ABSTRACT

Title of Thesis: SENSORY COMPUTING AND OBJECT PROCESSING ENTITY: ASSISTIVE ROBOTICS FOR HEALTHCARE

John Bachkosky VI, Alexandra Boukhvalova, Kevin Chou, William Gunnarsson, James Ledwell, Brendan McTaggart, Xiaoqing Qian, Nicholas Rodgers, John Shi, Jason Yon

Thesis directed by: Dr. Anil Deane
Institute for Physical Science and Technology

Team SCOPE has created an assistive robot for healthcare delivery. The robot is mobile, responds to spoken commands, and possesses Artificial Intelligence (AI). It extracts meanings about the patient's health from conversations and visual interactions. It summarizes these observations into reports that could be merged with the patient's Electronic Health Records (EHRs). This process aids healthcare professionals in delivering better care by augmenting attendance, increasing accuracy of patient information collection, aiding in diagnosis, streamlining data collection, and automating the process of ingesting and incorporating this information into EHR systems. SCOPE's solution uses cloud-based AI services along with local processing. Using VEX Robotics parts and an Arduino microcontroller, SCOPE created a mobile platform for the robot. The robotic platform implements basic motions and obstacle avoidance. These separate systems are integrated using a Java master program, Node-Red, and IBM Watson cloud services. The resulting AI can be expanded for different applications within healthcare delivery.

SENSORY COMPUTING AND OBJECT PROCESSING ENTITY: ASSISTIVE ROBOTICS FOR HEALTHCARE

By

Team SCOPE

John Bachkosky VI
Alexandra Boukhvalova
Kevin Chou
William Gunnarsson
James Ledwell
Brendan McTaggart
Xiaoqing Qian
Nicholas Rodgers
John Shi
Jason Yon

Thesis submitted in partial fulfillment
of the Gemstone Program,
University of Maryland, College Park
2017

Advisory Committee:
Dr. Anil Deane (Chair), Institute for Physical Science and Technology (IPST)
Dr. Beth St. Jeane, College of Information Studies
John Purtilo, FedCentric Technologies LLC

Acknowledgements

Team SCOPE would like to thank our librarian, Elizabeth Soergel, and all of the Gemstone Staff for their continuous support over the past four years in the research process. We would also like to thank our discussants, Dr. Beth St. Jeane and John Purtilo, and the Institute for Physical Science and Technology for their resources and computing licenses.

In particular, we would like to thank Dr. Anil Deane for his unwavering guidance and assistance. We could not have completed this project without you.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

With the rapid advances in computer technologies in recent decades, intelligent and robotic systems have come to permeate nearly every aspect of human life. In this context, we use the term robotic to refer to a technology that can perform physical tasks with some level of autonomy, and intelligent to depict the ability to retain and act on information. In general, such systems are implemented to perform tasks that either are sufficiently simple or can be performed more safely or effectively by a robotic system. Like most fields, the healthcare sector employs an increasing number of intelligent and robotic systems.

Team SCOPE has developed a distributed Artificial Intelligence (AI) solution that is capable of alleviating routine tasks of healthcare professionals for increased quality of care. The assistive robot developed through this project assets healthcare professionals by serving as an additional medical attendant to deliver better care. It prevents missing information input from patients, aids in diagnosis, streamlines data collection, and automates, the process of ingesting and incorporating this information into HER systems. SCOPE's solution uses cloud based AI services along with local processing to form a distributed intelligence.

Robotic technologies have been incorporated in the medical field in various capacities, and with varying levels of intelligence and autonomy. At the lowest end of the artificial intelligence spectrum are robots that are completely human-controlled. An important example of this is robotic-assisted surgery, such as the well-known da Vinci system. This type of robot translates the hand motions of a surgeon into smaller and more precise movements of small instruments seen by the surgeon on a magnified 3D vision system [1]. This allows for smaller

incisions and more precise procedures than would be possible using only human hands. Slightly more autonomous robots are used in hospitals to reduce the occurrence of hospital-acquired infections in patients. Robots by Xenex use high-intensity light administered by an autonomous robot to quickly disinfect hospital rooms, showing 70% reduction in occurrence of specific infections in research trials [2].

In addition to assisting with medical functions, robotic technologies have also been implemented to improve interaction with patients and consumers. For example, robotic prescription dispensing systems by ScriptPro have been incorporated into pharmacy operations to perform repetitive actions and thereby allow pharmacists to devote more time to handling more complex patient interactions [3]. While robots can clearly contribute to repetitive tasks like prescription filling, intelligent systems are increasingly entering the healthcare realm to take the place of complex human interactions.

Various intelligent robots, often called robot companions or therapeutic robots, have been developed to serve as social partners to treat particular mental health problems. PARO, a robot designed to look like a baby harp seal, can provide the benefits of animal therapy to patients in settings where it might be inappropriate to have live animals [4]. PARO uses tactile, light, auditory, temperature, and posture sensors to monitor its surroundings, feel when it is being held or stroked, and recognize the sound and direction of a voice speaking its name, greetings, or praise. It can also respond to these stimuli through sounds and movement, mimicking the behavior of a live animal. Similarly, humanoid robots that can interact intelligently with people are rapidly becoming more advanced, and can be introduced into domestic environments to

interact socially with patients. One of the most advanced and commercially available humanoid companion robots is Pepper by SoftBank Robotics [5]. With a plethora of sensors, Pepper can navigate the surrounding environment, recognize faces, and converse with humans. Furthermore, it can identify a person's emotional state based on voice, facial expressions, body movements, and words spoken, and will adapt its behavior accordingly. It also learns about the preferences, tastes, and habits of its owners through interactions. While the effectiveness of robotic companions in treating mental health disorders has not been studied thoroughly, case studies suggest that they can have positive effects for patients with autism spectrum disorders, age-related cognitive decline, and psychosocial disorders [6].

Robotics and artificial intelligence promise to become an increasingly present feature of the healthcare field, as considerable research efforts are pushing toward more advanced medical robotic technologies. One specific sector of healthcare into which intelligent technologies are poised to move is simple patient interaction in hospital and clinical settings. Healthcare providers have many tasks ranging from initial patient evaluations, researching medical literature, documentation, and more. Robotic assistants can perform tasks that are routine and require dense information mining so that healthcare providers can focus on the more personal and intricate aspects of patient care.

Given routine tasks in healthcare such as standard initial interviews and documentation and the prevalence of autonomous healthcare tools, we wanted to explore developing a robot that makes use of existing software and technologies through cloud computing. The addition of cloud computing could provide even more powerful intelligence and information processing. This lead

us to our research question; how can cloud computing be integrated with on-board processing to create a robot to assist with healthcare delivery?

Building upon recent developments in medical robotics technology, Team SCOPE has developed a robotic system that incorporates various components of artificial intelligence to perform simple patient interaction tasks in a clinical setting, dubbed NELI[1]. Team SCOPE's goal was to create a decentralized artificial intelligence system that is easily adaptable for different applications. The distributed model approach is defined by compartmentalizing the different functionalities of the robot into separate systems. Our decentralized model, which we termed as our "modular design model", is a hybrid of onboard and cloud processing, with some functions performed on remote servers. In the case of this project, the functions include visual object recognition, natural language interaction, and physical actuation (mobility). Each of these functions are integrated and managed by a main program. Functions that require advanced computational technologies can be implemented through cloud services. For example, tasks such as object recognition require machine learning capabilities which are very costly to implement locally in terms of computing power. However, in our system, the functionality can be performed on dedicated remote computational servers. By having the object recognition function in a separate system, it can be run through cloud services while other systems remain local.

Another advantage of the modular design model is the ability to easily adapt the robot through updates of individual components. Components can be interchanged, added, or removed to update the functionality of the whole system, with minimal maintenance necessary. For

[1] The name "NELI" is inspired from the Natural Language Interaction (NLI) component.

example, a new natural language interface can be easily added by updating the natural language cloud service. The main integration program would not necessarily need to be updated, and could provide the functionality of the new natural language interface with minimal changes.

This project focused on creating this hybrid-processing platform for medical applications. By applying this logic, the individual components can be updated for any possible application with ease. The robustness of the main integration program is an important factor, as it provides the bulk of the intelligence for NELI. Without the integration program, the intelligence within the individual components will not be able to perform any useful tasks. For the purposes of this project, the intelligence of NELI is the measure of its ability to interpret and act on the information received by the component functionalities.

Team SCOPE executed NELI by focusing on the visual object recognition, natural language interaction, and mobility functions as independent components in the modular design modular that can functions completely on their own outside of the robot implementation. Each component will be detailed throughout this paper as a subproject with an analysis on the technologies used, the components connection with the cloud computing platform, and how the components contributed to NELI's assistive robotic function.

The natural language interface is supplemented by a human avatar and all components are all tied together using a main integration program. Together, NELI can be analogized to a human assistant: the eyes and mouth are the visual and recognition and natural language interface, while the hands, legs and face are the mobility and avatar components. The main integration program is thus the brain, controlling each component. Each component provides

information to the brain, and the brain can output information and actions through these components as well. The system structure as described is visualized in Figure 1. The arrows connecting the components show how information will flow between them. For example, there is a two-way arrow between the Integration and Visual Object Recognition components. The main program will first request a visual search from the visual recognition services. The Object Recognition component will return search results back to the Integration program, which will be able to interpret the results.

Team SCOPE has created an assistive robot for healthcare delivery. The robot is mobile, responds to spoken commands, and demonstrated AI. It extracts meanings about the patient's health from conversations and visual interactions. It summarizes these observations into reports that could be merged with the patient's (EHRs). This process aids healthcare professionals in delivering better care by augmenting attendance, increasing accuracy of patient information collection, aiding in diagnosis, streamlining data collection, and automating the process of ingesting and incorporating this information into EHR systems. SCOPE's solution uses cloud-based AI services along with local processing. Using VEX Robotics parts and an Arduino microcontroller, SCOPE created a mobile platform for the robot. The robotic platform implements basic motions and obstacle avoidance. These separate systems are integrated using a Java master program, Node-Red, and IBM Watson cloud services. The resulting AI can be expanded for different applications within healthcare delivery.

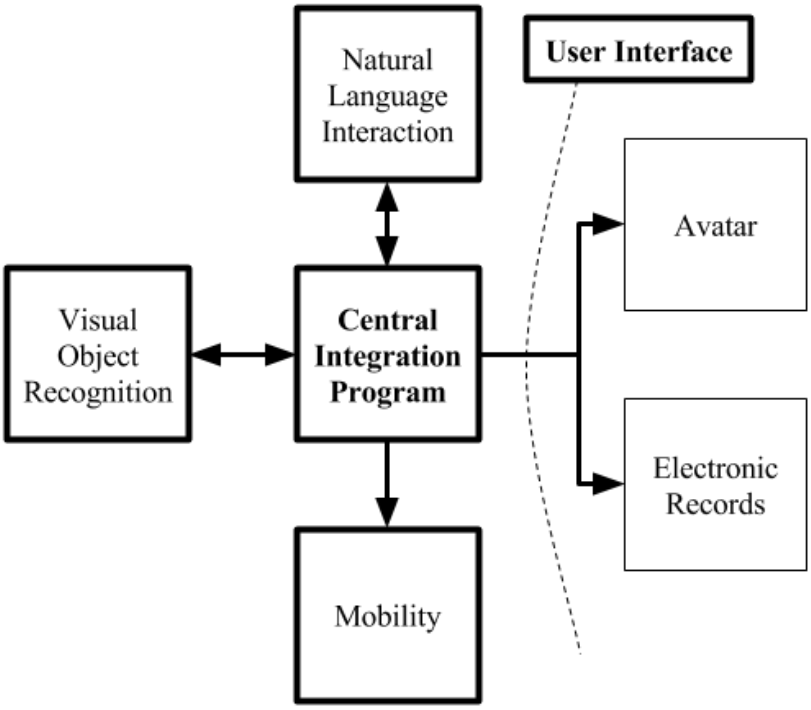


Figure 1 System structure of NELI. The direction of arrows represent the flow of information between the components

Chapter 2: Visual Object Recognition

2.1 Overview

The goal of the visual object recognition component is to identify people and prominent objects in the view of the robot. The subteam chose to use a cloud service to identify any object that the robot encounters rather than creating our own library that we would have had to train ourselves. This decision greatly reduces the algorithmic complexity of the object recognition process, without reducing effectiveness.

2.2 Watson Object Recognition

Team SCOPE's object recognition design leverages IBM Watson's powerful Visual Recognition service to identify objects seen in images taken by our robot. Since we wanted to be able to identify unknown objects and scenes, we had to go with a large cloud based service. Watson's Visual Recognition service uses deep learning, a machine learning technique that uses multiple levels of neural nets, to determine information about images. This object recognition service sends a series of keywords along with probabilistic weights assigned to a given image [7]. We chose IBM Watson over visual recognition services from other cloud platforms, like Microsoft Azure or Google Cloud Platform, for ease of integration with the other components of our design. Using a cloud service inherently introduces a lag in our response time and a dependence on internet access. The response time varies based on the internet connectivity and image size, but is not excessive.

IBM Watson's Visual Recognition attempts to find keywords that describe the entire

image, not individual objects in the image. For example, a picture of a room is classified as a "painting" or "landscape," instead of identifying the objects in the room. The service is constantly improving, of note to our subject area, Watson has learned to identify ambulances, hypodermic needles, pill bottles, and surgical instruments during the course of our project.

Team SCOPE solves this issue by using image segmentation. We identify regions of the image that correspond to individual objects and then send those individual parts up to IBM Watson's Visual Recognition. By this way, image segmentation is used to split up an image and get images of individual objects, which can then be identified by Watson.

Initial attempts at image segmentation involved using edge detection, color grouping, key point identification and heat maps. However, while these segmentation methods did outline and identify the objects in the scene, we could not find algorithms to effectively split up the image into images of the individual objects to send to Watson. For example, edge detection effectively outlines the objects in the image, but it is impossible to distinguish where the edge of one object ended and the edge of another object began. Similar problems occur with the other segmentation methods.

To solve this problem, we introduce the idea of super-pixels. We divide up the image into larger blocks called super-pixels. Then, we average the RGB values of the colors in each super-pixel to make each super-pixel one color. Afterwards, we use color distances to group adjacent super-pixels of similar colors. These groups are based on both similar adjacent colors and similarity to the color of the starting super-pixel. The groups are formed using a depth-first search algorithm. Starting from the top-left and going to the bottom-right, super-pixels that were

not grouped were used as a starting point for the depth-first search.

From there, the color of adjacent super-pixels that were not grouped yet are compared against the starting super-pixel. They are grouped with the starting super-pixel if the colors are a certain distance away from each other. Then, the adjacent super-pixels to the super-pixels that were grouped are then considered to be grouped. The process continues until there are no more super-pixels that are similar in color to the adjacent, grouped super-pixels or there are no more adjacent super-pixels that are within a certain distance from the color of the starting super-pixel. After forming these groups, the largest groups are chosen and a bounding rectangle is drawn around the group. These rectangles are sent to IBM Watson to be identified. This approach works for landscapes and images where the objects are all similar colors, but fails for objects of multiple colors.

For example, it can identify a lake in a landscape, but cannot identify a dog in another picture. Also, the thresholds for the size of the super-pixels, the minimum distance between colors for two super-pixels to be grouped and the minimum number of super-pixels in a group for the group to be sent to IBM Watson are hard to determine and vary based on the image. Thus, this approach does not work for objects that consist of multiple, dissimilar colors and the parameters for the algorithm are hard to identify and quantify.

After struggling with segmentation based entirely on colors, we decided to enhance our capabilities by combining the color image with a depth map generated by a Microsoft Kinect V2, a sensor that can take color and infrared depth images. We use the depth data to identify key regions in the color image and these smaller images can be more effectively processed by

Watson. This process helps us identify multiple objects in a single image captured by our robot. Using both the depth map and the color image provides information about the relative location of the identified objects.

2.3 Microsoft Kinect

We chose to use the Kinect V2 as our depth sensor because it is very economical for its numerous and unique functionalities. The Kinect has a 1080p 30Hz color camera as well as a 512 x 424 30Hz Depth Sensor, which allows us to identify key regions in the color image [8]. An IEEE report noted that "[in] summary, as a depth sensor with a relatively low price (much lower than the professional depth cameras or tracking systems like Vicon), Kinect for Windows sensor V2 shows acceptable performance" [9]. In addition to the depth data, the Kinect also provides a high definition camera. The Kinect can recognize up to six individuals and track two bodies, their associated joints, hands, and faces. Kinect can also determine traits of the users it is tracking, such as whether a user is happy, engaged, or wearing glasses, has his/her eyes closed or mouth open or moving, or is looking away [8].

The Kinect does not operate with the same accuracy at all ranges, angles, surfaces, and conditions. The Kinect's depth accuracy is within 2 mm directly in front of it. This means that the center of the image is the optimal zone for segmentation. At longer ranges and around the edge of the image, this accuracy is reduced. The Kinect also has reduced performance on reflective and IR-absorbing materials, like mirrors and computer monitors [9].

2.4 Segmentation Description

We begin by acquiring the initial depth data, which is saved as a matrix with the data in each position corresponding to the distance of that pixel from the capture plane. We found that our algorithm has difficulty detecting the edge between an object and the floor. This issue was solved by setting the depth data from the floor to zeros. We identified the floor plane using the Kinect's floor clip plane function and compared points in the depth data to the floor plane. If the distance was below a threshold, we deleted it. This replaced all values in the floor plane of the depth data with a zero value, which allows us to detect edges between objects and the floor effectively. By removing the floor, we were able to find the edge between an object and the floor very accurately.

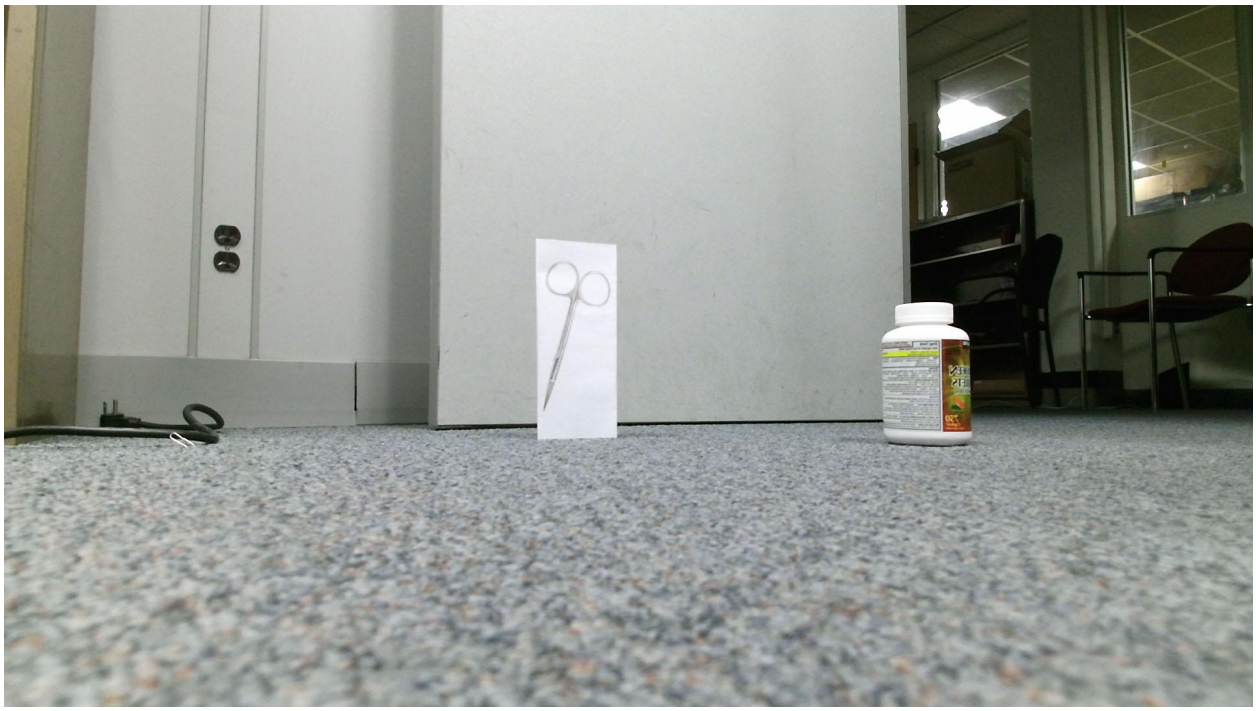


Figure 2 **Full color image from Kinect V2**

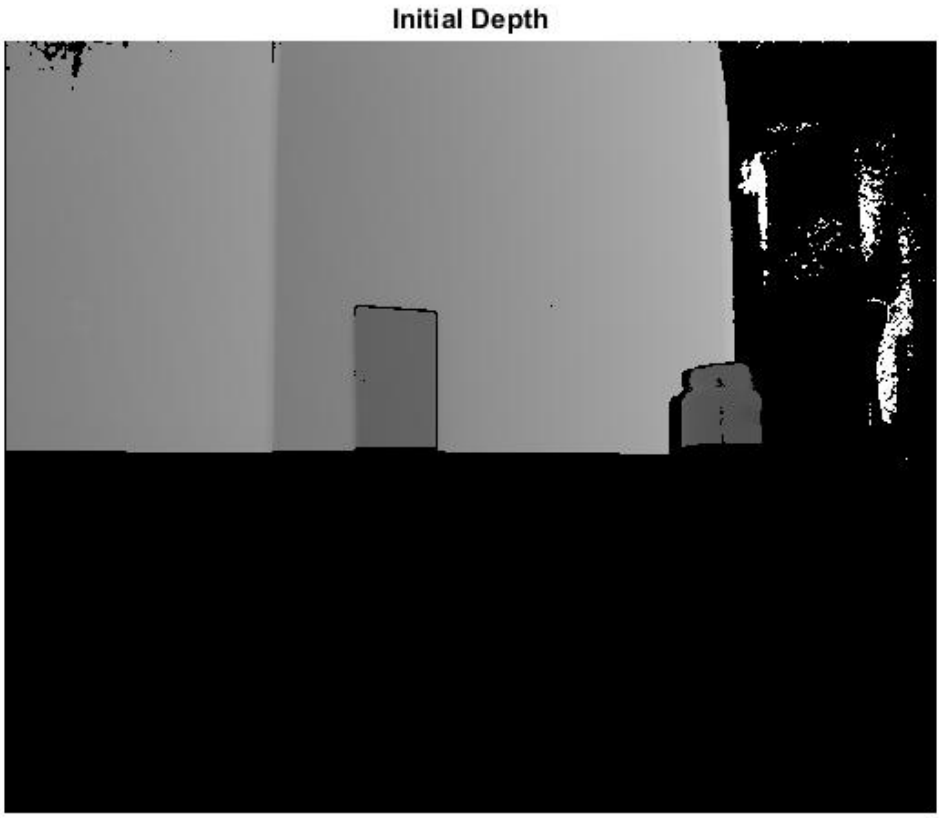


Figure 3 **Depth image from Kinect V2**

After getting the depth matrix, we perform real-time, local processing of the matrix using MATLAB. We began by cropping the exterior edges and regions that contained parts of the robot out of the matrix. The data becomes noisy around the exterior edges, especially when viewing long distances. We therefore remove the regions that contained the exterior edges from our depth matrix. We also remove the regions where our image captures certain parts of our robot. For obvious reasons, the robot does not need to identify itself. After removing those

regions, we removed data that was beyond a certain threshold. Only objects up to a certain range can be identified because the Kinect begins to lose accuracy beyond three meters [9]. Watson is more accurate with its search results when the sent image is larger, while the Kinect loses accuracy at longer range. Therefore, we can only identify images effectively up to a certain range.

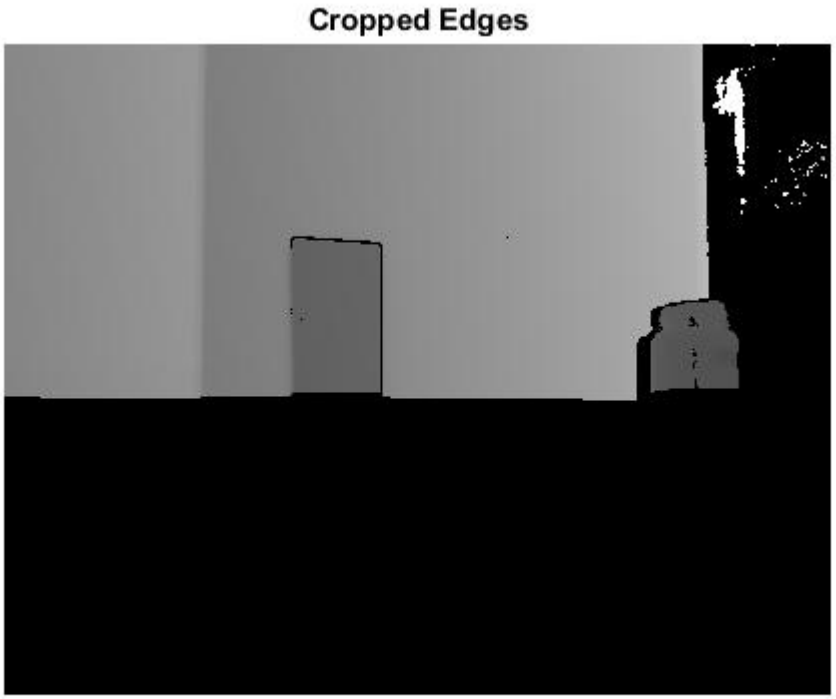


Figure 4 **Cropped edge image from Kinect V2**

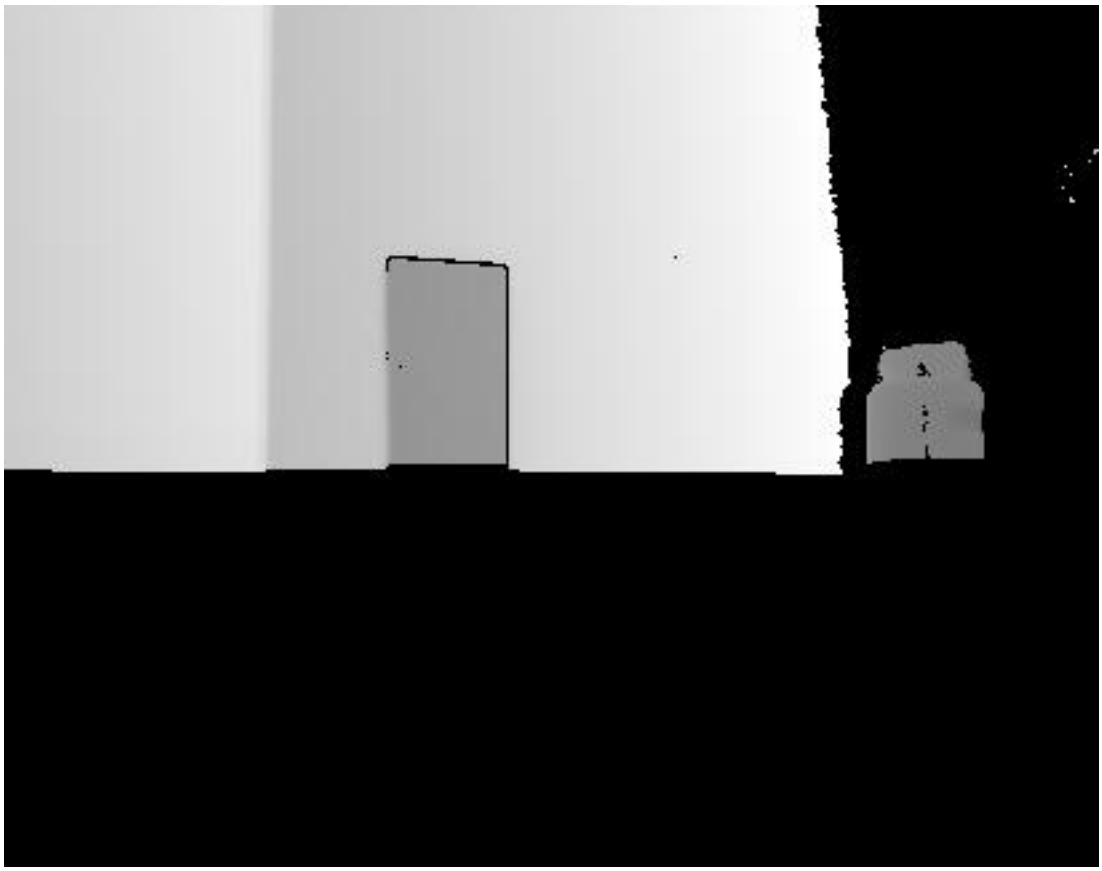


Figure 5 **Distance cropped from Kinect V2**

The next step is to identify edges of the object using a filter. MATLAB has two such filters, Canny and Sobel. The Canny filter identifies all the edges in an image, while the Sobel filter finds the exterior edges of objects. As the goal of our segmentation is to determine the

outline of objects, we went with the Sobel filter. This produces a binary matrix with edges having a value of one. We then apply morphological filters to the output in order to create solid region outlines in our depth matrix. Regions that are below a threshold size are removed.

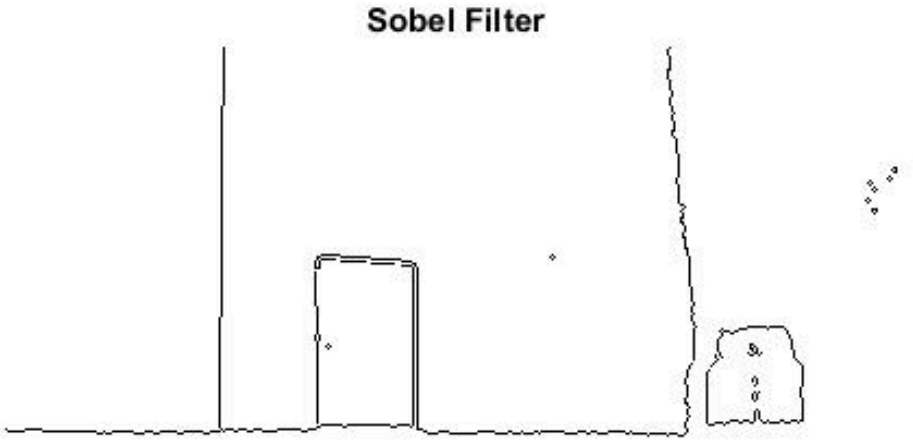


Figure 6 **Sobel filter applied to image**

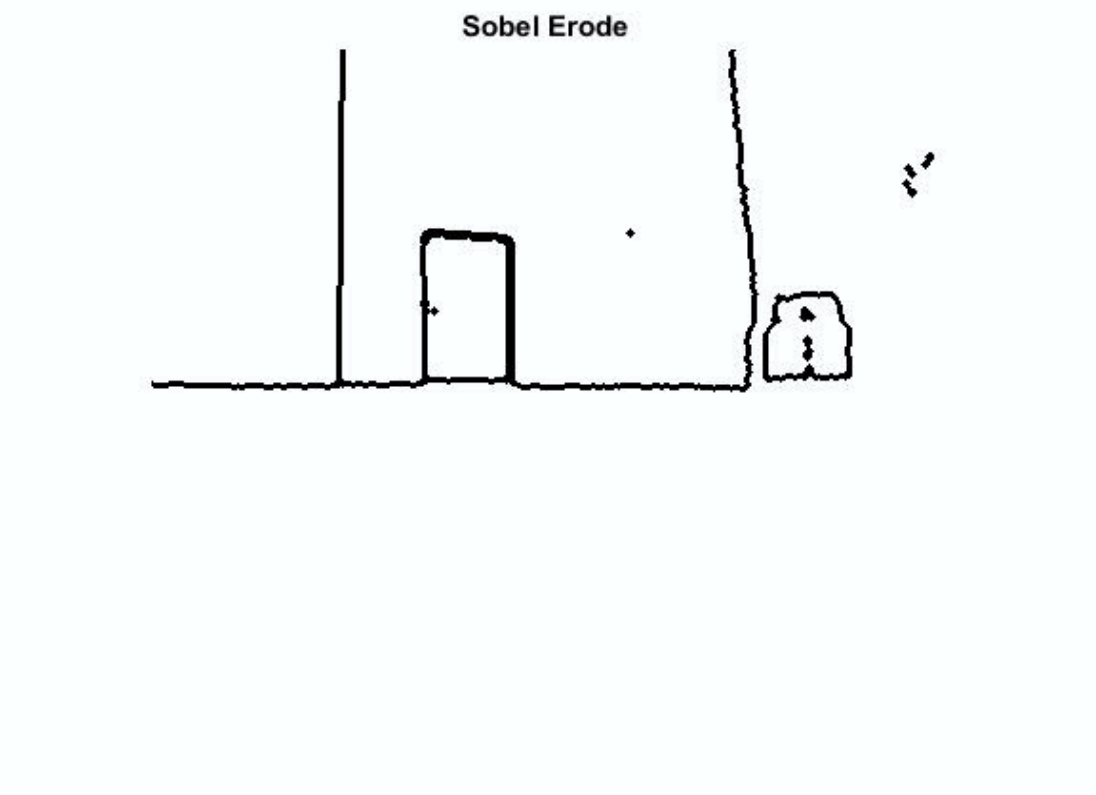


Figure 7 **Edges eroded**

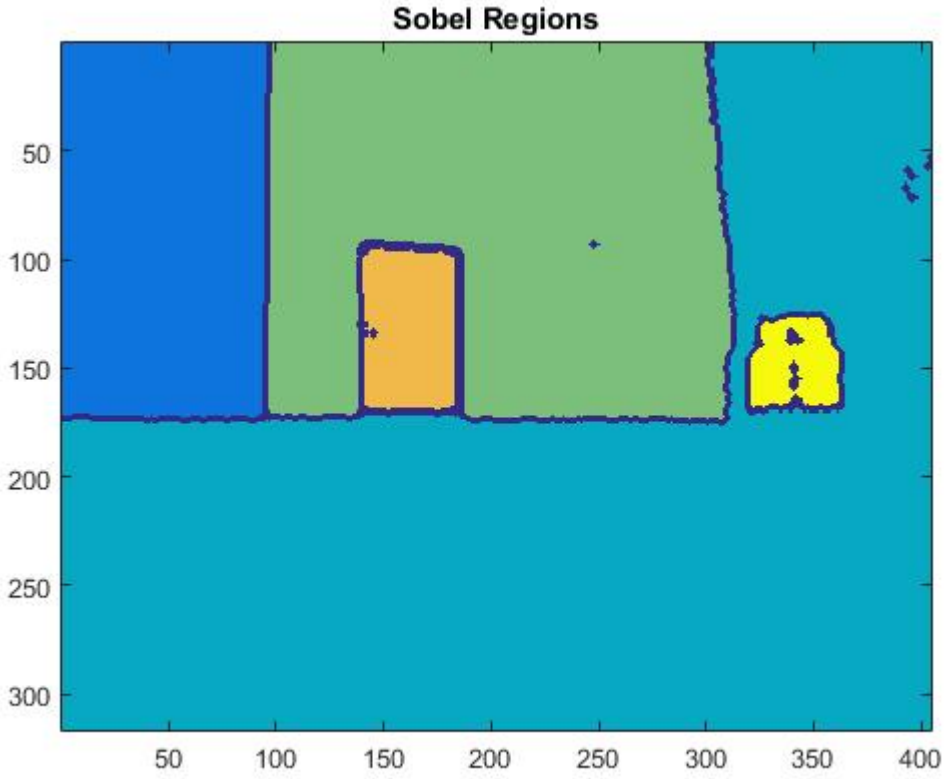


Figure 8 **Sobel Regions differentiated**

The next step is to determine the bounding boxes of the regions. We filter these results again to remove regions that encompass most of the image as they represent the background and removed regions rather than those we are looking for. We also group regions that are covered by multiple bounding boxes. We can then map these boxes to the color image and crop out sub-images. We are more interested in isolating the objects than finding their precise boundaries so we send a slightly larger image that still contains the full object. In our current implementation the segmentation process takes four seconds. This is primarily due to the overhead of starting

MATLAB.

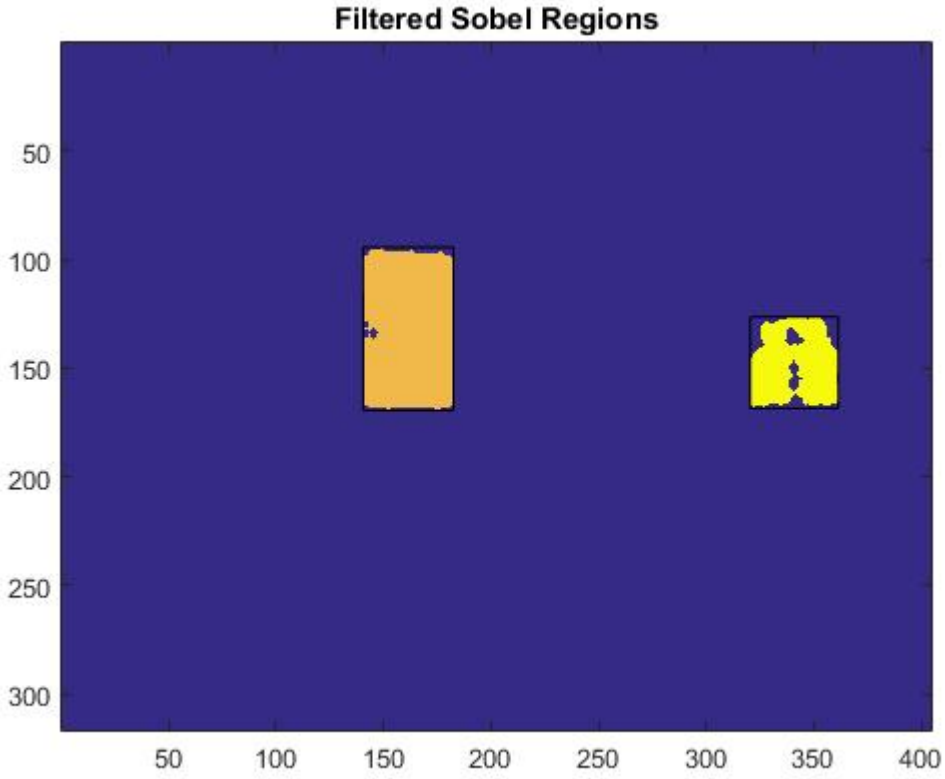


Figure 9 **Filtered Sobel Regions**

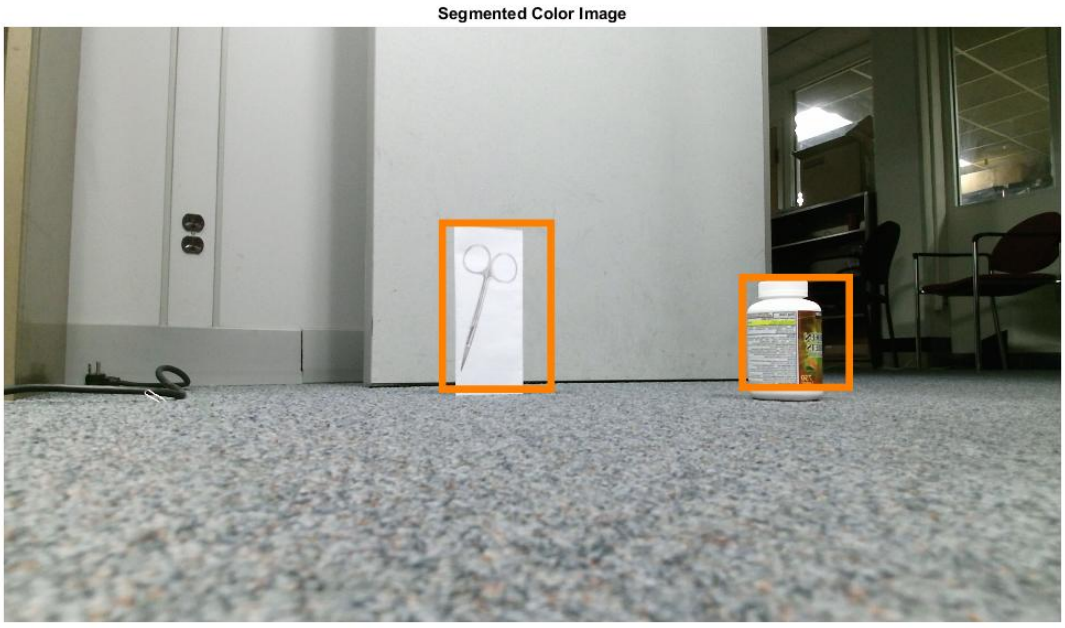


Figure 10 **Full color image segmented**

2.5 Testing

We tested the effectiveness of our system by analyzing a number of objects. We had three versions of each object, the original image, the image segmented by the MATLAB algorithm, and a manually cropped object, each one easier for Watson to identify. We sent the images to Watson in order and found the first one that correctly identified the object. A condensed example of a result is found in Appendix A. The results are summarized in Table 1. If Watson did not identify the object it is denoted as a failure. If it identifies the object the response and Watson's certainty are listed.

Image	Original	MATLAB Segmentation	Manually Cropped
Microwave	Appliance .691	Microwave .829	
Computer Monitor	Monitor .707	Television .846	
Lamp	Fail	Lamp .536	
Scissors	Fail	Scissors .683	
Pill Bottle	Fail	Pill Bottle .759	
Doctor(2D)	Fail	Doctor .721	
Ambulance(2D)	Fail	Fail	Ambulance .989

Table 1 Identification Results Summary

The results confirm that the Watson Vision Recognition Service is more effective at identifying segmented images. We also see that the image segmented by the MATLAB algorithm does segment the images appropriately. Our sample size is too small to make any sweeping declarations, but it does confirm the effectiveness of our system of local depth segmentation and cloud based object identification. Improvements to our sensors, segmentation algorithm, and object recognition services will enhance the performance of our system. Watson's Visual Recognition service has and will continue to improve. The Kinect is a very effective sensor, but it can be replaced by more accurate and higher resolution cameras and depth sensors. Improved segmentation is the focus of an entire field of study.

2.6 Future Work

As healthcare professionals may go from room-to-room for various reasons including meeting with patients, it can be beneficial for an assistive robot to be able to follow an individual. The Kinect skeleton-tracking algorithm involves a two-stage process: computing the

depth map and inferring the body position with trained machine learning. By using the projection of a known array of light from the infrared emitter, the Kinect determines depth values and then uses a randomized decision forest algorithm to infer the location of specific body parts. One joint in particular, the spine base, can be used to track the position of individuals as the orientation of the individual can be disregarded. Because detected skeletons also include tracking identification numbers, even bodies that leave the frame for short periods of time can be identified with the same tracking number. To follow the tracked individual, coordinates of the single specified joint must be continuously obtained. For simplicity, conditions for the robot to move forward and laterally are independent. The robot moves forward if the tracked joint exceeds a set following distance, and if the x-coordinate of that joint falls beyond a certain threshold, a command to move left or right is appropriately issued.

Chapter 3: Natural Language Interaction

3.1 Introduction to NLI

Natural Language Interaction (NLI) is the field of study that focuses on the interactions between a human and computer, using human language as a method of communication [10]. An NLI interface is used to create a more realistic conversation environment between humans and technology. NLI interfaces provide a more natural and intuitive way to interact with complex technologies, reducing the technical burden on the human user.

For this project, an NLI chatbot was developed as a method for a human to control the robot. As this robot is intended for use in healthcare, natural language provides the most comfortable and efficient method for communication. In our research, we developed NELI using IBM Watson cloud services to collect initial patient information, as well as provide the robot with commands. It provides intuitive control for the mobility, object recognition, and healthcare tools that, combined, are used for basic level medical care. The emotional connection between doctor and patient is an important aspect of healthcare. Often, this connection is removed when humans are replaced with technology. The NLI program facilitates this connection in an inherently emotionless system. The program will not replace the necessary patient-doctor interactions, but enhance them by providing a groundwork of information to allow for more focused conversation.

In addition to the aspect of comfort, it is also imperative that the NLI can extract the relevant information from the patient accurately and efficiently. The recording of patient symptoms is a relatively simple task. However, the NLI conversation should not be a time-

consuming process that collects irrelevant information. Subsequent sections will discuss previous studies regarding effective practices in collecting patient information, which informed our design of the NLI system.

3.1.1 Existing NLI Technologies

NLI is a rapidly developing field that looks to allow humans to interact with computer-based systems in the same way they would interact with other humans. Most research surrounding NLI has focused on creating advanced systems for database querying through either spoken or written natural language [11]. Early research efforts allowed users to type a sentence which would be analyzed the NLI system and used to retrieve desired information from an underlying database. Further research has aimed to broaden the range of possible query types for such a database and enhance the range of possible responses that can be provided by the system [12].

In recent years, NLI interfaces have been incorporated into various existing technologies. Many mobile electronics such as cell phones and tablets now feature virtual personal assistants. Well-known examples of this include Apple's Siri, Microsoft's Cortana, Amazon's Alexa, and Samsung's Bixby. These personal assistants essentially serve as natural language interfaces for interacting with and controlling the various facets of the mobile device. Most of the processing for these NLI systems takes place via cloud technologies and proprietary software. These systems cannot be incorporated into our project because of a lack of customization. At this point in time, the proprietary nature of these commercial offerings means that it is difficult for developers to repurpose them for different applications. In the future, with expanded developer

support, it may be useful to reevaluate these systems for this project.

NLI has been incorporated into a variety of systems to allow more efficient interaction between humans and machines. Many of these are robotic systems with the ability to perform mechanical motions or work, specifically within contexts that require some human input during these actions. Robot control through NLI has been simulated and demonstrated through systems that map spoken language onto a finite set of robot commands such as move, pick up an object, or request clarification [13]. In simple cases that involve robots with few capabilities, the NLI interface can also be made relatively uncomplicated, as it must be able to understand language that relates to each of its capabilities but can disregard other words.

Mobile robots present a particularly interesting challenge when NLI is incorporated due to the largely unrestricted nature of movement-related commands. It is possible to have a simple yet comprehensive NLI system for this application (i.e. a mobile robot could be fully controlled by a small set of movement commands). However, it is often desirable for the robot to develop an understanding of its surroundings in order to effectively navigate. While this can be largely accomplished through various sensors, NLI may also play an important role in teaching a mobile robot about its environment. Such a system has been demonstrated, using a natural language analyzer to collect spoken information describing objects in the surrounding environment and the desired route of travel for the robot [14]. By storing relevant information about the environment and path relative to objects within the environment, the robot can use its perception system to make decisions about its movements.

Demonstrated NLI systems are fundamentally limited in their interaction capabilities

when compared to humans. A typical NLI system will only be able to decipher language that can be fit into a predefined set of grammars and utilizes vocabulary from a known library [13]. However, under these limitations it is possible to create an NLI interface that is sufficient for the purposes of basic patient interaction.

Some companies have made NLI and voice recognition services available through cloud-based APIs. For example, Teneo is an NLI development and analytics platform by Artificial Solutions that allows businesses to incorporate a human-like conversational agent into online services with which customers interact. Furthermore, Teneo analyzes natural language data such as customer conversations to provide insight into the customer's sentiments and thoughts [15]. Similarly, Nuance Communications, Inc. offers a voice recognition software package called Dragon NaturallySpeaking, with SDKs to allow incorporation of voice recognition into various platforms [16]. Furthermore, IBM offers a variety of cloud-based API services for NLI development. These are based on IBM Watson, and will be discussed in detail later.

3.1.2 Applications of NLI in Healthcare

Within the field of healthcare, new technologies have shown utility in the gathering of patient information and diagnostic capabilities. In previous studies, tablet computers have been used to gather patient history upon entrance to the hospital. Patients answer a series of routine questions that help both patients and physicians by streamlining the conversation towards a chief complaint. The information collected is presented to the healthcare worker before the patient is seen in person. In healthcare, there is a constant difficulty balancing the quality of care provided, comprehensive documentation, and reduction of the length of patients' visits [17]. In these

studies, patients report that inputting symptoms and medical history to an automated system does not hinder the quality of care received. In fact, some report that the process assists with organization of thoughts and symptoms before seeing the physician [18]. The automated system answers a variety of questions the patients have about their symptoms. Importantly, the diagnosis process is performed later by the physician, based on the information recorded by the system. However, there is exploratory research into automating the diagnosis process. At the University of Texas MD Anderson Cancer Center, a prototype system that uses IBM Watson technology has been trialed for basic patient diagnosis. The system uses a vast database of past studies and patient records to uncover connections between patients and cancer diagnoses [19]. The technology was never used to treat actual patients at the Cancer Center and was halted by a failure of integration with the medical record system at Texas MD Anderson Cancer Center [20]. Development of Watson services for diagnostic tools and integration with outside databases is still an active area of research.

3.1.2 Choosing a Technology

Prior to deciding on IBM Watson as our NLI platform, we also considered using lower-level services, such as the Natural Language Toolkit (NLTK) and Wit.ai. The desired attributes of a technology were ease of use during development and integration options. To implement the component modular design of NELI, the technology chosen must be easily integrated into the main integration program. For these reasons, the current version of NELI relies solely on Watson services for its NLI interface.

NLTK

The Natural Language Toolkit (NLTK) is an extensive set of software and data for natural language processing applications developed at the University of Pennsylvania [21]. The software is open-source and written in Python, and is designed primarily as a teaching tool for natural language processing courses at the undergraduate and graduate level. As a result, development of NLI applications using NLTK functionality requires relatively low-level coding. The primary function we could have used from NLTK is the part-of-speech tagger, which must be trained using a pre-tagged corpus. This led to the first of a few limitations, as NLTK does not contain a pre-tagged corpus that incorporated commands (e.g. “Go to the door.”), so use of the part-of-speech tagger required development of custom training sets to allow for interpretation of commands.

The second helpful functionality provided by NLTK is a parser. This required development of a context-free grammar (CFG) to represent interpretable sentences, which created a slightly limited set of sentences that the NLI system could interpret (for example, incomplete or incorrectly formed sentences might not be parsed correctly by a formal CFG). Once a sentence is parsed, meaning the NLI system has determined which words correspond to which components of the sentence structure, NLTK offers little helpful functionality for our purposes. This is primarily because NLTK is designed for typical NLP purposes, such as analyzing large amounts of text of a particular form, usually to the point where basic conclusions can be drawn (e.g. categorization of text). Our NLI system requires a deep analysis of a single sentence, drawing all information from the sentence and either storing or responding to each

piece of information contained. This portion of the NLI functionality required hard coding of all analysis following the parse. Because of these limitations, NLTK was ultimately not incorporated in the current version of NELI. We decided to look further into options with more developed capabilities in interpretation and analysis of sentences.

Wit.ai

Wit.ai is the first high-level natural language development kit that was considered. Unlike NLTK, Wit.ai is a developer-targeted service that focuses on quick and easy integration into apps. As such, the service provides an easy-to-use GUI for definitions of intents and entities. Intents describe the purpose of a sentence, such as performing a specified task. This is determined by sentence structures, as well as entities in the sentence. Entities are defined keywords that the conversation is looking for in a sentence. This could be nouns of interest, or action words with verbs. Conversations, which are chains of user inputs and responses, are presented in a visually logical manner. The largest differentiator of this service is its ease of use.

Wit.ai provides a significantly different logic structure than NLTK. Text entry is collected locally, and processing is performed remotely in a cloud server. For the most part, only entity and intent detection logic is housed remotely. Once an entity is detected, the server follows the predefined program flow and executes local code. Importantly, the majority of the logic is defined in a local Node.js program. This program running on the local machine decides the actions to be taken by adding and removing “tokens”. Tokens are used by the cloud server to determine the context of a conversation, and where the conversation is to proceed.

A feature not seen in other cloud services is the ability to train the conversation system

through the inputting of examples. In the Wit.ai interface, one can enter sample sentences into the training system to create a backend of examples that massively increases the robustness of the entity detection algorithm. Other services require much more involved methods for creating a similar database. While Wit.ai proved to be easy to setup and robust, the service was ultimately not used because of problems with integration.

The local Node.js component would require additional integration tools that greatly increases the complexity of the integration step. For this reason, Wit.ai was deemed unsuitable for the project, as the integration step is a vital part of the project. Other than not fulfilling the specific requirements of this project, Wit.ai is a very solid NLP service that has many potential applications.

IBM Watson

Bluemix Watson services from IBM provide similar functionality to Wit.ai, but all the logic is run remotely on a cloud server. The flow of the conversation can be defined and controlled with minimal coding, which works well as a prototyping method. Within Bluemix, the Watson service allowed us to build various dialogs using tree-like diagrams, within a graphical user interface.

Watson is a very high-level cloud computing service that provides intuitive access to IBM's Watson Artificial Intelligence computational engine. IBM's provided API gives a very simple method of using Watson's natural language capabilities with minimal amounts of coding required. It is able to accomplish this by housing all required logic remotely in the cloud. The local component only sends message prompts and receives responses from the server. In

addition, because the Watson conversation service is part of the suite of Bluemix services from IBM, there are robust tools that can be used in the integration step. The advantages and methodology of integration will be discussed in future sections.

The provided online interface provides methods for defining entities and intents. The cloud computation engine automatically interprets natural language inputs for intents and extracts entities. Synonym detection is also easy to define. Unfortunately, Watson does not provide any backend database for synonyms of words and phrases. This means that many variations of phrases must be provided for entity detection to be robust enough for real life applications.

Dialog flows in Watson Conversations are created in a logic tree structure composed of nodes. Each node in the tree represents a response, which is triggered when an entity is detected in the user's response. At every node, specified responses are defined and presented as the response of NELI. The responses can be made conditional based on specific inputs of entity values.

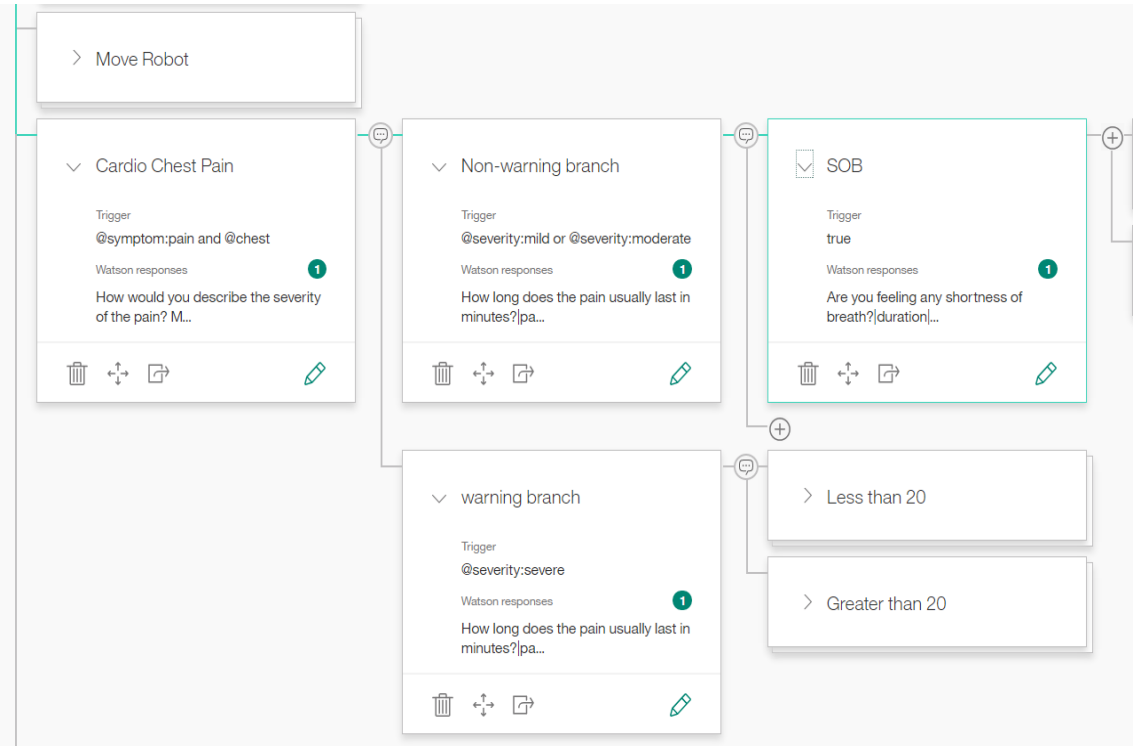


Figure 11 **Sample interface for dialog in the IBM Watson Conversation Service**

Figure 11 shows an example of a dialog flow created in Watson. Each rectangular box is a node, which is triggered by detection of a specific entity. The console interface was designed for the purposes of increasing ease of use, which presented some challenges for the purposes of this project.

A downside of using Watson is the lack of precise control over the dialog flow. As the service only provides a very high level interface, the conversation entirely depends on how Watson detects entities and intents. Unlike Wit.ai, there is no way of manually defining flow logic with code. Through trials of the service, Watson detects entities using a very simple search algorithm, which can potentially create unforeseen errors. However, if entities and synonyms are defined robustly, this should not prove to be a problem.

In addition to the lack of control, Watson also provides no analytical or statistical tools at all. For testing purposes, this provides the challenge of finding a way to accurately measure the success rate of the NLI program. On the same note, there is no automated method of training the program through trial inputs, as available in Wit.ai. This means that all possible synonyms and variations of phrasings must be predetermined during construction of the NLI program.

In the end, the Watson conversation service was chosen to create our implementation of the NLI program for NELI. The service focuses on ease of use, which made it easier to design and create conversation flows that are well-suited for this project. In addition, the tools for integration that come with IBM Bluemix was an important fact that was taken into consideration. In order for Team SCOPE to successfully integrate all of the functionalities for NELI, the integration process for a technology was deemed a priority.

3.2 Designing Dialogs Using Medical Literature

The NLP capabilities of Watson were demonstrated for a clinical application. The aim was not only to create a diagnostic program, but one that incorporates information filtering and data presentation for use by a physician. Robots are designed to optimize the time spent on routine tasks, with the goal of allowing for more meaningful time spent in interactions requiring full human capabilities. In the healthcare setting, this means less time spent by the physician asking routine questions regarding symptoms and patient history. Instead, more time can be spent building a connection with the patient, as well as producing a more in-depth diagnosis.

The most effective clinical conversations occur when both patient and doctor are contributing and asking questions equally. But a review of clinical communication studies found

that patients usually contribute little to the patient-physician interaction [22]. The primary finding was that patients do not ask many questions, which the review attributes to difference in education level. In order to encourage a time-efficient two-sided conversation, NELI can be used to screen for the routine symptoms and patient history. The questions asked can be related to commonly occurring chief complaints in general practice medicine. NELI can initially ask for basic symptom information, such as length of time, pain characterization, and severity. Based on the patient's responses, NELI can ask related questions that a physician may be interested in knowing. As such, the program can present a more comprehensive report to the physician for the actual patient visit.

Dialogs were created to collect information that could be useful in a primary care office prior to the patient seeing a physician. Due to time and complexity constraints of the project, the dialogs were centered on a very limited number of chief complaints but asked questions to create a detailed understanding of the symptoms surrounding these specific complaints. This could be expanded easily to accommodate complaints involving many body systems. We chose cardiac and respiratory complaints with an emphasis on chest pain. NELI holds a human-like conversation that begins with the patient mentioning his/her chief complaint. The chief complaint will guide the questions that are asked during the conversation.

After the conversation has concluded and all necessary patient information has been collected, a diagram of symptoms and patient history is created for the physician to examine before or during the face-to-face visit with the patient. We will discuss the generation of this diagram, or summary chart, in later sections. In addition to the general symptoms, the data

recording system also labels potential critical symptoms. The critical symptoms flag is triggered by any severe health warnings that are noted in the conversation. This can be useful in increasing the patient's priority to be seen.

3.2.1 Dialog Contents

Due to the breadth and complexity of patient-physician dialogues, only two dialogs were created, both relating to the chief complaint of chest pain. Chest pain is a prevalent topic in medical literature, and the associated symptoms are well-established. After reviewing literature regarding the etiologies of chest pain, dialogs for the cardiac and respiratory branches were created. Both branches are initialized with the same trigger chief complaint.

Due to the critical nature of cardiac chest pain, the NLI program defaults to first traversing the cardiac-based dialog. This dialog asks questions relating to symptoms that could suggest a diagnosis related to cardiac chest pain, such as severity of chest pain, duration of symptoms, palpitations, and shortness of breath. These symptoms have been listed as critical warning flags for cardiac chest pain and are often indicative of an emergency situation [23]. The dialog was formatted, as in Figure 12, with yes/no prompts and decision graph formats as used by previous medical dialogues and interview software [24][25].

After studying the parsing capabilities of the Watson Conversation service, we integrated questions with a yes/no response format with more flexible response structures. The cardiac responses that suggest potentially life-threatening symptoms are concentrated towards the front of the cardiac dialog. Further questions targeting cardiac symptoms were gathered from review articles on patient interviews that asked for common symptoms, such as palpitations, related to

heart disease [26][27]. Given that the natural language portion of the robot is focused on asking a set of questions to get the most pertinent information about the patient's chief complaint for the doctor to review before coming into the room, the dialogue front-loads the questions related to critical conditions to help rule out any emergency situations earlier in the visit.

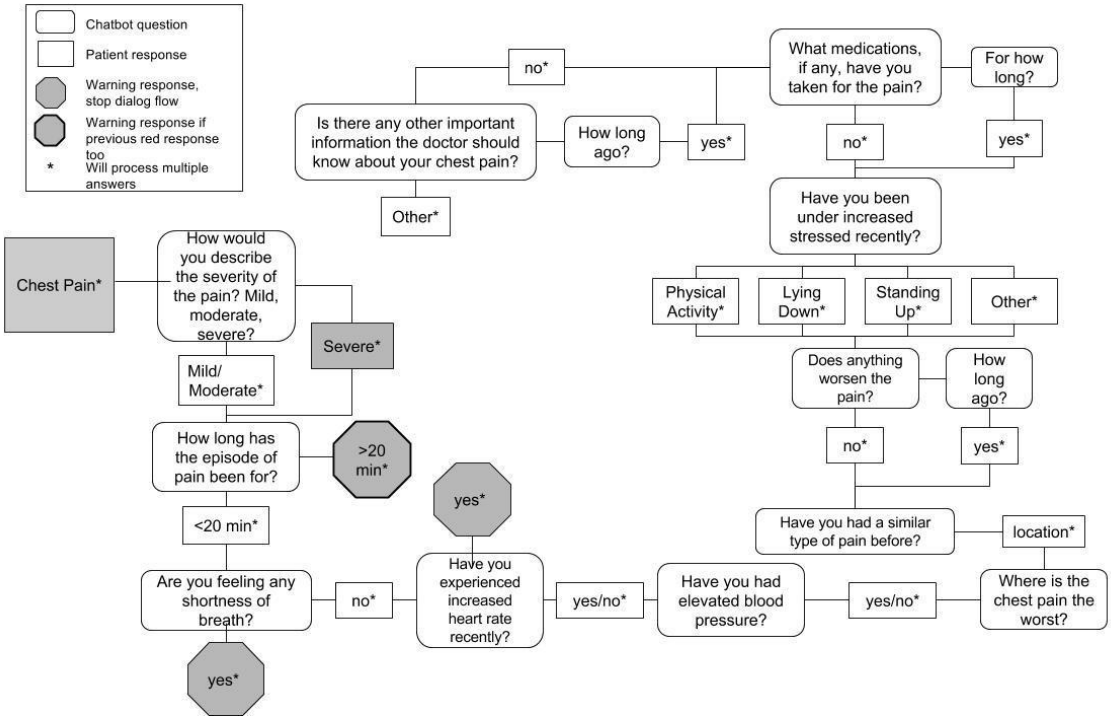


Figure 12 Dialog flow used in Watson's conversation service to process cardiovascular chief complaint. Flow diagram created with algorithms and routine symptoms relating to past medical and NLP literature

For practical purposes, not all symptoms related to cardiac etiology of chest pain were included, since the overall goal of the dialog is to streamline the beginning of the conversation. The dialogues created focus only on commonly asked questions regarding prevalent symptoms for chest pain noticed in the literature review. Analysis of program function and accuracy, as detailed in the testing section of the thesis, will dictate exact questions and topics included in

future iterations of the NLP program. The topic of the dialogues is flexible based on the overall goal of the robot.

Additionally, the program aims to minimize the length of the conversation by asking the most relevant questions based on statements made by the patient throughout the conversation. A ranking of critical symptoms and any other general symptoms for the respiratory dialog was developed from NIH references on warning signs of respiratory failure and common symptoms surveyed in clinical evaluations of airway obstruction [28][29]. The respiratory-based chest pain dialog asks the patient questions with the premise that their chest pain is not due to a cardiac etiology. Some of the questions later in the dialogue, such as exacerbation of pain are present in both dialogues.

Given that cardiac chest pain has more critical symptoms in the dialogue, we start the NLI flow with that dialogue. If the patient answers negatively to a sequence of at least three questions following the critical symptom list, the flow switches to the respiratory symptom dialog. This is done with the goal of not asking additional questions relating to cardiac symptoms if the patient's complaint may be related to a different organ system. The logic of switching between various etiologies in the form of different dialogues is the first step we tried to implement in increasing the efficiency of asking the correct questions for comprehensive triage summary documentation to provide to the physician when he/she enters the patient's room.

3.2.2 Inputting Dialogs into IBM Watson

Inputting the dialog branches into the Watson Conversation service required the creation of entities. For each response that the program can provide, there must be a trigger that has to be

predefined. The entities range from simple "yes/no" responses, to more informational responses, such as "severe/mild". To ensure NELI is robust in its ability to interpret patients' complaints, entities are given lists of possible synonyms. The lists of synonyms were defined manually. For example, the entity `@pain` can be detected when the user response contains words such as "hurt", "burn", "tightness", along with all the other grammatical tenses. Some examples of entities that were used are displayed in the table below.

Entity Name	Examples	Explanation
`@pain`	Hurt, burn, tightness	Used to identify that the patient is experiencing pain or discomfort of some sort
`@severity`	Mild, not bad, moderate, bearable, severe, pretty bad	Used to determine the severity of the patient's symptoms (such as pain)
`@chest`	Heart, pectoral, upper torso	Any general term that could be used to identity the chest region of the body

Table 2 Entity List

3.3 Testing Methodology for Dialogs in NLI Program

Once dialogs were fully formatted using the Watson Conversation service and connected through the robot's integration program, the NLI program was tested for expected question flow and syntactic flexibility. Team SCOPE simulated numerous patient interactions with the NLI system by collecting hypothetical responses to all of the questions that would be asked by the robot.

Surveys to collect these hypothetical patient responses were sent out to University of Maryland students, as specified in a project review with the University's Institutional Review Board. The survey can be found in Appendix D. All students responding did so electronically through a Google form that presented the hypothetical situation of them being a patient with

chest pain. Instructions prompted questions to be filled out within 10 minutes, in order to simulate normal response times to such questions. Survey takers were not given any indication on how to answer questions regarding symptoms other than imagining that they had chest pain. No sick patients or students with known chest pain symptoms were requested to participate in the surveys. The identity of all students taking the survey was kept anonymous to our research team, given that only the responses to the Google form and time of submission was made available to those processing the data. A total of fifty survey responses were collected for analysis.

The set of hypothetical responses from each survey was used as a single beta test of the NLI system. The students surveyed did not interact with the robot directly, rather members of the NLI research team input the answers as written in the survey into the conversation with the robot. The results of each beta test were presented in a summary chart produced by the integration program following the input of answers to all questions or the location in the conversation where the program could not process a response. The results were used to quantify the flexibility of the program in understanding a conversation held with a variety of people using different language to explain their chief complaint. The results also allowed the NLI research team to improve the prototype model based on errors noticed through program failures.

Once all the survey responses were collected and formatted, the testing process was automated to create each corresponding program output. The content of each survey response was individually stored in a file with a standardized format. A testing program then read the responses from the file, which were fed as inputs into natural language program. After the conversation finished, a screening result table is generated, recording the processed responses

from each survey taker and dialog flow that the conversation took. The table is recorded, serving as a summary chart for a physician, and used as a part of the testing analysis. The summary charts produced by the automated testing program format symptoms based on cardiac or respiratory causes and condense answers based on yes/no answers, responses regarding duration of symptoms, and general note-format responses.

After each test case was run, the conversation flow was manually inspected to find any errors that may have occurred. As a side effect of the way user responses were stored, the automated testing program sometimes had to be manually adjusted for different conversation branches that may have been taken. These adjustments were only made so that the testing program correctly inputted the appropriate user response for each question presented by NELI. Occasionally, the natural language program would encounter an error with the user responses. Such mistakes include spelling errors, formatting issues and other easily addressed issues. In these cases, the user responses were manually corrected to ensure that the program is able to finish execution. When correcting these errors, the meaning of the response as well as the syntax were kept the same, to ensure testing integrity. Our goal was ultimately to test the flexibility of the NLI program, so it was in our best interest to keep the responses as natural as possible.

3.4 Results of NLI Testing

3.4.1 Analysis of Overall NLI Program Completion Rate

The two questions we aimed to answer with the analysis of the analyzed data were how accurately the NLI program executes conversation and records responses based on a created

dialogue flow and how the program handles a variety of syntactic responses to the questions that it asks patients. In order to assess if the NLI system acted as expected, the program execution was tested for a number of errors that occurred and the resulting summary chart of symptoms for each dialog was inspected for accuracy.

Out of all the survey responses run through the NLI program, 84% completed full processing of the hypothetical conversation without error, and created appropriate summary charts based on the conversation (Figure 13). The other 16% of survey response sets resulted in error produced by the natural language program failing to run to completion. The exact patient responses that resulted in errors were recorded and compared qualitatively to find trends, and subsequently used to understand how the NLI system could be improved, especially with regard to flexibility and robustness. Many of the question types that resulted in errors when processing the survey responses were binary questions, which we define as questions expecting a solely "yes" or "no" type response. The error-generating responses were grouped into one of three groups: (1) answering binary questions with details, (2) answering binary questions with time or relative time period, and (3) responding with a non-binary affirmative answer.

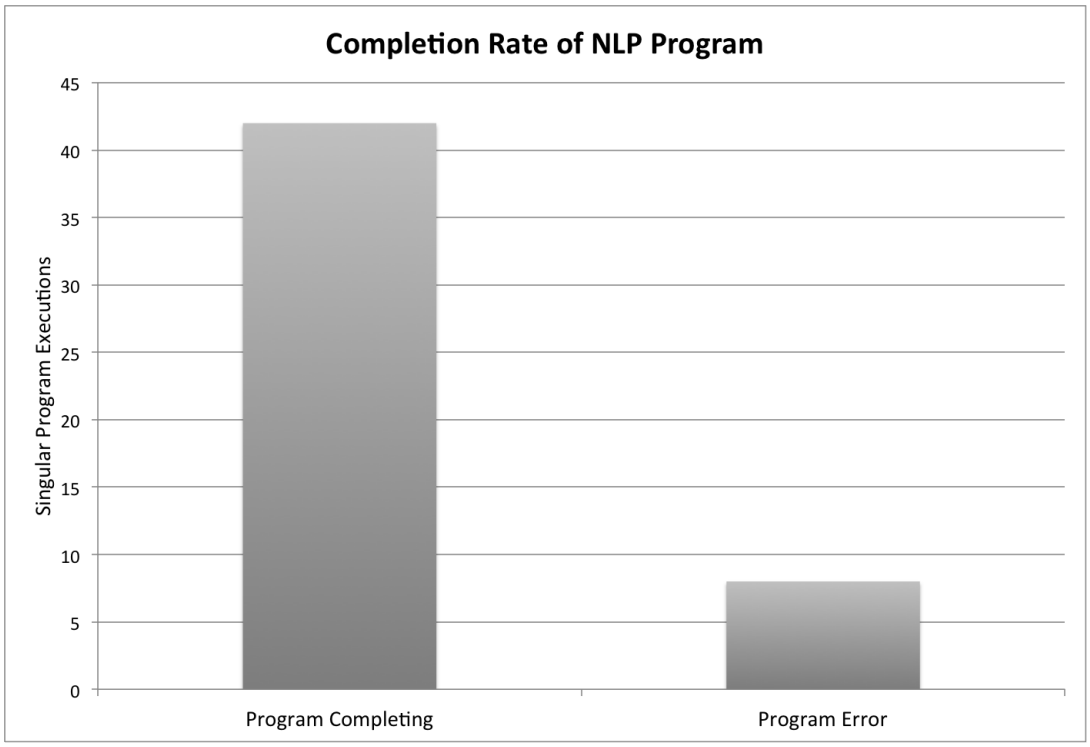


Figure 13 NLP Program runs resulting in successful completions versus error preventing processing of all dialog responses. Sample size for completion rate was 50 survey responses and 84% resulted in successful program completions

Out of the survey responses resulting in an error during program execution, the most common cause was answering a binary question with a time or relative time period (Figure 14). For example, this includes answering the question "Have you ever had this type of pain before?" with the response "First time". The errors resulting from answering with details or non-binary affirmative answers occurred with equal frequencies. An example of answering with details includes responding to the questions "Are you feeling any shortness of breath?" with "During the pain". An example of a non-binary affirmative answers is answering the same question with "every now and then".

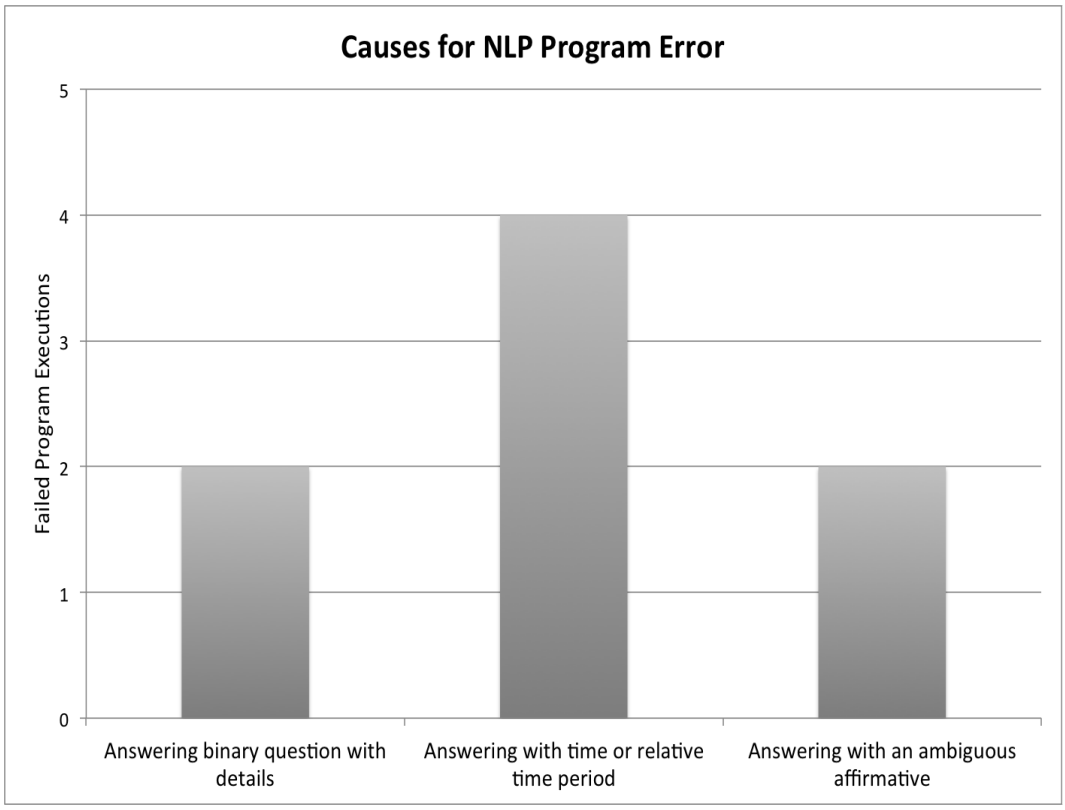


Figure 14 Causes of NLP error in program execution. Out of 8 runs resulting in errors upon program execution, the responses that cause errors were grouped into three general categories based on a qualitative analysis.

3.4.2 Analysis of NLP program responses

To evaluate the accuracy of the NLI system's operation with varying testing survey responses and syntactic flexibility, Team SCOPE compared the expected summary chart that should have been produced from survey responses and the summary chart produced by the NLI system. Examples of the summary charts produced by the program can be seen below for patients providing dialogues with error in program processing versus a successfully processed chief complaint of chest pain appearing to come from respiratory etiology (Figure 15). The summary charts produced by the robot after the hypothetical patient conversations were

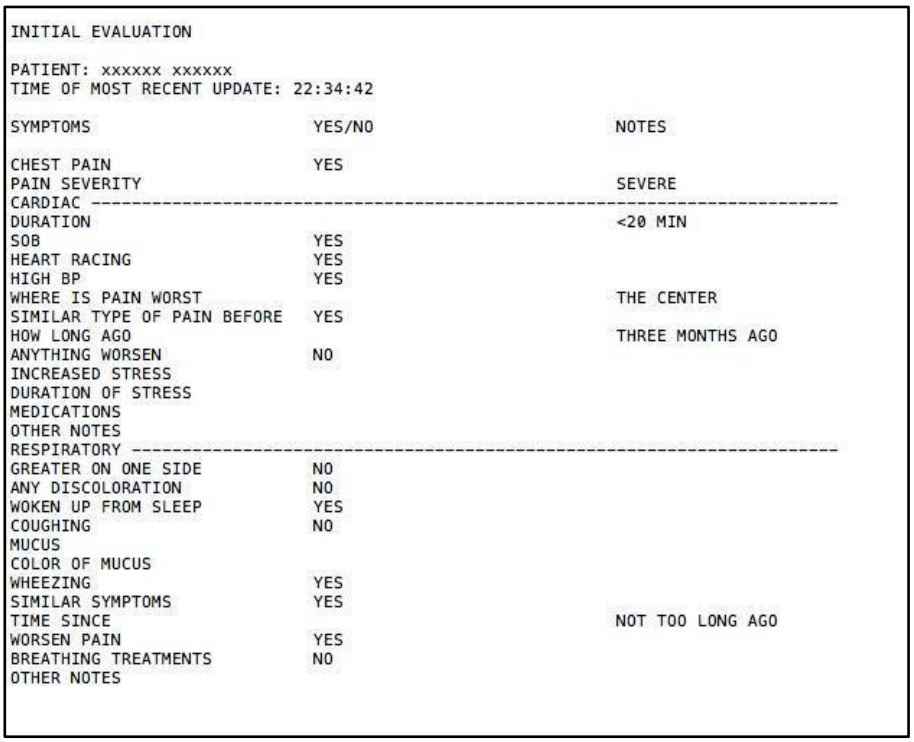
compared to the expected chart and categorized into three groups: (1) correct summary charts, (2) incorrect but complete summary charts, and (3) incomplete summary charts (generally caused by an error that stops program execution). Summary chart layout depicts shortened responses or commentary for physicians to digest prior to each patient encounter.

```
INITIAL EVALUATION

PATIENT: xxxxxx xxxxxx
TIME OF MOST RECENT UPDATE: 13:26:45

SYMPTOMS                    YES/NO                    NOTES

CHEST PAIN                  YES
PAIN SEVERITY                                         MILD
CARDIAC ------------------------------------------------------------------------
DURATION                                              5 TO 10 MINUTES
SOB
HEART RACING
HIGH BP
WHERE IS PAIN WORST
SIMILAR TYPE OF PAIN BEFORE
HOW LONG AGO
ANYTHING WORSEN
INCREASED STRESS
DURATION OF STRESS
MEDICATIONS
OTHER NOTES
RESPIRATORY --------------------------------------------------------------------
GREATER ON ONE SIDE
ANY DISCOLORATION
WOKEN UP FROM SLEEP
COUGHING
MUCUS
COLOR OF MUCUS
WHEEZING
SIMILAR SYMPTOMS
TIME SINCE
WORSEN PAIN
BREATHING TREATMENTS
OTHER NOTES
```



```
INITIAL EVALUATION

PATIENT: xxxxxx xxxxxx
TIME OF MOST RECENT UPDATE: 22:34:42

SYMPTOMS                    YES/NO                  NOTES

CHEST PAIN                  YES
PAIN SEVERITY                                       SEVERE
CARDIAC ------------------------------------------------------------------------
DURATION                                            <20 MIN
SOB                         YES
HEART RACING                YES
HIGH BP                     YES
WHERE IS PAIN WORST                                 THE CENTER
SIMILAR TYPE OF PAIN BEFORE YES
HOW LONG AGO                                        THREE MONTHS AGO
ANYTHING WORSEN             NO
INCREASED STRESS
DURATION OF STRESS
MEDICATIONS
OTHER NOTES
RESPIRATORY ---------------------------------------------------------------------
GREATER ON ONE SIDE         NO
ANY DISCOLORATION           NO
WOKEN UP FROM SLEEP         YES
COUGHING                    NO
MUCUS
COLOR OF MUCUS
WHEEZING                    YES
SIMILAR SYMPTOMS            YES
TIME SINCE                                          NOT TOO LONG AGO
WORSEN PAIN                 YES
BREATHING TREATMENTS        NO
OTHER NOTES
```

Figure 15 Summary charts of test patient/robot conversation produced automatically by NLP program. Top: Incomplete chart produced as a result of a program error at question about shortness of breath. Bottom: Complete summary chart traversing part of cardiac dialogue and full respiratory dialogue based on survey taker's responses.

Determination of accurate summary charts demonstrated accurate program function. Subteam members processed the expected result of each survey set by traversing each response through dialogues and constructing a summary chart through human analysis. This was completed independently of automated processing of the survey results and subsequent automated summary chart formation. The first criteria the NLI system aimed to satisfy was the ability to ask the most relevant questions based on the patient's responses.

To test this, Team SCOPE designed the NLI system to ask questions which would help determine the etiology (cardiac or respiratory) of reported chest pain, and choose a conversational path based on the answers to these questions. Because chest pain could stem from

a cardiac or respiratory-related cause, our definition of an efficient symptom-gathering program is that which asks the majority of simple history questions based on the most likely cause of the patient's chief complaint. This is opposed to asking the patient all questions pertaining to possible cardiac and respiratory symptoms.

Each person that participated in our anonymous survey could answer the questions from the NLP program with no constraints, except for answering the chief complaint prompt with some form of chest pain. Participants were not instructed on having a specific cause of chest pain and were not aware of the tests to be completed. From the responses gathered (n= 50), 18% of the hypothetical responses were expected to traverse the respiratory related question pathway and 82% were expected to traverse the cardiac related pathway. This pathway is entered after the front loaded critical symptoms questions about cardiac chest pain are asked. The NLP program is coded such that if the participant answers negatively to a series of non-critical questions, the program will deem a different etiology path for symptom questions.

Based on the created dialog flows, we traversed through each participant's survey responses and recorded the expected conversation flow (cardiac or respiratory) that was supposed to be taken by the program and recorded in the summary chart. Testing showed that 88% of program execution followed the expected symptom question path (Figure 16). Within the incorrectly traversed paths, all of them were cardiac-related chest pain complaints interpreted as respiratory-related chest pain.

During the process of determining the question path that was expected for each conversation, NLP members recorded what the correct output summary chart should be for each

hypothetical conversation. The correct output summary chart was based on human dialog analysis given strict traversal of the original dialog flows that were unchanged after program creation and execution. The human produced charts were compared against the automatically produced summary charts by the NLP program. Program executions amounted to 56% correct summary chart, with each symptom discussed during the conversation recorded as expected. Incorrectly recorded charts included common errors of incorrectly documented "other notes" section in the chart, incorrect recordings of symptom durations, and details for symptoms being recorded in the incorrect location on the chart.

The final assessment following question traversal accuracy and overall chart accuracy was chart completeness. Within the program execution, one of the ultimate goals is to ask a full set of symptom questions relating to one etiology that is likely to give a healthcare provider a comprehensive foundation for taking the patient's history. "Complete" is defined as having traversed and filled out a full cardiac or respiratory portion of the summary chart. Only one portion will be filled out for each conversation due to the design of the NLI program. Participants' answers are processed until the end as cardiac or deemed to be respiratory after a critical cardiac symptom questions. Within the testing executions, 76% of participant conversations produced complete charts for cardiac or respiratory chest pain. The other 24% of charts varied from being completely empty due to program error at the beginning of the conversation to missing summary responses for a few symptoms noted throughout the conversation. Incomplete charts were a result of individual symptoms not recorded or follow-up details for a symptom not being recorded.

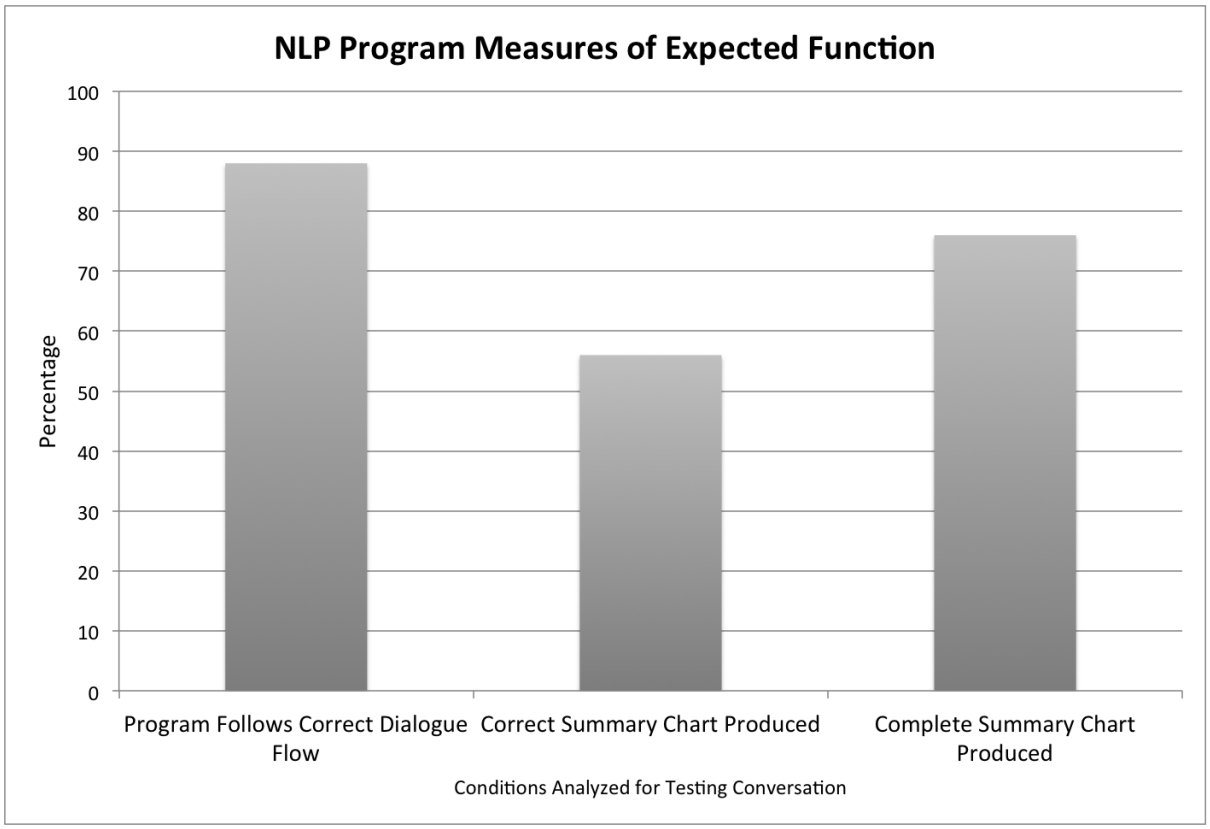


Figure 16 **Assessment of NLI program accuracy. Sample size of survey response sets used to test program accuracy was n=50. Out of those tests, 88% program runs followed the expected dialog path, 56% program runs produced the correct summary, and 76% program runs resulted in completed summary charts**

3.4.3 Limitations in Data Analysis

The program capability analysis completed for this version of NELI allows for error detection and improvement in future iteration, but is not sufficient for larger conclusions on patient interaction with medical software or human interaction with NLP programs. The population size of 50 was limited given that only a small portion of the total conversation set (16%) could be analyzed for program error and the set containing inaccurate charts (44%) was not large enough to formulate concrete trends on error-causing trends.

Testing participants who submitted conversation responses through survey are estimated

to all be within the 17-24 year old age range, given the sampling was completed through online university forums. Exact demographics of the population completing the survey responses used for testing is not known given the anonymity agreement in IRB documentation. Education level, age, and state of health are not able to be extrapolated to the general population that would use a robot like NELI in an assistive setting.

With the structure of the survey for testing participants, the dialog responses do not completely mimic real time conversation given time allotted for typing and ability to see questions as a set instead of processing one at a time. The general dialogs were created based on medical literature for this application, but dialogues created with guidance from professionals in the field may yield different results.

3.4.4 Future data analysis

For future iterations and development of the NLP program, the team would like to create a greater variety of testing dialogues with professionals in the field of NELI's application. To obtain more conclusive data on how to improve the processing power of the program, the team aims to sample a larger population consisting of ages across multiple generations, various education and health levels, and individuals with differing exposures to interactive technologies. The ultimate goal for testing and improving the program would be to have testing participants interact with the program as a part of the full robot in real time and gather automated responses from the program in addition to subjective responses of program use from testing participants. Human subject research approval would be required. The future analysis and testing for the NLP program would move from testing for errors to aid in program improvement to testing for total

program accuracy as well as patient's and physician's ease of use and overall satisfaction with NELI.

3.5 NLI Conclusions

Given accuracy and syntactic flexibility analysis, we were able to determine that the NLI program is able to process a conversation on a specific medical complaint and ask follow-up questions based on individualized concerns. In addition, the program also creates a concise form of data to present to a professional. Conclusions could not be made on the significance of these results given a small and concentrated testing population.

3.5.1 Findings

Despite the small quantity of failing test cases, trends could be gathered regarding phrasing and topics that caused failure; including but not limited to answers including affirmative and negative responses to binary questions and answering questions with details instead of affirmative or negative statements. Specific words and phrasing that resulted in program errors allow our program developers to modify and expand the intents and entities in Watson as necessary to minimize future program failure. The types of errors encountered in the program executions also inform the level of minimum breadth needed across intents and entities that are created with the Watson dialogues. The overarching goal of NLP programming is to create a robust system that can conduct and process the information in a conversation with any individual, so the errors and validation gathered in our testing allow for concrete changes to be made to a second version of the NLP program. We validated that the program can function

according to constructed dialog flows.

3.5.2 NLP Future Work

For more advanced versions of the program, we hope to incorporate certified terminology dictionaries to populate the entity and intent fields in the Watson conversation service. This will reduce the labor involved in having developers populate intent and entity fields based on results of beta testing as completed for the created NLP program in this project. Also, an automated connective to terminology dictionaries through certified databases or a collection of literature articles would allow for regular updates and training to the program's function without developer intervention.

To standardize the method of creating the dialogue flows and connecting them through an online conversation service for applications beyond clinical medicine, future work includes creating a standardized methodology to turn dialogue flows into conversation trees in online services. This include a simple interface with a dialog builder based on preset categories such as binary questions and various response categories. Creation of a dialog builder interface that can be systematically converted to an NLP program will expand possible applications and ease of creation for the NLP program. This would allow for professionals without extensive programming or technical background in computing services to create new NLP programs pertaining to a variety of topics.

Chapter 4: Mobility

4.1 Overview

In order to make NELI's visual object recognition and natural language interaction capabilities available for use as a medical assistant, it was necessary for Team SCOPE to establish mobility for the purposes of an autonomous robotic entity. In a hospital environment, many basic mobility modes are required, including maneuvering about and between rooms, following specific individuals such as doctors or patients, and moving particular objects such as medicines. These modes can be divided into basic motions which can be implemented via an Arduino microcontroller and the appropriate hardware.

4.1.1 Existing Applicable Technologies

Before beginning with the design of NELI's mobile robotic platform, an analysis of existing robotic solutions in hospital and medical fields was completed. A hospital environment presents a wide array of potential robotic tasks ranging from pharmaceutical retrieval to gathering current patient status, and it presents numerous challenges in terms of navigation and adaptability to the variety of potential tasks. This section explores several different existing robotic technologies that operate within medical environments. These existing technologies go beyond the current capabilities of NELI and represent some of the future directions that Team SCOPE can explore.

One of the most widely used mobile robotic platforms in hospitals is the TUG series of robots, manufactured by Aethon [30]. The TUG family focuses on performing delivery and

transportation tasks for a variety of applications. Hospitals are constantly moving materials, including medicines, supplies, and laboratory equipment between different rooms, floors, or even buildings. The TUG robots fill this void by automating the delivery of these materials, thus improving efficiency, and saving time for hospital personnel.

TUG robots operate in a variety of medical departments, including pharmacy, laboratory, nursing, food services, and environmental services. Within the nursing department, research has shown that nurses spend 30% of their time searching for medications, supplies, and lab results [31]. By limiting this time away from the patient's bedside, TUG robots improve the efficiency and quality of patient care provided by the hospital nursing staff. In the laboratory department, efficient and safe transportation is vital to the inner-workings of the hospital. Many lab specimens are time-sensitive or could be considered biohazards if they leak from their containers. The TUG robots eliminate the possibility of specimen leakage and have built-in safety protocol to assure that specimen are handled properly.

TUG utilizes a wide array of onboard sensors to navigate through hallways with crowds of people. These sensors include a pair of 180° laser sensors for object detection, and an array of sonar and infrared sensors to detect low-lying objects. For navigation, TUG is preloaded with a detailed map of the hospital it is working within, including pre-programmed routes. TUG keeps track of its location using an advanced odometry algorithm. Finally, TUG communicates with elevators and automatic doors by utilizing the hospital's existing WiFi services [32].

The Swisslog Robocourier is similar to the TUG in that it is also used for supply transportation in a hospital setting. However, the Robocourier is slightly less advanced in terms

of its mobility capabilities. The Robocourier is preloaded with a map of the facility, and each route is predetermined on-board when a request is given. The robot has a laser guidance sensor for basic obstacle evasion and to assure that navigation is accurate [33]. However, the Robocourier has limited ability to adjust its predetermined route for obstacle avoidance.

The InTouch RP-VITA is another popular mobile robotic platform used in hospitals. The RP-VITA is the first remote presence robotic solution for patient care, meaning that it acts as a physical entity allowing physicians to directly interact with patients from a remote location. The robot features an interface which allows a doctor to remotely conduct any relevant hospital task, including acute patient care.

The RP-VITA is the most mobile of InTouch's Remote Presence (RP) solutions. It is a tall robot, similar to the height of a human, and moves on wheels hidden under the robot's casing. The interface allows the remote doctor to control the movement of the robot to different rooms, as if the doctor were there themselves. A healthcare professional can also send the robot to any desired destination within the hospital. For navigation, the RP-VITA is equipped with a mapping system to handle movement while the remote care-provider is occupied and on-board sensors for obstacle avoidance [34].

4.2 Design

4.2.1 Constraints and Proof of Concept

As NELI's hardware was designed and built, a budget of $600 shared by the team could be utilized for parts and machining. Rather than having access to a large mechanical shop or lab,

NELI was constructed with hand tools in the same lab that the rest of the team used for coding and other components of the project. Thus, it was decided that a proof of concept would fit the requirements of the first iteration of NELI.

In order to complete a successful proof of concept, Team SCOPE had to trade off style and complexity for simplicity and basic functionality in terms of its mobile platform. That is, the four basic motions of moving forward, moving backward, turning left, and turning right were the first priorities for the robotic platform. These motions were perfected by using just two motor controllers, one for both wheels on each side of the robot.

4.2.2 Hardware

NELI is built on a VEX robotics 12” x 12” chassis with two wooden platforms. The first is mounted on the chassis, with the second constructed 10” above the body for additional hardware mounting, as shown in Figure 17 and Figure 18 below. The Arduino is set up on the lower platform along with the breadboards. The distance sensor is mounted on the front of the robotic chassis for collision prevention. The motor controllers are mounted on the underside of the lower platform, and are wired to the Arduino. A tablet stand is mounted on the lower platform and extends forward to hold the Kinect in a forward-facing orientation. The battery and remainder of the electronics are placed on the lower platform.

In addition, several hardware components were retooled from a previous version of Team SCOPE’s robotic platform named ALVORES, which stands for Autonomous Language and Visual Object Recognition External System. ALVORES used a Samsung tablet, which was mounted on top of the tablet stand facing the front of the robot, for the purposes of patient

interaction. The upper wooden platform was used as a mount for the Kinect, which could not be effectively manipulated. When designing NELI, both the tablet stand and upper wooden platform were repurposed in order to improve the functionality and general appearance of the robotic platform. The tablet stand was repurposed to hold the Microsoft Kinect, allowing for a wider and more manipulable range of view. The upper platform was retasked as a computer mount, so that a laptop or similar small computer, rather than a Samsung tablet, can be on-board for integrated processing with the artificial intelligence.

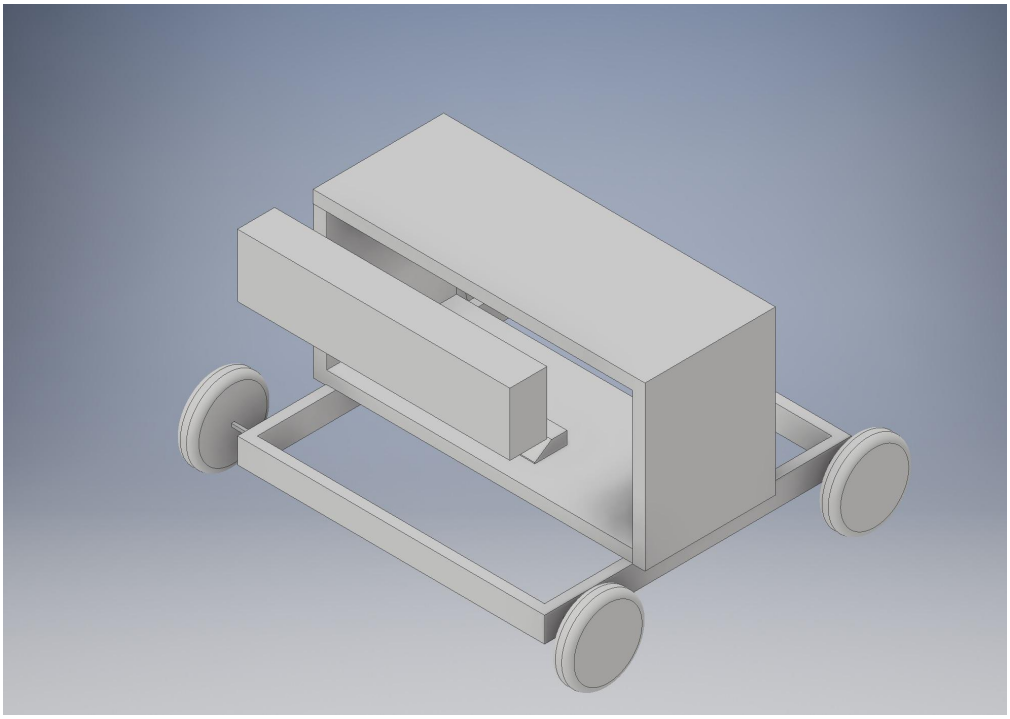
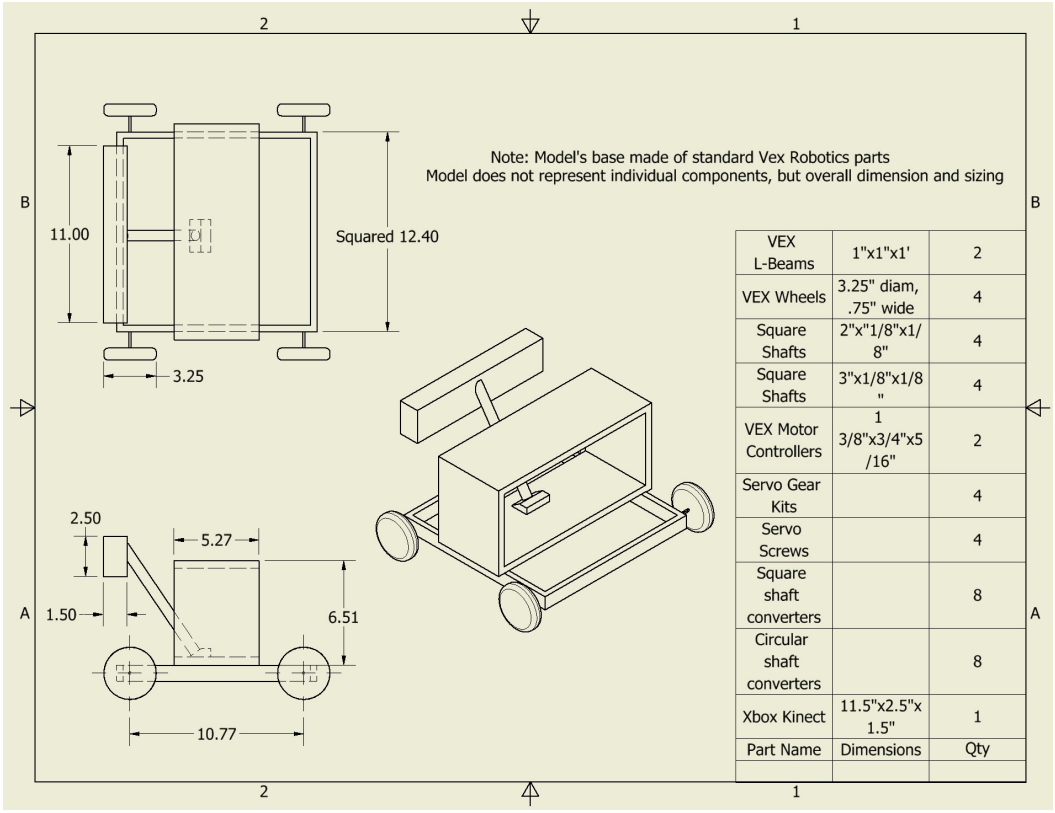


Figure 17 A basic CAD drawing of the current NELI design, featuring the Kinect mounted on a tablet stand in front of the two wooden platforms.



Part Name	Dimensions	Qty
VEX L-Beams	1"x1"x1'	2
VEX Wheels	3.25" diam, .75" wide	4
Square Shafts	2"x"1/8"x1/8"	4
Square Shafts	3"x1/8"x1/8"	4
VEX Motor Controllers	1 3/8"x3/4"x5/16"	2
Servo Gear Kits		4
Servo Screws		4
Square shaft converters		8
Circular shaft converters		8
Xbox Kinect	11.5"x2.5"x 1.5"	1

Figure 18 An engineering drawing showcasing views of NELI's structure from the top, side, and isometric views. This includes a bill of materials shown in the bottom right

4.2.3 Software

Team SCOPE uses an Arduino Uno as NELI's main microcontroller. The decision to use an Arduino was based on several factors, including the capabilities of the Arduino as well as the team's prior experience in using the Arduino for robotics purposes.

Firstly, the Arduino Uno has several input and output ports that are very important to NELI's functionality. A USB input port allows code to be uploaded from a team member's laptop and provides a power connection to the Arduino, allowing for a 5 Volt input voltage to the distance sensor. The Arduino Uno also provides 14 digital input/output pins, which can be used for receiving electrical signals from sensors or controlling motors and other actuators. Six of the

input/output pins have Pulse Width Modulation capabilities, a system by which the average voltage value is fed through the pin by switching the load on and off, allowing for signals of different durations to control the speed of NELI's various motors.

In addition, Arduino has its own software Integrated Development Environment (IDE) that allows users to implement programs on the microcontroller with relative ease. The Arduino IDE has a Servo library that is specifically designed for controlling servos and motors, which is particularly useful for Team SCOPE's purposes. This library includes functionality for attaching, detaching, reading from, and writing to a servo motor, which acts as a rotary actuator that can be set to a precise angle or speed. This allows for the control of a wide range of servo types, including both standard and continuous motion servos. For a standard servo, the library allows for the shaft to be positioned at any angle between 0 and 180 degrees. For a continuous motion servo, the Servo library allows the Arduino to set the rotation of the shaft to a particular speed, which is appropriate for controlling NELI's drive wheels. These factors particularly influenced Team SCOPE's choice to use an Arduino Uno for its main microcontroller.

For Team SCOPE's first robotic iteration, entitled ALVORES, a Raspberry Pi was to be used as a means of communication between Arduino and an Android tablet, which was the center processing unit for visual object recognition and natural language interaction. However, Team SCOPE had trouble connecting to the Arduino via the Raspberry Pi, and eventually eliminated the Raspberry Pi from the mobility design. A Raspberry Pi could also have been used in place of an Arduino as NELI's main microcontroller, but Team SCOPE's limited experience using the Pi made the Arduino Uno a more logical choice.

4.3 Arduino Code Overview

The Arduino code that Team SCOPE has developed to implement mobility functions is a simple finite state machine with up to eight possible states, as seen in Figure 19. Four of the states correspond to the four basic motions discussed in the previous sections. State 1 corresponds to forward movement, state 2 represents a left turn, state 3 is backward movement, and state 4 corresponds to a right turn. State 5 is the universal "Stop" state, which stops robotic movement upon command at any point. States 6 and 7 are available for controlling the orientation of the Microsoft Kinect. State 6 adjusts the Kinect's position about the z-axis, while state 7 adjusts the pitch of the Kinect. These states are not implemented in the current version of NELI due to hardware constraints. However, these states, as well as the mechanical infrastructure to implement Kinect motion, will be utilized in a future version of NELI. State 8 is general search state, designed for searching for a particular object in a room by spinning in circles and consistently taking pictures of the surroundings.

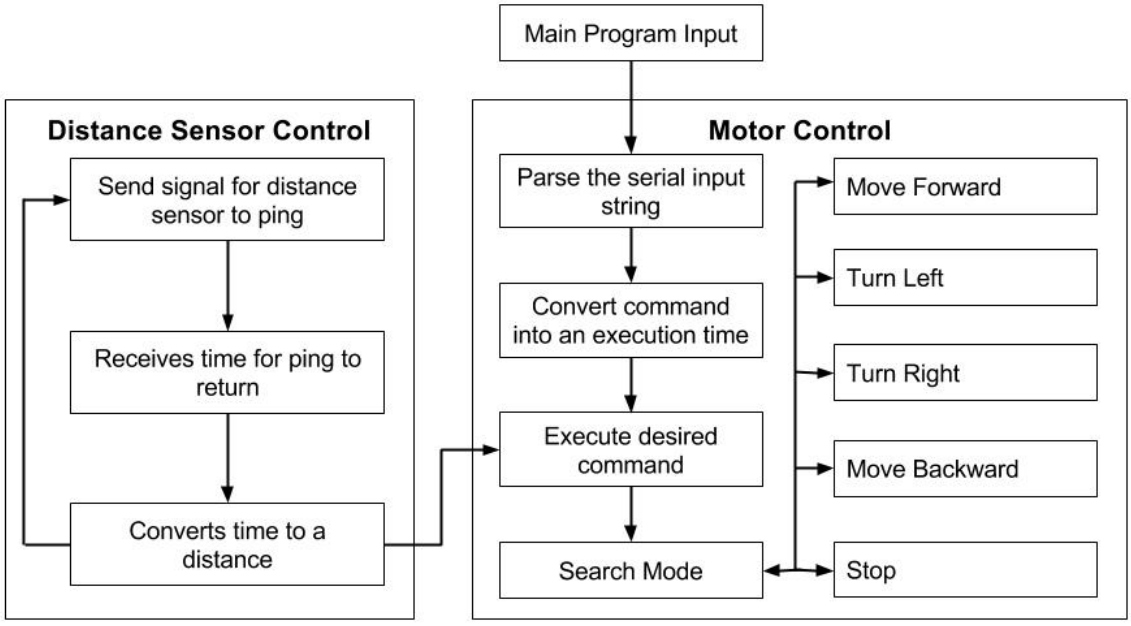


Figure 19 **Arduino Code Flow Chart. This flowchart separates the Arduino logic into two distinct functions. These include a distance sensor control function that is used throughout the Arduino code, and a motor control loop for reading in and executing the desired commands.**

The Arduino code traverses between these states according to a serial input coming from NELI's main program. The main program utilizes a Javascript that establishes serial communication with the Arduino, bypassing the Arduino IDE's built-in serial monitor. The Java program then sends a text (String) command of the following form:

x:yyyy

Here, *x* is a single integer representing one of the eight possible states implemented on the Arduino Uno. The variable *yyyy* is an integer number that will indicate the distance in inches to be traveled or the angle in degrees (in the case of turn left or turn right) of the desired movement.

In addition to the basic finite state machine implemented through the Arduino Uno, the

Arduino is programmed to control several key pieces of hardware, including motors and motor controllers made by VEX robotics, an ultrasonic distance sensor, and standard servos from Radio Shack. Within the loop function of the Arduino code, which runs continuously while a serial input is not received, the ultrasonic distance sensor is called to assure that NELI does not run into any walls, tables, chairs, or other common indoor obstacles. The distance sensor runs on a +5V input voltage and utilizes two input/output pins on the Arduino Uno. The TRIG pin on the distance sensor takes in a triggering signal from the Arduino. The input signal must be low, then brought high for at least 2 microseconds before returning to low. This triggers the distance sensor to send out an ultrasonic ping. When the ping is received, the sensor returns the time, in microseconds, it took for the ping to return to the sensor. The Arduino Uno reads this time from the sensor's ECHO pin. Then, the time is converted to tenths of an inch using the following equation:

$$distance = duration(\mu s) * (\frac{5}{74})$$

This equation is an approximation derived from the basic relationship between time, distance, and velocity as well as the accepted value of the speed of sound, approximately 340 m/s. Due to rounding conventions from the Arduino IDE, the Arduino will always round down to the next lowest unit of distance. Therefore, tenths of an inch are used to provide precision in distance sensing.

The distance sensor is also called in the serialEvent function within each of the eight Arduino states. When a serial input is received, the loop function is halted, which requires the distance sensor to be queried within the main function as well. This then caused a problem with

using the Arduino IDE's delay function when executing movements. During a delay, no other Arduino commands are processed. Therefore, the distance sensor could not be called during these periods of time, making it possible for NELI to crash into a wall or another obstacle. To eliminate this problem, the *millis()* command is used to keep track of time within the program. The millis command provides the time, in milliseconds, at any moment since the inception of the program. This allows the program to calculate time, turn the motors on and off, and query the distance sensor at any time.

The VEX robotics motors and motor controllers are also controlled through the Arduino Uno program. Each motor requires a +7.2V input voltage, which is provided by an external Lithium Ion battery. The motors are then controlled by a VEX motor controller, which utilizes a single input/output pin on the Arduino Uno. For the purposes of NELI, both wheels on each side of the robot are controlled by the same motor controller. Turning is simplified with this configuration, as the robotic platform can turn by having the wheels on opposite sides turn in opposite directions.

The motor controllers are controlled through pulses of differing durations. Every 20 milliseconds, the motor controllers expect to see a pulse of width between 1000 and 2000 microseconds. The middle of this range, 1500 microseconds, holds the motors in their current position or stops them from moving. Pulses of width less than 1500 microseconds turn the motors in the counterclockwise direction, while pulses greater than 1500 microseconds wide turn the motors clockwise. These inputs are given by the Arduino writeMicroseconds command, which writes the desired pulse width automatically every 20 milliseconds to each motor

controller.

4.4 Overall Performance

The first testing of the integrated mobility pieces was successful wherein the code passed along the necessary information to the different functional parts. While the control of the motion functioned properly, the mechanical parts themselves needed improvement in order to perform at the necessary standards, such as moving at reasonable speeds, and being able to turn. The original wheels chosen had too much friction and as a result, the robot moved at a slower pace than desired and could not turn on surfaces that were not completely flat and relatively frictionless. New wheels were chosen from VEX that had a much lower coefficient of friction and allowed the robot to move as desired. These wheels also have smaller wheels built-in, which roll sideways with respect to the forward direction of motion. This allows for omnidirectional movement, and makes turning NELI significantly easier.

Another notable change as the project advanced was the removal of motion of the Kinect. Originally it was to be mounted on a swivel for 180 degrees of motion to remove some of the strain of the robot's motion. This configuration allowed the chassis to remain oriented in one direction and the camera to turn and view something to the left or right. With the new wheels, however, we found that NELI had a negligible turn radius and there was no need for the added weight and complexity that comes with more motors, motor controllers, and coding. With the reduced weight, the robot moved faster and ran marginally more efficiently, taking less battery power to move.

4.4.1 Calibration

In order for NELI to operate as desired, calibration of both the distance sensors and the motors are required. Without this calibration, Team SCOPE risks the different motors running at different speeds, or the distance sensors producing inaccurate measurements, or NELI not accurately traveling the desired distance. The majority of this calibration was done with simple experimentation and testing.

To calibrate the distance sensors, three trials were conducted in which the distance sensor was moved incrementally away from a wall, with the returned value being compared to the real distance at each increment. To start the testing, the distance sensor was placed two inches from a wall. After making a measurement, the distance sensor was moved back 0.5" away from the wall until the sensor was 6" away. From there, the distance sensor was moved back by 1" increment until it was one foot away. Sensor calibration was limited to these short distances because the Arduino code only required action if the distance sensor detected something less than 8 inches away. After sensor values were acquired at each distance, they were plotted against the actual values. According to Figure 20, the data was then matched with a line of best fit, and the R^2 value of each trial was greater than 0.99, indicating a strong linear relation well within one tenth of an inch of the real distance.

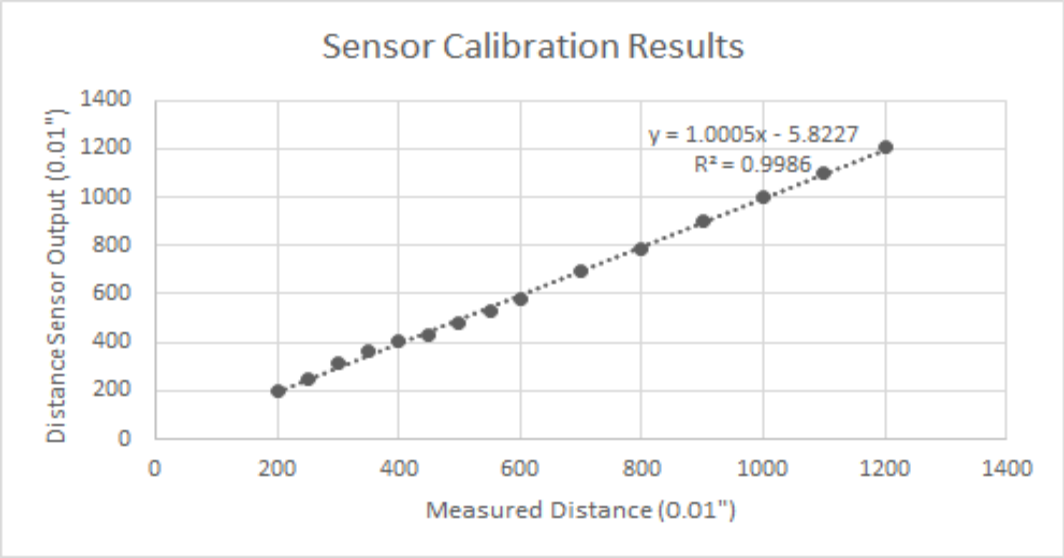


Figure 20 **Results of a single trial of distance sensor calibration. Distances were converted to hundredths of an inch for the sake of accuracy. The trendline has a slope very close to one and an R^2 value very close to one, indicating a strong one-to-one relationship.**

In addition, testing was necessary in order to find the appropriate constant of conversion between the distance-of-movement variable inputted by the main program and the time-of-movement variable used by the Arduino code. A conversion factor was found for both forward and backward movement and for angular movement, where the input from the main program is an angle rather than a distance.

Conversion factor testing was completed using similar methods to the ones used for distance sensor calibration. The Arduino code was edited to take in an input of time for the VEX motors to be turned on. Using a controlled course set-up with carefully measured markers every one foot of distance, the distance traveled in the given period of time was easily measured. These values were measured at differing increments of time, ranging from 100 milliseconds to 2 seconds. The relationship between time input and distance traveled was linear as expected, and is

extrapolated to cover any possible distance of movement. According to Figure 21, the linear relationship has a slope of 72.80, which represents the change in distance traveled with respect to a change in time input, and is used as the forward and backward movement conversion factor.

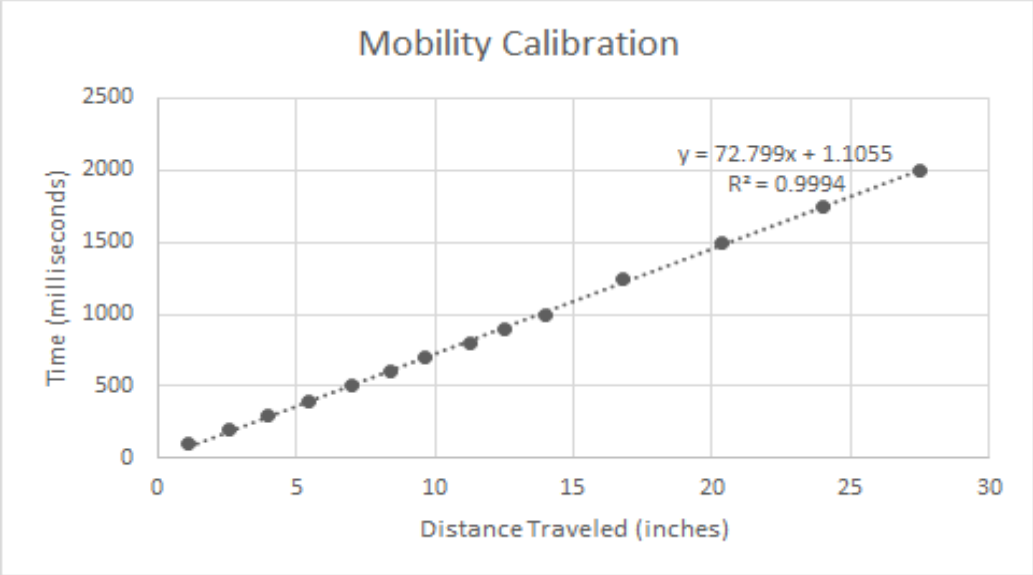


Figure 21 **Results of the mobility calibration test for forward movement. The slope indicates the conversion factor while the R2 value indicates a very strong relationship between motor operation time and distance traveled, particularly at shorter distances**

The angular conversion factor was found using similar methods. The strong linear relationship in forward motion was extrapolated to apply to angular movement as well. It was then found that a left or right turn for approximately 1850 milliseconds resulted in a full 180° turn of zero turn radius. Thus, the constant of conversion *angle_convert* = 10.28. That is, a turn of one degree would require NELI to operate in one of the turn states for 10.28 milliseconds.

4.5 Future Work

There are several avenues for further mobility development beyond the simple proof of concept currently implemented. Team SCOPE envisions a fully autonomous robotic platform designed to navigate safely throughout a hospital environment. This vision entails a laptop, which will operate the Kinect, the Arduino, and the Natural Language functionality, fully mounted to NELI's top platform. In addition, a "Follow Me" feature is a useful future direction. With this feature, NELI would be able to follow a care provider or a patient around the hospital. This would make NELI useful for carrying supplies and materials for a healthcare professional, or for tracking the progressions of physical therapy exercises through comparisons.

Another avenue of future development would be expansion to a larger, easily-identifiable mobile entity. The current NELI platform has a base that is only 12" x 12" and only stands about 18" tall with a computer mounted. In a hospital setting, a platform this small would be difficult for individuals to see and could become an obstacle for some healthcare professionals. A larger mobile platform would be easier to identify and interact with. A larger robotic platform also allows for several useful features, including a rack and pinion steering system or a mechanical arm for picking up and transporting medical supplies.

As NELI progresses towards an AI entity appropriate for implementation in a hospital environment, Team SCOPE must consider interfacing its AI capabilities with an existing mobility platform. Many robots that currently exist or are being developed, including the TUG series, the InTouch RP-VITA, or Aldebaran Robotics' Pepper, possess advanced capabilities that Team SCOPE cannot develop given limited time and resources. These robots include complex

mapping and route-planning capabilities as well as more advanced obstacle avoidance techniques, capabilities which are vital to robotics in a hospital environment. Therefore, as Team SCOPE's AI becomes more advanced, it may be appropriate to house the project's natural language and object recognition intelligence on an existing mobility platform.

Chapter 5: Integration

5.1 Previous Iterations

In the first robotic iteration, ALVORES, a Raspberry Pi was used as the means of communication between an Arduino and an Android tablet. The Raspberry Pi was chosen for its portability. The Raspberry Pi would handle all natural language interaction and image recognition requests to Watson and communication with the Arduino. However, when using the Raspberry Pi, we found that it did not have the processing power to run MATLAB image processing software, language processing software, and communication software simultaneously. Additionally, the Raspberry Pi had very few communication options with the Android tablet. These problems led to us to decide to replace the Raspberry Pi and Android tablet with a small laptop running a Microsoft Windows 10 operating system. The small laptop provides more processing power than the Raspberry Pi. The laptop has the capability to run the image recognition, NLI Watson services, and communicate with the Arduino simultaneously, which was a fundamental problem with the Raspberry Pi. It also can display the avatar and communicate with it, removing the need for the Android tablet. The switch to a laptop brought us to our second iteration, NELI, which is discussed more in the next section.

5.2 Avatar

In order to interact with the user and sync the conversation with a graphical representation of a medical practitioner, we designed an avatar using Adobe Flash Player, ActionScript 3.0, and the Microsoft Windows 10 operating system. The design of the avatar

focuses on simplicity and professionalism, while the software implementation focuses on multi-platform flexibility and web application execution.

The Adobe Flash Player runtime allows for quick and simple 2-D avatar animation. According to Adobe's published statistics on Flash Player usage, over 1.3 billion devices use Adobe Flash and over 3 million developers use Adobe Flash for web content [35]. The Adobe Flash Player runtime uses ActionScript 3.0 (AS3), which is an object-oriented programming language based on ECMAScript, the international standardized programming language for scripting [36]. AS3 was specifically designed for rich web applications. The Adobe AIR runtime enables the avatar to be packaged into native applications for Windows, our chosen operating system, while still remaining as AS3 code [37]. As many members of the team are proficient in the Windows operating system and own Windows computers, we chose to launch the program via a web application running on Adobe AIR. Lastly, we are using the Adobe Flex SDK and the FlashDevelop IDE because they are both open source and compatible with the Adobe suite. Flex is a "...highly productive, open source application framework for building and maintaining expressive web applications that deploy consistently on all major browsers, desktops, and devices", which is compatible with our web application [38], while FlashDevelop is specifically designed as a web development IDE [39]. By centering the software design on a web application, the avatar is runnable on all devices connected to the Internet. Therefore, future applications of this project can extend to other aspects of assistive health, providing open-source access and multi-platform functionality.

While researching existing avatar libraries for AS3 and the Flash runtime, we decided on

a simple design. The face of NELI should be inviting yet professional. We were also searching for AS3 lip synchronization libraries, thus enabling the avatar to talk. The first avatar we tested was created by Majid Hejazi, released on March 2014 [40].

Figure 22 Image of Avatar

It includes five avatar designs with face animations, template interface fonts and buttons, mp3 file playability, and two pre-made audio files for testing. Two of the sample avatar designs are shown above in Figure 22. The library provided exactly what we needed to create the face of NELI. For the first iteration of the avatar design, we will be exclusively using this library. Future iterations would provide a greater degree of customizability to the avatar's appearance. For example, language settings could be edited from a settings bar on the avatar's interface. In addition, skin tone, gender, and clothing could also be changed to appeal to different audiences. For example, a pediatrics office could perhaps set the avatar to appear younger, or a hard-of-hearing doctor's office could add captions to synchronize with the spoken responses of the avatar.

5.3 Methodology: Main Program

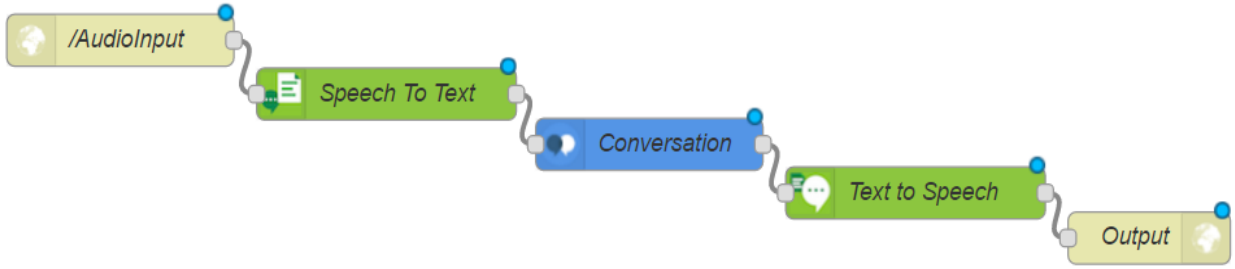


Figure 23 **Node-RED flow used to communicate with NLI cloud service and Image Recognition cloud services. The flow go from left to right where input is received on the left and sent through various nodes until an output is sent back by the far right nodes. This is an example flow for NELI's conversation function.**

In the second iteration, NELI, the laptop runs a main program written in Java. We decided to write the main program in Java because it could communicate with the NLI and image recognition services through Node-RED, the Arduino Uno through serial I/O, and the avatar through file I/O using many libraries that are available. Also, Java has many built-in libraries that many other features in the main program, such as voice input from the user, require. Since Java is so readily accessible and is object-oriented, future work and improvements become easier because of simple class additions to the project.

For the NLI component, the main program is always listening for user voice input. Java takes in audio input from the user through the Javax Sound Sample library. The program collects the user's audio input, converts it to a .WAV audio file and sends the input to the NLI program on Node-RED. The recording of the user's voice input begins when the volume goes above the ambient level and stops automatically once the volume drops to the ambient level. Next, the program sends the converted .WAV audio file through Node-RED to be processed and put through our NLI program on the Watson server. This is shown in Figure 23 in the HTTP request

called /ConversationSpeechInput. Node-RED responds with a message that contains both the text response that NELI should reply to the user, and any other pertinent information that is used by the main program. The NLI's string response always takes the form of:

speech response | additional tags

For example, if the user input was, "move forward one foot", the NLI response would be "Okay moving forward", along with "forwards", "one", and "foot" attached to the response string. The program uses that response string to complete any assigned tasks, and verbalizes the response to the user through text-to-speech functionality. Some tasks that can be executed include various movements options, recording patient symptoms for a doctor, and communicating with the Kinect to take and return a segmented image. Other tasks, such as the patient inquiry dialog, are all handled through the NLI Watson server. The symptoms received from the NLI for generating a report for a doctor about a patient are stored locally in the main program and a report is generated at the end of the dialogue for the doctor.

For the image recognition component, after the main program gets a string response from the NLI relating to image recognition such as finding an object, the main program will start executing the task. When it needs an image to be taken, it will run the Kinect executable file, which will take an image and segment the image. Then, the image will be sent up to Watson image recognition using Node-RED. This is shown in Figure 23 in the HTTP request called /ImageRecognition. The image is then classified and the results are interpreted by the main program. The main program will use these results to complete the given task.

For the mobility component, after the main program gets a string response from the NLI

relating to movement, it will use the RXTX serial library to write a string to the serial port on the laptop. The string will have two numbers; one representing an enumerated movement command and the other representing the time required to move in that direction. The Arduino code will be constantly listening for strings written to the serial port. When it gets a command, it will parse the command and execute it, moving the robot for the given amount of time.

For the avatar component, when the NLI program returns a string that requires a voice response back to the user, the Node-RED program will automatically return a .WAV audio file to the main program. The audio file is then converted by the main program into a .MP3 file format using the built-in Audio Encoding libraries in Java. The resulting audio file is then placed into a folder. Each audio file is given a number increasing from 1 in its filename. The Avatar program will be constantly checking the folder for an audio file with the next number. Once the main program puts the audio file with the next number in its file name, the Avatar program will automatically play the file. To avoid the audio file being recorded by the main program as user voice input, the main program pauses for the duration of the audio file before continuing.

The main program connects the NLI, image recognition, mobility, and avatar components of the project together. It starts by getting voice input from the user. Then, the audio is processed by the NLI component. Based on what the user is asking the robot to perform, it will execute the command. If the task related to image recognition, it will run the Kinect executable to get the images and process them using Node-RED. If the task is related to mobility, it will send commands to the Arduino, which will move the robot. Any audio response the robot must give to the user will be done by the Avatar component.

5.4 Mapping System

For NELI to navigate without many sensors, a mapping system is needed in the main program. We decided to hardcode the blueprints of where NELI can move. This would need to be changed depending on the environment NELI is operating in. There are many other mapping systems and sensors that could be implemented for a better mapping system depending on the environment and these should be taken into consideration when actually implementing NELI in a real world environment. In an actual office setting, these blueprints would simply map out the rooms so that NELI knows where it can physically travel. The blueprints are divided in the code into a coordinate system where each coordinate represents a target location. Our mapping system is based on a directed graph. A directed graph is a commonly used computer science tool that has nodes of data and edges with weights that point towards other nodes. In our directed graph, each node represents one of the coordinates within the blueprint and the edges are weighted with a distance and direction towards the target node. A Java program that we created will then use Dijkstra's algorithm to find the shortest path. We implemented the mapping system using Java because our main program is also in Java; having both in the same language allows for easy interfacing and a very short time delay to access the mapping system from the main program. Dijkstra's algorithm will be discussed in detail shortly. This mapping system will allow for remote web-based access, which will show the blueprints as well as NELI's current location on those blueprints. Additionally, future work would allow for remote movement commands to be implemented from the blueprints. These blueprints will continue to be hardcoded for the current iteration of the mapping system, however future iterations of the mapping system may be able to

use markers throughout the building or other systems such that a direct hardcoding of the blueprint is not needed.

Dijkstra's algorithm is a well-known algorithm used to solve the shortest paths for a given graph with vertices/nodes and edges [41]. We will be using a variant of Dijkstra's algorithm to find the shortest path from a single source node, NELI's current location, to a single destination location, wherever NELI is told to go. The node at which NELI starts is called the initial node. The distance of node D is the distance from the initial node to node D, the destination node. Dijkstra's algorithm will assign some initial distance values and will try to improve them step-by-step. The steps for the algorithm are listed below:

1. Assign to every node an initial distance value: zero for our initial node and infinity for all other nodes.
2. The initial node will be set to the current node. Mark all other nodes unvisited. Create a list of all unvisited nodes.
3. For the current node, look at all of its unvisited neighbors. Neighbors are any nodes the current node has a direct edge to. For our purposes, all nodes represent a location that NELI could fit in and all neighbors are only a distance of 1 away from the current node. Looking at all neighbors, they will be assigned new distance values based on the distance value of the current node plus one. To clarify, say the current node is E and is marked with a distance of 7. All neighbors with a distance value greater than 8 will be reassigned to have a distance value of 8.
4. After step 3, mark the current node as visited and make sure it is removed from the

unvisited list. This way, it will never be checked twice.

5. If the destination node D has been marked as visited, then stop. The algorithm has finished and the shortest path will be returned as a list of nodes that NELI should traverse.
6. If the destination node has not been marked visited, select the unvisited node that is marked with the smallest distance, set it as the new current node, and go back to step 3.

The mapping system as described above is functional but could be improved greatly. One study looking at existing indoor navigation systems and the many different technical and usability problems they face found that using a Near Field Communication (NFC) based indoor navigation system called NFC Internal helped to eliminate many of these problems. NFC Internal enables easy data transfer by physically touching special tags with a NFC enabled device [42]. This could be very useful for NELI if it is going to be put into implementation at a hospital or at a general doctor's office. There are other mapping systems and techniques that could be implemented that would greatly improve NELI's movement system and the best ones would need to be selected based upon specific project needs. One paper describes an, "implementation of a people detection system for a robotic platform able to perform Simultaneous Localization and Mapping" [43]. Another paper, "presents a robust mapping system with Kinect V2 for automated photovoltaic cleaning system" [44]. Both of these techniques/algorithms could be useful to improve NELI's movement system based upon a specific use for NELI.

5.5 Discussion/Future Work

Overall, the integration of the various components was successful. The main Java program had the capability to communicate to Node-RED and through Node-RED could access the necessary image and natural language cloud based services. Additionally, the main program could send text to the Arduino Uno and therefore could execute movement commands. Lastly, the main program could execute both the Avatar program for the interface and the necessary MATLAB programs for image analysis. This was shown through a test that was done within our lab. NELI was tasked, “Find the ambulance”, where the ambulance was a printed picture of an ambulance hung on a nearby wall. NELI successfully moved to the picture using the IBM’s cloud-based Watson image recognition service. After analyzing the image, NELI responded with, “Found the ambulance”.

Despite being a simple test, it clearly shows that all functions of NELI are correctly integrated and work to the extent of finding a picture of a designated object. While everything did function correctly, there is also a lot of room for improvement within the integration program of NELI. The audio input and output algorithm was still slightly inconsistent, in the sense that sometimes NELI does not get the entirety of the users input command. Additionally, the user must wait some time after the output ends prior to when NELI starts listening for input again. These two issues could be fixed with the fine tuning of the audio input/output algorithm. Another area for improvement is the image recognition executable. The executable does work correctly, but takes a significant amount of time to complete and produce a response. This problem can be improved by optimizing the executable code. Additionally, as technology and cloud based

services improve, NELI should be updated to use the most efficient programs available.

Chapter 6: Conclusion

6.1 Discussion of Results

Team SCOPE has demonstrated a simple implementation of an intelligent personal assistant through cloud computing services, with functionalities that are compartmentalized such that they could be modified in a modular fashion. Furthermore, the functionalities have been tailored for an application in the field of healthcare. The chest pain conversation and object recognition tests served as a proof of concept for NELI's ability to converse intelligently based on unique conversations and ability to improve upon IBM Watson's object recognition accuracy with on-board processing. Intelligent assistants could create a significant positive impact by performing some of the routine work typically performed by nurses, physicians, and physician's assistants. NELI's capabilities have been individually tested, demonstrating their accuracy and the viability of more comprehensive versions.

A natural language-based interface has been designed and realized using IBM's Watson conversation technology. The NLI was tested using sample responses for how accurately patient information is recorded for the physician. The results showed that simple responses were recorded much more accurately than complex responses. The visual object recognition capabilities have been implemented using a Microsoft Kinect, Watson cloud computing services, and image segmentation Matlab code running locally. The functionality was tested with an assortment of images to measure the robustness of the algorithms. NELI's mobility platform runs local code on an Arduino microcontroller. The Arduino communicates with the rest of the system to facilitate the actuating abilities of the robot. This allows for the basic movement and

distance sensing required for the robot to operate autonomously and effectively in a hospital or clinical environment. The accuracy of the movements performed by this subsystem has been analyzed and shown to be appropriately precise for the purposes we have suggested.

These subsystems are incorporated through a main integration program, which uses Node-RED to link together the various Watson services used. The program establishes communication with the local Arduino microcontroller and Microsoft Kinect to provide additional functionality. The output of the system is linked to an avatar program which provides the system with a human-like face. In terms of quantifying the success of the project, the integration program shows that each component of NELI is able to work together as intended. Though there are small kinks that have arisen occasionally, the system is able to facilitate communication between the components of NELI.

Though the functionality of the system is limited at the moment, this proof of concept project shows the potential of such a setup. Each component of NELI can be updated and changed without affecting the other components. New components can be added independently, by only changing the main integration program. This project also demonstrated the applicability of the robot for healthcare applications. The NLI program in particular provided an useful way of collecting and classifying patient information for the physician. However, the modular nature of the system allows for easy adaptability for other applications. This design choice is an asset that will allow for new technologies to be integrated into the robot's functionality.

6.2 Future Work

Future iterations of NELI have been discussed at length, and several additions and

improvements have been proposed as the project continues. The first is a “follow me” feature. Not implemented in the current NELI iteration, a feature that would allow the physical robot to follow a specific target, namely a healthcare professional, would be instrumental in allowing immediate access to information the robot observes, and being able to use it when required. This functionality can be implemented with the visual recognition and mobility components.

We have not addressed this, but it is within the mission for NELI to connect to electronic health records with all the issues of security, the Health Insurance Portability and Accountability Act of 1996 (HIPAA), authentication, and in heterogeneous systems. Because NELI is modular, these additional modules can be constructed and NELI’s architecture allows them to be integrated. The IBM system will have to be HIPAA certified to deploy NELI into the healthcare setting.

In addition to expanding the NLI and visual object recognition components with electronic health records and databases, we would like to transform NELI into an expert system. This entails involving professionals in the healthcare field to help with NLI conversation dialogue development and medical image databases. With their help, we could expand the conversation capabilities beyond standard chest pain conversations. We would like to explore developing a simple interface that allows a professional to review necessary questions and various responses for medical symptoms. This interface could lead to a more automated and robust dialogue development system.

Another improvement to the system overall would be the use of a standard chassis for the physical base of the robot. With the current aforementioned system, there is a problem of friction

and inability to steer whilst moving forward. A chassis with a rack and pinion steering system, not dissimilar from a car, would be a more efficient method of travel. This would allow NELI to avoid obstacles while maintaining forward motion instead of having to stop and make several calculated turns. It could instead simply make a small adjustment when noticing an object ahead of it in its way and alter its course slightly.

NELI should be fully integrated into the database that maintains EHRs. Using a secure connection, NELI would upload the information it gathers from its own observations to help compile and complete the patient's file. Connection to a doctor or nurse's phone via Bluetooth connection for alerts such as a status update on a patient or reminder to administer more medicine would also be beneficial. NELI's ultimate goal of being a full-featured assistant for healthcare professionals requires automation and integration with existing productivity systems

REFERENCES

[1] "da Vinci Surgery | Robotic-Assisted Surgery." [Online]. Available: http://www.davincisurgery.com/. [Accessed: 10-Mar-2017].

[2] "Xenex," *Xenex*. [Online]. Available: https://xenex.com/. [Accessed: 10-Mar-2017].

[3] "Robotic Prescription Dispensing Systems - ScriptPro." [Online]. Available: https://www.scriptpro.com/products/robotic-prescription-dispensing-systems/. [Accessed: 10-Mar-2017].

[4] "PARO Therapeutic Robot." [Online]. Available: http://www.parorobots.com/. [Accessed: 10-Mar-2017].

[5] "Who is Pepper?," *SoftBank Robotics*. [Online]. Available: https://www.ald.softbankrobotics.com/en/cool-robots/pepper. [Accessed: 10-Mar-2017].

[6] L. D. Riek, "Robotics Technology in Mental Health Care," *arXiv:1511.02281 [cs]*, pp. 185–203, 2016.

[7] "IBM Watson Visual Recognition." [Online]. Available: https://visual-recognition-demo.mybluemix.net/. [Accessed: 10-Mar-2017].

[8] "Microsoft Windows Dev Center: Kinect hardware." [Online]. Available: https://developer.microsoft.com/en-us/windows/kinect/hardware. [Accessed: 10-Mar-2017].

[9] L. Yang, L. Zhang, H. Dong, A. Alelaiwi, and A. E. Saddik, "Evaluating and Improving the Depth Accuracy of Kinect for Windows v2," *IEEE Sensors Journal*, vol. 15, no. 8, pp. 4275–4285, Aug. 2015.

[10] G. G. Chowdhury, "Natural language processing," *Ann. Rev. Info. Sci. Tech.*, vol. 37, no. 1, pp. 51–89, Jan. 2003.

[11] I. Androutsopoulos and M. Aretoulaki, "Natural Language Interaction," Jan. 2005.

[12] B. Webber, A. Joshi, E. Mays, and K. McKeown, "Extended natural language data base interactions," *Computers & Mathematics with Applications*, vol. 9, no. 1, pp. 233–244, Jan. 1983.

[13] Huda Khayrallah, Sean Trott, and Jerome Feldman, "Natural Language For Human Robot Interaction: Proceedings of the Workshop on Human-Robot Teaming at the 10th AMC/IEEE International Conference on Human-Robot Interaction, Portland, Oregon." University of California, Berkeley International Computer Science Institute, 2015.

[14] J. Marciniak and Z. Vetulani, "Ontology of Spatial Concepts in a Natural Language Interface for a Mobile Robot," *Applied Intelligence*, vol. 17, no. 3, pp. 271–274, Nov. 2002.

[15] Artificial Solutions, “Natural Language Interaction |,” *Natural Language Interaction | Artificial Solutions*.
[16] Nuance Communications, Inc., “For business | Printer Management, Security & Cost Recovery for Business | Nuance | Nuance.” [Online]. Available: http://www.nuance.com/for-business/index.htm. [Accessed: 12-Mar-2017].
[17] C. V. Almario *et al.*, “Computer-Generated Vs. Physician-Documented History of Present Illness (HPI): Results of a Blinded Comparison,” *Am J Gastroenterol*, Dec. 2014.
[18] S. Arora, A. D. Goldberg, and M. Menchine, “Patient Impression and Satisfaction of a Self-administered, Automated Medical History Taking Device in the Emergency Department,” *Western Journal of Emergency Medicine*, vol. 15, no. 1, Jan. 2014.
[19] Jim Newman, “MD Anderson Taps IBM Watson to Power,” *The University of Texas MD Anderson Cancer Center*, 18-Oct-2013. [Online]. Available: https://www.mdanderson.org/newsroom/2013/10/md-anderson--ibm-watson-work-together-to-fight-cancer.html. [Accessed: 10-Mar-2017].
[20] The University of Texas System Administration, “Special Review of Procurement Procedures Related to the M.D. Anderson Cancer Center Oncology Expert Advisor Project.”
[21] Steven Bird, Ewan Klein, and Edward Loper, *Natural Language Processing with Python*. O’Reilly Media, 2009.
[22] J. Harrington, L. M. Noble, and S. P. Newman, “Improving patients’ communication with doctors: a systematic review of intervention studies,” *Patient Educ Couns*, vol. 52, no. 1, pp. 7–16, Jan. 2004.
[23] S. J. Knoble and M. R. Bhusal, “Electronic diagnostic algorithms to assist mid-level health care workers in Nepal: A mixed-method exploratory study,” *International Journal of Medical Informatics*, vol. 84, no. 5, pp. 334–340, May 2015.
[24] A. Corbel, D. Baud, A. Chaouch, J. Beney, C. Csajka, and A. Panchaud, “Utility of an Algorithm to Increase the Accuracy of Medication History in an Obstetrical Setting,” *PLoS ONE*, vol. 11, no. 3, p. e0151205, 2016.
[25] D. Zakim, “Development and significance of automated history-taking software for clinical medicine, clinical research and basic medical science,” *J Intern Med*, vol. 280, no. 3, pp. 287–299, Sep. 2016.
[26] M. A. Probst *et al.*, “Emergency Physicians’ Perceptions and Decision-making Processes Regarding Patients Presenting with Palpitations,” *J Emerg Med*, vol. 49, no. 2, p. 236–243.e2, Aug. 2015.

[27] H. Yan, Y. Jiang, J. Zheng, C. Peng, and Q. Li, “A multilayer perceptron-based medical decision support system for heart disease diagnosis,” *Expert Systems with Applications*, vol. 30, no. 2, pp. 272–281, Feb. 2006.

[28] “What Are the Signs and Symptoms of Respiratory Failure? - NHLBI, NIH.” [Online]. Available: https://www.nhlbi.nih.gov/health/health-topics/topics/rf/signs. [Accessed: 13-Mar-2017].

[29] J. G. Teeter and E. R. Bleecker, “Relationship between airway obstruction and respiratory symptoms in adult asthmatics,” *Chest*, vol. 113, no. 2, pp. 272–277, Feb. 1998.

[30] © 2017 Aethon All Rights Reserved, “Aetheon: New Hospital Construction,” *Aethon*. [Online]. Available: http://www.aethon.com/hospitalconstruction/. [Accessed: 13-Mar-2017].

[31] © 2017 Aethon All Rights Reserved, “More Efficient Nurses, Happier Patients,” *Aethon*. [Online]. Available: http://www.aethon.com/nursing/. [Accessed: 13-Mar-2017].

[32] © 2017 Aethon All Rights Reserved, “How the TUG Automomous Mobile Robot Works,” *Aethon*. [Online]. Available: http://www.aethon.com/tug/how-it-works/. [Accessed: 13-Mar-2017].

[33] “RoboCourier® Autonomous Mobile Robot - Swisslog.” [Online]. Available: http://www.swisslog.com/en/Products/HCS/Automated-Material-Transport/RoboCourier-Autonomous-Mobile-Robot. [Accessed: 13-Mar-2017].

[34] “InTouch Health and iRobot to Unveil the RP-VITA™ Telemedicine Robot at Clinical Innovations Forum,” *InTouch Health*, 24-Jul-2012.

[35] “Statistics | Adobe Flash runtimes.” [Online]. Available: http://www.adobe.com/products/flashruntimes/statistics.html. [Accessed: 27-Feb-2017].

[36] “ActionScript 3.0 overview | Adobe Developer Connection.” [Online]. Available: http://www.adobe.com/devnet/actionscript/articles/actionscript3_overview.html. [Accessed: 27-Feb-2017].

[37] “Adobe - Adobe AIR.” [Online]. Available: https://get.adobe.com/air/. [Accessed: 27-Feb-2017].

[38] “Free, open-source application framework | Adobe Flex.” [Online]. Available: http://www.adobe.com/products/flex.html. [Accessed: 27-Feb-2017].

[39] “FlashDevelop.org - Welcome.” [Online]. Available: http://www.flashdevelop.org/. [Accessed: 27-Feb-2017].

[40] Majid Hejazi, “AS3 Talking Avatar library for Starling,” *CodeCanyon*, Mar-2014. [Online]. Available: https://codecanyon.net/item/as3-talking-avatar-library-for-starling/7173912. [Accessed: 23-Mar-2017].

[41] D. E. Knuth, “A generalization of Dijkstra’s algorithm,” *Information Processing Letters*, vol. 6, no. 1, pp. 1–5, Feb. 1977.

[42] B. Ozdenizci, K. Ok, V. Coskun, and M. N. Aydin, “Development of an Indoor Navigation System Using NFC Technology,” in *2011 Fourth International Conference on Information and Computing*, 2011, pp. 11–14.

[43] A. T. Angonese and P. F. F. Rosa, “Integration of People Detection and Simultaneous Localization and Mapping Systems for an Autonomous Robotic Platform,” in *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*, 2016, pp. 251–256.

[44] M. Li *et al.*, “Fast and robust mapping with low-cost Kinect V2 for photovoltaic panel cleaning robot,” in *2016 International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2016, pp. 95–100.

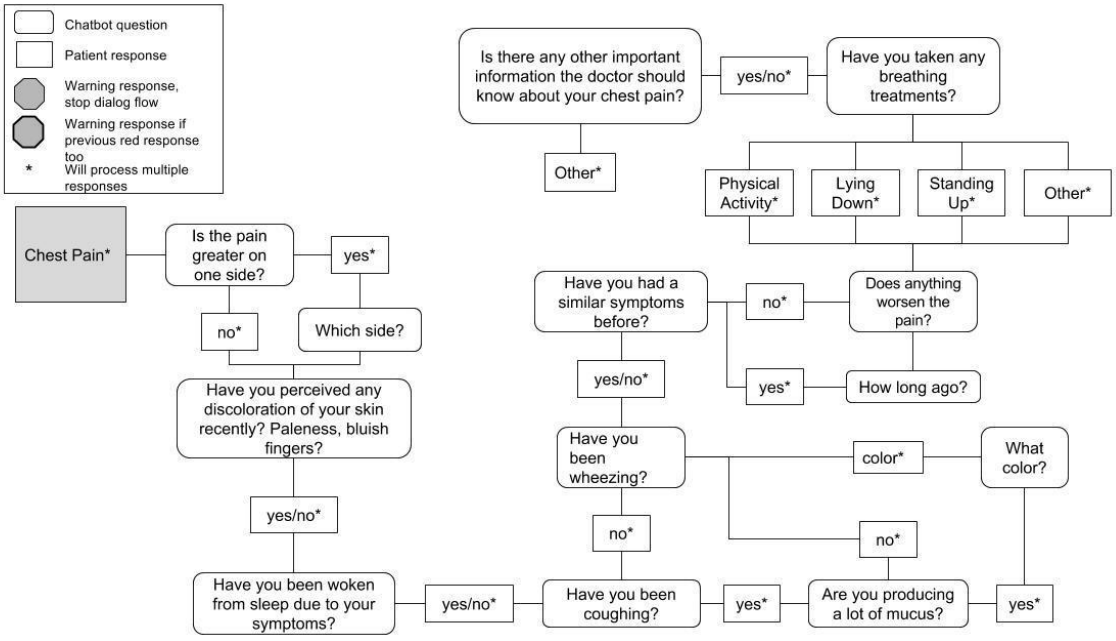
APPENDICES

Appendix A: Watson Responses to Object Recognition Test

Microwave (3D):
Auto Segmented:
Microwave, .829, /appliance/microwave
Appliance, .83
CRT screen, .637, /electronic device/video display/CRT screen
Video display, .638
Electronic Device, .639
Device, .639
Coal black color, .97

Raw:
Appliance, .691
Gas Heater, .576, /device/gas heater
Device, .654
Computer, .561, /device/machine/computer
Machine, .564
White goods, .53
Ash grey color, .815
Charcoal color, .746

Appendix B: Watson Conversation Respiratory Charts



Dialog flow used in Watson's conversation service to process cardiovascular chief complaint. Flow diagram created with algorithms and routine symptoms relating to past medical and NLP literature.

Appendix C: IRB Research Determination Letter

UNIVERSITY OF MARYLAND

INSTITUTIONAL REVIEW BOARD

1204 Marie Mount Hall
College Park, MD 20742-5125
TEL 301.405.4212
FAX 301.314.1475
irb@umd.edu
www.umresearch.umd.edu/IRB

DATE:	February 10, 2017
TO:	Anil Deane, PhD
FROM:	University of Maryland College Park (UMCP) IRB
PROJECT TITLE:	[1027367-1] Gemstone Research on Sensory Computing and Object Processing Entity: Testing natural language interaction chatbot with hypothetical dialog responses
SUBMISSION TYPE:	New Project
ACTION:	DETERMINATION OF NOT HUMAN SUBJECT RESEARCH
DECISION DATE:	February 10, 2017

Thank you for your submission of New Project materials for this project. The University of Maryland College Park (UMCP) IRB has determined this project does not meet the definition of human subject research under the purview of the IRB according to federal regulations.

We will retain a copy of this correspondence within our records.

If you have any questions, please contact the IRB Office at 301-405-4212 or irb@umd.edu. Please include your project title and reference number in all correspondence with this committee.

This letter has been electronically signed in accordance with all applicable regulations, and a copy is retained within University of Maryland College Park (UMCP) IRB's records.

Appendix D: Survey used to gather dialogue testing responses

Conversation Entry

Thank you for your interest in assisting with Gemstone Team SCOPE's research! Your responses to this survey will be used to beta-test a natural language interaction program functioning as a virtual physician's assistant. You will be writing two conversations in the form of responses to physician's questions. Please spend no longer than 10 minutes on these together - think about how you would answer these prompts in everyday conversation and that is how you are advised to write your responses.

The goal of this exercise is for you to simulate that you are having a conversation with the robot. You will repeat the conversation twice, however pick a different type/source of chest pain for each.

Your only requirement is to MENTION SOME FORM OF CHEST PAIN AS THE CHIEF COMPLAINT. However, all other responses are up to your discretion. Please try to be realistic in replicating what your responses would be in a clinical setting.

* Required

1. **What is your chief complaint?** *

2. **How would you describe the severity of your pain? Mild, moderate, severe?** *

3. **How long does the pain usually last in minutes?** *

4. **Are you feeling any shortness of breath?** *

5. **Have you experienced increased heart rate recently?** *

6. **Has your blood pressure been higher than normal?** *

7. **Where is the chest pain the worst?** *

8. **Have you had a similar type of pain before? if so, how long ago?** *

9. **Does anything worsen the pain?** *

10. **Have you been under increased stressed lately?** *

11. **What medication, if any, have you taken for the pain?** *

12. **Is the pain greater on one side?** *

13. **Have you had any perceived discoloration of your skin recently? Paleness or blueish fingers?** *

14. **Have you been woken from sleep due to your symptoms?** *

15. **Have you been coughing? if so are you producing a lot of mucus?** *

16. **Have you been wheezing?** *

17. **Have you had similar symptoms before?** *

18. **Have you taken any breathing treatments?** *

19. **Is there any other important information the doctor know about your chest pain?** *

Conversation #2

20. **What is your chief complaint?** *

21. **How would you describe the severity of your pain? Mild, moderate, severe?** *

22. **How long does the pain usually last in minutes?** *

23. **Are you feeling any shortness of breath?** *

24. **Have you experienced increased heart rate recently?** *

25. **Has your blood pressure been higher than normal?** *

26. **Where is the chest pain the worst?** *

27. **Have you had a similar type of pain before? if so, how long ago?** *

28. **Does anything worsen the pain?** *

29. **Have you been under increased stressed lately?** *

30. **What medication, if any, have you taken for the pain?** *

31. **Is the pain greater on one side?** *

32. **Have you had any perceived discoloration of your skin recently? Paleness or blueish fingers?** *

33. **Have you been woken from sleep due to your symptoms?** *

34. **Have you been coughing? if so are you producing a lot of mucus?** *

35. **Have you been wheezing?** *

36. **Have you had similar symptoms before?** *

37. **Have you taken any breathing treatments?** *

38. **Is there any other important information the doctor know about your chest pain?** *