

Santa Clara University

Scholar Commons

Electrical and Computer Engineering Senior
Theses

Engineering Senior Theses

Spring 2023

Project Hermes: The Socially Assistive Tour-Guiding Robot

Aly Khater

Justin Sun

Fernando Camou

Follow this and additional works at: https://scholarcommons.scu.edu/elec_senior



Part of the [Electrical and Computer Engineering Commons](#)

SANTA CLARA UNIVERSITY

Department of Electrical and Computer Engineering

I HEREBY RECOMMEND THAT THE THESIS
PREPARED UNDER MY SUPERVISION BY

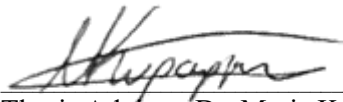
Aly Khater, Justin Sun & Fernando Camou

ENTITLED

Project Hermes: The Socially Assistive Tour-Guiding Robot

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING



Thesis Advisor: Dr. Maria Kyrarini

June 14, 2023

Date



ECE Department Chair

06/16/23

Date






4 - Project Hermes_AK_JS_FC_Senior_Design_Final_Report

Final Audit Report

2023-06-16

Created:	2023-06-16
By:	Darcy Yaley (dyaley@scu.edu)
Status:	Signed
Transaction ID:	CBJCHBCAABAA0d-K40DvE_vZCcLvrDon9_bX-TpeEaye

"4 - Project Hermes_AK_JS_FC_Senior_Design_Final_Report" History

-  Document created by Darcy Yaley (dyaley@scu.edu)
2023-06-16 - 9:30:51 PM GMT
-  Document emailed to Shoba Krishnan (skrishnan@scu.edu) for signature
2023-06-16 - 9:31:34 PM GMT
-  Email viewed by Shoba Krishnan (skrishnan@scu.edu)
2023-06-16 - 9:57:05 PM GMT
-  Document e-signed by Shoba Krishnan (skrishnan@scu.edu)
Signature Date: 2023-06-16 - 9:57:27 PM GMT - Time Source: server
-  Agreement completed.
2023-06-16 - 9:57:27 PM GMT

Project Hermes: The Socially Assistive Tour-Guiding Robot

By
Aly Khater, Justin Sun & Fernando Camou

SENIOR DESIGN PROJECT REPORT

Submitted to
the Department of Electrical and Computer Engineering
of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements
for the degree of
Bachelor of Science in Electrical and Computer Engineering

Santa Clara, California

Spring 2023

Abstract

With the reduced amount of availability of a labor force for non-technical tasks, service robotics has grown to be used in place of human labor to handle these tasks. There have been various studies on the impact of using robotics in a sociological context. The use of service robots in a social and labor environment recognizes the need of cohesive Human-Robot Interaction (HRI). In this senior design project, we delve into the thought process of using a service robot in place of a human for tasks that are normally reserved for humans. These tasks outline design considerations when performing emotional-centric activities and the need to deliver an effective and efficient service. Codenamed as Project Hermes, we developed a guided tour robot that will provide an interactive routine. Using the robot's array of sensors and motors, the routine consists of navigating from one room to another, providing an audible explanation of each room, answering visitor questions, and moving on. With the robot's embedded microphones, the robot is capable of limited interactions with humans, providing feedback and performing tasks accordingly. Once the core functionalities are developed, Hermes will be evaluated in a real-world environment to garner data and feedback. With all these considerations in hand, the design of the service robot needs to cover many of these areas for our framework. To address this need, we outline the ideas and considerations for the task.

Acknowledgements

We would like to thank both our faculty advisors Dr. Maria Kyrarini and Dr. Andrew Wolfe, for their immense support and encouragement throughout the duration of our project. Additionally, we would like to thank Santa Clara University and the School of Engineering for providing all the resources and funding necessary for the future successful completion of this project.

Contents

1	Chapter 1: Introduction	1
1.1	Background and Motivation	1
1.2	Problem Statement	2
1.3	Research Objectives	3
2	Chapter 2: Hardware and Software	4
2.1	Hardware	4
2.2	Software	6
3	Chapter 3: Project Design and Methodology	8
3.1	Overview	8
3.2	Robot Motion	10
3.2.1	Directed Motion	11
3.2.2	Expressive Gestures	13
3.2.3	Implementation and Challenges	14
3.3	Control and Navigation Algorithms	15
3.3.1	Complexity Analysis	17
3.4	Human-Robot Interaction	19
3.4.1	Text-to-Speech	19

3.4.2	Speech-to-Text	20
3.4.3	Implementation and Challenges	21
3.5	Integration	23
3.5.1	Motion and Speech Integration	23
3.5.2	Motion-Speech Implementation and Challenges	23
3.5.3	Network Integration	25
4	Chapter 4: Experimental Results	27
4.1	Test Environment	27
4.2	Robot Motion Accuracy	28
4.3	Navigation Accuracy	30
4.4	Speech Recognition Quality	35
4.4.1	VOSK API	35
5	Chapter 5: Hermes Evaluation	36
5.1	Experimental Protocol	36
5.2	Evaluation Metrics	38
5.3	User Feedback	39
6	Chapter 6: Standards and Constraints	40
6.1	Standards	41

6.2	Technical Challenges and Constraints	42
6.3	Privacy	43
6.4	Cost	44
6.5	Economic Considerations	45
6.6	Safety	45
6.7	Sustainability, Environmental Impact, and Social Impact	46
7	Chapter 7: Conclusion and Future Work	47
7.1	Lessons Learned	47
7.2	Future Work	48
7.3	Final Remarks	48

List of Figures

1	NAO6 humanoid robot (left) with camera locations (right)	5
2	Overview of the Hermes framework	8
3	NAO6 Robot with X and Y axes. Z axis points outwards.	12
4	Axis particular to the robot head. [1]	14
5	Robot strayed from the X axis	15
6	Curved QR code	24
7	Small-scale QR code	25
8	Jetson Nano	26
9	TP-Link Archer GX90	27
10	Walk Test setup.	29
11	Demonstration Setup with Initial Angles	31
12	Demonstration Setup of the Nao robot with a straight path and 3 QR Codes	32
13	Demonstration Setup of the Nao robot with one-corner path and 5 QR Codes	33
14	Demonstration Setup of the Nao robot with Two-Corner path	34
15	Output using VOSK	36
16	Location of Demonstration. SCDI 4th floor hallway.	37
17	Demonstration Setup of the Nao robot and 90 degree angle in the path	38

List of Tables

1	NAO6 Robot Specifications	5
2	Examples of the functionality of Hermes	11
3	Major Functions and Their Complexities	17
4	Walk Test Results	29
5	Straight-Line Test with 3 QR Codes	32
6	One-Corner Test with 5 QR Codes	32
7	Two-Corner Test with 5 QR Codes	33
8	User Feedback	39

1 Chapter 1: Introduction

1.1 Background and Motivation

Appreciable advances in robotics and Artificial Intelligence have birthed and expanded the subfield of socially assistive and service robots. Because of their extreme precision, consistency, computational power, and lack of need for compensation, socially assistive robots have a great capability to increase the quality of life of humans. The aim is to show that they have their place in the world by programming the NAO robot to give guided tours of the fourth floor of the Sobrato Campus for Discovery and Innovation (SCDI) at Santa Clara University (SCU).

The central motivation of this project is to bring a socially-assistive humanoid robot, called Hermes, to life, which requires less training and is more fiscally efficient. One of the major trends of the future is the coexistence between humans and robots. There are many tasks and environments that are not suitable for humans to conduct but are capable for robots. With the assistance of robots, more possibilities will be created in numerous fields. This will enrich the quality of life by allowing humans to focus on more creative or higher-complexity endeavors, and there are a few studies of this idea put into action.

A study from the School of Computer Science and Psychology at the University of Birmingham, United Kingdom observed the effect of the robot's personality on the visitor's ability to retain the information from a robotic tour [2]. The study first had an emotionless robot give visitors a tour of a museum and then tested the visitors on the information given on the tour. The experiment was repeated using two collaborative robots with cheerful personalities, and the investigation showed much better results on the assessment. The reason for employing two robots instead of one was inspired by another cognitive study mentioned in this study that stated that humans are more likely

to retain information that is given by two human speakers. By making two robots human-like and personable, the benefits discovered in the cognitive study can be enjoyed, and tours of all kinds can be more effective without requiring an additional human tour guide.

In another study from eight years ago, researchers from the University of Tokyo, Japan sought to develop a museum tour-guiding robot's ability to recognize and adjust to spatial orientations [3]. Spatial orientations relate to the collective shapes that people naturally take when they engage in a group conversation or are listening to someone speak. People subconsciously reorient themselves during group conversation in such a way that they can listen to the speaker without obstructing others, and by improving the robot's ability to recognize the appropriate stance, tours, and other human-robot interactions can seem more natural. With our study accompanying previous ones, we could refine the functionality of service robots to have the best effect on the quality of life for all living things.

1.2 Problem Statement

In demonstrating that socially-assistive service robots have their place in the world, we need to showcase an instance of a robot performing a task that is considered by humans to be of relatively low complexity. The task under investigation is tour guiding the fourth floor of SCDI at SCU. The robot guides visitors to the various exhibits, gives a description of each exhibit, listens to questions, and responds accordingly. The tour concludes when all the exhibits have been visited and the robot should indicate when this point is reached.

1.3 Research Objectives

The execution of this project involves the research and development behind the three main objectives of motion, navigation, and Human-Robot Interaction (HRI). Each main objective requires us to learn about the capabilities provided by the robot's hardware as well as methods for controlling and testing its many applications to improve the robot's behavior. A very helpful tool for testing and adjusting behaviors or actions rapidly is the Graphical User Interface (GUI) named Choregraphe. We used it in the early stages to run simple tests that familiarized us with the robot's inherent abilities. These were our areas of research pertinent to the development of the solution to our problem.

- Motion: this is the most fundamental objective to accomplish and research on how to control the robot's limbs will be needed.
- Navigation: we will need to establish QR code recognition to act as the "trigger" for giving a description of each exhibit. Camera vision will also be researched to be able to detect the QR codes and to maintain alignment as the robot walks.
- Human-Robot Interaction: questions and answers are very useful in tours for clarifying or expounding upon information and building rapport with visitors, so the robot should have a reasonably abbreviated version of this ability as well.
- Portability: it will be necessary for the robot to have strong internet connectivity and low latency as it traverses the floor during the tour.

This senior design project aims to investigate the use of a service robot in a live environment and human-robot interaction in a sociological setting. This senior design project report begins in Chapter 1 with an introduction of the problem the senior design project aims to address. Chapter

2 showcases the hardware and software features of the NAO robot. Chapter 3 demonstrates the proposed design of the project. Chapter 4 presents the results of approaches with motion and speech. Chapter 5 discusses evaluation metrics and user feedback. Chapter 6 discusses the inherent limitations and constraints of the robot. Chapter 7 concludes with the anticipated outcomes of the senior design project and a summary of the report.

2 Chapter 2: Hardware and Software

This chapter will discuss the hardware and software tools that were utilized in this project.

2.1 Hardware

In this subsection, the hardware that was used in this project is introduced.

NAO6 Robot: The robotic platform that is selected for the project is NAO6 robot, created by Aldebaran. NAO is a small humanoid robot and has 25 degrees of freedom. It is also equipped with 2 cameras, 4 microphones, and 2 speakers. The specifications of the NAO robot are summarized in Table 1. The NAO robot that is used in this project is called Sofia and it can be seen in Figure 1.

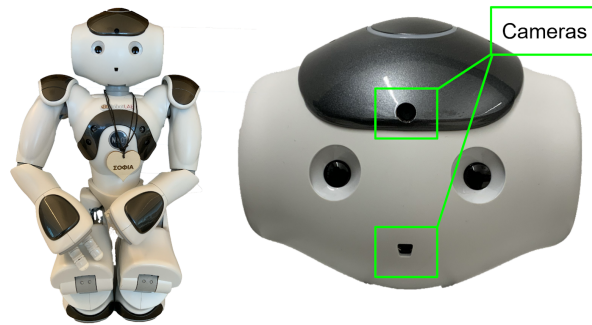


Figure 1: NAO6 humanoid robot (left) with camera locations (right)

Table 1: NAO6 Robot Specifications

Feature	Specification	Location
Height	58 cm	-
Weight	5.6 kg	-
Degrees of Freedom	25	-
Tactile Sensors	7	3 on top of head, 2 on back of each hand, 2 on front of each foot
Ultrasonic Sonars	2 (40kHz, 20-80 cm range)	1 on chest plate
2D Cameras	2 (Resolution: 2560x1920 at 1fps or 640x480 at 30fps)	1 on forehead, 1 at mouth
Omnidirectional Micro-phones	4 (Frequency range: 100Hz - 10kHz)	2 on top of head, 2 at rear of head
Loudspeakers	2	1 on each side of head
Connectivity	Wifi and Bluetooth	-
Battery Runtime	90 minutes on full charge	-

2.2 Software

This section introduces the software that was used in this project.

NAOqi: NAOqi [4] is a core software development kit (SDK) developed by the Aldebaran group for their humanoid robots, including the NAO6 robot. It provides a comprehensive set of features and tools designed to facilitate programming and control over these robots. We leveraged the capabilities of the NAOqi API to perform functions on the NAO6 platform. NAOqi provides developers with several functions, including Joint motor control, Text-to-speech capabilities, and Status updates such as joint position.

Choregraphe: Choregraphe [5] is a desktop application developed by the Aldebaran group, utilizing the NAOqi Framework without the need to write a line of code. The application provides a series of pre-programmed functions to create behaviors on the NAO6 robot platform, test routines on a simulated environment and robot, and to monitor and control the NAO6. Choregraphe is linked to the NAOqi Framework, easing the interaction with NAOqi. Choregraphe serves as an initial stepping stone to ease developers into the use of the NAOqi Framework.

Python: Python [6] is a high-level, interpreted programming language known for its ease of use and readability. Python is considered one of the most popular and widely used programming languages globally. Python version 2.7 was released in 2010. It provides developers with various functionalities for data processing, complex mathematical calculations, and a variety of libraries to use. Python 2.7 end of life date was in 2020, so it serves to support legacy codebases that are written in Python 2.7.

OpenCV: OpenCV [7] is an open-source library focused on real-time image processing and computer vision applications that is a widely-adopted tool used by developers, researchers and companies worldwide. OpenCV version 3.4.4 was released in 2018. The OpenCV library offers a

multitude of functions and algorithms, ranging from image reading and writing to color detection, facial recognition, and object detection. OpenCV is capable of processing images and videos to detect objects, track movements, and more. OpenCV equips researchers and developers with the capabilities to develop complex image processing and computer vision applications.

Gstreamer: GStreamer [8] is an open-source multimedia framework allowing developers flexible multimedia processing that supports a wide variety of media-handling components. Utilizing its modular, pipeline-based architecture, it operates on the principle of streaming data through elements. Gstreamer features network streaming and supports a wide range of protocols for audio and video streaming.

VOSK: Vosk [9] is an offline open-source speech recognition toolkit that offers offline capabilities, making it suitable in cases where internet connectivity is not always available. Vosk provides developers with efficient and accurate speech-to-text functionalities across a myriad of languages. Vosk has the capability of recognizing speech from various audio sources, ranging from files to data streams from microphones in real-time.

Quick Response (QR) Codes: QR codes are two-dimensional barcodes that can store a wide variety of information types. QR Codes are utilized in numerous sectors due to their ability to store a significant amount of information and their easy readability using cameras and barcode readers. Compared to one-dimensional barcodes, which store information horizontally, QR Codes store data both horizontally and vertically, allowing for more room to hold larger amounts of information. QR Codes also have a fast reading response time, hence the name ‘Quick Response.’

3 Chapter 3: Project Design and Methodology

3.1 Overview

For an overview of Project Hermes, we have the slated Project Statement. Can robots effectively provide assistance and guidance in various settings to enhance efficiency and user experience? The objective of Hermes was to develop an interactive robot tour guide. This section will go into detail on completing the said objective. Figure 2 provides an overview of the developed Hermes framework.

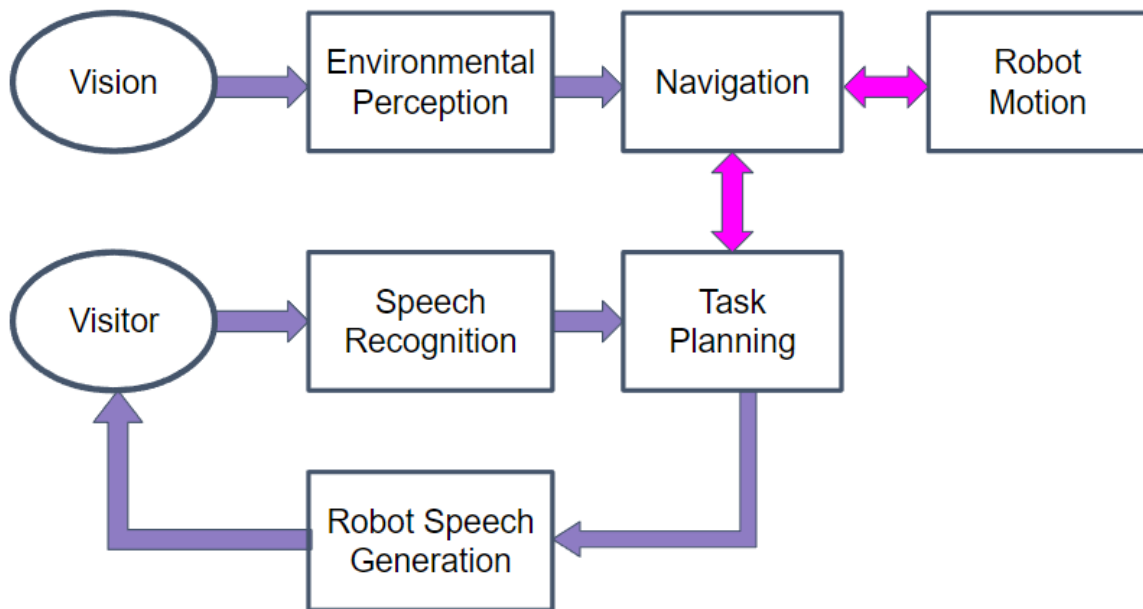


Figure 2: Overview of the Hermes framework

Examples of the functionalities of each module in the framework are provided in Table 2. The framework consists of the following modules:

Vision: The robot has two cameras, one located on the top of its head and one located at its mouth.

The position of the cameras provides a field of view of 67.4 degrees. The cameras support a stream of 2D images of one frame per second(fps). The top camera can stream at a resolution of 2560x1920, or a stream at 30 fps at a resolution of 640x480. The bottom camera can stream at a resolution of 1280x960, or stream at 15 fps at a resolution of 640x480. The cameras are considered as the vision of the NAO6 robot, as the robot receives the stream of images of the environment.

Environmental Perception: The NAO6 robot takes data from its vision to process what environmental landmarks show in its field of view, such as humans, furniture, or QR codes. QR codes, short for quick response code, is a machine-readable code that stores information as a two dimensional barcode. Using the robot's camera, the robot will be able to read the QR code to get information about its environment. Hermes utilizes OpenCV's capabilities to decode the QR codes and assign a response to reading the QR codes.

Navigation: The navigation focuses on how the robot determines its current location and how to get to the next desired position. Hermes utilizes OpenCV to navigate via color detection. By following a red carpet, which will be further explored later in the report, Hermes has the ability to navigate through diverging paths, allowing navigation around corners and objects.

Robot Motion: The robot motion includes the possible movements of the robot, such as walking and standing. Provided by the NAOqi API, the team has control over all basic motor functions to provide movement, such as walking or motions to perform a gesture. Robot Motion provides what code is needed to provide the movements required to perform in Navigation.

Visitor: The visitor is our user. Primary interaction from the visitor to the robot will be via speech. Possibility of having multiple languages provided by the visitor aimed at the robot. Currently, the robot has the capability of interacting with a user through gestures.

Speech Recognition: The robot needs to be able to process the speech from the visitor to perform

its next step. Hermes aimed to use VOSK, an offline speech recognition toolkit that provides a massive catalog of vocabulary. The robot needs to be able to understand the vocabulary being used by the Visitor and recognize what is being said.

Task Planning: The task planning will include a predefined guided tour. The robot has the capability to stop in front of a designated QR code to give a response according to the data stored in the QR Code. Given an actionable command from the QR code, the robot processes the data from the QR code to give an actionable response. The task planning is done through a series of Python code functions to have a deterministic route from its current command to the next. Hermes will perform the actionable command, then move forward to navigate through the path until it reaches its next destination.

Robot Speech Generation: Convey if the robot understood the request from the visitor, or relay information about the room given by the QR code placed next to the room. From Hermes task planning, the robot provides speech based on whether it was a request from the visitor or a command from a QR code. The request from a visitor will either prompt the user to repeat the question should the robot not understand the question, or respond with a comprehensive answer. From the QR code, the QR code will have information attached to it to provide the robot speech about the current room.

3.2 Robot Motion

Motion pertains to the ability of the robot to move from one location to another or to perform physical tasks. This functionality is integral in Hermes' application, contributing to task completion and the ability to interact with a robot. Applications in Hermes are the use of directed movement and expressive gestures, enabling the NAO6 to demonstrate its current state or communicate intuitively.

Table 2: Examples of the functionality of Hermes

Block	Example Functions
Vision	The robot is looking at the floor. A QR code is in front of its cameras
Environmental Perception	The robot processes the image, either looking for a QR code to determine its next action, or utilizing color detection to detect the path.
Navigation	From the color detection, the robot can adjust its movement to follow the path.
Robot Motion	Perform the necessary motions to reach its next destination.
Visitor	Visitor asks the robot a question. “Tell me about Lab 4007”
Speech Recognition	Robot understands the visitor’s question and processes what was asked
Task Planning	Robot plans its next action, in this case, plans to output a speech response to the visitor.
Robot Speech Generation	Conveys to the visitor information about Lab 4007

To perform motions, the team utilized the NAOqi API to facilitate the movement of the NAO6 robot platform. The API hosts various modules specifically designed for robot motion, specifically the NAOqi Motion API.

3.2.1 Directed Motion

In the project, we utilized the NAOqi Motion module to direct our robot’s movement. The Motion module allows control over the robot’s movement, individual joints, locomotion, and body posture. We were able to program the robot to move to specific locations, continuously walk forward, and maintain its position along a path.

The core function we employed for directed movement is ‘void ALMotionProxy::move()’, which is part of the NAOqi API’s ALMotion module. This function provides an interface to control the robot’s locomotion. The function takes three constant float parameters: ‘x’, ‘y’, and ‘theta’ [10].

The 'x' and 'y' parameters represent the robot's desired movement in meters per second along the respective axes of the robot's coordinate system. Positive values of 'x' move the robot forward, while negative values move the robot backwards. Positive values of 'y' move the robot to the left, while negative values move the robot to the right. [10]

The 'theta' parameter represents the robot's movement in radians per second along the Z-axis. Positive 'theta' values move the robot clockwise, while negative values move the robot counter-clockwise. [10]

For our project, we mainly utilized 'ALMotionProxy::move()' to facilitate the directed motion of the robot. For example, we utilized the values of 0.3 for 'x', 0 for 'y', and 0 for 'theta' to provide the robot movement to walk forward. Figure 3 provides a representation of the axes from the robot's perspective.

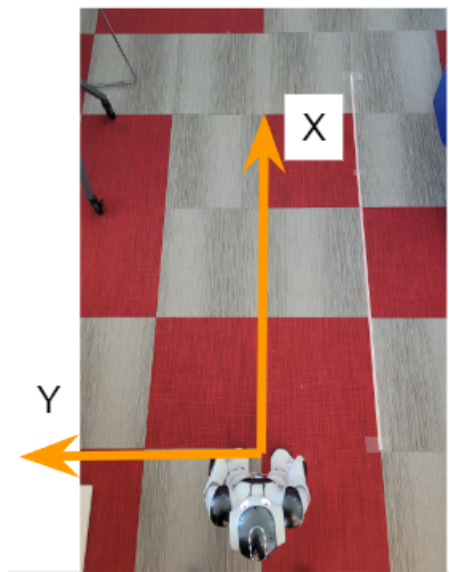


Figure 3: NAO6 Robot with X and Y axes. Z axis points outwards.

3.2.2 Expressive Gestures

We also utilized NAOqi's capabilities to provide our robot with expressive gestures. These gestures were designed to mirror human-like expressions and body language, contributing to a more robust and interactive human-robot interaction. This involved programming specific movement routines by adjusting each individual joint and synchronizing movement sets with other robot functionalities, such as speech.

The core function to serve for our expressive gestures is `'void ALMotionProxy::angleInterpolation()'`, allowing us precise control over individual joints, enabling us to create expressive gestures and more human-like movements. The function accepts four parameters: `'names'`, `'angleLists'`, `'timeLists'`, `'isAbsolute'`. [11]

The `'names'` parameter refers to the joints or multiple joints that are to be moved. It can be the name of a single joint, such as "HeadYaw" or "HeadPitch", or the multiple joints, such as "Head" which encompasses both the pitch and yaw of the head itself [11]. The `'angleLists'` parameter denotes the target angles, in radians, for the joint. Each joint has a limited range of motion. The angles are references to the axis that pertains to the joint itself. See Figure 4 for an example [11]. The `'timeLists'` parameter is the amount of time in seconds at which the joint should reach the target angle. Moreover, the `'isAbsolute'` parameter is a Boolean value that specifies whether the target angles pertain to the relative frame or the current frame. If true, the movement to the angle is seen from the relative frame.

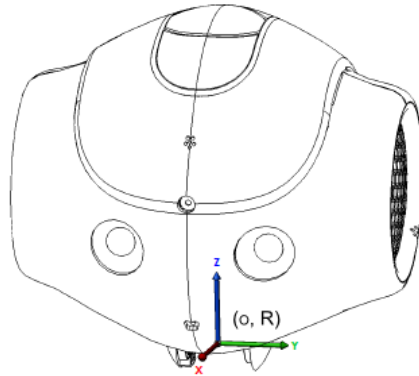


Figure 4: Axis particular to the robot head. [1]

We utilized ‘ALMotionProxy::angleInterpolation()’ to perform a hand-waving gesture. We utilized numerous function calls to facilitate a fluid movement from standing at attention to raising the hand to smoothly wave the hand. We controlled 6 joints and actuators to perform this motion, 5 joints of which are in the left arm, and 1 actuator that is in the left hand. As many of the joints are represented in their own function call or, at maximum, a two-element array, the movements had to be synchronized to allow the robot to perform multiple movements across its joints synchronously.

3.2.3 Implementation and Challenges

In implementing these motion functionalities, we faced several challenges, including robot drifting. As detailed in Section 3.2.2, utilizing the function ‘ALMotionProxy::move()’ with the parameters set to move the robot forward only in the X-axis still had the robot drifting in the Y-axis, even with no prior inputs for the Y direction. See Figure 3 to view the X and Y axes.

We performed a series of experiments to combat the problem of drifting, by performing a walk test across a myriad of surfaces, which is detailed in the Results section of the report. In short, the walk test gave valuable data that the drifting problem is inherent to the robot itself, and not due to the

surface it is walking on. The solution was to utilize computer vision to adjust its trajectory to stay on a straight path by using a colored surface as our means of navigation. The solution is further detailed in the next section for Controls.

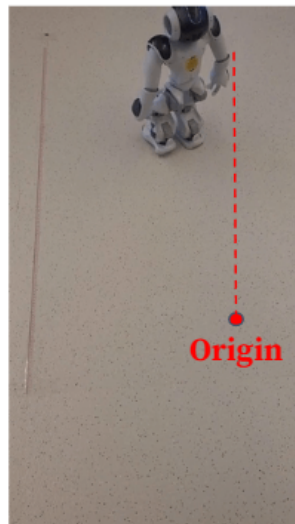


Figure 5: Robot strayed from the X axis

Despite the challenges, we were successful in achieving our motion objectives by utilizing the NAOqi API and building our control methods.

3.3 Control and Navigation Algorithms

Algorithm 1 demonstrates the high-level Python structure of the program: Hermes gives a tour of the *HMI*² lab at Santa Clara University. This program involves computer vision techniques and various services provided by the NAOqi library. The core concept is to detect QR codes in the robot's vision field and adjust the robot's path based on color hue. The program uses OpenCV for video capture and processing, and NAOqi to control the robot.

The script starts by importing necessary libraries and initializing some constants, services, and

variables. The program communicates with the robot using the NAOqi library, establishing proxies for services like `ALMotion` for general movement, `ALBasicAwareness` for disabling unnecessary head movement, `ALTextToSpeech` and `ALAnimatedSpeech` for speech generation, and `ALRobot-Posture` for posture control. After waking up the robot and adjusting the head's position, the robot starts moving forward. The main control flow takes place inside a loop that runs indefinitely until manually stopped. Within this loop, it checks if the camera is ready to capture frames. If so, it sets up video capture and adjusts certain properties of the captured frames.

For every frame, it checks if the frame is not empty, and then converts it from BGR (Blue, Green, Red) color space to HSV (Hue, Saturation, Value). There are eight specific points located in the middle of the frame's height to check the hue values. Based on this value, it classifies the hue into one of six color categories: Red, Orange, Yellow, Green, Blue, or Violet. In this program, the robot will follow the color Red, if the boundary points detect different colors than Red, an error is present and the robot will rely on PID control to adjust its direction. The program then checks if it's time to detect QR codes. If a QR code is detected, depending on the code's predefined data, different speech outputs are produced, and the robot's motion is temporarily stopped. For example, if QR code '1' is detected, the robot introduces herself as Sofia and gives a brief description of the lab. When QR code '5' is detected, Sofia ends the tour and crouches down.

After the QR code checking and processing, the script runs a loop back to the eight specific points in the frame, assessing the hue value of each point. Again, if the hue value is above a certain threshold, the robot adjusts its movement depending on whether the point is on the right or left side of the frame. The robot continues to move forward if it's not already moving and the tour is not finished. Every processed frame is displayed in a window titled "Nao Bottom Camera Color Test". The loop is terminated if the 'q' key is pressed.

Once the main loop is exited, the program releases the video capture, resumes the robot's aware-

ness, changes its posture to "Crouching", and destroys all OpenCV windows.

3.3.1 Complexity Analysis

The complexity of the program can be broken down into time and space complexity. The time complexity refers to the computational complexity that describes the amount of time taken by an algorithm to run, as a function of the size of the input to the program. The space complexity is the amount of memory used by the algorithm (including the input values to the algorithm) to execute and produce the result. Table 3 explains the complexity of the major functions.

Function	Description	Complexity
cv2.VideoCapture	Captures video from the camera	Hardware-dependent (Constant per frame)
cap.set	Sets properties of the video capture	Constant
cap.read	Captures a frame from the video feed	Hardware-dependent (Constant per frame)
cv2.cvtColor	Converts the color space of the frame	$O(n)$ (n = number of pixels)
detector.detectAndDecode	Detects and decodes QR codes in a frame	$O(n)$ (n = number of pixels)
motion_service.stopMove, move, setFootSteps	Robot control functions	Constant
for loop	Performs operations and calculations based on the frame's data	$O(1)$

Table 3: Major Functions and Their Complexities

Overall, the main factor affecting the time complexity of this program is the number of pixels in the frames being processed, which affects the 'cvtColor' and 'detectAndDecode' functions. Given that the size of the frames being processed remains constant, the time complexity for each frame can be considered to be $O(1)$, making the time complexity of the whole program $O(n)$ where n is

Algorithm 1 Hermes Motion-Navigation-Speech

Require:

Import Required Modules; Initialize IP and Port; Initialize Robot Variables; Initialize Camera Variables; Initialize QR Variables; Initialize PID Control Variables; Setup Robot Walk Test Variables; Setup Robot Head Posture

Ensure:

Robot Head Still, "Head" Stiffness=1.0; Robot "Standing"

StartWalking()

while True **do**

if CameraIsReady() **then**

 SetupVideoCapture

end if

 frame ← ReadFrameFromCamera()

if FrameIsEmpty(frame) **then**

continue

end if

 hsv_frame ← ConvertFrameToHSV(frame)

if TimeToDetectQR() **then**

 Decode QR Code; Output Motion and Speech; Update Variables

end if

for each i in Eight Points of the Frame **do**

 hue_value ← GetPixelHue(hsv_frame)

 error ← CalculateError(hue_value)

if HueIsAboveThreshold() **then**

 UpdatePIDFlag

if PIDFlagIsAboveThreshold() AND RobotIsMoving() **then**

 adjust ← AdjustRobotMovement()

 lastError ← error

 ResetPIDFlag

if FirstPointIsRight() **then**

 MoveFootStepsRight

end if

if FirstPointIsLeft() **then**

 MoveFootStepsLeft

end if

end if

end if

end for

if RobotIsNotMoving() AND TourNotFinished() **then**

 MoveForward

end if

function DISPLAYFRAME(frame)

end function

if 'q' is pressed **then**

break

end if

end while

ReleaseVideoCapture; ResumeRobotAwareness; CrouchRobot; CloseDisplayWindow

the number of frames processed. The space complexity is also $O(n)$, where n is the size of each frame (since each frame is stored in memory), plus the space required for the various variables and data structures used in the program.

3.4 Human-Robot Interaction

Human-Robot Interaction (HRI) is a fast-expanding field that is responsible for the various means of communication between humans and robots. Humans and robots can exchange information in a few ways—face recognition (using a camera), speech recognition (using a microphone and speakers), touch detection (using tactile sensors), and more. For this project, the only necessary means of communication is speech recognition because the main form of interaction during a guided tour is verbal communication. Both components of speech recognition, speech-to-text and text-to-speech, were implemented in Python 2.7 using the API of the robot’s software development kit, NaoQi, which provided libraries of methods that controlled the robot.

3.4.1 Text-to-Speech

For the robot to be able to generate speech and respond to a visitor’s question or remark, it needs text-to-speech capabilities. We were able to establish this functionality using the API from the ALTextToSpeech module, mainly the `ALTextToSpeechProxy::say()` function, which only had as a parameter the string it was to speak. There is also a function that changes the language in which the robot speaks, making it capable of conversing in the wide range of languages the robot supports.

A concern of the team was that the robot would not sound as human when it speaks because when we first heard it speak, the robot’s voice had a small inflection in its tone of voice, it did not pause enough between some words, it did not enunciate as well as a human, and its volume could not

change as it spoke to simulate emotion. However, these are things that could be changed via other methods. For example, a NaoQi function named "The ALTextToSpeechProxy::setParameter()" can change the pitch, speed, and volume of the robot's voice. Another method named "sayToFile()" saves a file that contains the recording of the robot's speech and is stored in its file system. After various parameters were adjusted, the robot spoke well, and we could move to our next objective, speech-to-text.

3.4.2 Speech-to-Text

The vision behind HRI was that throughout the tour, the robot would arrive at each exhibit, give an explanation of its significance, respond to any questions, and move to the next stop. For the robot to be able to respond to questions it must first be able to interpret them. For the robot to interpret human speech, it needs to first recognize it and convert it to text, which it then parses to look for keywords to construct an interpretation. Conversational capability is something incredibly complex to develop in a robot if the goal is to assimilate human conversation as much as possible, but for our purposes, it was assumed that most conversation would pertain to the tour site and exhibits. This is why our intended approach was to predetermine and save a list of responses to common or potential questions and have an algorithm map the visitor's actual question to the closest match in the list of saved responses.

To give the robot the ability to distinguish between words and identify keywords, we applied the same approach as with Text-to-Speech, but this time interacting with the API of the AL-SpeechRecognition module. This module works by using the robot's memory to keep a list of its "vocabulary," or words to recognize. Whenever the speaker is heard, whichever word in its vocabulary that best matches what is heard is placed in a key in memory, WordRecognized. The WordRecognized key holds a list of words in its vocabulary along with their corresponding confi-

dence level, sorted in decreasing likelihood of a match. There are many methods that interact with the speech recognition engine, including one to change the language of the speech to recognize, another to set the vocabulary of the robot, and one to start listening.

NaoQi also includes a list of events, which trigger callback functions where the occurrence of an event is processed. Our main focus was the `WordRecognized` event because, as a human spoke, the robot would have to find the best match between the spoken word and vocabulary word and process the event of a match in a callback function. Since it would be costly (memory-wise and computationally) to set the entire dictionary as part of the robot's vocabulary and have the robot process every spoken word, we would instead only include keywords related to the subject of the tour as part of the robot's vocabulary. Because not every spoken word is being processed, the callback function would work to build a sentence, albeit broken, of only keywords, from which a rough interpretation of what was spoken can be constructed. This interpretation would then be mapped to a hard-coded response that best suits the question. Only a handful of distinct interpretations can be generated because of the assumption that questions would only be related to the tour, which makes it feasible to hard-code responses about the exhibits, faculty members, or SCDI itself.

3.4.3 Implementation and Challenges

The development of text-to-speech was surprisingly simple, requiring the use of only a couple of methods and trial and error testing. With repeated testing, we found a combination of speed, pitch, and pauses that made the robot sound more natural. While actually implementing the speech-to-text module, however, we ran into problems that required a different approach to circumvent.

To begin testing the speech-to-text module, we included only the word, "miller," in the robot's vocabulary and would speak that word to trigger a callback function that made the robot respond,

“I heard miller.” As we repeated the word “miler” to the robot as clearly as possible while the speech recognition engine was active, we noticed that the callback function would not be triggered, leaving us confused and doubtful of the correctness of our implementation.

However, there were a few instances in which we accidentally forgot to terminate the program, and after a long period of silence a thud caused by a falling phone would somehow trigger the callback function, causing the robot to repeat, “I heard Miller.” After that thud, it seemed that the implementation was finally working because the callback function was executed every time we repeated “Miller.” After a while, this would no longer be the case, and the implementation would stop working again. In other instances, the robot would even repeat, “I heard miller” twice or thrice for every time the word “miller” was spoken. This left us very confused, so we decided to have the program also print the result located in the WordRecognized key in memory. We tested the functionality again and noticed that there was a low confidence regardless of whether we repeated “miller” or any other word. Only after a long silence and a random thud would the program work properly and display the result with accurate confidence levels depending on the word spoken by the user.

This prompted us to reach out to RobotLabs, the manufacturers of the Nao robot, and ask for assistance on our implementation. They did not find any causes to our faulty implementation, but they could not test and debug it either. After searching the internet for cases of the same problem and potential solutions, we did not find anything specific to our situation because most of these cases incorporated Choregraphe, which had code that could not translate easily to a standalone implementation. The growing impatience caused by this obstacle and approaching deadline caused us to seek alternative solutions to establishing speech-to-text functionality. We explored the use of Google Cloud services and VOSK API, and the results we observed are covered in greater detail in Chapter 4 of this report.

3.5 Integration

3.5.1 Motion and Speech Integration

The final step of our project is about joining all the parts we have worked on into one whole system. This includes blending tasks related to motion control, speech handling, and QR code recognition into one smooth process.

When it comes to controlling movement, we used NAOqi APIs and OpenCV. NAOqi APIs are like a bridge that helps us give commands to the robot's moving parts. At the same time, OpenCV is a computer program that helps us 'see' and understand QR codes, which we chose to use for this project. By 'reading' these codes, the NAO 6 robot can decide what to do next, creating a smooth flow based on the QR code information it gets. We also worked on handling speech, which involves turning spoken words into written text and vice versa. We achieved this by using VOSK, NAOqi APIs, and Google Speech. VOSK is a tool that helps turn speech into text, while NAOqi APIs help the robot understand this text and decide how to react. Google Speech is another tool that helps to improve the robot's understanding and use of speech in real-time. By linking these parts together - controlling movement with NAOqi APIs and OpenCV, and handling speech with VOSK, NAOqi APIs, and Google Speech - we created a complete system.

3.5.2 Motion-Speech Implementation and Challenges

In the implementation of the motion-speech integration, we found that the physical parameters of the QR code played a significant role in the success of the system. Based on our testing results, the size, color, and placement of the QR code directly affected the performance of our system.

The programming environment of the NAO robot API gave us a challenge. It only supports up

to Python 2.7, hence we were limited to using OpenCV 3.4.4, a version of the library released in 2018. This version is considerably outdated by computer vision standards, which advance rapidly. For instance, Python 2.7 does not include the function to detect QR code with curvature (Figure 6). Despite this restriction, we were still able to use it effectively for QR code recognition. Another obstacle was the limitations of the robot's camera. The camera resolution and frames per second (FPS) rate are quite low compared to modern standards. This made the detection of small QR codes difficult (Figure 7), as the camera struggled to accurately capture and process them.



Figure 6: Curved QR code

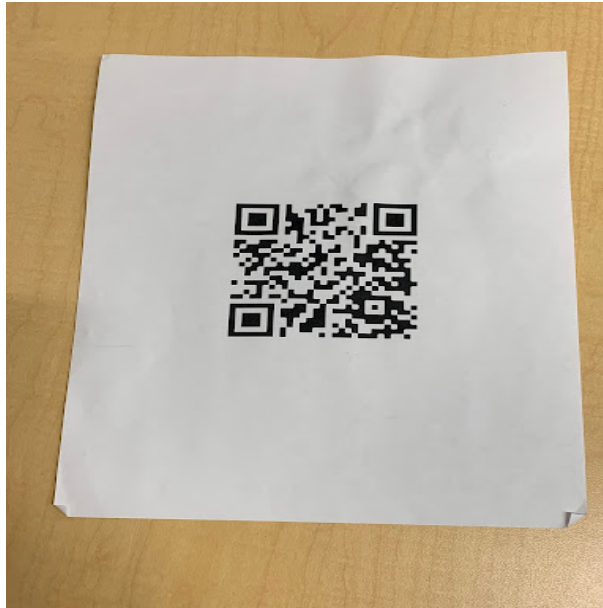


Figure 7: Small-scale QR code

To mitigate these challenges, we focused on optimizing the physical parameters of the QR code. We decided to use a larger, A-4 size QR code in a bright red color to maximize visibility. Keeping the QR code flat also ensured a more stable detection by the camera. Despite these limitations and challenges, our strategic adjustments demonstrated that successful integration is possible with thoughtful design considerations and efficient use of available resources.

3.5.3 Network Integration

One of the final visions is to adapt the project in any location with appropriate network approaches. The initial network approach was to use Jetson Nano as a portable computer. We successfully installed Ubuntu 18.04, OpenCV, and Python 2.7 onto the Jetson Nano. This combination of software provided the necessary platform for our robot's operation and interface. We also assigned a static IP to our robot, establishing a consistent point of network contact. However, integrating

the NAOqi API presented an unexpected challenge. The provider of the API, Aldebaran, only supports Linux 64, Windows 32, and Mac operating systems with their executable files. None of these matched the ARM 64 architecture of the Jetson Nano. Yet, we proposed a solution involving the source file of NAOqi: running it directly from the Jetson Nano could potentially generate an ARM 64 compatible binary file. We believed this would enable us to compile the NAOqi and Python scripts directly on the Jetson Nano, bypassing the architecture compatibility issue.



Figure 8: Jetson Nano

While striving for this solution, we simultaneously explored other network options. The second approach we considered was utilizing existing university networks, like Eduroam or BroncoFi. While the availability of these networks was a plus, the main challenge that emerged was network latency. This lag, if not properly managed, could lead to accumulated errors in our system's performance, affecting the overall efficiency and accuracy of our robot.

This led us to our final, and ultimately successful, approach - the use of a portable router. After

much research and testing, we settled on the TP-Link Archer GX90. This compact device provided us with a swift and steady network, ideally suited to our project's needs. It ensured consistent, low-latency communication between our system components, enabling our robot to operate optimally. Thus, through a process of research, trial, error, and adaptation, we arrived at a robust solution for our network integration challenge.



Figure 9: TP-Link Archer GX90

4 Chapter 4: Experimental Results

4.1 Test Environment

The test environment for all tests consisted of the NAO6 Robot and a PC to run Python scripts. The tests were run in the Human-Machine Interaction & Innovation (*HMI*²) laboratory located on the 4th floor of the Sobrato Campus of Discovery and Innovation (SCDI) at Santa Clara University.

There was general lighting present in the laboratory.

Robot Motion Accuracy: We utilized different floor coverings and carpets, using a 2 meter long measurement tape to detail our x-axis and another measurement tool to measure the y-axis.

Navigation Accuracy: We utilized red floor coverings and QR Codes. Our measurements would come into the PC from the Python script, measuring out the Hue-Saturation-Value colors at eight distinct pixel values on the robot's camera.

4.2 Robot Motion Accuracy

In this section, we present our findings on the accuracy of the robot's motion as controlled by our implementation of the NAOqi API functions.

We tested the accuracy of the robot's movement by performing a walk test. Our experimental variable was the floor surface, which would be changed after 10 trials. Our setup consisted of giving a command to the robot to walk along the X-axis 1 meter forward. The robot was given no rotation. After the completion of one trial, we would measure the actual distance the robot traveled in the X direction and the deviation from the Y-axis. We would then calculate the angle from its starting position to end position and compare them to our target angle. The setup can be seen in Figure 10.

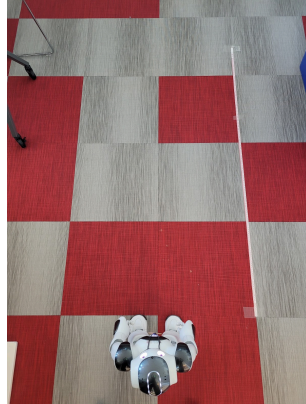


Figure 10: Walk Test setup.

The robot is placed in a designated starting position, with measuring tape laid out on the side of the robot to indicate the X-Axis. The flooring was changed every 10 trials the test whether the surface the robot is walking on accounts for the drifting problem. The flooring that were utilized are rubber, wood, and tiled biofelt.

Table 4: Walk Test Results

	Average Distance Travelled (cm)	Average Angle (degrees)	RMS Error Dis- tance (cm)	RMS Error Angle (degrees)
Rubber	105.51 ± 1.31	8.54 ± 4.5	5.66	9.67
Wood	104.47 ± 0.92	3.22 ± 1.96	4.55	3.72
Tiled Biofelt	99.06 ± 1.26	2.95 ± 2.51	1.52	3.79

In Table 4, we measured the mean and standard deviation from the intended path. With a goal of traveling 1 meter with 0 angle deviation. Using the values for mean and standard deviation, we calculated our Root Mean Square (RMS) error. Our testing showed that when given a command to move 1 meter forward, the surface that had a standard deviation within that range was Tiled Biofelt. When calculating the average angle, the closest value to 0 degrees is also Tile Biofelt. Each surface also has our Root Mean Square(RMS) error, which demonstrates our error that built

up across 1 meter.

These results confirmed that our approach to using different surfaces is not feasible over a longer movement period. The drifting problem would still be present and cascade over a longer distance. This demonstrated a need to shift away from relying purely on the NAOqi API for precise control for robotic movement, and finding a solution to combat the drifting problem. This led to us using OpenCV to give Hermes precise movement and the elimination of the drifting affecting the project as a whole.

4.3 Navigation Accuracy

To provide a complete guided tour, the robot requires to follow a red carpet while performing motion and/or speech tasks based on given QR codes. The navigation accuracy depends on the success rate of the test run. A successful test run means that the robot needs to be able to perform a complete guided tour and reaches the final position without missing any QR code. To measure the navigation accuracy, we posit the robot at the initial position of the carpet with different angles from -30 degrees to +30 degrees, with 0 degrees being the center line (Figure 11). We demonstrate three scenarios: The Straight-Line Test, the One-Corner Test, and the Two-Corner Test. The test environment is the *HMI*² lab. The test conditions are listed below:

- The carpet must be tightly attached to the ground to avoid folded surface.
- QR codes must be evenly separated with a distance of at least 2 feet between two QR codes.
- QR codes must not be placed near the initial position, or at each corner, with a distance at least 2 feet away from the corner(s).

- The test environment must avoid irrelevant red-color objects other than the red carpet and the red QR codes to reduce distraction.

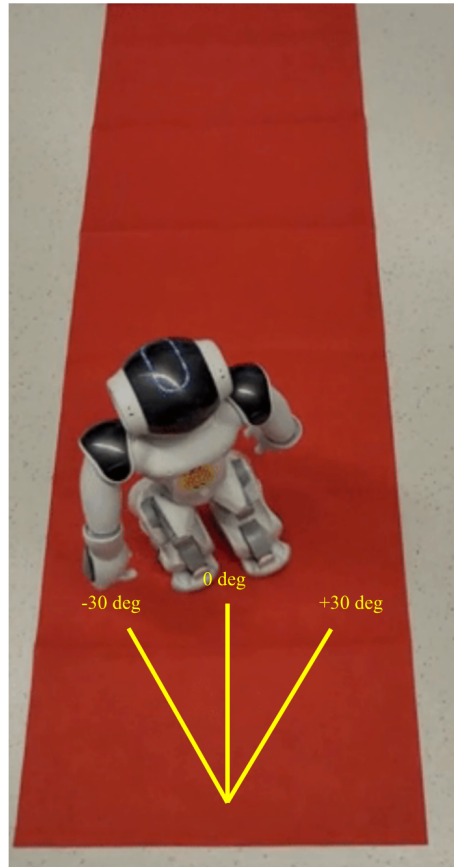


Figure 11: Demonstration Setup with Initial Angles

The straight line test consists of 3 QR codes and is shown in Figure 12. The robot starts from different starting angles (-30 degrees, 0, and 30 degrees) and for each angle, 5 trials are performed. Table 5 demonstrates the result of the Straight-Line Test with an average success rate of 0.933. In the One-Corner Test scenario (Figure 13), the robot can perform the task with a success rate of 0.66, shown in Table 6. Lastly, Figure 14 offers the demonstration of the Two-Corner Test, and Table 7 shows the Two-Corner Test has a success rate of 0.466.

Table 5: Straight-Line Test with 3 QR Codes

Starting Angle (deg)	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Success Rate
-30	Yes	Yes	Yes	No	Yes	0.8
0	Yes	Yes	Yes	Yes	Yes	1.0
+30	Yes	Yes	Yes	Yes	Yes	1.0



Figure 12: Demonstration Setup of the Nao robot with a straight path and 3 QR Codes

Table 6: One-Corner Test with 5 QR Codes

Starting Angle (deg)	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Success Rate
-30	Yes	Yes	No	No	Yes	0.6
0	No	Yes	Yes	Yes	Yes	0.8
+30	No	No	Yes	Yes	Yes	0.6

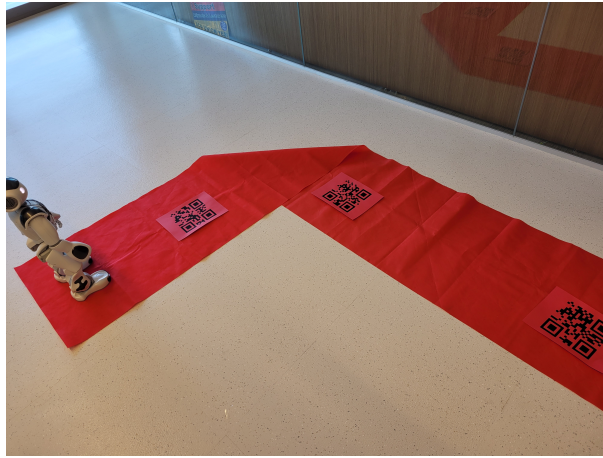


Figure 13: Demonstration Setup of the Nao robot with one-corner path and 5 QR Codes

Table 7: Two-Corner Test with 5 QR Codes

Starting Angle (deg)	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Success Rate
-30	No	Yes	Yes	No	No	0.4
0	Yes	Yes	No	No	No	0.4
+30	No	Yes	Yes	No	Yes	0.6

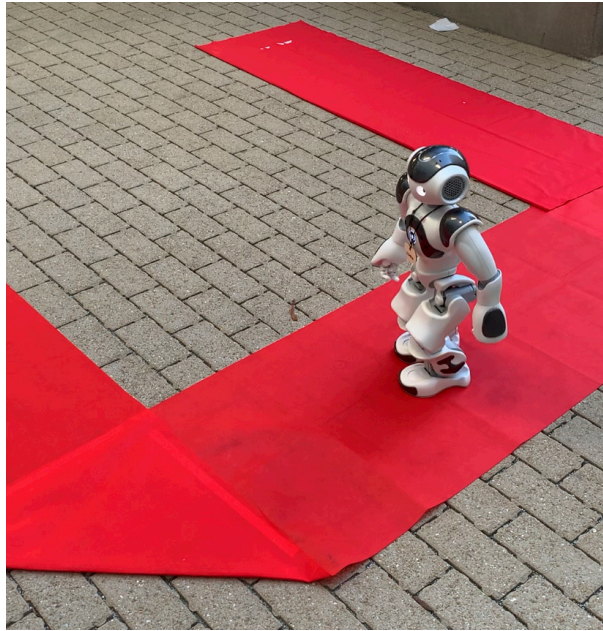


Figure 14: Demonstration Setup of the Nao robot with Two-Corner path

Based on the test run results, they demonstrate that the structure of the test environment influences the success rate. In the scenario where the path is a straight line, the robot can perform the task with the highest success rate. The only failed trial was due to missing reading of the first QR code. The cause was an unusual turn while it was performing PID control. In the scenarios where one or more corners are present, the robot was able to perform with a relatively acceptable success rate. Two major failure reasons were 1) skipping one or more QR code(s) and 2) following the wrong path-at the corner, it will detect and move toward the ahead path (which is perpendicular to the current path) rather than the current path. In addition, the robot usually overheats every other 45 minutes to 60 minutes of operation. Therefore, some test runs were invalid and required retest. Overall, we predict that as the structure of the test environment becomes more complex, the robot's performance will decrease with a relationship between linear and logarithmic decay.

4.4 Speech Recognition Quality

Our initial implementation for speech-to-text used the NaoQi API and speech recognition engine to look for whenever a word in the robot's vocabulary is heard. Because this approach was not working properly and we were unable to find the cause, we decided to forgo the use of the API and instead incorporate Google's speech-to-text service, which would require the robot to stream the microphone data to the Google Cloud via the internet. Using Google Speech-to-Text, including streaming to the cloud required Python 3.8, and our robot could not support past Python 2.7, so we could not simply stream the microphone to the cloud directly from the robot. Instead, the data would have to be streamed from the robot to our main PC, which would stream the data to the Cloud. The result of whatever speech was decoded would also have to take this path of communication, and ROS Topics for both directions of communication would have to be implemented. With a fast-approaching deadline, this would be very difficult to implement, so we sought a simpler solution, ideally one where there was no need to stream data across the internet. This is where VOSK, an offline speech recognition API, became a more viable option.

4.4.1 VOSK API

VOSK would not be as accurate at decoding speech as Google's speech-to-text is, but it would use the microphone data to generate text and place it in a string that could then be parsed, interpreted, and mapped to the best-matched response. Unfortunately, time ran out before we could actually implement this approach, but we successfully tested a proof-of-concept on our main PC in which we spoke into a connected microphone, stored the converted text in a string, and generated a simple response to the text. Had there been more time, the same idea would be applied to the robot and we would have been able to program the algorithm that generates an interpretation and determines

the response. The output can be seen in Figure 15.

```
{  
  "text" : "a robot can you please tell me more about the miller center"  
}
```

The miller center is the center for social entrepreneurship. Miller Center is at the forefront of the global social entrepreneurship movement, with a robust track record of accelerating social enterprises since 2003. Social entrepreneurs around the world are innovating solutions to tackle our most pressing social problems and addressing the United Nations Sustainable Development Goals (SDGs). We believe that social entrepreneurship is leading the way in transforming markets for good and ultimately propelling us to end poverty.

Figure 15: Output using VOSK

5 Chapter 5: Hermes Evaluation

As Hermes requires to interact with humans, it is important to evaluate the human perspective of a tour guided robot. Therefore, we conducted a user study, where participants are to observe the robot performing a demonstration of giving a tour in an enclosed setting.

The focus of this study is to evaluate the usability and applications of such a service robot in contemporary society. We sought feedback on whether users could see a service robot being used in the future to assist with tasks. Additionally, we set to investigate that with a humanoid robot performing an action, humans would be receptive to service robots throughout contemporary life. The feedback will improve the actions of the robot for an improved cohesive demonstration to make it more natural, and to enhance and keep engagement with the participants.

5.1 Experimental Protocol

We set up a space for the robot to maneuver and perform a routine as users observe from a distance. A pilot study was conducted utilizing a simple routine for the NAO6 robot to perform on the SCDI

4th floor hallway as seen in Figure 16. The user study was approved by Santa Clara University's Institutional Review Board(IRB), IRB protocol number: 23-04-1935, allowing us to perform a study on how the robot interacts with humans. We had 6 participants. The pilot study aims to gather an initial dataset to drive changes into the robot in the usability and applications realm.



Figure 16: Location of Demonstration. SCDI 4th floor hallway.

Participants were asked to observe a routine delivered by the NAO6 robot. The routine of the NAO6 robot is to follow a set path with a 90-degree angle in the path as shown in Figure 17. There were QR codes present along the path for the NAO6 robot to output speech at the designated points along the path. Afterwards, participants were asked to fill out a questionnaire on a 1-5 Likert scale to gather their feedback on the robot's usability and applications.

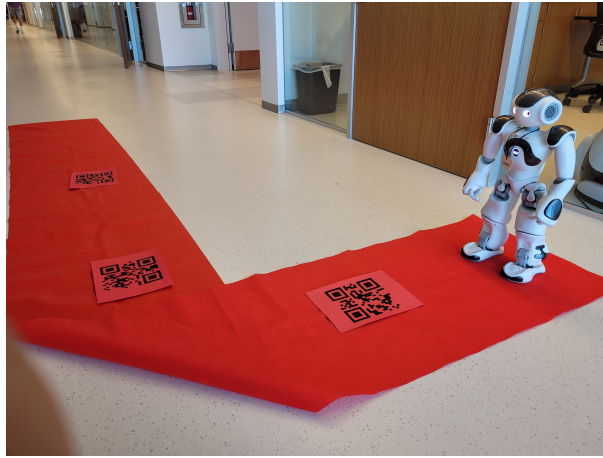


Figure 17: Demonstration Setup of the Nao robot and 90 degree angle in the path

5.2 Evaluation Metrics

Following the experimental protocol, a questionnaire was developed as the study moves onto participants that are partaking from demonstration given by the NAO6 robot. The use of subjective metrics provides feedback from the participants. The statements are as follows:

Q1: I was comfortable observing the humanoid robot's interaction during the demonstration.

Q2: I believe humanoid robots like the one demonstrated could be integrated into various aspects of daily life.

Q3: I have trust in the reliability and safety of the humanoid robot based on the demonstration.

Q4: I believe it is important for humanoid robots to possess human-like appearance and behavior.

Q5: I have ethical concerns or limitations arising from widespread adoption of humanoid robots in society.

Q6: I believe the demonstrated humanoid robot has the potential to enhance social interactions.

- Note: Scale 1 means the participant has the most negative attitude towards the sentence; scale 5 means the participant has the most positive attitude towards the sentence.

5.3 User Feedback

To evaluate our system, a preliminary study was conducted with 6 participants. The study was anonymous and no personal information was collected. Table 8 includes the mean, standard deviation, and descriptions of the questionnaire results.

Table 8: User Feedback

Questions	Mean	Standard Error	Description
Q1	5	0	This score demonstrates that all participants were comfortable with Sofia performing her demonstration.
Q2	4.5	0.25	While the score is relatively high, it also shows that there are doubts seated with having robots integrated throughout the modern day.
Q3	4.16	0.47	Our measures into having Sofia be reliable and safe for all users are shown in this score.
Q4	3.67	0.55	Our lowest score, the value shows that a humanoid robot that has human characteristics is not universally essential. This also shows us that participants may place more value on functionality and reliability over appearance.
Q5	3.33	1.55	Relatively diverse scoring, split between a score of 5 and a score of 3. Some participants exhibit some ethical concerns related to having robots in contemporary society, while others welcome the inclusion of robots without any concerns about widespread robot adoption.
Q6	4	0.33	Demonstrates the strong agreement between participants that the design of Hermes does have the potential to enhance our interactions between humans and robots. The score gives us a strong foundation to continue working towards.

From the questionnaire results presented above, we see a generally positive response to the robot performing as a guided tour. While robot demonstrations are welcomed and certain design aspects are favored, there are several concerns about full integration, robot appearance, and societal impacts. It is crucial for all robot developers to focus on addressing these concerns, providing clear guidelines, and emphasizing functional reliability over human-like appearances.

6 Chapter 6: Standards and Constraints

It is important to manage expectations and understand that our robot has some limitations. The first limitation is its size. Aside from being welcoming to young children, it is harder to notice which makes it more likely that someone could accidentally collide with it. It does walk like a human, but, because of its size, much slower than one, which will extend the length of the tours it gives. There is another limitation on its ability to converse because it is computationally demanding and has too much algorithmic complexity. Once we develop a simple working model, others may improve upon it. The limitations of this robot are understood by many and this project may serve as a proof of concept for when the hardware is more capable.

For any project, limitations are unavoidable and difficult to predict until you have invested enough work. For example, we did not imagine that our robot would be limited by its hardware until we gained control over motion and began to think about having it traverse most of the fourth floor of SCDI for the duration of the tour. It was during the final stages of testing robot motion control that we realized that the servo motors which moved the robot's joints were overheating due to overuse. We were at first confused about how there was overuse when we only made it walk a few steps back and forth for a few runs. We later realized that the servo motors are still facing a load even when they are not moving because they still have to overcome the torque due to the weight

of the robot's limbs to maintain their current positions. This observation combined with the slow walking speed of the robot made us realize that it would not have the capacity to guide the tour without stopping to cool down in between. We then shifted our objective to a proof-of-concept where the functionality was demonstrated in an abbreviated version of the tour.

6.1 Standards

Although the task of tour guiding may seem to be of low complexity for humans, it is a complicated ability to develop in a robot, with many moving parts and deserving of much testing. If the robot is to have such impressive characteristics after only a year of development, it will be necessary to build upon existing technology. This is why using standards that others worked tirelessly to establish is important. The most important standards on which we relied were the Python programming language and Wi-Fi. It would be impossible to implement our desired functionality for the robot using machine code or even Assembly language. Using the Python language syntax and its interpreter, along with the fact that the programming language is the same on any computer that supports it, we can ensure that this robot can be controlled from any computer in any building with a stable connection. This portability depends on the use of another standard, Wi-Fi, which is the communication protocol between the robot and the computer loading code it. If the means of communication between the computer and the robot were to be developed from scratch using no standards, the length of the project could not have been a year or less. If the robot's task is to be repeatable and portable, it is useful to use the standards of Python and Wi-Fi because they ensure consistency across many different scenarios.

6.2 Technical Challenges and Constraints

Throughout the course of our project, we faced various technical limitations which presented unique challenges. However, each constraint also served as a valuable learning experience.

One of our first challenges was the software constraint posed by Python 2.7. The Nao robot library, a critical component of our project, only supports up to this version of Python. This limitation constrained our ability to utilize more advanced functions that were available in later versions, such as those capable of detecting curved QR codes. As we proceeded, we had to carefully select compatible functions and libraries that could work effectively within this programming environment.

In addition, we had to consider the hardware limitations of the Nao robot. The robot could easily overheat after 30-45 minutes of continuous operation, particularly in the left leg, posing risks to both performance and hardware integrity. Moreover, its physical dimensions and structure restricted its speed of movement. This presented a challenge in scenarios where faster response times were desirable.

The third limitation was that the NAOqi API only offers a restricted set of functions for Python. This constrained the range of programming options of the project, requiring us to carefully strategize the functionality and structure of our algorithm within the given limitations.

On the networking end, identifying the optimal networking approach remained as a challenge. As we experimented with various strategies, it became clear that a balance needed to be considered between stability, latency, and compatibility with our robot's computing platform.

Finally, we faced challenges associated with the complexity of our programming functions. Many functions exhibited linear or even higher-order complexity, raising potential efficiency issues. It was crucial for us to design our algorithms wisely to avoid nested loops that could significantly

decrease the performance of our system.

Nevertheless, despite these challenges, we learned and adapted at every step of the project. Each limitation not only tested our problem-solving skills but also broadened our understanding of effective system design, resource management, and algorithm optimization. In retrospect, these limitations and challenges have shaped our project into a more resilient and versatile system than we initially planned.

6.3 Privacy

Privacy concerns associated with robot utilization are a primary consideration. Unlike mobile phones, which are owned by users, robots in banks or hospitals, for instance, are controlled by third parties. It is concerning if the information collected by robots, such as facial images, voices, or even pictures of private information, is stored or shared with unauthorized parties. Addressing these concerns, particularly with regard to information sharing for navigation and voice recognition, is a priority for our team and other researchers.

In our project, the NAO 6 robot features two cameras: one on the forehead and another on the mouth. The forehead camera is the primary camera with a 480P resolution, while the mouth camera serves as a secondary camera with a 360P resolution. Although both cameras lack high resolution, they can still scan QR codes and recognize and record human movements. The NAO 6 robot's software also enables basic machine learning, allowing the robot to recognize patterns and potentially record the daily schedules of people on the fourth floor of the SCDI building. Furthermore, the NAO 6 robot includes microphones that record voices and convert them into text. As with mobile phones, which can record human voices and tones, detailed privacy policies and agreements must be established and discussed. Additionally, the NAO 6 robot is equipped

with Wi-Fi and Bluetooth modules, enabling communication via networks and Bluetooth. Privacy concerns must not be overlooked with any Wi-Fi-enabled device, as there is a risk of information being accidentally exposed, leading to potential misuse.

While personal information should generally be considered private, humans should have the right to access selected data when robots communicate with them. The widespread integration of networks within robots and automated systems increases the risk of unauthorized individuals stealing information through robot cameras and sensors. Addressing the ethical considerations of privacy policies for implementing robots and automated systems is an urgent issue that requires attention.

6.4 Cost

The development and deployment of sophisticated, reliable, and user-friendly tour-guiding robots can be expensive. This cost is often passed on to customers or tour operators. Additionally, maintaining and repairing these robots can be costly, particularly if specialized technicians or replacement parts are needed.

In this project, we used two strategies to relatively reduce the cost of operation. First, we utilized open-source software. Our team employs several open-source software including Python, OpenCV, and ROS to reduce development costs. Open-source software provides access to a wide range of resources and tools, allowing developers to build upon existing technologies rather than starting from scratch. Second, Santa Clara University can collaborate with robot manufacturers such as SoftBank Robotics and Aldebaran Robotics to access resources for research and development. This can help us to reduce the cost of innovation and development.

6.5 Economic Considerations

Robotics and automation are advancing to a stage where they can replace humans for various tasks, and it is a good idea to evaluate the repercussions of developing a robot that can guide a tour in the place of a human. While our development is not yet at this stage, it will not be long before another development is, and considering the possible changes it will have on the job market ahead of time will allow us to prepare for the transition and lessen the rippling effects. The most compelling concern is that it will reduce job availability for humans because most tour guide jobs will be handled by a robot. However, this concern fails to take into account the fact that new jobs will be created which undertake the protection and maintenance of each robot guide, as well as the testing and training of these robots for their continued advancement. The increase in robot-centered duties corresponds to the decrease in available tour guiding jobs, so the job availability differential between before robot integration and after is greatly reduced, and the shock on the job market can become a gradual change.

6.6 Safety

Because these robots are expensive to develop, ensuring their protection and that of others is paramount. The NAO robot is quite delicate and may operate in the vicinity of a stairway, so if it falls down the stairs, it will be a total loss of investment. It is also small in size, making it very easy for people coming out of an elevator or door to fail to notice it, strike it, and injure themselves and the robot. This is why future iterations of tour guide robots need to be programmed to keep track of “caution areas”, where before entry it scans for moving objects or boundaries that it should never cross. Only when it is safe should the robot proceed through the “caution areas.” This along with other techniques should prevent the robot from placing itself in a treacherous position where

large investments are at stake. In our project the robot used color detection and a red carpet to maintain its alignment down the path, making it very easy for humans to notice the robot and avoid colliding with it. Other variations that do not use this approach will have to install caution signs or another visual stimulus to make everyone aware of the presence of a robot so that they know to walk with caution. The other half of the equation is the safety of humans. It may not weigh very much when concerning small humanoid robots, but what about when the robot is human-sized and is capable of inflicting serious damage on the user? An existing robot of this size developed by Aldebaran Robotics is called Pepper, and these along with other developers are incorporating preventative measures into human-sized robots such as automatic slowdowns of its limbs or motion when its sensors detect a person nearby.

6.7 Sustainability, Environmental Impact, and Social Impact

The question of whether the integration of tour guide robots is sustainable is key for knowing whether this is a step in the right direction for our survival on this planet. Sustainability in paying for the power consumption of these robots is important and seems promising as chips are able to execute their functions using less power as technology evolves. The goal is for the cost of the power consumption of as many robots one employs to become equal to or less than the cost of employing humans for the job instead.

Whether the manufacturing of these robots will damage the planet is dependent on the materials used to make its components at each level of complexity as well as the techniques used to fuse everything together. This means that hardware chips, motors, sensors, and materials to fuse and protect everything from the outside need to be made using carbon-neutral methods. Luckily there is a great push for all technology to have these qualities, so generations of these robots should be

progressively more sustainable.

Many also wonder about the societal impacts of employing socially assistive robots in the workforce. The worry is that children will adapt to interacting with more robots than ever before, which can cause unprecedented consequences after generations. It is important to clarify that the goal should not be to make robots ubiquitous and render human work obsolete, but rather to enrich human life and productivity. They should be used to enhance children's learning and workforce productivity, not to replace human interaction.

7 Chapter 7: Conclusion and Future Work

After much deliberation, testing, and readjusting, we were able to develop a proof of concept of a tour guide robot. Others can use this project and our findings as inspiration for a different approach or as a platform upon which to build and reach new heights of capability.

7.1 Lessons Learned

There were many lessons learned in the duration of this project, but none demanded more attention than time management. We learned that we should never assume that an initial approach to solving a problem will work because by allocating time based on this assumption there is no room for error or a change of plans. It is necessary to assume that there will be a change of approach for almost every objective and allocate time specifically for the event that the initial approach fails. The initial approach for robot motion failed, additionally requiring a control algorithm and the camera system for keeping it aligned with its trajectory. The initial approach of using the NaoQi API for the speech-to-text module also failed, requiring a completely different approach with VOSK.

Another important lesson was to check for hardware compatibility before doing anything. Knowing this would have saved us much time and frustration when trying to integrate the Jetson Nano as part of the robot. We were unable to use the NaoQi SDK because the computer architectures of the robot and the Jetson Nano were incompatible. When we finally learned about their incompatibility, we had already invested much effort (which could have been applied elsewhere) in trying to force something that was never going to work.

7.2 Future Work

Although this thesis marks the end of the duration of this project, there remains some curiosity about what the robot is missing or what other abilities it could ascertain. Because the speech-to-text module was not fully implemented on the robot, the next step would be to apply the proof of concept that uses the VOSK API to the robot. This would require secure shelling (SSH) into the robot's brain, downloading the VOSK SDK onto the robot, and adding to the main program a method that streams microphone data to a place in memory that will contain the text conversion of all the spoken words. This method should keep a list of keywords that shape the robot's interpretation and check which keywords are spoken by the user to determine the best response. The conversational capabilities will remain somewhat limited because of the sheer complexity of this task, but the important thing is to build a platform from which other teams can launch.

7.3 Final Remarks

This was a fantastic opportunity for gaining insight into how real-world projects are managed and executed, how unexpected problems arise and get fixed or circumvented, and how technical concepts that only those developing the project understand are presented to uninvolved people for

securing backing and making the final product sellable. We also had the privilege of working with our advisor, Dr. Maria Kyrarini, who specializes in robotics and human-machine interaction and was able to nudge the team in the right direction when an obstacle was encountered. We are grateful for the opportunity and hope for more breakthroughs in this field.

References

- [1] *Body Frames*, Aldebaran. [Online]. Available: http://doc.aldebaran.com/2-8/family/nao_technical/masses_naov6.html
- [2] *Human Interaction and Improving Knowledge through Collaborative Tour Guide Robots*, 2019. [Online]. Available: 10.1109/RO-MAN46459.2019.8956372
- [3] *Social interaction with visitors: mobile guide robots capable of offering a museum tour*, IEEJ Transactions on Electrical and Electronic Engineering. [Online]. Available: <https://doi.org/10.1002/tee.23009>
- [4] *NAOqi - Developer Guide*, Aldebaran. [Online]. Available: http://doc.aldebaran.com/2-8/index_dev_guide.html
- [5] *Choregraphe Suite*, Aldebaran. [Online]. Available: <http://doc.aldebaran.com/2-8/software/choregraphe/index.html>
- [6] Python Software Foundation. [Online]. Available: <https://www.python.org/>
- [7] OpenCV. [Online]. Available: <https://opencv.org/>
- [8] GStreamer. [Online]. Available: <https://gstreamer.freedesktop.org/>
- [9] Alpha Cephei. [Online]. Available: <https://alphacephei.com/vosk/>
- [10] *Locomotion control API*, Aldebaran. [Online]. Available: <http://doc.aldebaran.com/2-8/naoqi/motion/control-walk-api.html>
- [11] *Joint control API*, Aldebaran. [Online]. Available: <http://doc.aldebaran.com/2-8/naoqi/motion/control-joint-api.html#control-joint-api>