Dixon: TurtleBot Voice Commands

Alex Rhodes
Cleveland State University
Email: awrhodes9723@gmail.com

Sujay Bajracharya
Cleveland State University
Email: sujaymanb@gmail.com

***Abstract*—Voice commands are an accessible method of interacting with a robot. This report describes Dixon, a project that allows the use of simple voice commands to control a TurtleBot. It details the various components of the system, as well as how they are implemented, and describes the results of how successful the system was. In addition, the paper also outlines the various challenges and limitations of the project and describes future work on how to improve the concept.**

***Index Terms*—TurtleBot, voice commands, speech recognition, Dixon**

I. Introduction

Dixon is a ROS package that enables a TurtleBot to act on voice commands issued by a user [1]. Dixon can respond to simple movement commands that are either local or global, local meaning relative destinations (e.g. "forward," or "backward") and global meaning destinations that are predefined (e.g. Room 202).

The main objectives of this project were to develop a package that processes vocalized phrases such as "move forward," and "go to x destination" and generates command signals based on them. Secondarily, we wanted to create a web application that allows a user to interact with the ROS package remotely.

The reasoning behind this project was that currently, human-robot interaction (HRI) is quite obtuse and mostly limited to programming. Vocal, haptic, and somatic input methods are much more intuitive to users that cannot program. As we are both quite interested in Natural Language Processing (NLP), we decided to focus on a voice input system.

Dixon has four main components: the speech recognition component, parser, voice or feedback component and the navigation component. For speech recognition we used pocketsphinx to implement the speech recognition for Dixon. There is also a pocketsphinx package for ROS that publishes the results of the speech recognition to the topic.

For the parser, we utilized a shallow semantic parser to perform a keyword search on the transcription returned by pocketsphinx. The parser searches for command and destination keywords such as "go," and "forward," and generates a command based on words it detects. It then generates a response for the user, either an affirmative or negative response.

Once a valid command has been generated, the navigation module checks whether the required movement is local or global. If the command is for local movement, e.g. "forward" or "backward", then the navigation node publishes the corresponding Twist messages to the topic "mobile_base/command/velocity". On the other hand, if the command is for global movement, i.e. move to a location on the map, then Dixon navigates to the location using actionlib "goals". If the command is "stop", then the command velocity is set to zero for local movement, and a goal to the current position is sent for global movement.

In order to provide vocalized feedback to the user, the University of Edinburghs Festival speech synthesizer was used. Festival is a free and open source speech synthesizer, and the ROS package sound_play provides an interface to send the synthesizer text to be synthesized.

The paper is organized in the following way. Section II, III, IV, and V describe the speech recognition, parser, navigation and voice components of Dixon respectively along with the topics they subscribe and publish to. Section VI describes the results of the tests conducted to measure the effectiveness of the system. Section VII talks about the limitations and challenges faced during the development of Dixon. Finally, section VIII contains a description of the future work and the conclusion to the report.

II. Speech Recognition

A. pocketsphinx

Dixon uses the pocketsphinx ROS package for speech recognition. Pocketsphinx is a more portable implementation of CMU Sphinx, developed at Carnegie Mellon University (CMU), which is a Hidden Markov Model (HMM) based speech recognition toolkit [2] [3].

B. Topics Published

The pocketsphinx package for ROS publishes the results of the speech recognition to the topic "/recognizer/output". We designed our parser node to subscribe to this topic in order to parse the command.

III. Parser

A. Shallow Parser

Our parser performs a keyword search on the transcription returned by pocketsphinx. It searches for command such as "go," "move," and "head," and destinations such as "forward," "backward," "alpha," and "beta." "Alpha" and "beta" are placeholder names that could easily be replaced with real destination names.

TABLE I
MESSAGE COMPOSITION

Field	Description	Data Type
command	Command	String
destination	Destination	String
local	Local Destination Flag	Bool
x	Destination X Coordinate	Float32
y	Destination Y Coordinate	Float32

B. Command Message Structure

[t] The parser generates a command based on the keywords detected of the following form: command, destination name, local flag, destination x coordinate (if applicable), destination y coordinate (if applicable). Table 1 describes the structure of the command message. The command can be either a movement command ("move," "go") or a stop command ("stop," "abort"). The destinations can be general directions ("forward," "backward") or predefined points on the map that have names attached. The local flag is a boolean value that informs the navigation node whether the destination is local or global, and the X and Y coordinates of the destination which correspond to the location on the map.

C. Topics Published and Subscribed

The parser subscribes to the topic "recognizer/output". This topic contains the result of the speech recognition performed by the pocketsphinx node. The string message from this topic is used to generate the command. The parser node publishes to two nodes. First, it publishes responses to the "dixon_response/response" topic which is used by the voice node to generate feedback to the user. Second, the command message is published to "Dixon/command" which is used by the navigation node to carry out movement actions.

IV. NAVIGATION

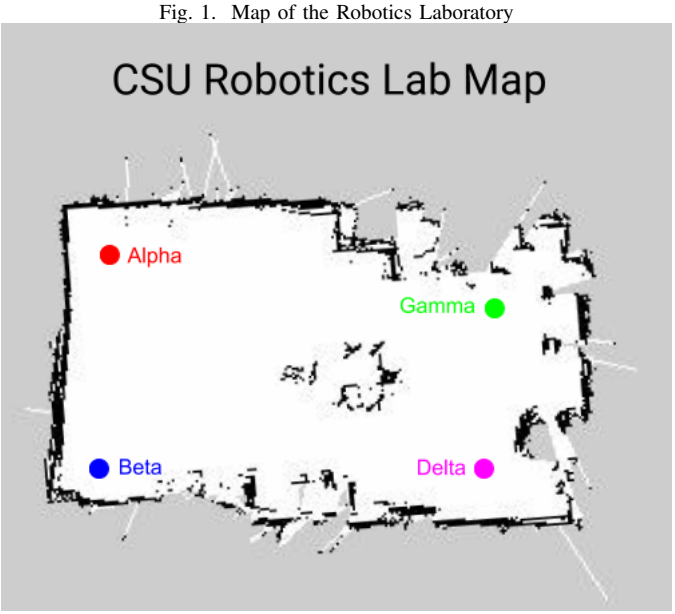
A. RVIZ

RVIZ is the "Ros-Visualization" package that can be used for various mapping and navigation related tasks. It allows the map to be viewed and even set navigation goals or view the progress when creating a map. Along with gmapping, we used RVIZ to create a map of the lab where the tests were conducted.

B. gmapping

To create the map of the lab shown in figure 1, we used the gmapping package for ROS which provides a wrapper for the "Gmapping" Simultaneous Localization and Mapping (SLAM) algorithm implemented by OpenSLAM. Gmapping is a Rao-Blackwellized particle filter based SLAM algorithm to create occupancy grid maps which we used for robot navigation [4] [5]. The ROS package allows us to use the "Astra" depth camera on the TurtleBot to create the map.

Fig. 1. Map of the Robotics Laboratory



C. Navigation Stack

amcl is the ROS package that implements the "Adaptive Monte-Carlo Localization" to track the position of the robot on a known map [6]. We use the amcl_pose topic to get the current position of the robot on the map. It is part of the ros navigation stack that we use for global movement commands

D. actionlib

The actionlib stack is a package that provides a client-server interface to send tasks to the robot. A task could be sending a navigation goal, for example. We used actionlib for the global movement portion of the navigation node which involves moving to given coordinates on a map.

E. Dixon_nav Node

The dixon_nav node carries out the navigation command on the robot once the parser has found a valid command and published it. The node first checks whether the "local" flag is set. If the flag is True then the node checks the "destination" part of the message to see what direction the robot needs to move in. Then, the linear direction variables and the variable that keeps track of which type of movement is being carried out are set accordingly. The node constantly publishes the velocity to the "mobile_base/command/velocity" in a second thread when local movement is required. If the local movement is not set then the node uses actionlib to send a navigation goal to the coordinates that are in the message. If the command is stop, the node checks which type of movement is being carried out currently. If local movement, it sets the velocity to zero and for global movement it sends a goal to the Dixons current coordinates.

Fig. 2. The results to the global commands with built-in mic tests.

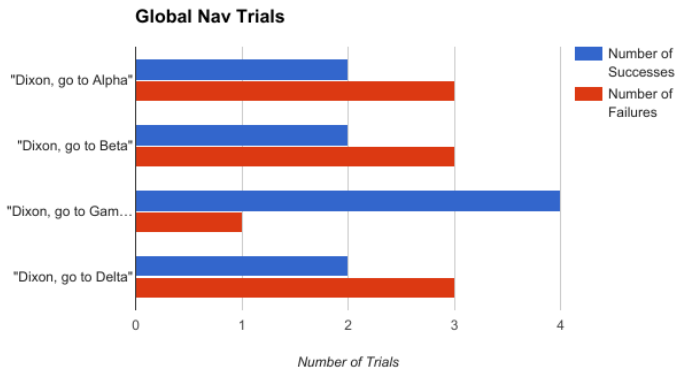
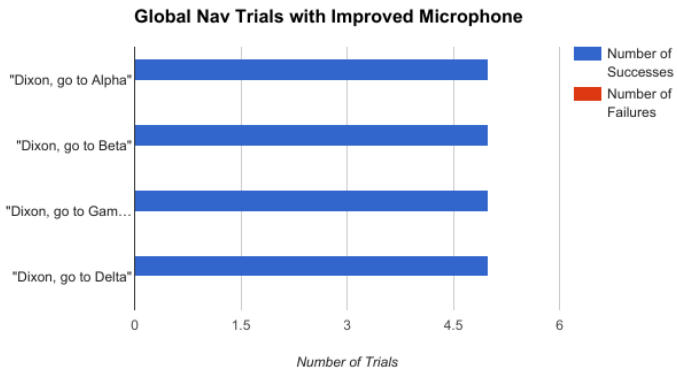


Fig. 3. The results to te global commands with improved mic tests.



F. Topics Published and Subscribed

The navigation node subscribes to the topic "Dixon/command" which contains the command message published by the parser. It also subscribes to the "amcl_pose" topic to get the current position of the robot on the map. Then, if local navigation is required then the node publishes to the topic "mobile_base/command/velocity". Otherwise, the node uses actionlib which uses its own respective nodes.

V. Voice

A. sound_play

University of Edinburghs Festival is a framework for building speech synthesis systems [7]. Festvox is a voice synthesization implementation of Festival that can be used for text to speech. Sound_play is a ROS package that enables a robot to play sounds, either pre-made sound bites or synthesized speech created using the Festvox implementation of Festival.

If a string is published to a topic that sound_play subscribes to, it will generate speech using Festvox. We used this feature of sound_play to synthesize Dixons voice in the dixon_voice node of our package.

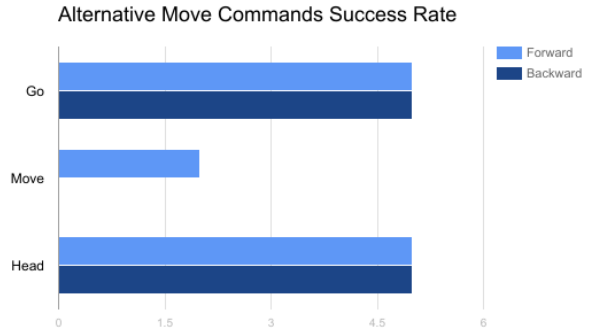
B. Topics Published and Subscribed

The voice node subscribes to the "dixon_respose/response" topic which contains the response generated by the parser. The string contained in the message is then used along with sound_play to generate a voice response. These topics are illustrated in figure 5.

VI. Results

In order to test the accuracy of Dixons command recognition, we performed several trials. In the first trial, we tested four individual global movement commands five times each for a total of twenty trials using the built-in microphone of the netbook running ROS. The commands given were "Dixon, go to Alpha," "Dixon, go to Beta," "Dixon, go to Gamma," Dixon, go to Delta. In figure 2 the results of this trial are shown. Destinations "Alpha," "Beta," and "Delta," all had the same results, with two successful recognitions. "Gamma,"

Fig. 4. The results to the alternative command keyword test.

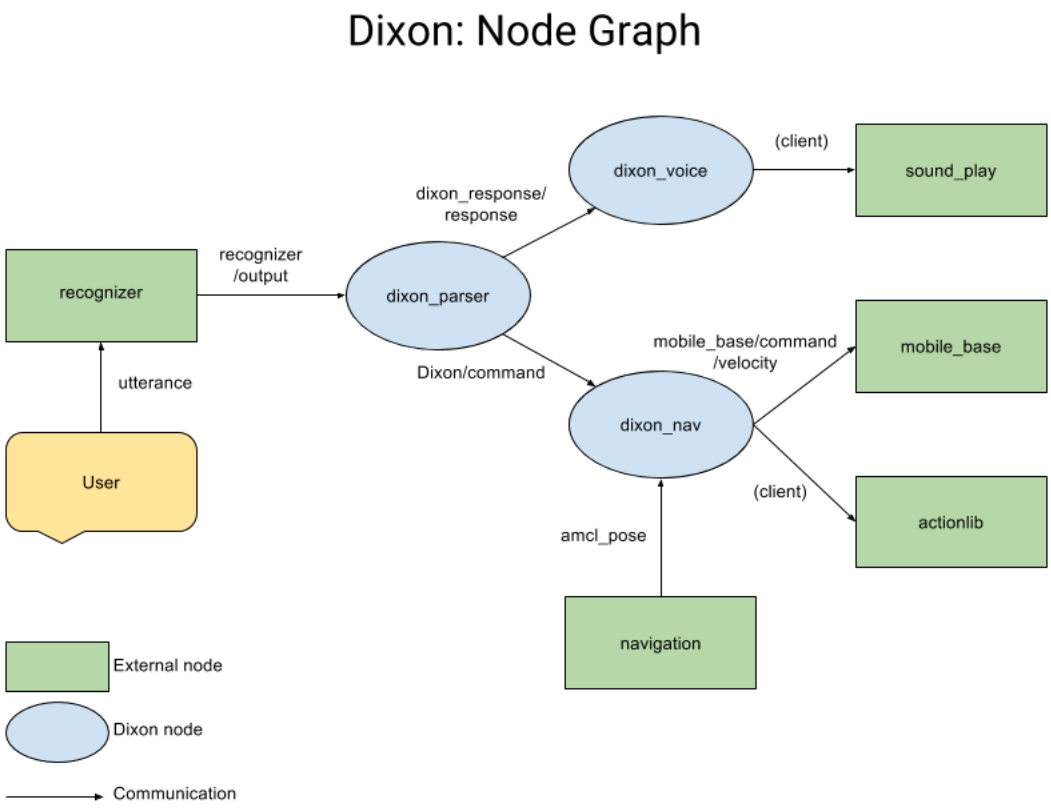


however, had four successful recognitions. In addition, for every successful recognition the command given was carried out without error. A possible reason for this disparity in successes is the word "Gamma." As will be seen in a later trial, pocketsphinx also has a better rate of transcribing the word "go" instead of an alternative like "move." We hypothesize that these results may be correlated, as the words "gamma" and "go" both start with the letter "g" and are followed by a vowel, possibly making them easier for pocketsphinx to transcribe. This is just conjecture however, as we have not performed trials to test this hypothesis.

In general, the overall success rate was not satisfactory. The main reason for the low success rate was the poor quality of the built-in microphone on the netbook. The microphone also had a limited effective range of approximately 10 cm. In order to improve the speech recognition, we carried out the trials again with an external microphone. The microphone was found to have a longer effective range. figure 3 shows the success rates with the new microphone. It is significantly improved as all 20 trials were successfully recognized.

Besides the hardware limitations we found that there were some issues recognizing particular words and their success rate was lower than others. For example, figure 4 shows the results of the tests for alternate move commands. The commands

Fig. 5. Graph of all active nodes in the Dixon ROS package.



"Go" and "Head" had a similarly high success rate with all 10 trials, for "backward" and "forward" successfully recognized. However, "move" had an anomalously low recognition rate.

VII. Limitations

A. Hardware Limitations

The main hardware limitation we ran into during development and testing was the microphone. We found that the quality of the microphone being used for voice to text can make or break a transcription. A lower quality microphone suffers from quite a limited range; The built-in microphone we used initially was only capable of a maximum range of 10 cm. Not only does the quality of the microphone make a difference, but the placement as well. The microphone being placed in or on the robot causes it to pick up vibrations of the robot moving and interferes with voice transcription.

In addition, most personal assitant products on the market currently that utilize voice transcription come equipped with an array of microphones that allows for accurate noise cancellation and voice detection from many angles at far distances. We, however, only used a single microphone during development and testing.

B. Recommendation for Future Hardware

If we continue to develop Dixon in the future, we plan on seeking out a better microphone setup. A high quality long range microphone placed far away from the mobile base seems like it would provide the best results.

C. Mapping Limitations

Our initial plans were to map out the second floor of the Cleveland State engineering building, however due to time constraints we were not able to do this. Instead, we mapped out the Cleveland State Robotics Laboratory and used placeholder global destination names ("alpha," "beta").

Considering Dixon can successfully navigate the laboratory using the placeholder names, it is safe to assume that if a map was provided of a larger area and destinations were defined, that Dixon would be able to navigate this new area as well.

D. Web-app Limitations

Due to time constraints, we were not able to implement a web app that can interface with Dixon so that users can provide voice commands remotely. In addition to time constraints, we are both fairly inexperienced with web development regarding implementing speech recognition, so the decision to forgo the

web app was made. This decision allowed more time to be dedicated to the core goals of the project.

VIII. Future Work and Conclusion

A. Future Work

The current state of Dixon works as a prototype voice command system that can be applied to mobile robots like a TurtleBot. In order to improve upon this project, various potential ideas can be explored. An improved parser with deeper semantic parsing and natural language processing can be introduced [8]. This will allow the robot to carry out complex tasks and be more accurate in its understanding of the tasks. Similarly, for improved task planning we can use a dialog system to get more information from the user as well as use a commonsense reasoning system to improve performance and accuracy [9]. Additionally, a web app can be developed that allows the issuing of commands remotely. Finally, improved hardware like a microphone array can enhance the speech recognition.

B. Conclusion

In conclusion, the goal of this project was to create a simple voice command system in order to give movement based commands to a TurtleBot. In addition, a web app that allowed a user to remotely issue commands to a TurtleBot was planned. The reasoning for these goals is that communicating via vocalized inputs is much more intuitive and easy to use than current methods, such as programming. Although other input methods for communication with robots can be equally as intuitive, we found that dealing with vocal inputs proved to be the most compelling, yet achievable route. Our primary goal of creating a simple voice command system was accomplished, and exists in a working prototype state. Our secondary goal was not completed in any sense, due to time constraints and inexperience, however.

Dixon consists of the pocketsphinx voice-to-text ROS package, a shallow semantic parser, a voice synthesization node that utilizes the sound_play ROS package, and a navigation node. In this report, we describe the technical details of these components, as well as the implementation of the system as a whole.

Acknowledgment

This project was done as part of the CIS 493 Autonomous Intelligent Robots class at Cleveland State University. We would like to that Dr. Shiqi Zhang for providing us the opportunity to work on this project.

References

[1] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, 2009.

[2] D. Huggins-Daines, M. Kumar, A. Chan, A. W. Black, M. Ravishankar, and A. I. Rudnicky, "Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 1, pp. I–I, IEEE, 2006.

[3] P. Blunsom, "Hidden markov models," *Lecture notes, August*, vol. 15, pp. 18–19, 2004.

[4] G. Grisettiyz, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 2432–2437, IEEE, 2005.

[5] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[6] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, no. 343-349, pp. 2–2, 1999.

[7] A. W. Black and K. A. Lenzo, "Building synthetic voices," *Language Technologies Institute, Carnegie Mellon University and Cepstral LLC*, vol. 4, p. 2, 2003.

[8] J. Thomason, S. Zhang, R. J. Mooney, and P. Stone, "Learning to interpret natural language commands through human-robot dialog.," in *IJCAI*, pp. 1923–1929, 2015.

[9] S. Zhang and P. Stone, "Corpp: Commonsense reasoning and probabilistic planning, as applied to dialog with a mobile robot.," in *AAAI*, pp. 1394–1400, 2015.