinity Field*

List of figures

1.0 Introduction

Medical Rehabilitation is a way of assisting people who have reduced body functions due to sports injury, infections, genetic disorders, accidents and traumatic injuries, so that they can recover through specified goals according to the impairment. These goals are defined by environmental factors and the type of rehabilitation therapy. Rehabilitation is categorized into six different sections which are physical, occupational, speech, respiratory, cognitive and vocational [15]. In addition, the method of providing rehabilitation is said to be either inpatient where the individual will be continuously monitored and treated in a hospital or outpatient in which case the individual will schedule private visits to the clinic or therapist. According to the World Health Organization, 2.4 billion people profit from rehabilitation around the world [17]. Each person can plan their own process of rehabilitation according to specific needs validated by both the therapist and the patient. Involving the patient in creating the plan and educating has been encouraged by health professionals, as it creates better outcomes and provides better understanding of the patients own health improvements [16].

The process of understanding what the individual requires in rehabilitation can be defined through five steps, namely: identifying the problem, relating the problem to the varying and limiting factors, outlining the goal and selecting suitable measures, planning interventions and finally, evaluating outcomes [18]. The field of rehabilitation has been growing throughout the years through Assistive Technology (AT). Assistive technology for rehabilitation is a device or tools use for the aid of the individual to either assist recovery goals or daily life functions. The first type of AT is the self-propelled wheelchair that was invented in 1665 by Stephan Farffler [19]. Other well-known AT devices are crutches, hearing aids and dictating devices which can all be used in different types of rehabilitation processes.

Moreover, with accessibility to computer technology all types of rehabilitation have more tools and facilities to help patients and perform different treatments. Current advancements focus on robotics, mobile applications, virtual reality and gaming platforms. The benefits of the addition of this technology start from making the patients more motivated and interested in improving their cognitive functions to saving time on travelling and helping patients reach their goals much faster and more safely. Mossrehab technology consists of 25 robotic devices to assist patients in recovering their neurological deficits [4]. The devices are attached directly to the part of the body requiring assistance, in order to exert out the correct movement. Mossrehab also introduced simple games that accomplish physical goals by introducing simple body movements to improve cognitive functions.

The idea of this project will be a start, which will enable building a rehabilitation system with robotics to aid specifically with physical therapy. The system will be a robotic rehabilitation therapist with an interactive feedback system. There are two ways in which robotics can be used to support patients. One way is an equipment that helps with the movement of muscles through body-part attachment that has been developing well in the past few years. The other is an interactive robot that can act like a companion to the patient while therapy takes place. Necoro is a robotic cat released in 2004 which was designed to interact with patients as a friend in order to help them with their emotional well-being [8]. However, the main objective of this thesis is to create a system where a robot can play the role of the therapist to the patient and give feedback on movements performed by the patient along with an evaluation on the practical applications. It will be much faster than going to the therapist and can be done during the patient's own time as the robot can act as a stay-at-home personalized therapist coach.

The Nao robot, which is an autonomous humanoid robot, will be used for the implementation of this process throughout the thesis. This project begins by analyzing the capabilities of the Nao robot and adapting it to the role of a physical therapist. Next, the focus will be more on the computer vision technology that can be used to provide feedback to the patient. This will involve object and movement detection with an analysis of each physical activity to provide feedback to the patient. The reason behind this focus is not only to understand the limits of the computer analysis through the humanoid but also evaluate the efficiency of the process that can be done towards the completion of the system. The therapist coach must definitely be able to guide the patient in the right direction as any wrong pattern to the exercise can resort to a negative outcome of the patient's recovery process. Therefore, identifying positive and negative patterns of exercises is essential to the systems success.

This thesis will commence with two introductory chapters that will give the background of the robot that will be used for practical application and a history of already existing projects on robotics rehabilitation. Secondly, the main features of deep learning methods that are used in this project will be outlined. Thirdly, detailed information about the system that will be implemented along with the breakdown of step-by-step processes will be provided. The next

chapter magnifies the specific step which uses computer vision in the rehabilitation system and the implementation of its functions. In addition, the code will be outlined through which simple communication and interaction with the Nao robot is allowed more efficiently.
Furthermore, it will show the concept and progress of the system that would achieve the physical therapy robot. The following chapter will show the results of the implemented functions with detailed test cases to validate the system. The conclusion chapter will summarize the project as well as analyze the next steps that could be taken to complete and improve the system. In the discussion chapter, an evaluation will be performed to understand ethical side and the questions raised in healthcare robotics sector.

2.0 Rehabilitations Robotics

The use of robotics especially in therapy has been one of the key advancements in medical rehabilitation. Countries all over the world have budgeted significant sums in the past twenty years to create tools and instruments to help individuals with physical impairments, continue their daily routines. These tools can be either building an adaptable environment around the patient in the form of smart homes or use of robotics to directly aid the patient's impairment. The use of computer technology is said to revolutionize the field of rehabilitation, as the number of patients increases every year.

To begin with, there are 795, 000 patients in the United States and 1.1 million patients in Europe who suffer from strokes every year [20]. *Figure 1.0* shows that the research conducted on AT devices is increasing every year and the interest in improving this technology has been growing steadily each year.

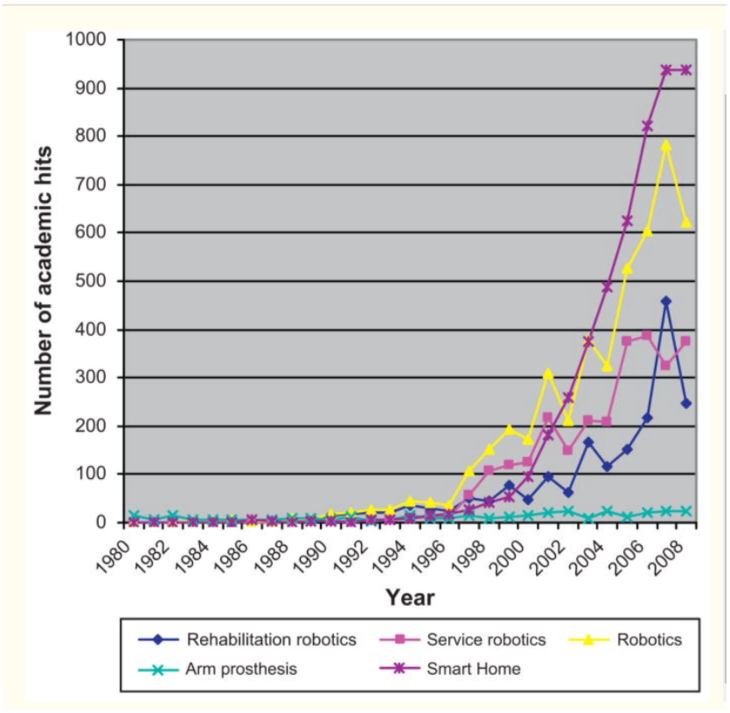


Figure 1 Research over the years on AT devices [20]

2.1 Brief History

This chapter will show the previous studies and work applied through AI and humanoid technology for the general supportive care of patients who require assistance in their daily routine. The assistance given to patients vary depending on the objectives of the study, but the applications used in the projects coincide with similar technologies such as AI, sensory and motor functions and computer vision.

1. The Spartacus telethesis was an experimental system that started in 1975, by researches in France. This was a model of a telethesis for quadriplegics [21]. The word telethesis was given as it was a robotic manipulation system for the handicapped without any attachment to the body. The controller of the device was adapted to the patient either through body-enabled prosthesis or a joystick [22]. Through fast developments, the Spartacus system was able to move, mimic and adapt to the environment through the controlling of the patient.

2. The MIT-MANUS robot designed by the Massachusetts Institute of Technology in 1998 was created to aid neuron-rehabilitation and recover sensory motor functions. The mechanics were built as a body-part attachment for the arm and shoulder, providing movement in the horizontal plane [23]. The MIT-MANUS robot trial was tested by

over 250 patients during a twelve-week period and the results were safe with no detrimental effects [24], [27].

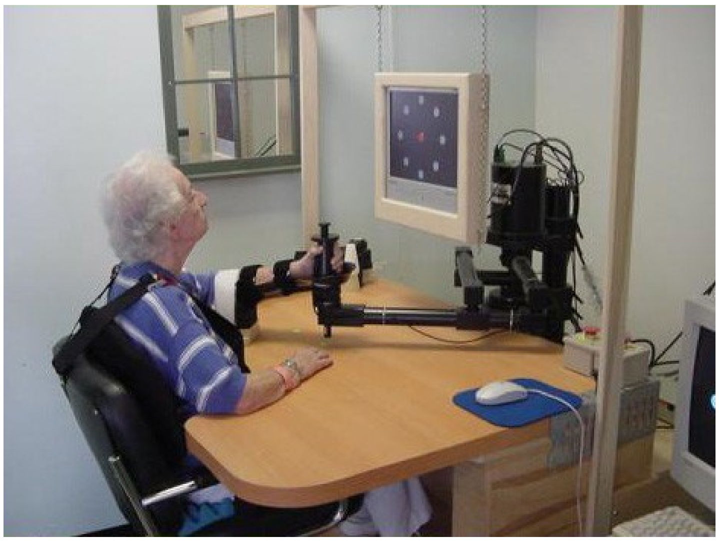


Figure 2 Stroke Inpatient during Therapy at the Burke Rehabilitation Hospital (White Plains, NY). Therapy is being conducted with a commercial version of MIT-MANUS (Interactive Motion Technologies, Inc., Cambridge, MA). [24]

3. The humanoid robot ARMAR III with a height of 70 meters and an upper body resembling a humanoid. The goal of this project is to produce a robot that is capable of interacting with a household surrounding and manipulate objects assimilated around it. The two main features were perception and motor functions that aided the humanoid to recognize objects as well as avoid collision. During the CeBIT and Automatica exhibitions, along with the previously specified features, the humanoid was able to use speech recognition, 3D face and hand detection and basic hand movements. [28]

4. Cody is an autonomous assistive robot that mimics hand movements in order to clean a patient after a bed bath. When the area required to wipe is selected, image capture and coordinates of the capture is used to determine the area of the body that needs to be wiped. [29]

5. Care-O-Bot 3 is aimed to be a service robot to humans in small activities in a household environment. It is used to clone the simple tasks of a butler and, therefore, the robot includes a tray for moving objects, flexible arms and hips to imitate bowing down. [30]

6. RIBA is a robotic assistance specifically designed for the nurse care industry. It consists of tactile sensors in the upper arm, shoulder, hand and forearm and is developed to carry

patients from one point to another. RIBA is 1.4 meters high and when tested it was able to carry a load of up to 63 kg. It contains a DC motor with omnidirectional wheels and can be used up to 1 hour continuously. [31]

2.2 The Nao humanoid robot

The Nao is an autonomous-based humanoid robot that was created by Aldebaran Robotics. The company was founded in 2005, then sold to Softbank Robotics, a French company, in 2015. Softbank Robotics has released three types of robots: Pepper, Nao and Whiz. The Whiz is a vacuum with the ability to memorize patterns of up to 600 cleaning routes [32]. The Pepper robot is a 120 cm humanoid that can support facial recognition and detect basic human emotions. Moreover, Pepper has many similar functions as the Nao robot with sensors and a fully programmable platform [33]. The pepper robot is used all around the world for testing different concepts due to its easily accessible features and sensors. They are mainly used in greeting people or answering frequently asked questions in companies, hotels and hospitals. One example of its usage can be seen in Japan where the pepper is used for greeting customers in hotels to promote social distancing due to covid-19 pandemic. In addition, it is used as a mediator in hospitals to reduce direct contact with covid-19 patients and hospital workers. The whiz is also used in cleaning the floors of different hotels. [34]

The Nao was released in 2008. It is used worldwide in teaching children, practical applications by programmers and university students, as a practical learning tool, a companion and to experiment on implementing specific methods on its existing hardware. The purpose of this robot is to be a tool for the general public where people can use or implement everyday activities. The robot has a quad core CPU, 4 GB DDR3 (Double Data Rate 3) RAM with Atom E3845 as the processor (9). It has a height of 57.4cm and a width of 27.5cm. In addition, a lithium-ion battery is used for charging the robot. The robot can be connected to a computer through the Wi-Fi or Ethernet cable, directly attached to the computer. The benefit of this robot is that it can be used as a tool, as it is built with the ability to move and rotate joints as well as different sensors that allow interaction and communication with humans.

There are motors that aid movement, speakers, four directional microphones, two 2D cameras to perceive objects and LEDS that are already built into the robot. Furthermore, it has Force Sensitive Resistors (FSR), ultrasound sensors, a gyro meter, an accelerometer and joint position sensors that coordinate the movements. The contact and tactile sensors aid with the rigid and free movement capabilities. Apart from this, the humanoid contains seven touch sensors in its hands, feet and head as well as sonars and an inertial unit to identify the surrounding and detect itself in space. The company has also added a USB port to which we can connect another microcontroller (Ardiuno) or another sensor according to the requirements of the user. The different sensors and motors of the robot are shown in *figure 1*.

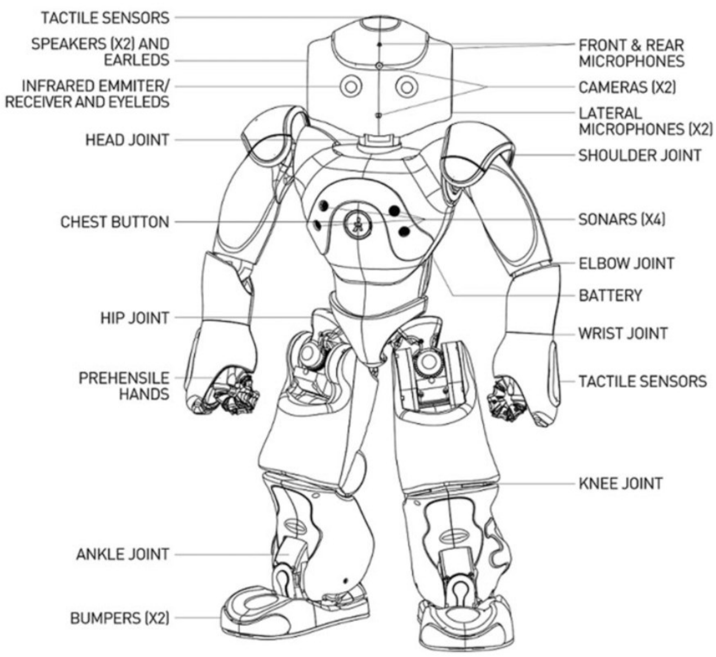


Figure 3 Nao Robot

https://www.researchgate.net/figure/The-Nao-robot-The-camera-used-for-recording-the-child-activity-is-the-one-on-the-top_fig2_325544590

The robot comes with its own prebuilt functions and programming application which is known as Choregraphe Suite. It is an application where you can create a behavior for the robot with a graphical interface, python or C++. The company has also released an SDK (Software Development Kit) in both python and C++. This also contains the prebuilt functions and APIs (Application Programming Interface) that allow the robot to interact [35]. These two main interactions occur through communication which can be done in different languages and movements: from walking to dancing. Naoqi is the operating system of the robot that controls and runs everything within the robot. The operating system is what contains the API modules which are used to control the behavior of the robot. These API modules of the OS contain the core APIs, along with emotion, interaction engine, motions, audio, vision, people perception and sensors and LEDs [36]. Therefore, this is a major tool that can be used for several different requirements as it contains a wide variety of functions.

If one wants to build a system to perform a specific task, a behavior is created in choregraphe and uploaded to the robot. This behavior uses functions that already exist in the OS or the programmers can add their own python or C++ code according to the purpose for the robot. The built-in functions which are known as box libraries in choregraphe can be categorized as follows:

1. Animation: This contains functions of movements during speech, the moods of interaction which are positive, negative or neutral. Moreover, a few entertainment movements such as dancing, music (guitar, saxophone sounds), sports and animal actions.
2. Speech: This consists of speech settings such as volume, speaker language and recognition language as well as the blocs to which the user writes what the Nao will say (text or animated speech).
3. LEDs: The LEDs attached to the Nao are controlled through these functions. Some examples are the ability to change the color, blinking and twinkling of the eyes.
4. Multimedia: This contains the ability to access emails, recordings, videos and images external to the application in order to use them for a particular purpose.
5. Movement: This includes all movement calibration which means the user can access the motors, the postures (sitting, standing), the orientation (look, point), safety measures (fall detection and anti-collision), and also detect objects for obstacle avoidance.
6. Sensing: The control of vision and hearing which means that touch, sonar and human-understanding related functions are seen here.
7. Programming: This consists of all technical built-in functions, such as a timer or delay, which give the Nao a few seconds to process certain data. It also includes mathematics, memory information, logic and other program-specific functions. The user can also find the empty python script with which one can introduce new functions that are not already within the Nao.

These built-in box libraries have different features of their own and the written code can also be accessed by simply double clicking the bloc. To better understand the features, if the hands in the movement module were selected, the features that can be changed within it would be the choice of which hand (left, right, both) and action (open, close). The box libraries also have the option to select the number of variables along with the type of variable that will be on the input and output of the bloc (bang, string, number).

Choregraphe also helps with testing different programs and shows exactly what the robot can see and perceive. The figure below shows the choregraphe platform that is used for creating programs and uploading behaviors into the Nao.

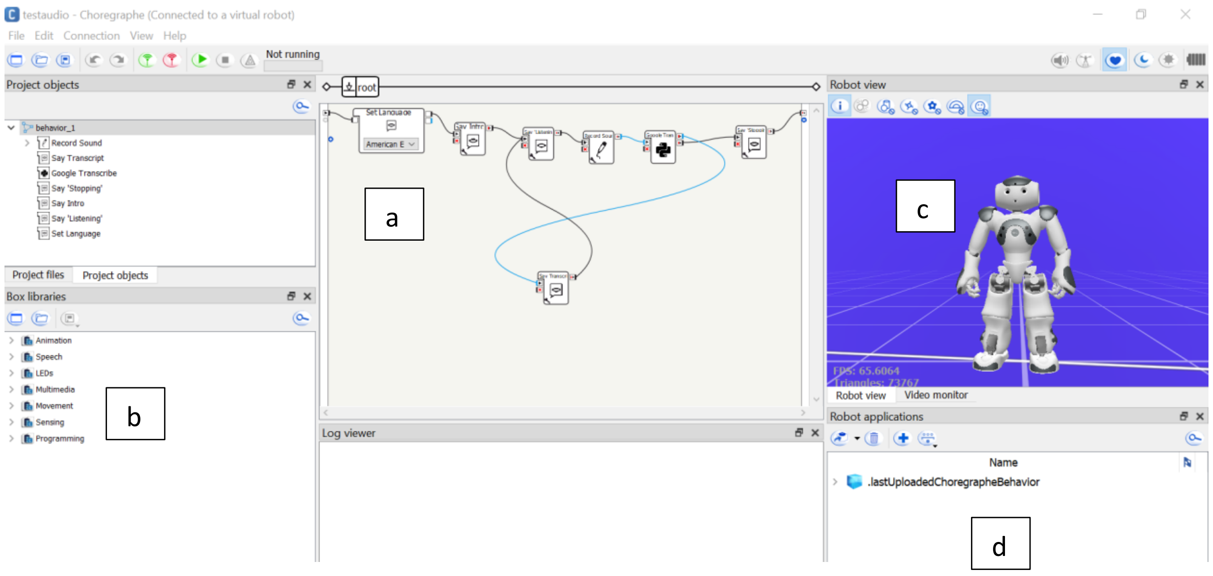


Figure 4 Choregraphe Suite by Aldebaran Robotics

The descriptions of the four sections a-d of the figure above are as follows:

a. The main area of programming where the user can create the behavior and a closed looped system which will then be uploaded to the robot. The blocs must each be connected in a closed loop system for the upload to work. The user can find the code of each bloc along with its different settings by double clicking it.

b. The box libraries that are used to create the behavior. These libraries are every function that can be used with the robot and each upgraded version will have an additional function or improvement to the previous box libraries.

c. The positioning of the real or the virtual robot. The user can also change the menus to see what the robot can see from its video camera in the case of the real-life robot. The video camera menu also contains a set of options that is used for facial recognition. This means that it has the option to pause a frame, study it and add it to a vision recognition database.

d. Shows the behavior that was uploaded to the robot along with any additional uploads which can be a song or even the language that is currently used for the behavior.

There are different versions of this robot, the latest being version 6. Each version has added built-in functions and improvements to the system. The additional improvements help in creating new behaviors to the robot.

3.0 AI Methods for human Interaction and feedback analysis

In order for the system of a rehabilitation therapist to work, the Nao humanoid must be able to process and interact in a similar way to a human. This means that human like interaction must be immediate and valuable analysis must take place. To converse, analyze and provide feedback there needs to be machine learning process as it gives the best outcomes. This chapter will provide background into the topic of AI and the methods that can be used to approach the system.

3.1 Machine Learning

The father of computing, Alan Turing created the first test to verify if a computer can be intelligent through his review paper published in 1950 [60]. Even though there is a lot of arguments towards the theory, it is still considered to be a foundation in proving the presence of intelligence in a computer. AI is the science of creating a machine that would be capable of duplicating the works of a human mind, namely their problem solving and decision making skills [61]. It has brought speech recognition and image analysis amongst other tasks through computer science into a whole new level of understanding which has made jobs thought impossible for a computer to do, possible. When a computer is given a task, it must be given precise detail to produce an output, but this wouldn’t work for some practical applications like OCR (optical character recognition) where there can be thousands of ways in which a single letter is written. In this case the usual way would be to define the thousand possibilities but due to machine learning, this problem is eliminated. One of the first papers that studied machine learning was by Arthur Samuel who successfully created the self-learning checkers game [62]. Machine learning is a method to mimic human behavior through models that has been trained with data by providing explicit instructions. These models are trained to map predicted outputs by understanding the features of the data given to train itself. Therefore, these data are presented in feature vectors that undergo feature processing [63].

There are four main different approaches of machine learning: [64], [65]

- Supervised learning: This is when the data given contains labels. Labels are what can define or classify features. Hence, the dataset will contain the feature vector and the labels. Since labels are provided the data can be classified into different sets according to the features. Some examples of supervised learning are Classification and Regression problems. Classification is when the output variable is classified, like if the features given are of a box or circle. Regression is when the features indicate a logistical value such as price of apartment according to the features inside it.
- Unsupervised learning: This type of learning contains no labels in the dataset, in other words, the algorithm learns without an instructor. A simple example is audience segmentation, where the population of 15 to 20 year old spend more time outdoors whereas an audience of 30 to 50 spend their time indoors. Two examples of unsupervised learning are association and clustering algorithms. Clustering algorithm is where the data is grouped according to a specific common manner of the features.
- Semi supervised learning: this type of learning is a mix of supervised and unsupervised learning but there will only consist of some data that is already labelled. Most data will be unlabeled.
- Reinforcement learning: this is when a learning agent perceives the environment and assesses actions which can be in the form of rewards or consequences. In basic terms, it is a student learning without the teacher. The learning curve will be based on the actions of the student which can result in a reward or consequence [66].

Deep Learning (DL) which is a part of machine learning is guided by the architecture of the human brain with the creation of artificial neural networks. Current examples of deep learning methods are autonomous vehicles, sentiment analysis and virtual assistant. DL is a subset of machine learning, that is a part of Artificial Intelligence, showing tremendous effects in the past decade through different fields. ML works through human intervention where the features or data given is defined initially for the algorithm to be trained. However, DL does not require any human intervention as it uses the neural networks to distinguish the features by itself as seen in the figure below.

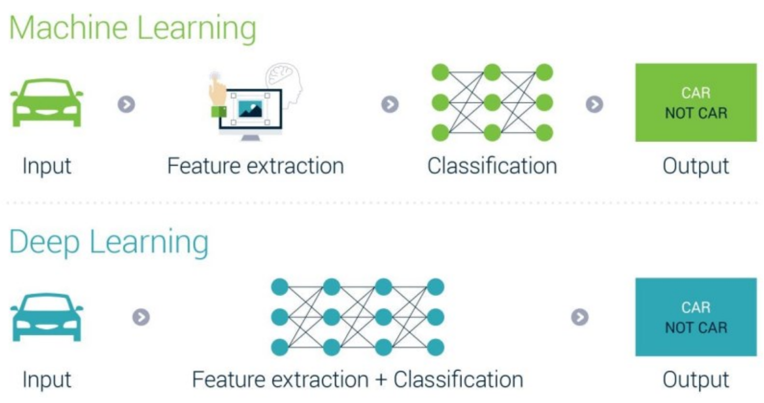


Figure 5 Machine learning Vs. Deep learning [69]

3.2 Neural Networks

Neural networks are what is behind DL methods and its architecture is motivated by the pattern of neuron connections in the brain. Artificial Neural networks contain nodes that corresponds to a neuron of the brain, and these are the known as the processing units [67]. These units form layers that overall create a network structure. The layers can be divided as the input, output and the hidden layers. The input layer gathers the information, while the output layer sends out the signal. The hidden layers send information within the input and output layers as seen in the figure below:

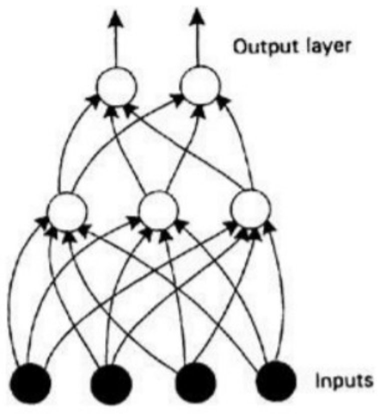


Figure 6 Neural Network [68, page 14]

The connecting channels are defined by a weight, which is a single number, so that each input from the input layer is multiplied by the weight before it is transferred to the next layer of nodes, by comparison to a threshold value. The nodes are mathematical functions that act on the input. The network also contains an activation function that determines the activation of certain nodes depending on the input, before it is passed onto the next layer. All this together provides an artificial neural network. NN can train through both supervised and unsupervised learning. Deep learning can contain two networks which are convolutional neural networks and fully connected neural networks. Fully connected neural networks are the layers to which all nodes of one layer are connected to the nodes of the next layer. However, CNN does not have all input layer connections and, therefore, the weights for a layer are much smaller. This means that the number of parameters is reduced with the most minimized loss of quality. As a result, it can learn in a fluid manner, so it is very well-known of image data analysis and natural language processing.

3.3 Natural Language processing

One major example that uses CNN is NLP (Natural Language Processing). Natural Language means the languages that are used in daily communication (English, French, and Swedish) by humans, whereas artificial languages are the mathematical and programmable languages. NLP is a set of computational techniques that is used to analyze natural language for linguistic analysis. Current examples that use NLP are chatbots, sentiment analysis, speech recognition (Siri, Cortana), spell checking, machine translation and even the advertising matching done automatically on the computer. All these examples require understanding of human language. NLP requires various steps in analyzing the text which are [70], [71]:

- Word tokenization: This is where words are broken into several tokens, which are small units that will be processed in the next step. The text, “my name is” can be separated into three tokens.
- Stemming: This is when all words are normalized into their root formula. The words “larger, largest, largely” all contain the root word large.
- Lemmatization: This step performs a morphological analysis of the words using a dictionary to relate all words back into its original word form. Both words ate and eaten are derived from the word eat.
- POS tags: Parts of Speech tags are used to define the tokens into different parts of speech which can be in the form of noun, verb, adjective, article amongst other POS.
- Named entity recognition: This is to recognize certain words that could be a name of a person, city or company. Sometimes POS can categorize different words in multiple

ways. The word act can be both a verb and a noun, which is why named entity recognition is introduced.

- Chunking: In the end all the separated tokens are grouped together into chunks to understand the sentences or phrases

The importance of NLP for this project is so that the human-like interaction between the Nao robot and the patient is more successful, as it is important for the robot to understand the emotions of the patient as well as the sentences said by the patient. One main tool that provides this form of communication is known as Chabot. It is a current example of NLP that communicates and understands human language. It is seen in customer service chats on different websites.

3.4 Computer Vision

Computer vision is a field of study that uses AI to understand digital images and perform actions or obtain knowledge from them [72]. The ML algorithms can be used for two different applications. One is to improve the perception of the internal representations of the image and the other to obtain information from the representation of the image. This is one field that uses CNN to accomplish analyzing images. The study of computer vision through AI began in the 1960s. Since then, more data have become available to train models for different purposes. Some examples that use computer vision are: [73]

- Facial recognition: Using certain measurements and angles of the face, images can be analyzed to identify different people. Facial recognition is used in the public sector for security purposes.
- Object detection and tracking: This is widely used in autonomous vehicles to detect objects and avoid collisions as well as in the medical field to detect different forms of cells and cellular structures.
- Image classification: This can identify certain features of the images and class them into different groups.

The figure below shows a CNN architecture for image analysis to identify an apple on a tree. It consists of convolutional layers, pooling layers and fully connected layers. The convolutional layers merge two sets of operations. In other terms, it takes the input and uses filters to create feature maps. The deeper the layers are, the more complex feature extractions take place. The pooling layer reduces the number of parameters by down sampling. An example of pooling is max pooling where the height and weights are reduced by taking the maximum value of a pooling window. However, the depth does not change. The layers also perform a non-linear activation function (ReLU). These layers focus on creating the feature maps. The next layer is a fully connected layer that completes the CNN in order to classify and provide the output. [75]

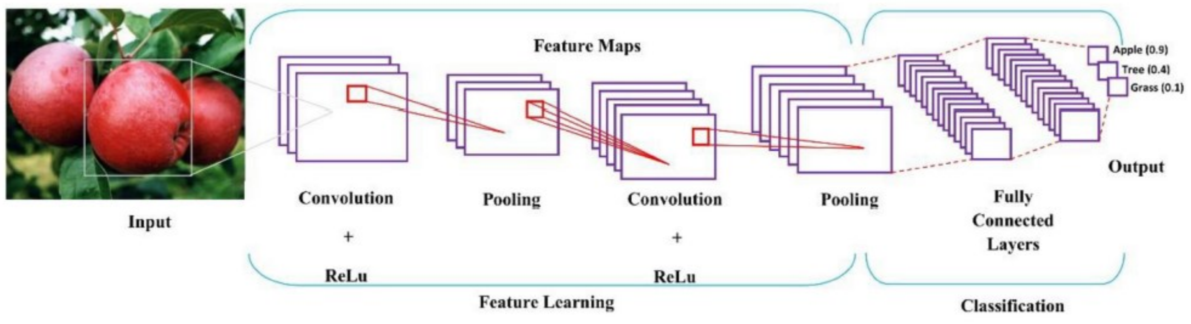


Figure 7 Architecture of a CNN for image classification [74, page 5]

Analyzing images is a vital section in this project, as the Nao needs to not only visualize but also understand what the patient is doing. This simple architecture will help understand the method of image analysis implemented in the next chapter.

4.0 Rehabilitation system prototype

The process of having and interactive robot to act as a therapist means that it must be able to communicate, visualize and give specific feedback to the user. In order to perform this task, we can categorize the system into three main parts, as shown in *figure 5*.

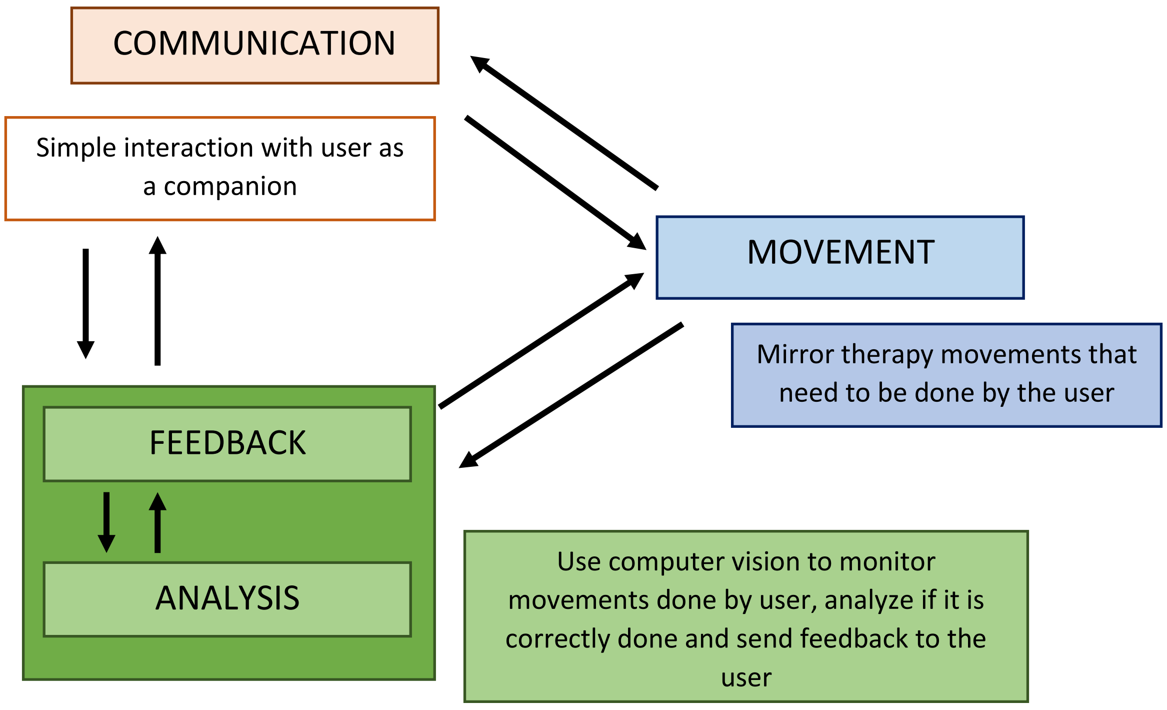

COMMUNICATION
Simple interaction with user as a companion
MOVEMENT
Mirror therapy movements that need to be done by the user
FEEDBACK
ANALYSIS
Use computer vision to monitor movements done by user, analyze if it is correctly done and send feedback to the user

Figure 8 Rehabilitation System Basic Prototype

The three parts interact with each other in order to create a system that can act as a fully functioning therapist. The first step is communication and interaction where the robot would achieve human-like interactions. Firstly, it will greet the patient and have a normal conversation to understand the well-being of the patient. Next, it will start the exercise scheme that will be predefined by the physical therapist in charge of the patient. Even though the humanoid will act as a therapist, the physical therapist of the patient is in charge of analyzing the needs of the patient and preparing recovery goals for the patient at the start. Then the therapist will predefine a set of exercises that will be performed and analyzed by the robot. The main objective of the robot is to prepare and act during the physical therapy sessions.

The robot will explain each step while showing how it is performed and the patient will mimic the exercise. Then it will observe and analyze each exercise completed by the patient and provide feedback as to whether the patient is following the exercise correctly. In order for this system to be a success, the humanoid must be able to understand the correct movements and provide possible solutions to the patient for correct posture orders. It must also be able to motivate the patients when they are performing well. In addition to performance evaluation of the patient, the interaction capability of the robot must comprehend the well-being of the patient. This means that the robot must be able to understand or act when the patient is in pain or if the patient has a problem during one of the exercise sessions.

Figure 6 shows further analysis of the system:

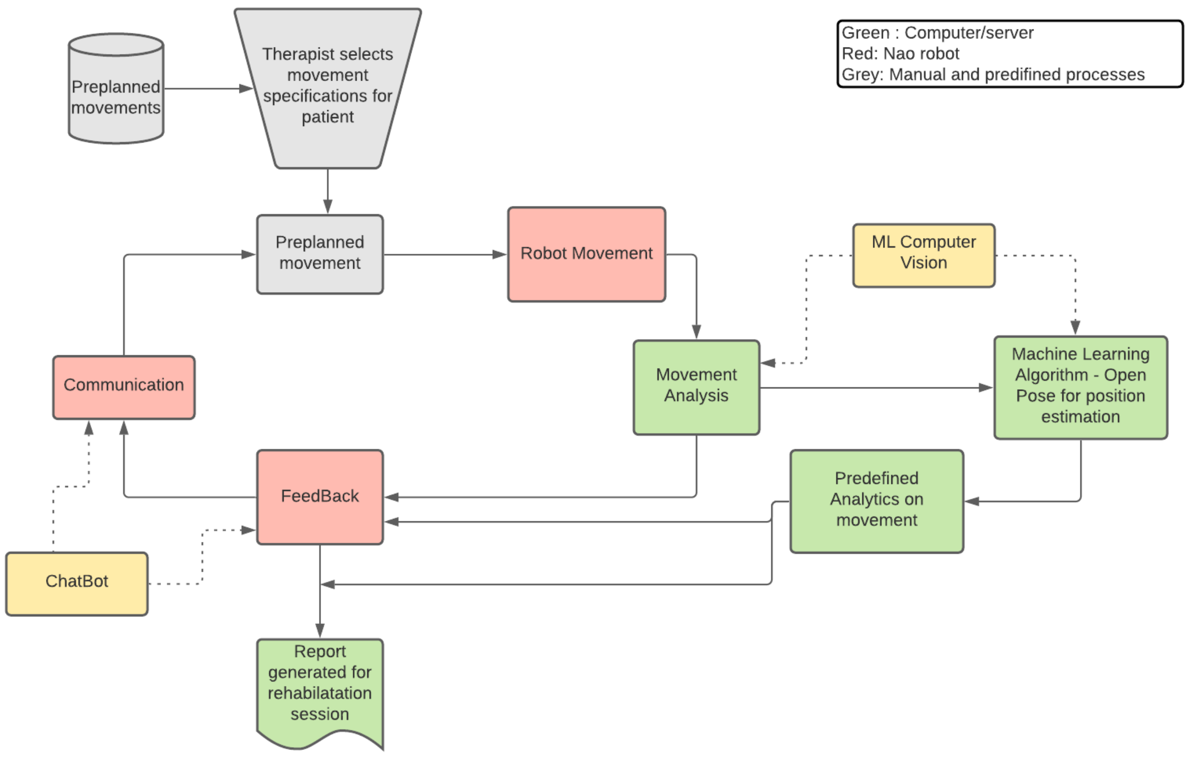

Preplanned movements
Therapist selects movement specifications for patient
Green : Computer/server
Red: Nao robot
Grey: Manual and predifined processes
Preplanned movement
Robot Movement
ML Computer Vision
Communication
Movement Analysis
Machine Learning Algorithm - Open Pose for position estimation
FeedBack
Predefined Analytics on movement
ChatBot
Report generated for rehabilatation session

Figure 9 Deep dive into the system prototype

As seen in the diagram above, the system involves different aspects of technology including programming robot movements, creating an exercise database, designing user interfaces and applying AI technology to interact and analyze the patient. The main focus in the implementation part of this thesis will be on the concept of feedback and analysis, as it is vital to understand what exactly the users do in order to give them more assistance to reach their physical goals. The first step of the feedback system is to understand what the robot can see and how we are able to analyze the data through computer vision. This will enable the Nao to verify if the patient is performing the movement correctly and encourage them or if the patient does the movement incorrectly, respond in such a way that the patient is able to correct the movement. As seen from figure 6, the two main technologies for feedback and analysis are machine learning computer vision algorithm and Chatbot for patient-robot interactions.

In order to visualize the functioning of the system, figure *7* demonstrates a use case for patient x whose therapist has given the T exercise as a part of the recovery goals exercises. The test case shows the user interaction and the analysis performed by the robot to provide accurate feedback. The green blocs are the patient's side of the conversation whereas the yellow blocs are the robot's. The grey boxes show the posture analysis of the patient.

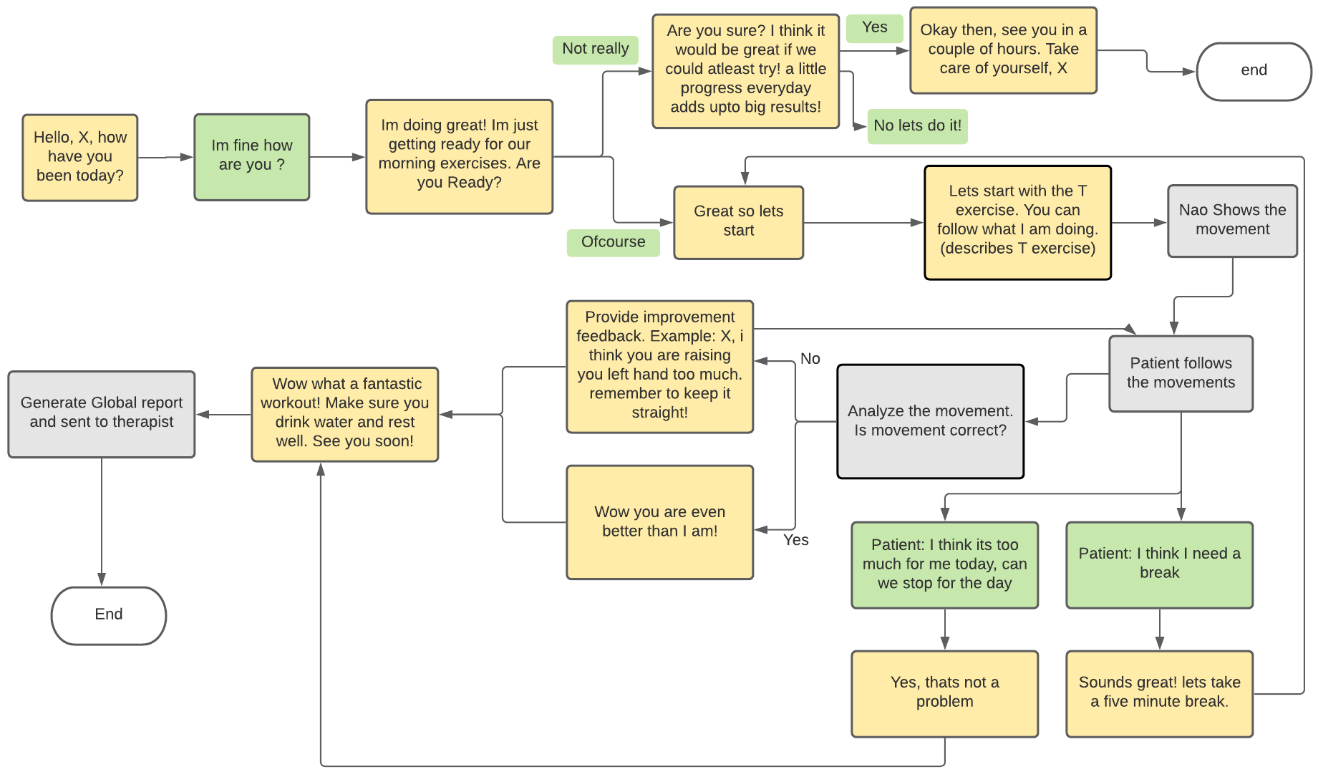

Hello, X, how have you been today?
Im fine how are you ?
Im doing great! Im just getting ready for our morning exercises. Are you Ready?
Not really
Are you sure? I think it would be great if we could atleast try! a little progress everyday adds upto big results!
Yes
Okay then, see you in a couple of hours. Take care of yourself, X
end
No lets do it!
Ofcourse
Great so lets start
Lets start with the T exercise. You can follow what I am doing. (describes T exercise)
Nao Shows the movement
Patient follows the movements
Analyze the movement. Is movement correct?
No
Provide improvement feedback. Example: X, i think you are raising you left hand too much. remember to keep it straight!
Yes
Wow you are even better than I am!
Wow what a fantastic workout! Make sure you drink water and rest well. See you soon!
Generate Global report and sent to therapist
End
Patient: I think its too much for me today, can we stop for the day
Patient: I think I need a break
Yes, thats not a problem
Sounds great! lets take a five minute break.

Figure 10 Test case of patient X performing exercise T

Therefore, to implement certain functions, the code needs to be inserted into the behavior model of the choregraphe. The language used in choregraphe for this thesis is python version 2.7. The python code for the built-in functions can also be visualized and changed according to the requirements given. A new python script can also be added to the program, if it is something that is outside the built-in functions. To test this system, the Nao robot is used. The Nao performs actions based on the behaviors that are uploaded directly through choregraphe.

There are also instances where the program can exist outside of the choregraphe platform. These are complex programs that will require a larger amount of computational power, such as machine learning models. This type of code will need to be connected to the robot through its IP address where different API modules can be accessed. This is because the processing power of the robots OS is less and will slow the runtime of the machine learning models if it is directly processed on the robot. For the main focus of this system to be tested, the camera of the Nao robot needs to be used outside the choregraphe platform for posture analysis. The Audio and Speech APIs will also be required for the communication part of the system.

4.1 Text-To-Speech Enhancement

The Audio module of the Nao OS contains a text-to-speech API (ALTextToSpeech) that has the ability to understand what the user says. It contains the test-to-speech engines which identify the words based on the selected language. Different languages use different engines to understand what the user is saying. Apart from English and French, the Nao robot supports more than 10 different languages in its platform [37]. Most of the languages use either the ACAPELA or NUANCE engine to synthesize the text. The Nuance speech engine was created by Nuance Communications, an American company founded in 1992 [38]. Consisting of many neural networks, it is used mainly for commercial purposes to provide personalized customer experience in various industries. Acapela was created by a European company, the Acapela Group. In 2003, three European companies were combined to form the Acapela Group. (Babel Technologies, Elan Seech and Infovox) [39]. These two engines process the audio and convert speech to text in the Nao operating system.

As communication is a vital part of the system, the robot must be able to understand exactly what the user is saying in its entirety. Therefore, Google Speech API is introduced to the system as a first step. The current API that already exists in the operating system takes a longer time to process the text and respond according to the test given. The ALSpeechToText module also has difficulties identifying words in different accents and at lower volume levels. Google Speech engine contains speech-to-text that is easily accessible and widely used as it possesses an accurate model. In practice, the patients who will use the system will not have the same voice. Hence there needs to be accurate conversion of different accents, voices and volumes for successful communication between the robot and the patient.

4.1.1 Google Speech API

Google Speech API is made of advanced neural networks in deep learning in order to understand spoken words (6). The main functionality that will be used from this API by the Nao is the Speech-to-Text (STT) functionality, where we can convert recognized sounds into words. It also has the ability to translate or understand over 120 different languages. This API is being used all over the world and is a very well-trained neural network that can understand voice even in different accents. The benefits of the API are the ability to customize it to words in specific fields, because it has a vast vocabulary as well as speech adaptation, recognition through different input devices, automatic punctuation, language detection, filtration of noise and words along with recognizing different speakers. In addition, it can also output real-time streaming of speech (6). The most important advantage of this API to the Nao robot is the ability to replace its current speech package and give a more effective output to the user.

Google Text-to-Speech python libraries exist that can easily be imported into the code. However, the limitation of the system that is used in the robot, the version of python in the application is not compatible to import the existing Google speech library and can only import pre-installed libraries to the system. Another method would be to directly access the Nao, but root access was not possible and the behavior application must first be able to run on the computer. In order to complete this task, a python script was created with path to packages necessary for Google speech as well as the credentials required to access the packages.

Every python script written in choregraphe has four main methods in the class. These are initialization, deletion, on loading, on unloading and on starting. These methods are used to declare and divide the code. The method ‘on starting’ is mainly regarding the code that will be uploaded into the Nao while the program is running. The created python script will then be connected within a closed loop system and uploaded to the robot. The closed loop will contain the listening section where the robot will listen to what the user says and then the Google Speech API is applied [40].

The Google Speech-to-Text contains three different forms of recognizing speech:

- Synchronous Recognition is when the audio of a duration of one minute is analyzed and processed altogether and sent as an output directly.
- Asynchronous Recognition can take in audio input for up to a duration of 480 minutes, which means that it is used for long running operations.
- Streaming Recognition uses bi-directional communication and has the capability to simultaneously capture, analyze and provide output on live audio. This means that while an output is being sent, the new audio will already be in the process of being analyzed.

To begin with, a test behavior was created where the robot is able to send audio to the Google Speech STT and repeat what the user had said. In order to perform this, asynchronous recognition was used. This uses REST (Representational State Transfer) and gRPC (google

Remote Procedure Call) communication. REST generally includes a creation of .json files whereas gRPC framework allows bi-directional communication [41].

The main parts of the test code are provided below. It shows how Google speech API was connected to the Nao robot and tested.

The python library paths were given to the robot.

```
GCS_PYTHONPATH = '/usr/lib/python2.7/site-packages'
GCS_CRED_PATH = '/home/nao/.config/gconf/transcribe-from-file-5f48bc943a22.json'
```

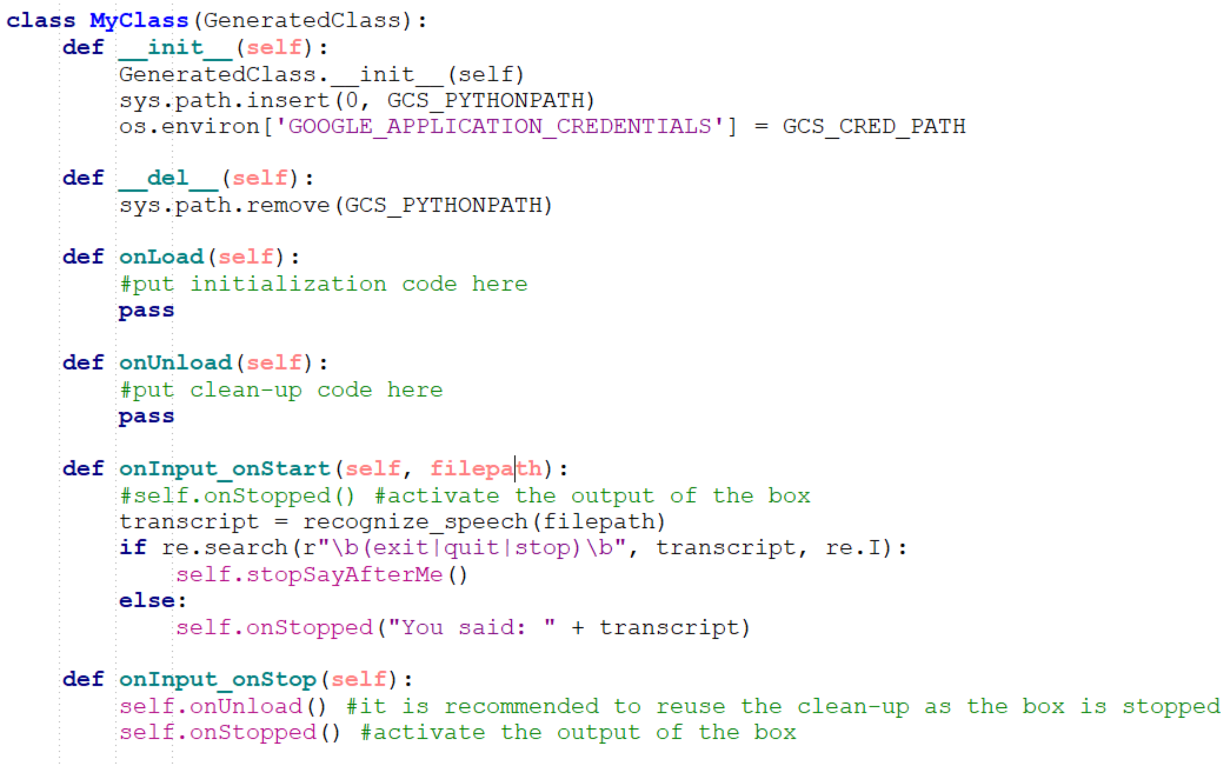
Figure 11 Python Paths

Then, creating a function that connects with the Google speech API and returns a .json file. At the input, the Nao will listen to the audio and convert it to a recording audio file. This part is already completed through the built-in box libraries. The audio file is then processed to return the transcript of the .json file. The .json file contains two important pieces of information. One is the transcript which is the actual text that was processed from the audio file. The second is the confidence level which shows the accuracy of the text. In the function below, the transcript is received and a response is returned. If the robot did not hear anything, the output of the transcript is given as none and the Nao says the user to repeat the sentence again.

```
"""
Takes in the path to the audio file.
Returns the string of text to be spoken.
"""
def recognize_speech(filepath):

    import google.cloud.speech_v1 as speech

    client = speech.SpeechClient()

    with io.open(filepath, "rb") as audio_file:
        content = audio_file.read()

    audio = speech.types.RecognitionAudio(content=content)
    config = speech.types.RecognitionConfig(
        encoding=speech.types.RecognitionConfig.AudioEncoding.LINEAR16,
        sample_rate_hertz=RATE,
        language_code="en-US",
        audio_channel_count=4
    )

    response = client.recognize(config=config, audio=audio)

    print( str(response.results[0].alternatives[0].transcript) )

    if response.results[0].alternatives[0].transcript == None:
        return "I didn't hear anything!"

    return str(response.results[0].alternatives[0].transcript)


def DEBUG_print(dict, func_loc, dict_name, temp):
    print("------------------")
    print( dict ) # simple output
    print("------------------")
```

Figure 12 Google Speech API Implementation

The main class of the program that contains the initialization, to which the Google credentials are added. At the end of the process the paths will be removed. At the start of the function, the recognize speech class will be used and the transcript will be repeated by the robot.



```
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)
        sys.path.insert(0, GCS_PYTHONPATH)
        os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = GCS_CRED_PATH

    def __del__(self):
        sys.path.remove(GCS_PYTHONPATH)

    def onLoad(self):
        #put initialization code here
        pass

    def onUnload(self):
        #put clean-up code here
        pass

    def onInput_onStart(self, filepath):
        #self.onStopped() #activate the output of the box
        transcript = recognize_speech(filepath)
        if re.search(r"\b(exit|quit|stop)\b", transcript, re.I):
            self.stopSayAfterMe()
        else:
            self.onStopped("You said: " + transcript)

    def onInput_onStop(self):
        self.onUnload() #it is recommended to reuse the clean-up as the box is stopped
        self.onStopped() #activate the output of the box
```

Figure 13 Google Speech Addition into the Nao

4.2 Human Pose estimation

Deep learning models can be used to recognize movements of patients and analyze them in order to give feedback to patients in physical therapy. Computer vision technology has created algorithms that can recognize and detect objects. The next step in the system is creating a method, that is able to detect the patient and provide a useful output that will aid the analysis of the exercises. The model must be able to both detect the patient and recognize the position accurately. As an example, if patient X is performing the T posture exercise, the robot must be able to detect the positioning of the arms, legs and the joints in the body to evaluate the performance of the patient. In order for this to be possible, human pose estimation through deep learning is used. Pose estimation through computer vision is having greater accuracy levels due to convolutional neural networks [42].

To create a pose detection model, two approaches can be used and they are known as bottom-up and top-down [42].

- The top-down method first detects the person through object detection, creates a bounding box around the person and then performs point estimation. Finally, it performs a calculation to creates the pose. The drawback of this approach is that the rate of process is proportional to the number of people in the image as the pose estimation will analyze sequentially for each person [43].
- The bottom-up approach detects all the joints and parts of the body (all the points) first and groups them together to create one pose structure.

Different approaches for human pose estimation already exist in both 2D and 3D format. The DeepPose released in 2014 was the first main methodology that applied neural networks into human pose estimation. This approach used Deep Neural Network (DNN)-based regression to predict different body parts [44]. It contains four convolutional layers and three fully connected layers along with the L2 loss to train the model. However, the fixed input size limited the model to observe finer details and so, the cascade regression was introduced. This polished the pose and provided much more accurate results [45]. Efficient object localization using CNN, convolutional pose machine, human pose estimation with iterative feedback are some of the different models created for human pose estimation [44].

4.2.1 Open pose

The model that is used in this thesis is the OpenPose Estimation model. The OpenPose model uses image recognition and AI-centered analytics to create human pose estimation outputs. Pose estimation of a human body takes the aid from the skeleton system to create a set of points that will represent the alignment of a human body. A relevant set of points or coordinates united through a connection will create a pair that in whole will create a full skeletal structure of the human posture on the image frame [46]. Normal object detection models identify and create a box around the detected object, but pose estimation can specifically find the connecting key points of the object.

OpenPose is one of the models that have the ability to detect the joint position pairs in the human body, including hand, face and foot. Depending on the requirements, OpenPose is able to create the joint analysis in one single frame. In total, 135 key points are detected which, in other words, are the points or coordinates creating the pairs of the body. The model can be used in python, c++ and unity plugin implementations [47]. OpenPose started with 2D single body pose estimation and now it can perform the following analysis [48]:

1. Real-time 3D single-person key point detection
2. Real-time 2D multi-person key point detection with different sets of key points for different parts of the body. For the whole body along with foot, there are 15, 18 and 27 key point detection models. In addition, 21 key point model for hands and 70 key points for the face.
3. Single person tracking for smoothing and increase rate.
4. Calibration toolbox for camera parameters (intrinsic, extrinsic and distortion)

2D pose estimation creates the 2D spatial location of the human body and is mostly used on images and videos, whereas 3D pose estimation identifies locations in 3D space and can be used in animation industries or virtual and augmented reality [49]. OpenPose uses the bottom-up approache where it detects the key points of every person in the image. The CNN pipeline creates confidence maps and part affinity fields that will identify the key points of the body [47].

At the initial stage OpenPose uses VGG-19 network, a baseline CNN network, which consists of 19 layers in total (16 convolutional layers, 3 fully connected layers, 1 soft max and 5 max pool layers) [50], [51]. This stage is the feature extraction layer where the network creates feature maps. Next the network contains a multiple stage CNN.

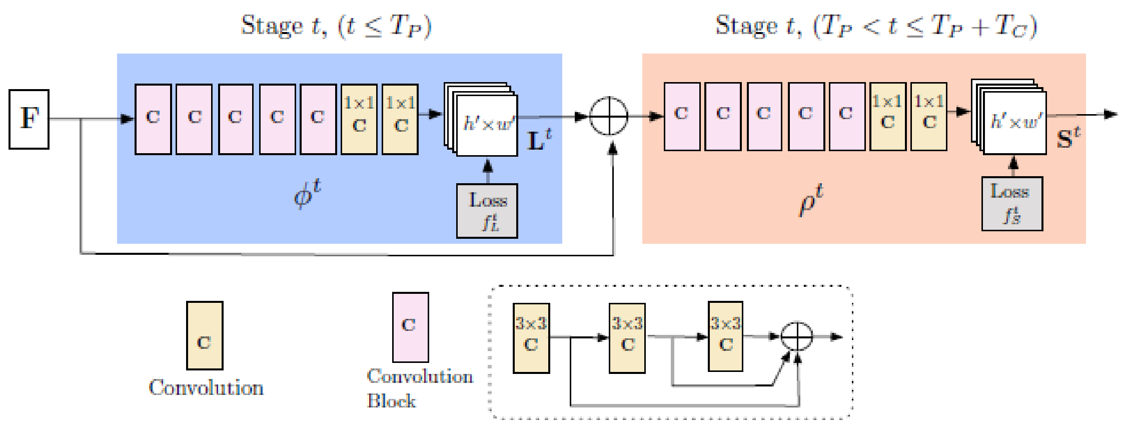


Figure 14 Multi-stage CNN [43, page 3]

The first branch, predicts the degree of association, meaning which two parts combined produce a pair. This is the Part Affinity Fields (PAFs) and this stage produces 38 different outputs. PAFs is a set of flow fields that encodes pairwise connections between body parts. Each body pair has a part affinity field that uses the following set where C is the number of pair body parts and Lc is the 2D unit vector of start and end joint at a particular pixel [52]:

$$\text{the set } L = (L_1, L_2, \ldots, L_C) \text{ where } L_c \in R^{w \times h \times 2}, c \in 1 \ldots C.$$

This stage refines L^t from the feature maps that was output in the previous step [47]:

$$\mathbf{L}^t = \phi^t(\mathbf{F}, \mathbf{L}^{t-1}), \ \forall 2 \leq t \leq T_P,$$

The next branch predicts a set of 18 confidence maps, each for each part of the body. Confidence maps is the probability that a point of a body can be located at a given pixel of the image and follows the set definition below [52]:

$$S = (S_1, S_2 .., S_J) \ where \ S_j \epsilon R^{w*h}, j \epsilon 1 ... J$$

where J is the number of body parts locations.

Confidence maps design heat maps that show where different parts of the body is, whereas PAFs give the direction so that pairs can be created. PAFs from the previous layers enhance the prediction of the confidence maps [47]:

$$\begin{aligned} \mathbf{S}^{T_P} &= \rho^t(\mathbf{F}, \mathbf{L}^{T_P}), \ \forall t = T_P, \\ \mathbf{S}^t &= \rho^t(\mathbf{F}, \mathbf{L}^{T_P}, \mathbf{S}^{t-1}), \ \forall T_P < t \leq T_P + T_C, \end{aligned}$$

These are trained through the mean square error method. After this multiple stage network, there will be different candidates for each part of the body. As an example, the point for the neck will consist of different positional points. Therefore, to find the most accurate position of a particular part, a refinement is done through non-max suppression (NMS) to find the peak value for that body part (maxima). The next step is finding the pairs to which a bipartite graph is created using the greedy algorithm. This graph uses the PAFs and a line integral formula to design a weighted bipartite graph. This graph contains all possible connections between two different parts of the body. The table sorts them from maximum to minimum. When this table is parsed, the highest weightage value will confirm which pair contains the highest score for the strongest bond. Finally, all the pairs created will connect to produce the full human skeleton structure. If there are multiple people, the merging of pairs occurs with the pairs that contain common vertices. The coordinates of the key points join in kinematic model state where the body is represented by dots and lines. [43] [52] [47] [53] [54]

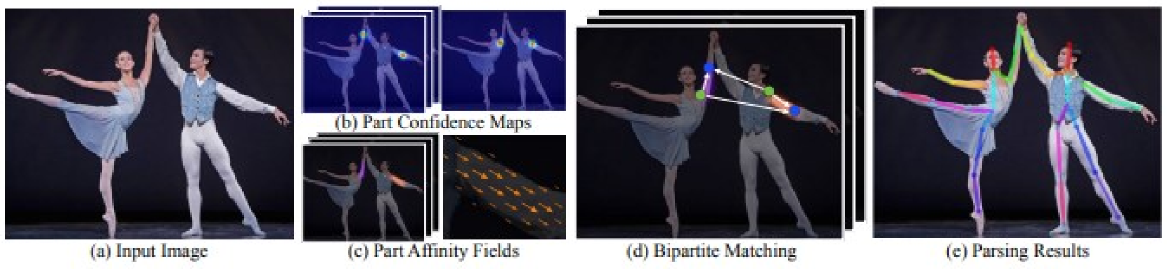


Figure 15 Complete Pipeline [43 page 2]

There are other methods of human pose estimation. These are alpha pose and mask R-CNN. Both these models are top-down methods. However, as seen in the graph below, the OpenPose model has no runtime effect (straight constant line) even if there is more than one person in the image to detect, whereas the alpha pose and mask R-CNN runtimes increase with the number of people.

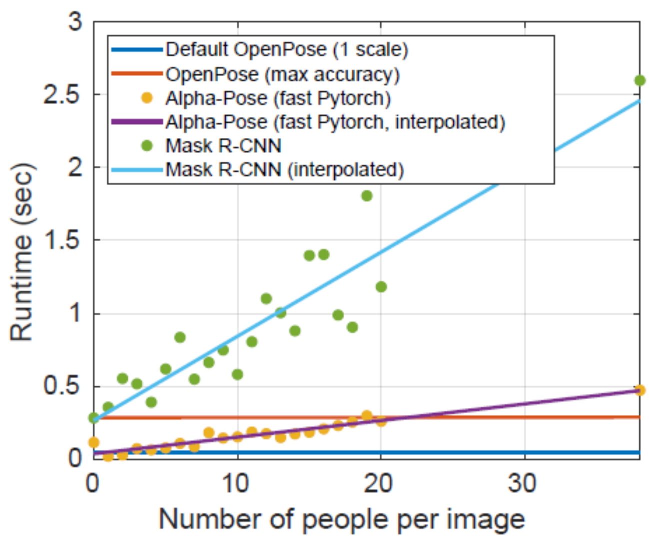


Figure 16 Runtime Proportionality [47]

For the rehabilitation system, OpenPose 2D real-time key point estimation was implemented. The implementation was done in Pycharm using python language. As OpenPose is a CNN network, it requires a certain amount of computational power (as both CPU and GPU are used). It supports images, videos and different cameras that will live stream visuals. In addition, it also supports different Nvidia versions and is compatible with Windows, Mac and Ubuntu. [48]

To increase the speed of processing, the model is created outside of Choregraphe, meaning the model is run outside of the Nao robot. Hence, the initial step is to evaluate the connection capability to perform the analysis of the patient. The input required by the OpenPose estimation is the visual image seen by the Nao robot. A connection must be established on the camera of the robot to absorb the input. Next, an analysis must be performed on the image that provides positive, negative or neutral feedback. The feedback will finally be sent to the Nao in the form of conversational language. The flowchart below describes the system connections:

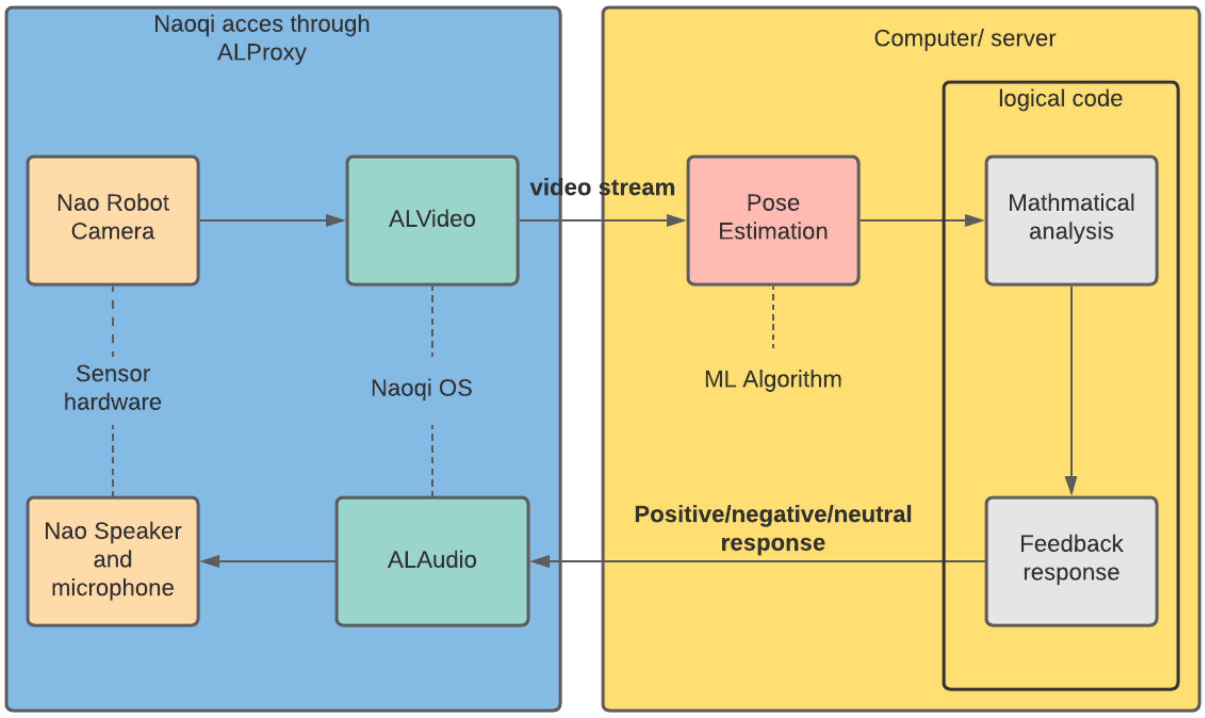


Figure 17 Prototype system connections

The following approaches must be considered to setup the system above:

- The key point model system of OpenPose that will be most efficient for the system, as there are 15, 18 and 27 key point models.
- The logical calculations of posture.
- The connection through IP address to the proxy service.

These together will create the concept of the system and can be verified using the Nao robot. The code uses the example of patient X performing exercise T in order to validate the concept.

4.2.2 Connection establishment

The following code is written to establish the connection between the computer and the OS of the Nao robot using the IP address. This will allow the access to the hardware integrated in the robot.

ALproxy gives access to all the modules that can be used in the Nao robot [55]. Therefore, to establish a connection between the modules, ALproxy is imported from the Naoqi OS.

```
from naoqi import ALProxy
```

ALProxy connects to the audio and video module of the Nao, which can be seen in the code below. This requires the name of the module, the IP address and port.

```
tts = ALProxy("ALTextToSpeech", "192.168.0.158", 9559)

tts.say("Hello everybody today we will be doing the T stretch exercise, so
now i will analyse your position, please lift your arms to the shoulder
level until your body forms a T position")

# get NAOqi module proxy
videoDevice = ALProxy('ALVideoDevice', "192.168.0.158", 9559)
```

The connection established can be tested via the following code:

```
from naoqi import ALProxy
tts = ALProxy("ALTextToSpeech", "<IP of your robot>", 9559)
tts.say("Hello, world!")
```

Figure 18 Test program [56]

Once the connection is completed, the sensors can be calibrated on the requirements. The voice, volume and supported languages along with other methods in Text-To-Speech module can be used to regulate the audio of the Nao robot. The most important input is the video from the camera and a well-calibrated image must be sent to the computer in order to create the image in a window frame. The method ALVideoDevice gives the means to select the camera, frames per second, resolution and the buffer required.

```
# subscribe top camera
AL_kTopCamera = 0
AL_kQVGA = 1      # 320x240 http://doc.aldebaran.com/2-
1/family/robots/video_robot.html#cameraresolution-mt9m114
AL_kBGRColorSpace = 13    # Buffer contains triplet on the format 0xRRGGBB,
equivalent to three unsigned char
# ALVideoDeviceProxy::subscribeCameras(Name,  CameraIndexes,  Resolutions,
ColorSpaces,  Fps)
captureDevice = videoDevice.subscribeCamera("test", AL_kTopCamera,
AL_kQVGA, AL_kBGRColorSpace, 30) # 30
```

Once the sensor input is calibrated, the image is created in a new window frame.

```
# create image
width = 320
height = 240
image = np.zeros((height, width, 3), np.uint8)
```

4.2.3 Open pose model selection and implementation

OpenPose allows the analysis of a person and provides a kinematic model output where coordinates of joints and body structure can be recognized. This will pave the way for understanding what type of posture the patient is performing. Further breakdown of the posture must be able to evaluate if the patient is in the correct position. The existing models were implemented and connected to input frame of the Nao video camera.

OpenPose model contains different experimental models, and they can be used depending on the requirement of the problem. The two main models are body_25 and the coco model. There are also a few variations of body_25 model. However, the main difference between the two models are the output format of body parts. The coco model was the original model that identifies 18 points, whereas body_25 identifies 25 points in the image. The figure below shows the different points identified by the models.

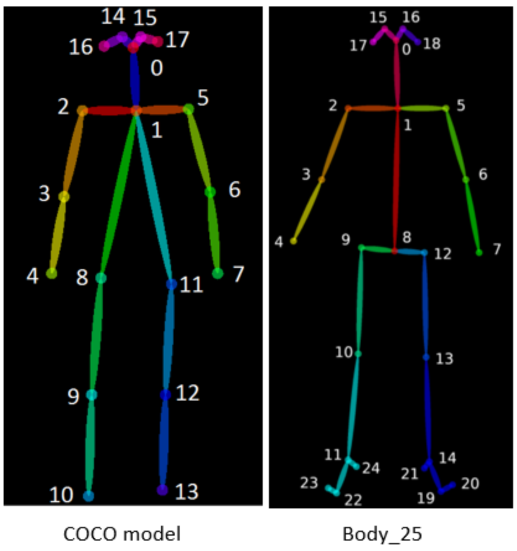


Figure 19 OpenPose models [78]

The coco model was used for this implementation. Firstly, the weights file (caffe model) and prototxt format model were imported and read into the memory. The points are defined by values and pairs were established as follows:

```
# Specify the paths for the 2 files for coco
protoFile = "pose_deploy_linevec.prototxt"
weightsFile = "pose_iter_440000.caffemodel"

# Specify the paths for the 2 files for body_25
#protoFile = "pose_deploy.prototxt"
#weightsFile = "pose_iter_584000.caffemodel"

# net = cv2.dnn.readNetFromTensorflow("graph_opt.pb")

# Read the network into Memory
net = cv2.dnn.readNetFromCaffe(protoFile, weightsFile)

thr = 0.2 #threshold
BODY_PARTS = {"Nose": 0, "Neck": 1, "RShoulder": 2, "RElbow": 3, "RWrist": 4,
               "LShoulder": 5, "LElbow": 6, "LWrist": 7, "RHip": 8, "RKnee": 9,
               "RAnkle": 10, "LHip": 11, "LKnee": 12, "LAnkle": 13, "REye": 14,
               "LEye": 15, "REar": 16, "LEar": 17, "Background": 18}

POSE_PAIRS = [["Neck", "RShoulder"], ["Neck", "LShoulder"], ["RShoulder", "RElbow"],
               ["RElbow", "RWrist"], ["LShoulder", "LElbow"], ["LElbow", "LWrist"],
               ["Neck", "RHip"], ["RHip", "RKnee"], ["RKnee", "RAnkle"], ["Neck", "LHip"],
               ["LHip", "LKnee"], ["LKnee", "LAnkle"], ["Neck", "Nose"], ["Nose", "REye"],
               ["REye", "REar"], ["Nose", "LEye"], ["LEye", "LEar"]]
```

The code above defines the points for the coco model. However, for the body_25 model there will be additional points to both body parts and pose pair lists. The body parts are the points that will be created on the frame to complete the kinematic model, while the pose pairs are the joining parameters of the points that will finalize the skeletal structure. The pose estimation class takes in each frame of the streamed video and designs a heat map to find the local maximums in order to simplify the samples. If the confidence is higher than the threshold specified, then a point is created. These points are in the form of (X, Y) coordinates and, therefore, the function can return the image and the coordinates. Once the coordinates are found, the model uses openCV to generate lines between the points. The coordinate values are output in the form of (x, y, confidence_scores). These values are created in a multidimensional list and can be accessed using the points list created in the code. The position of each list within the multidimensional list is equal to the value defined for each body part that was initially established. This means that points [0] will output the coordinates and confidence score of the nose. The pose_estimation class returns both frame and the coordinates which will be used for the logical calculations.

```
def pose_estimation(frame):
    frameWidth = frame.shape[1]
    frameHeight = frame.shape[0]
    net.setInput(cv2.dnn.blobFromImage(frame, 1.0 / 255, (width, height), (0, 0, 
0), swapRB=False, crop=False))
    out = net.forward()
    out = out[:, :19, :, :]
    assert (len(BODY_PARTS) == out.shape[1])

    points = []
    for i in range(len(BODY_PARTS)):
        # Slice heatmap of corresponding body's part.
        heatMap = out[0, i, :, :]

        # Originally, we try to find all the local maximums. To simplify a 
sample
        # we just find a global one. However only a single pose at the same time
        # could be detected this way.
        _, conf, _, point = cv2.minMaxLoc(heatMap)
        x = (frameWidth * point[0]) / out.shape[3]
        y = (frameHeight * point[1]) / out.shape[2]
        # Add a point if it's confidence is higher than threshold.
        points.append((int(x), int(y)) if conf > thr else None)
        # coord(points)
        # if we print coordinates here it will not work because append occurs 
meaning data keeps adding so there will be an error

    #a = points
    #x1 = points[2][1]
    #print(x1)
    for pair in POSE_PAIRS:
        partFrom = pair[0]
        partTo = pair[1]
        assert (partFrom in BODY_PARTS)
        assert (partTo in BODY_PARTS)

        idFrom = BODY_PARTS[partFrom]
        idTo = BODY_PARTS[partTo]

        if points[idFrom] and points[idTo]:
            cv2.line(frame, points[idFrom], points[idTo], (0, 255, 0), 3)
            cv2.ellipse(frame, points[idFrom], (3, 3), 0, 0, 360, (0, 0, 255), 
cv2.FILLED)
            cv2.ellipse(frame, points[idTo], (3, 3), 0, 0, 360, (0, 0, 255), 
cv2.FILLED)

    t, _ = net.getPerfProfile()
    freq = cv2.getTickFrequency() / 1000
    cv2.putText(frame, '%.2fms' % (t / freq), (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 
0.5, (0, 0, 0))

    return frame, points
```

One important calibration to the frame is that the entered width and height of the frame must be scaled to the width and height parameters of the OpenPose model.

4.2.4 Logical analysis

Since the coordinates of the key points are one of the output variables, the system can use distances and angles to formulate the position structures through fixed and moving key points. To test the program, the example of the T exercise was used. The following diagram is the posture of the T position. One important fixed position, which is the nose, can be observed. Each diagram shows one positive and negative image of the T exercise posture.

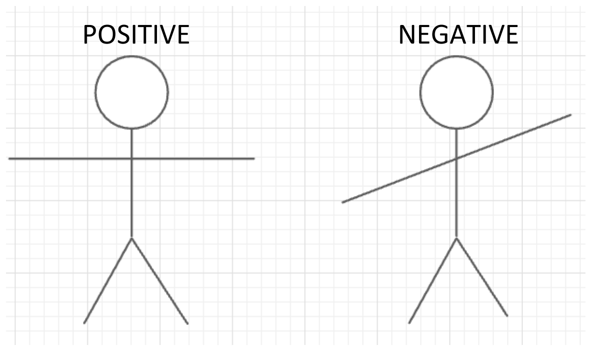


Figure 20 Positive and Negative Posture cases

During the T exercise, the arms and the shoulders are moved into a horizontal straight line and all joints must be approximately aligned into a straight line. Two functions to calculate the angles and distances between two coordinates can be used to mathematically identify the posture positions of the patient, as the specific posture such as the T exercises contains a precise angle and distance with relative to a fixed position. The fixed position in this case is the nose. The following equation is used to find the Euclidean distance between two coordinates (x_1, y_1) and (x_2, y_2) [57]:

$$d = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}$$

The following cosine equation is used to find the angle [58]:

$$c^2 = a^2 + b^2 - 2ab\cos(C)$$

In order to make the calculations scalable, the threshold or the median values for each calculation are established within that particular frame itself. *Figure 17* classifies the angles and distances that need to be considered for the T exercise:

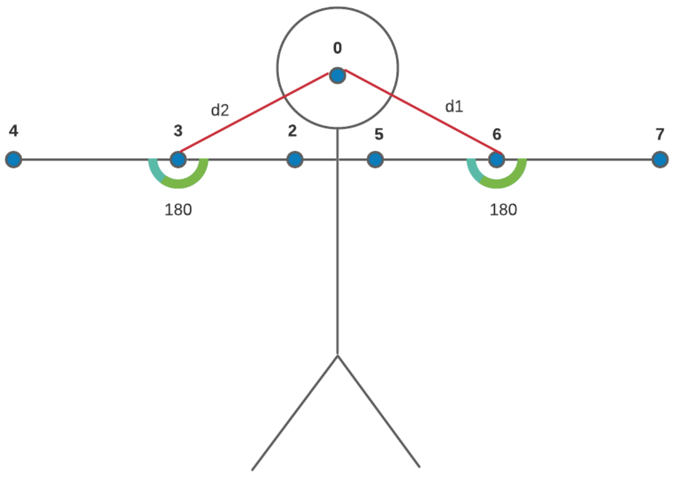


Figure 21 Logical Analysis for T exercise

According to the model specification of body parts and pairs, the above figure shows the numerical value given to each joint. Zero is the nose which is a fixed point on the face. Four and six are the wrists. Three and two are the elbows. Two and five are the shoulders. In the formation of the T exercise, the distances d1 and d2, which can be considered the hypotenuse of a triangle, can be compared to a theoretical value with an error margin. The theoretical value allows scalability of the distances. This means that the height of the patient will not affect the final comparison or evaluation. The theoretical comparison takes the opposite side of the triangle (figure below) which is the length of neck to nose. This value does not change for each frame, only for different individuals. For different individuals, the opposite is taken and used to calculate the theoretical hypotenuse result each time the patient is on the frame. Finally, it is compared with the practical values output through the image.

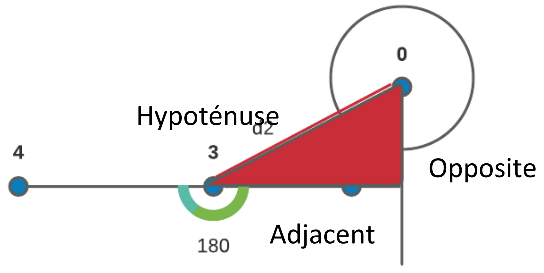


Figure 22 Theoretical Analysis for comparison

The functions below are written to calculate the distances and angles to verify the T exercise. If the lengths d1 and d2 are approximately equal to the theoretical value and if the angle of the elbow is almost 180 degrees, then the posture of the T exercise is passed.

```
def distance(cx1, cy1, cx2, cy2):
    # euclidean distance method to find distance between 2 points
    # since the output values are at a fast rate sometime there is no value hence
    The coordinate 0,0 is given to remove errors
    if cx1 == None: cx1 = (0, 0)
    if cy1 == None: cx1 = (0, 0)
    if cx2 == None: cx1 = (0, 0)
    if cy2 == None: cx1 = (0, 0)
    return math.sqrt(abs(cx2 - cx1) ** 2 + abs(cy2 - cy1) ** 2)

def angle(p0x, p0y, p1x, p1y, p2x, p2y):
    if p0x == None: cx1 = (0, 0)
    if p0y == None: cx1 = (0, 0)
    if p1x == None: cx1 = (0, 0)
    if p1y == None: cx1 = (0, 0)
    if p2x == None: cx1 = (0, 0)
    if p2y == None: cx1 = (0, 0)
    try:
        a = (p1x-p0x)**2 + (p1y-p0y)**2
        b = (p1x-p2x)**2 + (p1y-p2y)**2
        c = (p2x-p0x)**2 + (p2y-p0y)**2
        angle = math.acos((a+b-c) / math.sqrt(4*a*b) ) * 180/math.pi
    except:
        return 0
     return int(angle)
```

These functions are then used with theoretical values to evaluate the posture of the patient. They are called into the main function where the streaming coordinates are added and the calculations are completed.

```
threlbow = 80 #threshold value
threlbownose = 86 #threshold value
# opposite = distance(x1, y1, x2, y2)
adjacent = distance(x2, y2, x3, y3)
hypotenuse = distance(x3, y3, x1, y1)
#angles
anglehand = angle(s1, s2, x3, y3, w1, w2)
theoryresult = int((threlbownose * adjacent) / threlbow)
print(hypotenuse, theoryresult, anglehand)
#if hypotenuse in range(theoryresult -10, theoryresult+10) and anglehand in
range(178, 182):
   #  tts.say("wow, you are doing a good job")
if hypotenuse < theoryresult -10:
    tts.say("could you lower your hand a little")
elif hypotenuse > theoryresult +10:
    tts.say("could you lift your hand a little")
elif anglehand < 165:
    tts.say("make sure not to bend your hand")
else:
    print("good")
    tts.say("wow, you are doing a good job")
```

4.2.5 Sensory input and output

Video

After the connection to the Nao robot's video camera is made, a new frame window with the OpenPose analysis must be designed. The image of the stream is requested through getImage Remote, a method that exist in the ALVideo API. The image is captured and translated into matrix format which can then use openCV libraries to create the frame [59]. Before the output is given, the OpenPose model is run through the image to create a skeletal structure on top of the image with dots and lines. Finally, the OpenPose estimation can be seen in the window.

```
# get image
result = videoDevice.getImageRemote(captureDevice)

if result == None:
    print 'cannot capture.'
elif result[6] == None:
    print 'no image data string.'
else:

    # translate value to mat
    values = map(ord, list(result[6]))
    i = 0
    for y in range(0, height):
        for x in range(0, width):
            image.itemset((y, x, 0), values[i + 0])
            image.itemset((y, x, 1), values[i + 1])
            image.itemset((y, x, 2), values[i + 2])
            i += 3

    # show image
    #check, frame = image.read()
    estimated_image, coordinates = pose_estimation(image)
    cv2.imshow("nao-top-camera-320x240", estimated_image)
```

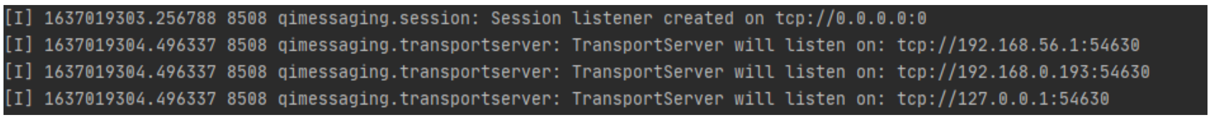
Audio

An if-else statement is maintained after the logical analysis of the posture. Since the robot's microphone is connected, the message required to be said is sent to the robot. The following statements are four test cases given to ensure that patient can succeed in the proper posture.

```
if hypotenuse < theoryresult -10:
    tts.say("could you lower your hand a little")
elif hypotenuse > theoryresult +10:
    tts.say("could you lift your hand a little")
elif anglehand < 165:
    tts.say("make sure not to bend your hand")
else:
    print("good")
    tts.say("wow, you are doing a good job")
```

6.0 Results

The rehabilitation system prototype gives the idea of a fully functioning therapist and in order to validate this concept, one of the first steps is a successful analysis of the posture. The implementation in the previous chapter enabled successful establishment of the connection between the robot and the machine learning model that is processed in the computer.



```
[I] 1637019303.256788 8508 qimessaging.session: Session listener created on tcp://0.0.0.0:0
[I] 1637019304.496337 8508 qimessaging.transportserver: TransportServer will listen on: tcp://192.168.56.1:54630
[I] 1637019304.496337 8508 qimessaging.transportserver: TransportServer will listen on: tcp://192.168.0.193:54630
[I] 1637019304.496337 8508 qimessaging.transportserver: TransportServer will listen on: tcp://127.0.0.1:54630
```

Figure 23 connection established in pycharm

It also showed the application of Google Speech API to the Nao robot behavior. The purpose of Google Speech connection was to create a practical environment, allowing the system to develop in the future steps. The behavior was successfully tested and had the ability to listen, record the audio for 10 seconds at a time before passing it to Google Speech API through the written python script explained in the previous chapter. Using the built-in library functions, the following blocs were used:

- Set language: The language is set to English as the entire test conversation will be conversed with the English language.
- Say: In this bloc, the text that needs to be said initially can be written as, "I will repeat after you, I'm listening". This means that this bloc library parameter contains the speed, voice shaping and the text that needs to be said. The input for this bloc from the previous bloc is a bang which is an event that does not represent any data.
- Record sound: The sound will be recorded and saved in the Naoqi OS which will then be used for speech analysis.
- Google script: This is the code that was implemented to parse the recorded sound through Google Speech.
- Say text: Once the audio is analyzed, the transcript that is generated will be the input for say text. The parameters that can be changed in this bloc are speed and volume, as the input is already the string that would be spoken.
- Say stop: this bloc stops the program once the Nao hears the word "stop" or esc is clicked.

Once the behavior is completed, it will run the cycle as shown in the figure below until the user says stop or the program is stopped.

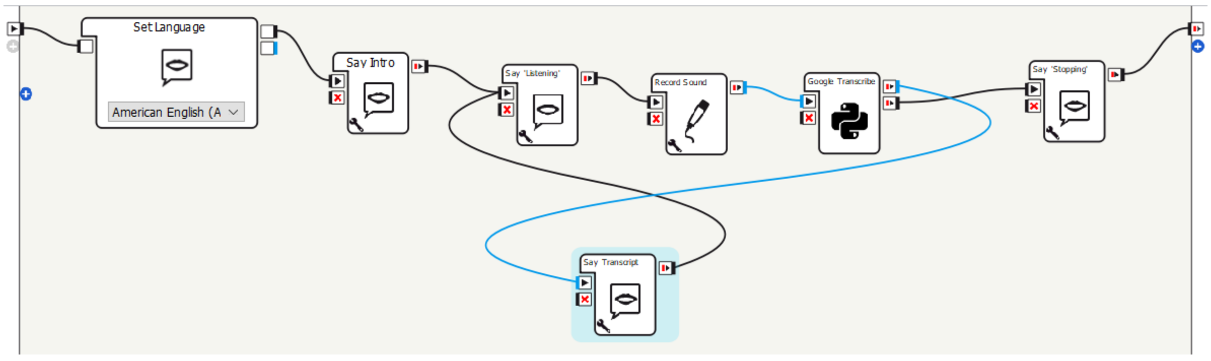


Figure 24 choregraphe behavior for Google Speech API

The next implementation step was to establish the OpenPose model and to evaluate the effectiveness and the possible outcomes of the program on the example of the T exercise. The table below shows the test cases used to verify the actions of the posture, using the logical calculations that were completed. As described in the previous chapter, the angle of the elbow must be straight and the values d1 and d2 need to be similar with the theoretical value.

Figure 25 Test cases for T exercise

Test case	**Logical comparison with error margin**	**Result**	**Outcome**
Angle of elbow	Angle < 180 (-/+) 15	Negative	Hand is bent
Distances d1 and d2	d1/d2 < theory value -10	Negative	Hand needs to be lowered
	d1/d2 > theory value +10	Negative	Hand needs to be lifted
Angle of elbow	165 > angle < 195	Positive	Good Job
Distances d1 and d2	d1/d2 == theory result +/- 10	Positive	Good Job

The calculation in the code defines negative results. If these results are not the case, then the end conclusion is the positive case. Hence the positive comparisons are not written in the code directly. However, as all the negative results are fully described in the code, the implementation showed effective results. The results are also scalable, which means that the height of the person does not affect the calculation. This was also tested on multiple persons. The diagram below displays the video frames with the test cases above that can be seen through the camera of the Nao. The cases are as follows:

- Test case 01: Positive result

  The diagram shows the correct T exercise posture in the video frame to which the Nao corresponds as doing a good job.

- Test case 02: Negative result

  The second case is when the hand is lower than the baseline of the T position (hypotenuse is greater than the theoretical hypotenuse) to which the Nao responds "lift your hand a little"

- Test case 03: Negative result

  The third case is when the elbow is bent in such a way that the angle is less than the marginal value. It should be kept in mind that the angle can be also greater, however, that would need the human elbow to completely bend downwards, which is physically not possible. The Nao recognizes the position and tells the user not to bend the elbow.

- Test case 04: Negative result

  The final test case shows when the hand is lifted up, which means the hypotenuse is less than the theoretical value to which the Nao responds as lowering the hand.

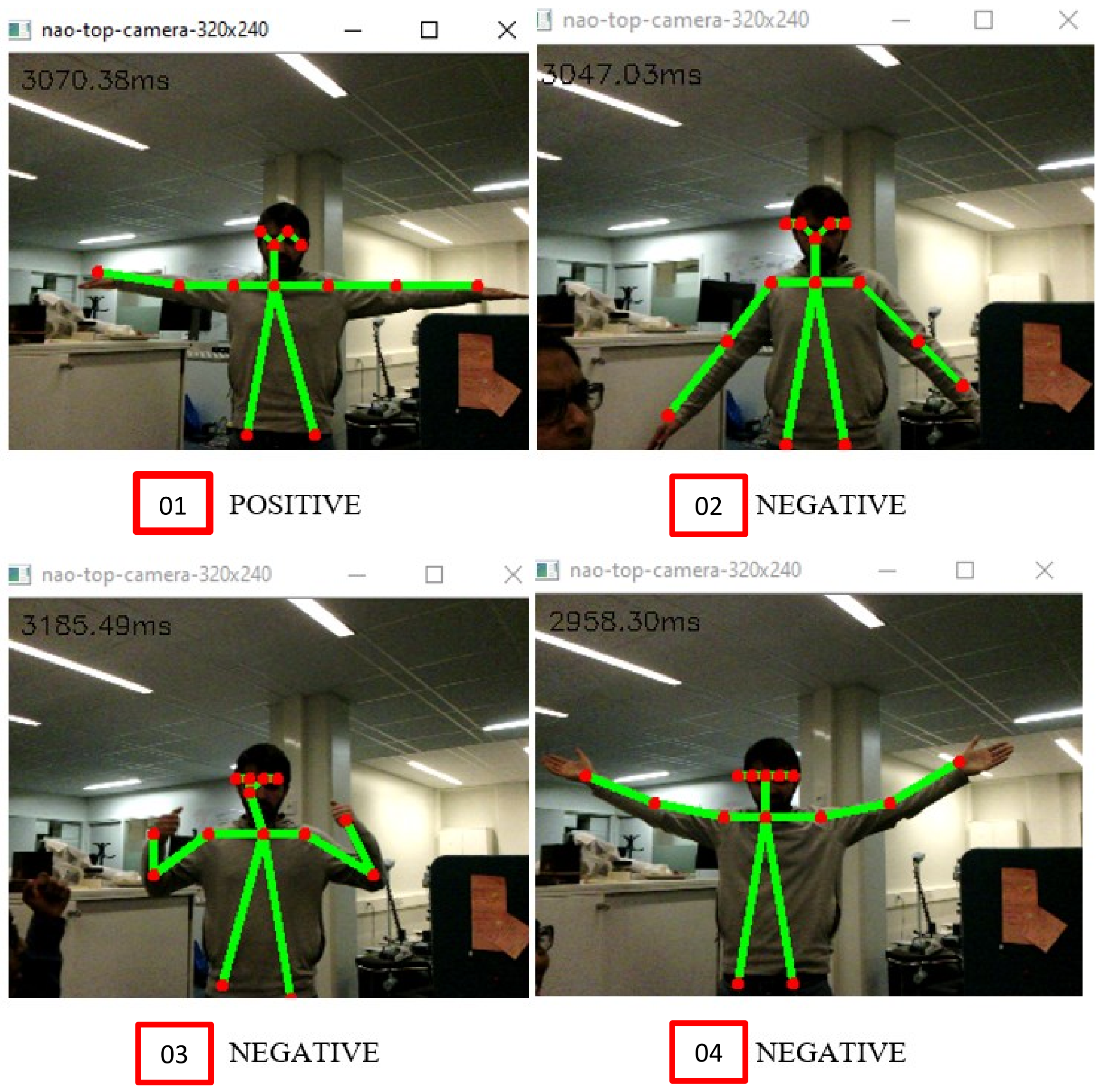


Figure 26 Video frames output

The values below represent three outputs of the logical analysis and what the Nao will say according to the explanations given above for each test case.

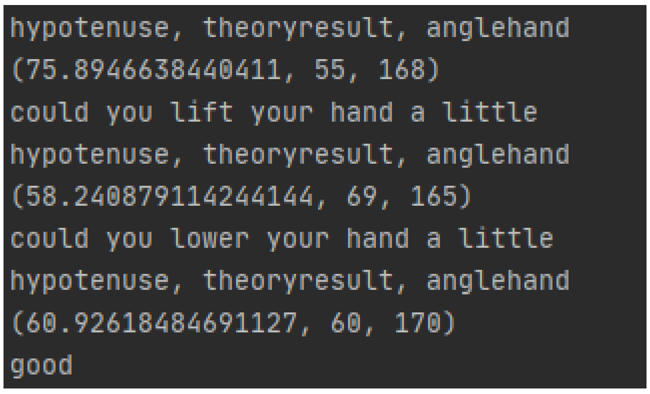

```
hypotenuse, theoryresult, anglehand
(75.8946638440411, 55, 168)
could you lift your hand a little
hypotenuse, theoryresult, anglehand
(58.240879114244144, 69, 165)
could you lower your hand a little
hypotenuse, theoryresult, anglehand
(60.92618484691127, 60, 170)
good
```

Figure 27 mathematical results and outcomes

7.0 Conclusion

Rehabilitation in the medical field is a large domain with the purpose of aiding patients in their recovery. The robotics technology is a key instrument in assisting the recovery process. The implementation of a stay-at-home customized humanoid therapist could be the next step in aiding physical therapy. This project uses already existing systems and tools, such as the Nao robot and machine learning algorithms, to output a demo of what the global objective would look like. The system prototype in this thesis was created to understand the variables that are required, in order to create a humanoid therapist robot. The system is amplified and one specific subject was mainly implemented, image analysis using already existing models customized to this project. The implementation of the program using python version 2.7 was successful and a demonstration was created in order to visualize the future steps that will be introduced to create a well-functioning humanoid therapist.

7.1 Future work.

As the proof of concept was established for visual analysis of an exercise, the system still has many processes that need to be established for a fully functional interface. The following can be categorized as a set of tasks that needs to be completed according to the analysis done in the rehabilitation prototype chapter.

- One focus would be to create a database of all exercises which will include positive and negative test cases as well as a user interface. The physical therapy of the body requires different movements in different body parts. It also depends on the type of rehabilitation required by the patient and the sets of exercises derived by the doctor. Therefore, creating a database with multiple exercises and a user interface through which the doctor can select the set of exercises can be seen as a future addition to the system.
- A Chatbot which will be customized to physical therapy practices could be created in order to have a successful human-like interaction. One surrounding variable which is Google Seech API implementation was done in the thesis. This implementation was to enhance the speech analysis of the robot, but the Chatbot will bring the conversational aspect of the therapist as NLP has become very powerful and the robot requires flexible understanding of human language.

- Report generation: Even though the system is mostly autonomous, the doctor of the patient should be given feedback regarding the process of rehabilitation of the patient. The Nao generating a report that shows the exercises completed and the outcomes on the part of the patient is a method that can be implemented to the system as well. This is to ensure the physical therapy sessions are productive and to evaluate a timeline for the patient recovery process.

8.0 Discussion

This thesis deals with a system that is using robotics. Robotics is implemented in different sectors for different purposes. The very first industry that used robotics was manufacturing where they had motors and actuators to perform a certain function [77]. Now, it is used for research, education, military, security, entertainment as well as healthcare. The advancements of robotics have revolutionized many fields, but the ethics of robotics is a topic that has begun to be studied recently. Robot ethics is understanding the issues and morality regarding robotics technology, as it is increasingly becoming more autonomous. In the healthcare sector, robots are not only used as labor machines but also as companions that understand human behavior to a certain extent. The ethical concerns in the healthcare system can relate to the following topics [76]:

- Human replacement which deals with the question: purpose of robots in healthcare. Is it to replace the work done by human care givers or to assist them in performing certain tasks? Will this be a problem that causes unemployment problems and what are the consequences that follow?
- Quality of care: When robots such as companions are introduced, does it necessarily mean that they provide the same type of care as given by a caregiver to the patients? Emotional care given by machines already come with a difficulty, as some patients refuse to accept or fear the implications.
- Autonomy & safety: Understanding the autonomy of robots in the healthcare system is important for both safety and security. The issue that needs to be addressed is how autonomous the robot must be? Should it be fully capable to take over a certain task or should it always be under supervision?

- Responsibility: If there is a situation that causes a dangerous scenario to the patient, what exactly should be the role in terms of responsibility? Should the human in charge of the robot remain responsible and how to exercise responsibility?
- Deceptive feeling: Humanoid robotics already come with the question of artificial illusion of emotion when dealing with robot companions. Is it morally right or wrong?
- Trust: Is the term trust applicable for robots as they are becoming more autonomous. How should we deal with understanding the responsibility given to a robot, in terms of giving them patients to take care of.
- Privacy and data protection: The data that is collected for healthcare robotics raises a question as to who should have the data, where should it be stored and who should have access to data.

These questions above are already taken into consideration and are studied when implementing robotics into the healthcare system. However, it needs to be noted that some of these questions must be understood before fully implementing a robotic system to the healthcare sector, as it is essential to define these subjects to prevent issues that can arise in the future.

References

1. A Guide to Different Types of Rehabilitation Therapy, Health Across Oklahoma, 23rd May 2018 available at: https://integrisok.com/resources/on-your-health/2018/may/a-guide-to-different-types-of-rehabilitation-therapy

2. Rehabilitation, Medilineplus, 17 September 2018, available at: https://medlineplus.gov/rehabilitation.html

3. "Rehabilitation Technology." Gale Encyclopedia of Nursing and Allied Health. . Retrieved April 17, 2021 from Encyclopedia.com: https://www.encyclopedia.com/medicine/encyclopedias-almanacs-transcripts-and-maps/rehabilitation-technology

4. Robotics, Mossrehab, available at: https://www.mossrehab.com/technology

5. Softbank Robotics, Technical overview, available from: https://developer.softbankrobotics.com/nao6/nao-documentation/nao-developer-guide/technical-overview

6. About Occupational Therapy, WFOT, 2012, available at: https://wfot.org/about/about-occupational-therapy

7. Bass. Haley ,Three Ways Technology Has Improved Physical Therapy Treatment, Concentra, 01/12/2018, available at: https://www.concentra.com/resource-center/articles/three-ways-technology-has-improved-physical-therapy-treatment/

8. www.megadroid.com. (n.d.). *Necoro*. [online] Available at: https://www.megadroid.com/Robots/necoro.htm [Accessed 17 Nov. 2021].

9. Softbank Robotics, Aldebaran Documentation, available at: http://doc.aldebaran.com/2-4/software/choregraphe/choregraphe_overview.html

10. Softbank Robotics, Aldebaran Documentation, available at: http://doc.aldebaran.com/2-4/naoqi/audio/altexttospeech-api.html

11. Softbank Robotics, Aldebaran Documentation, available at: http://doc.aldebaran.com/2-4/naoqi/audio/altexttospeech-api.html?highlight=language%20api#ALTextToSpeech::languageTTS

12. Nao, Robots your guide to the world of robotics, available at: https://robots.ieee.org/robots/nao/

13. Google, Google Cloud, Speech to text, Available at: https://cloud.google.com/speech-to-text#section-6

14. IBM, Computer Vision, available at: https://www.ibm.com/topics/computer-vision

15. Stephenson, B., 2018. *Rehab Select.* [En ligne] Available at: https://blog.rehabselect.net/7-types-of-rehabilitation-therapy

16. Dreeben, O., 2010. Dans: *Patient education in rehabilitation* . s.l.:David Cellar, p. 5.

17. Anon., 2020. *World Health Organization.* [En ligne] Available at: https://www.who.int/news-room/fact-sheets/detail/rehabilitation

18. Chapter 4 Rehabilitation. (n.d.). [online] Available at: https://www.who.int/disabilities/world_report/2011/chapter4.pdf [Accessed 12 Sep. 2019].

19. HISTORY.PHYSIO. (2019). *History of the Wheelchair*. [online] Available at: https://history.physio/history-of-the-wheelchair/ [Accessed 17 Nov. 2021].

20. Krebs, H.I. and Volpe, B.T. (2013). Rehabilitation Robotics. *Neurological Rehabilitation*, [online] pp.283–294. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4688009 [Accessed 29 Oct. 2019].

21. Guittet, J., Kwee, H., Quetin, N. and Yclon, J. (1979). *BPR 10-32, Fall*. [online] *Bull . Prosth . Res*, pp.69–105. Available at: https://www.rehab.research.va.gov/jour/79/16/2/guittet.pdf [Accessed 17 Nov. 2021].

22. telethesis. (n.d.) *McGraw-Hill Dictionary of Scientific & Technical Terms, 6E*. (2003). Retrieved November 17 2021 from https://encyclopedia2.thefreedictionary.com/telethesis

23. Lum, P.S., Burgar, C.G., Loos, M.V. der, Shor, P.C., Majmundar, M. and Yap, R. (2006). MIME robotic device for upper-limb neurorehabilitation in subacute stroke subjects: A follow-up study. *The Journal of Rehabilitation Research and Development*, 43(5), p.631.

24. Krebs, H., Ferraro, M., Buerger, S.P., Newbery, M.J., Makiyama, A., Sandmann, M., Lynch, D., Volpe, B.T. and Hogan, N. (2004). *Journal of NeuroEngineering and Rehabilitation*, 1(1), p.5.

25. IBM Cloud Education (2020). *What is artificial intelligence (AI)?* [online] IBM. Available at: https://www.ibm.com/cloud/learn/what-is-artificial-intelligence

26. Skansi, S. (n.d.). *Undergraduate Topics in Computer Science Introduction to Deep Learning*, Page 15. [online] Available at: http://103.47.12.35/bitstream/handle/1/1296/PS3%20Introduction%20to%20Deep%20Learning_%20From%20Logical%20Calculus%20to%20Artificial%20Intelligence%20%28%20PDFDrive%20%29.pdf?sequence=1&isAllowed=y [Accessed 17 Nov. 2021].

27. www.researchgate.net. (n.d.). *(PDF) ARMin-Design of a novel arm rehabilitation robot*. [online] Available at:

https://www.researchgate.net/publication/4170812_ARMin-Design_of_a_novel_arm_rehabilitation_robot [Accessed 18 Nov. 2021].

28. Asfour, T., Regenstein, K., Azad, P., Schröder, J., Bierbaum, A., Vahrenkamp, N. and Dillmann, R. (n.d.). *ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control*. [online] Available at: https://h2t.anthropomatik.kit.edu/pdf/Asfour2006b.pdf [Accessed 18 Nov. 2021].

29. King, C.-H., Chen, T., Jain, A. and Kemp, C. (n.d.). *Towards an Assistive Robot that Autonomously Performs Bed Baths for Patient Hygiene*. [online] Available at: https://smartech.gatech.edu/bitstream/handle/1853/37075/iros10_auto_clean.pdf [Accessed 18 Nov. 2021].

30. Joseph, A., Christian, B., Abiodun, A.A. and Oyawale, F. (2018). A review on humanoid robotics in healthcare. *MATEC Web of Conferences*, 153, p.02004.

31. Mukai, T., Hirano, S., Nakashima, H., Kato, Y., Sakaida, Y., Guo, S. and Hosoe, S. (n.d.). *Development of a Nursing-Care Assistant Robot RIBA That Can Lift a Human in Its Arms*. [online] Available at: http://rtc.nagoya.riken.jp/sensor/Wiki/papers/IROS2010_Mukai.pdf [Accessed 18 Nov. 2021].

32. Meet Whiz - Softbank Robotics. (n.d.). *Meet Whiz, the collaborative cobot - Softbank Robotics*. [online] Available at: https://www.meetwhiz.com/ [Accessed 18 Nov. 2021].

33. SoftBank Robotics (2019). *Pepper the humanoid robot | SoftBank Robotics EMEA*. [online] Softbankrobotics.com. Available at: https://www.softbankrobotics.com/emea/en/pepper.

34. Meisenzahl, M. (n.d.). *Softbank's famous robot Pepper is helping helping enforce social distancing and greeting COVID-19 patients around the world*. [online] Business Insider. Available at: https://www.businessinsider.com/softbank-pepper-robot-coronavirus-japan-and-germany-2020-5?r=US&IR=T#hotels-in-japan-are-temporarily-being-used-to-house-coronavirus-patients-with-mild-symptoms-1 [Accessed 18 Nov. 2021].

35. Softbankrobotics.com. (2017). *NAOqi Core | SoftBank Robotics Developer Center*. [online] Available at: https://developer.softbankrobotics.com/nao6/naoqi-developer-guide/naoqi-apis/naoqi-core#naoqi-core [Accessed 18 Nov. 2021].

36. developer.softbankrobotics.com. (n.d.). *NAOqi APIs | SoftBank Robotics Developer Center*. [online] Available at: https://developer.softbankrobotics.com/nao6/naoqi-developer-guide/naoqi-apis [Accessed 18 Nov. 2021].

37. developer.softbankrobotics.com. (n.d.). *Supported languages | SoftBank Robotics Developer Center*. [online] Available at: https://developer.softbankrobotics.com/nao6/nao-documentation/nao-developer-guide/supported-languages [Accessed 18 Nov. 2021].

38. Nuance Communications. (n.d.). *Text-to-Speech (TTS) Engine in 119 Voices | Nuance | Nuance*. [online] Available at: https://www.nuance.com/omni-channel-customer-engagement/voice-and-ivr/text-to-speech.html [Accessed 18 Nov. 2021].

39. Acapela Group. (n.d.). *Acapela Group | Company timeline*. [online] Available at: https://www.acapela-group.com/about-us/company-timeline/ [Accessed 18 Nov. 2021].

40. Google Cloud. (n.d.). *Speech-to-Text basics | Cloud Speech-to-Text Documentation*. [online] Available at: https://cloud.google.com/speech-to-text/docs/basics.

41. Yonego. (2021). *Why milliseconds matter*. [online] Available at: https://www.yonego.com/nl/blogs/why-milliseconds-matter/ [Accessed 18 Nov. 2021].

42. viso.ai. (2021). *Human Pose Estimation with Deep Learning - Ultimate Overview in 2021*. [online] Available at: https://viso.ai/deep-learning/pose-estimation-ultimate-overview/.

43. Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S.-E. and Sheikh, Y.A. (2019). OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp.1–1.

44. AI & Machine Learning Blog. (2019). *A 2019 guide to Human Pose Estimation with Deep Learning*. [online] Available at: https://nanonets.com/blog/human-pose-estimation-2d-guide/.

45. Toshev, A. (n.d.). *DeepPose: Human Pose Estimation via Deep Neural Networks*. [online] Available at: https://arxiv.org/pdf/1312.4659.pdf.

46. viso.ai. (2021). *A Guide to OpenPose in 2021*. [online] Available at: https://viso.ai/deep-learning/openpose/ [Accessed 18 Nov. 2021].

47. GeeksforGeeks. (2020). *OpenPose : Human Pose Estimation Method*. [online] Available at: https://www.geeksforgeeks.org/openpose-human-pose-estimation-method/.

48. GitHub. (2020). *CMU-Perceptual-Computing-Lab/openpose*. [online] Available at: https://github.com/CMU-Perceptual-Computing-Lab/openpose.

49. viso.ai. (2021). *Human Pose Estimation with Deep Learning - Ultimate Overview in 2021*. [online] Available at: https://viso.ai/deep-learning/pose-estimation-ultimate-overview/.

50. TIWARI, R.S. (2021). *Transfer Learning — Part — 4.0!! VGG-16 and VGG-19*. [online] Medium. Available at: https://becominghuman.ai/transfer-learning-part-4-0-vgg-16-and-vgg-19-d7f0045032de [Accessed 18 Nov. 2021].

51. Sec, D.I. (2021). *VGG-19 Convolutional Neural Network*. [online] All about Machine Learning. Available at: https://blog.techcraft.org/vgg-19-convolutional-neural-network/ [Accessed 18 Nov. 2021].

52. Hua, Q. (2019). *[CVPR 2017] OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. [online] Medium. Available at: https://towardsdatascience.com/cvpr-2017-openpose-realtime-multi-person-2d-pose-estimation-using-part-affinity-fields-f2ce18d720e8.

53. Wei Yang (2016). *Human Pose Estimation by Deep Learning*. [online] Available at: https://www.slideshare.net/plutoyang/human-pose-estimation-by-deep-learning [Accessed 18 Nov. 2021].

54. BeyondMinds. (2020). *An Overview of Human Pose Estimation with Deep Learning*. [online] Available at: https://beyondminds.ai/blog/an-overview-of-human-pose-estimation-with-deep-learning/.

55. developer.softbankrobotics.com. (n.d.). *AL::ALProxy Member List | SoftBank Robotics Developer Center*. [online] Available at: https://developer.softbankrobotics.com/alalproxy-member-list [Accessed 18 Nov. 2021].

56. developer.softbankrobotics.com. (n.d.). *Using the API - Making NAO speak | SoftBank Robotics Developer Center*. [online] Available at: https://developer.softbankrobotics.com/nao6/naoqi-developer-guide/other-tutorials/python-sdk-tutorials/using-api-making-nao-speak [Accessed 18 Nov. 2021].

57. Cuemath. (n.d.). *Distance Between Two Points- Definition, Formulas & Examples*. [online] Available at: https://www.cuemath.com/geometry/distance-between-two-points/.

58. Mathsisfun.com. (2017). *The Law of Cosines*. [online] Available at: https://www.mathsisfun.com/algebra/trig-cosine-law.html.

59. docs.opencv.org. (n.d.). *OpenCV: Mat - The Basic Image Container*. [online] Available at: https://docs.opencv.org/3.4.14/d6/d6d/tutorial_mat_the_basic_image_container.html [Accessed 18 Nov. 2021].

60. (Page15) Skansi, S. (n.d.). *Undergraduate Topics in Computer Science Introduction to Deep Learning*. [online] Available at: http://103.47.12.35/bitstream/handle/1/1296/PS3%20Introduction%20to%20Deep%20Learning_%20From%20Logical%20Calculus%20to%20Artificial%20Intelligence%20%28%20PDFDrive%20%29.pdf?sequence=1&isAllowed=y [Accessed 17 Nov. 2021].

61. IBM Cloud Education (2020). *What is artificial intelligence (AI)?* [online] IBM. Available at: https://www.ibm.com/cloud/learn/what-is-artificial-intelligence.

62. A. L. Samuel (1959), Some Studies in Machine Learning using the game of checkers, IBM Journal, available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5392560

63. Michal Konkol, Brainy : A machine Learning Library, Natural Language Processing Group, Department of Compute Science and Enginnering, University of West Bohemia. Available at: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.722.6418&rep=rep1&type=pdf

64. Oladipupo, T. (n.d.). *X Types of Machine Learning Algorithms*. [online] Available at: https://pdfs.semanticscholar.org/c4ae/802491724aee021f31f02327b9671cead3dc.pdf.

65. Jason Brownlee (2016). *Supervised and Unsupervised Machine Learning Algorithms*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/.

66. db0nus869y26v.cloudfront.net, Machine learning and data mining: Reinforcement learning, Available at: https://db0nus869y26v.cloudfront.net/en/Reinforcement_learning.

67. Ben Krose, Patrick van der Smagt (1993), An Introduction to Neural networks, CiteSeer, Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.493

68. Gurney, K. and York, N. (1997). *An introduction to neural networks An introduction to neural networks*. [online] Available at: http://www.macs.hw.ac.uk/~yjc32/project/ref-NN/Gurney_et_al.pdf.

69. hackernoon.com. (n.d.). *Machine Learning 101: How And Where To Start For Absolute Beginners | Hacker Noon*. [online] Available at:

https://hackernoon.com/machine-learning-101-how-and-where-to-start-for-absolute-beginners-si2530ar [Accessed 18 Nov. 2021].

70. edureka (2018). *Natural Language Processing In 10 Minutes | NLP Tutorial For Beginners | NLP Training | Edureka*. *YouTube*. Available at: https://www.youtube.com/watch?v=5ctbvkAMQO4. NLP

71. Liddy, E. (2001). *Natural Language Processing Recommended Citation*. [online] *Center for Natural Language Processing School of Information Studies (iSchool)*, p.2001. Available at: https://surface.syr.edu/cgi/viewcontent.cgi?referer=https://scholar.google.com/&httpsredir=1&article=1019&context=cnlp. NLP

72. www.ibm.com. (n.d.). *What is Computer Vision? | IBM*. [online] Available at: https://www.ibm.com/topics/computer-vision#:~:text=Computer%20vision%20is%20a%20field.

73. Nicu Sebe, Cohen, I., Ashutosh Garg, Huang, T.S. and Springerlink (Online Service (2005). *Machine Learning in Computer Vision*. Dordrecht: Springer Netherlands.

74. Tang, B., Pan, Z., Yin, K. and Khateeb, A. (2019). Recent Advances of Deep Learning in Bioinformatics and Computational Biology. *Frontiers in Genetics*, 10.

75. www.youtube.com. (n.d.). *Convolutional Neural Networks*. [online] Available at: https://www.youtube.com/watch?v=qwgtQiOjy8M&list=PL_sCWom8F1Awbz4NFFHf2TyJbwTBeMMNa&index=12 [Accessed 18 Nov. 2021].

76. Stahl, B.C. and Coeckelbergh, M. (2016). Ethics of healthcare robotics: Towards responsible research and innovation. *Robotics and Autonomous Systems*, [online] 86,

pp.152–161. Available at: https://www.sciencedirect.com/science/article/pii/S0921889016305292.

77. Lin, P., Abney, K. and Bekey, G.A. (2011). *Robot Ethics: The Ethical and Social Implications of Robotics*. [online] *Google Books*. MIT Press. Available at: https://books.google.fi/books?hl=en&lr=&id=oBb-lt3l4oYC&oi=fnd&pg=PA17&dq=ethics+behind+robotics&ots=ywdRu2B-_q&sig=otv8F9bYeSDYQjXy9eGQwRPEE0Y&redir_esc=y#v=onepage&q=ethics%20behind%20robotics&f=false [Accessed 18 Nov. 2021].

78. cmu-perceptual-computing-lab.github.io. (n.d.). *OpenPose: OpenPose Doc - Output*. [online] Available at: https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/md_doc_02_output.html#pose-output-format-coco [Accessed 18 Nov. 2021].