https://doi.org/10.2352/ISSN.2470-1173.2017.19.AVM-019
© 2017, Society for Imaging Science and Technology

Milpet – The Self-Driving Wheelchair

Samuel Echefu, Jacob Lauzon, Suvam Bag, Rasika Kangutkar, Amar Bhatt, Raymond Ptucha

Abstract

Over two million people in the United States rely on the use of a wheelchair to perform daily tasks. Joystick controls on motorized wheelchairs have improved the lifestyles of so many, but are of little value to the visually impaired or patients with restricted hand mobility. Often times these wheelchair users must rely on caretakers to assist them with their mobility, thus limiting their independence. Milpet is an effective access technology research platform centered around improving the quality of life of those confined to wheelchairs. By expanding Milpet's control interface to include speech recognition, those who cannot benefit from a joystick are given new freedoms. Utilizing a map of its environment, localization is performed using LiDAR sensor scans and a particle filtering technique. In addition to simple movement commands such as "turn left", "stop", and "go faster", the speech interface along with localization and navigation modules enable patients to navigate more complex commands. For example, commands such as "take me to the kitchen" instruct Milpet to autonomously drive to the specified location while avoiding walls and other obstacles. This self-driving wheelchair is a huge leap in improving the quality of life for the mobility impaired who cannot benefit from a joystick.

Keywords: A*, autonomous driving, obstacle avoidance, occupancy grid, omni-vision, LiDAR, localization, particle filters, ROS, speech recognition

Introduction

Whether living at home or an assisted living facility, wheelchair-bound patients oftentimes rely on caregivers or loved ones to assist them with getting around. Patients with medical conditions such as Parkinsons disease, visual impairment, brain injury, dementia, and even old age have a hard time operating the traditional joystick interface of most motorized wheelchairs. These patients run the risk of hurting themselves and others due to the possibility of collision with walls, stationary objects, and even other people. A caregivers role includes assisting with the mobility needs of these patients. Wheelchair users who cannot properly use a joystick need to rely on someone else for the sake of getting from one location to the next. This greatly lowers the amount of independence, and as one patient put it, her privacy. This paper introduces an effective access wheelchair research platform to investigate methods of improving the quality of life of these disabled users.

Milpet, or the Machine Intelligence Laboratory Personal Electronic Transporter, is an effective access wheelchair that aims to replace traditional joystick commands with an intuitive, user voice activated control system. Milpet is equipped with an omni-vision camera, a single channel LiDAR, sonar sensors, encoders, and a microphone. Milpet can be controlled wirelessly using Bluetooth through an Android phone application. The app includes a button to switch between Bluetooth manual control and autonomous mode. With the Bluetooth control mode, the user or their caretaker can manually drive the wheelchair. The latter autonomous mode gives the wheelchair user the freedom to control Milpet through human speech commands. Users can say directional commands such as "go forward" or "turn around" and Milpet will move accordingly.

When in autonomous mode, Milpet is provided with a map of its environment. Using the map, Milpet performs localization using vision and LiDAR sensor scans and a particle filtering technique. Given a trained localization model based on its environment, Milpet users are able to specify a destination for autonomous navigation using speech commands. The shortest path to the end destination is calculated using the A* path planning algorithm which then translates the corresponding movement commands back to the wheelchairs motor controllers. As Milpet takes the user to the specified goal location, it avoids obstacles using the LiDAR and sonar sensors. This potentially increases the safety for the wheelchair user. By allowing the user to simply say where they want to go, wheelchair patients who heavily rely on their caretakers now have an alternate and intuitive means of executing daily tasks.

In the following sections this paper will briefly go over related works, an overview of Milpet's system, our test setup, results, and the conclusion with future work.

Related Work

There have been many attempts at designing intelligent wheelchairs. The Mobility Aid for Elderly and Disabled People (MAid) [9] is a wheelchair that was designed to go through areas with heavy crowds and foot traffic. MAid specializes in obstacle avoidance through the use of laser scanners and dead reckoning navigation with the use of the wheel encoders attached to the wheelchair. The NavChair [14] assists the user by avoiding obstacles, performing automatic wall following, and assisted doorway traversal. The MIT intelligent wheelchair [16] uses sensors to perceive the wheelchairs surroundings and has a speech interface of its own to perform simple commands. The CoPilot [7] adds sensors to wheelchairs to assist with doorway traversal. It detects doorways and uses a sample based planner to safely traverse in and out of rooms. The Robotics Engineering Department at Worcester Polytechnic Institute developed a wheelchair [15] that uses voice control, facial expressions, and a Bluetooth joystick controller to interface with their wheelchair. Their system uses the Robot Operating System (ROS) as the software framework. ROS controls the user interfaces as well as navigation modules. The onboard sensors consist of infrared, sonar and a single channel laser scanning rangefinder. Some samples of these systems are shown in Fig. 1.

Speech recognition is a widely researched topic in the field of Human Computer Interaction (HCI). There are many speech-

Figure 1. *Effective access wheelchairs and accessories; (top left) MAid [9]; (bottom) Navchair [14]; (top right) CoPilot [7].*

based applications that inspired the Milpet speech interface. IWASR [13] is a Vector Quantization-based isolated-word automatic speech recognition system. It aimed to help customers call a central service center by responding in a natural manner. Speech recognition systems have military applications as well. The SCORPIO [6] is a small robot used for transportation and booby trap disposal. Operators can control the robot using a joystick interface and speech interface. The speech interface was built with a Hidden Markov Model system and uses a small language model and dictionary for the speech recognition. The speech recognition interface is useful for hands free operation which allows operators to handle multiple tasks. Speech Emotion Recognition (SER) [10] is another field in HCI that aims to enable natural interaction with computers by speaking. It also aims to detect and understand subtle clues such as micro expressions to determine emotion. This research helps to understand and tackle the needs of patients in therapeutic centers.

The common smartphone user has incorporated the use of speech recognition software for proprietary applications. Major corporations such as Google, Apple, and Microsoft have research labs dedicated to providing the end user with top of the line speech recognition engines on their consumer platforms. Voice assistants are widely used as means of performing normal functions on a smartphone just by using the sound of your voice. Voice Assistants such as Siri, Google Now, and Cortana grant IPhone, Android, and Windows phone users, respectively, the ability to interact with and use their phones to perform everyday tasks. They allow users to make phone calls, search the web, organize calendars, set alarms, and much more. In early 2016, Google publicly released their cloud speech API. This helped to improve efforts in boosting the Voice UI market [1]. Ruan et al. [11] reported that voice recognition is almost 3$\times$ faster than texting.

System Overview

Milpet is a re-purposed Quickie S-11 medical wheelchair with attached sensors as seen in Fig. 2.

The joystick interface was removed and replaced with a custom control system and laptop. Microcontrollers connect to the wheelchair motors for movement, and connect to the wheel encoders for movement data collection [17]. A LiDAR sensor, sonar sensors, and an omni-vision camera are connected to the wheelchair to collect information about the wheelchair's surroundings. A microphone is also mounted on the wheelchair for speech commands.

Hardware

The motor controllers used for Milpet are the Talon SR motor controllers (shown in Fig. 3), which are used for high performance speed control. These controllers can provide up to 100 amps of peak current and up to 60 amps of continuous current which is quite a bit more current than is required to move the each motor on the wheelchair. One Talon SR controller is used per motor for a total of two individual motor controllers. The power distributed to these motor controllers is controlled by a relay which only turns on once the control system is enabled and the safety button is released. This prevents the wheelchair from moving until the system is completely safe to move.

The control logic for the system comes from an Arduino Due microcontroller, which takes in movement commands from the Robot Operating System (ROS) and outputs a PWM signal to the motor controller in order to control the differential motors on the wheelchair. Sensor readings from several low cost sonar sensors (shown in Fig. 4) mounted to the outer edge of the wheelchair are also read. These sensors are mounted strategically to point in the cardinal directions about the wheelchairs center without interference from the user or chair. These sensors are used to detect obstacles, both stationary and dynamic, which is sent as an interrupt to the main controller to stop the motors or avoid the obstacle.

In order to allow the system to detect position and velocity, two separate rotary encoders (shown in Fig. 5) were installed to the back of each motor by removing the breaking mechanism of the motors and inserting the encoders directly onto the motors' geared axles. These encoders interface with an Arduino Due to record information of how far and how quickly each wheel of the wheelchair has moved in order to give the overall system a better idea of its speed and position over time.

A Hokuyo light detection and ranging (LiDAR) sensor, shown in Fig. 6, was mounted to the wheelchair in order to aid ROS in the creation of maps of the areas that the wheelchair travels in, as well as, to detect obstacles. The LiDAR itself provides a one-dimensional slice of the depth of its environment within a 270° field of view. The particular sensor provides millimeter accuracy up to a distance of 30 meters. The maps that are created using this sensor ultimately aid the wheelchairs autonomy in lo-

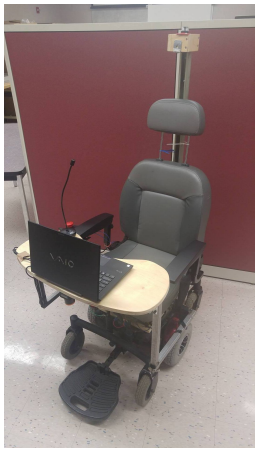

Figure 2. *Machine Intelligence Laboratory Personal Electronic Transporter (Milpet).*

Figure 3. *Talon SR Motor Controller*

Figure 4. *Low cost sonar sensor*

calizing its current position in order to allow it to navigate to a destination of interest for its user.

To power the chair and the multitude of components on it, a power board was created. This power board takes in 24V from the wheelchairs battery supply and converts it into 5V and 3.3V supplies using on board linear regulators, a 12V supply using an external isolated power converter, and a variable power supply. The 5V and 3.3V power lines are used to power the Arduino and any other small voltage, low current devices, such as the encoders. The 12V supply is used to power the under glow LED lights, car horn, and LiDAR. The variable supply is used to power a laptop running ROS and Pocketsphinx with an attached USB microphone.

Localization

Localization on Milpet is achieved using data from either the vision or LiDAR sensors. This data is fed into an adapted particle filter algorithm, similar to [18]. The localization algorithm uses the known map and outputs an (x,y) coordinate that represents the wheelchair's location on the map. The algorithm utilizes a Support Vector Machine (SVM) that must first be trained within the environment where it will deployed. The training process consists of dividing the map into ground truth locations, or waypoints. These waypoints are evenly spaced throughout the entire map and represent a series of classes that can be predicted by a classifier. The wheelchair must then be driven to each of these predefined

Figure 5. *Encoders used for velocity/position detection*

Figure 6. *Hokuyo Scanning Laser Range Finder (LiDAR*

waypoints to gather ground truth data from the sensors. The data must then undergo a series of pre-processing steps and is then used to train a linear, multi-class SVM to predict the waypoint based on the sensor input. The model is trained outside of the wheelchair, in MATLAB, using LibSVM and later transferred to the wheelchair for run-time use.

Once the model has been trained, an iteration of the adapted particle filter algorithm is run whenever there is new data from the sensor. The algorithm starts by randomly initializing X particles randomly throughout the valid spaces in the map. Then, given new sensor data, the data is read in, the same pre-processing that was applied during training is applied, and the data is passed to the pre-trained SVM. The multi-class SVM is able to return a probability for each waypoint which is used to calculate a weight for each particle. The particle weights are calculated with (1).

$$W_i = \sum_{i=1}^{k} \left(1 - \frac{d_i}{D}\right) * P_i; \text{ for } k \text{ nearest waypoints} \tag{1}$$

Equation (1) shows the weight of each particle W_i is calculated by selecting the k nearest neighboring waypoints, calculating their Euclidean distance d_i from the particle, and weighting the probability of that waypoint. The probability of the waypoints are predicted by the SVM as the distance d_i over the total distance, D, to all k classes. These weighted probabilities are then summed to set the weight for each particle. After the weights are calculated, a new batch of particles is sampled with replacement based on the new weights. Ninety-five percent of the particles are drawn in this way, however, the remaining 5% of particles are drawn from the original, random batch of particles. This is done to prevent the location estimate from converging to an inaccurate location. Using data from the encoders, the resampled particles are updated based on movement of the wheelchair since the last iteration. Finally, the location is estimated by creating a 2D mesh of the top 75% of particles and selecting the peak value of the mesh.

This algorithm was tested using data from two different sensors. Data from the omni-vision camera was tested in a MATLAB simulation and data from the LiDAR sensor was used for testing on the wheelchair. Both sensors had a different set of preprocessing steps, but the algorithm remained the same for each.

Omni-vision Camera Data

The omni-vision camera that Milpet is equipped with consists of a low-cost USB camera pointing upwards into a small 360° mirror. This is a much cheaper solution than a full 360° camera but results in a crude image that requires some extra pre-

processing. Fig. 7 shows the raw image from the camera as well as the resulting images after the first few pre-processing steps.

Figure 7. *Pre-processing steps of the omni-vision camera. Top left is the raw image, top right is the result of centering the image, and bottom is the unwarped image.*

An example of the raw images from the omni-vision camera is shown in the top left of Fig. 7. The first pre-processing step was to center the useful part of the image and crop out the remaining parts. This was done using a manual crop in MATLAB. Since the camera did not ever shift, the same crop could be applied to every image. The next step was to unwarp the image to achieve images like the example in the bottom of Fig. 7. Finally, the images were converted to grayscale and underwent histogram equalization to help mitigate differences in light levels. For dimensionality reduction, Locality Preserving Projections [5] was used. This algorithm was chosen because it is a semi-supervised learned algorithm that allows for the same reduction to be applied to both testing training images. The reduced images are used to train the SVM to be utilized in the particle filter algorithm.

LiDAR Data

A LiDAR scan consisted of a 1081 length vector of range values that covered 270° in front of the wheelchair. At each ground truth location three scans were taken: one facing forward, one facing slightly to the right, and one facing slightly to the left. These scans also required some pre-processing before passing them through the SVM. Each scan contained a few spikes where the sensor did not get a reading. These spikes were removed and replaced with an adjacent valid scan value. In addition, the mean value of each scan was subtracted from the entire scan. Gaussian noise ($\mu = 0, \sigma = 0.01$) was added to each scan to create five additional scans to expand the training data set. Fig. 8 shows the LiDAR scan before and after the pre-processing stage.

Navigation

The A* algorithm is a very common algorithm often used in navigation of autonomous robots and even other domains like networking to find the best route between two stations. It is a

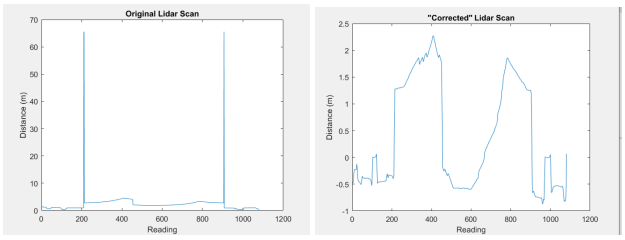

Figure 8. *LiDAR scans before (left) and after (right) pre-processing.*

corollary of the famous Dijkstra's algorithm. One of the famous applications of A* can be seen in the navigation planning of the Google smart car using Google maps with a lot of other sensor data incorporated inside the algorithm. In this paper, the algorithm has been modified and implemented in a similar essence of the Google self-driving car. Given a known map, the algorithm is used to determine the path from the wheelchair's current location to the coordinates corresponding to the users desired destination. Different locations on a map can be represented as graph nodes. The A* implementation uses a best-first search and finds the least cost path to get from a starting location to an ending destination. A* proceeds through the least cost path while keeping a sorted priority queue of different path segments. The A* cost function is shown in (2).

$$f(x) = g(x) + h(x) \tag{2}$$

The algorithm uses a knowledge-plus-heuristic cost function of node x (usually denoted by $f(x)$) to determine the order in which the search visits nodes in the tree. The cost function is a sum of two functions:

- The past path-cost function, which is the known distance from the starting node to the current node x (usually denoted by $g(x)$).
- A future path-cost function, which is an admissible *heuristic estimate* of the distance from x to the goal (usually denoted by $h(x)$). The heuristic function used in this paper is the Manhattan distance.

Joshi et. al [3] made a revision to the A* algorithm to include keeping track of the steps taken to get to the initial node. By having each node keep track of its previous node, the steps taken to get to the final destination can be obtained by backtracking from the destination after executing the full algorithm.

This algorithm was modified for the use of navigating a mobile robot and was adopted for use in Milpet. The major change to the algorithm was limiting the possible directions Milpet could go to while traversing through nodes. In Joshi et al.s study, the robot was able to travel in four directions: forward, backward, left, and right. However, in this paper backwards movement was eliminated due to the risk factor associated with a wheelchair moving in reverse. Hence the cost function was created as *[2, 1, 1]* corresponding to left, forward and right turn. The weight corresponding to turning turning left was given a higher value following the US system, where vehicles travel on the right side of the road, thereby making the right turn easier compared to the left turn. Based on this fact, the wheelchair should always try to calculate its path keeping the cost minimum and always choosing a right

turn over a left turn whenever possible. The algorithm was also expanded to take 3D input to consider the wheelchair pose along with the x and y coordinates of its location. These limits and modifications were made while keeping the goal of obtaining a shortest path to any end destination while keeping the path at a minimum cost.

Several trials were conducted both for the mapping and the navigation. In most of the cases the obstacles were mapped correctly on the 2D map. The wheelchair was also able to reach the end goal on the grid based on both user defined positions and predefined goal positions on the map in such cases. A simple 2D grid map is shown in Fig 9, where 0s represent free space for navigation and 1s are grid cells occupied by obstacles. The calculated path from the algorithm has also been shown in Fig 10.

```
[[0, 0, 1, 0, 0, 0],
 [0, 0, 1, 0, 0, 0],
 [0, 0, 0, 0, 1, 0],
 [0, 0, 0, 1, 0, 0],
 [0, 0, 0, 1, 0, 0]]
```

Figure 9. *5x6 2D binary grid map*

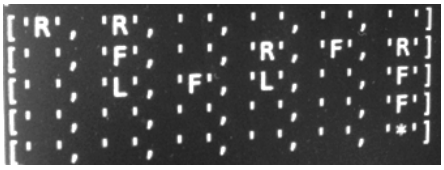

Figure 10. *Calculated path by the A* algorithm on the 2D map in Fig. 9 showing the move directions from the (1,1) grid cell as the initial position to the goal position *.*

Mapping

Milpet uses a two dimensional occupancy grid to localize itself. In an occupancy grid, the world (area to be navigated) is represented as an array where each cell of the array is a possible position. Generally, white spaces show free space where the robot can travel whereas black spaces show obstacles i.e. where the robot cannot go. Since homes, malls, etc have blueprint images readily available, Milpet makes use of these instead of building its own map as in Simultaneous Localization and Mapping (SLAM). By providing a bitmap to the map node, two occupancy grids are created by it: one for localization and one for path planning. The localization map is used by the localization-node and the path planning map is used by the navigator-node.

Since a blueprint image is basically a 2-D array with a birds-eye view of the world, it is easily related to an occupancy grid. In the map node, the bitmap is first filtered with a threshold at 100, so that all the pixels are either black (0) or white (255). The input resolution of the image is set such that the value of one pixel represents $w \times w$ sq. distance in the physical world. The required output resolution $y \times z$ sq. distance is set as well. If the input resolution is finer than the output resolution the map shrinks and vice versa. Before changing resolutions we must make sure that each cell can be $y \times z$, i.e. *mod(y,w)* and *mod(z,w)* should be zero, if it is not, the array is padded with zeros (black) to make it valid before being sent for conversion. The formulas (3-6) are used for calculation of the required padding and the updated input array.

$$h = (-1 * mod(iC, oRes) + oRes \tag{3}$$

$$inputPadMat = hCat(inputMat, zeros(iR, h)) \tag{4}$$

$$v = (-1 * mod(iR, oRes) + oRes \tag{5}$$

$$inputPadMat = vCat(inputMat, zeros(iC, v)) \tag{6}$$

where,
h = amount of padding needed horizontally
v = amount of padding needed vertically
iC= number of columns in input array
iR = number of rows in input array
$oRes$= resolution of output array
$zeros(w,y)$ = matrix of w by y dimensions
$mod(x,y)$ = returns remainder of x / y
$hCat(n,m)$ = horizontal concatenation of n, m arrays
$vCat(n,m)$ = vertical concatenation of n, m arrays
$inputMat$ = input array without padding
$inputPadMat$ = input array after padding

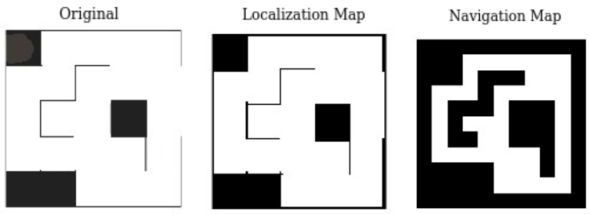

Figure 11. *Occupancy grids at various scales.*

For localization with particle filters, a resolution of 4 inches is needed, so the output resolution for the localization map was 1 pixel (px) corresponding to 4×4 sq inches. Meanwhile, the navigator-node makes use of the A* algorithm, where each cell represents a position of the robot. Thus, one cell should be the size of the robot. Milpet measures to approximately 36×36 sq. inches. Consequently, 36×36 is the output resolution for the navigation map. In Fig. 11, the original map with input scale 1, localization map with scale 4, and navigation map with scale 36 is shown, where the term scale means $1px = y \times y$ sq. inches. As the scale increases the black area increases; small obstacles get bloated up as the scale is increased.

Speech Recognition

Automatic Speech Recognition (ASR) is a method by which people can communicate with computers using their voice. ASR transforms speech into text for computers to parse and react depending on what phrase was said. There are two kinds of speech recognition approaches: Isolated-word and continuous. Isolated-word speech recognition systems operate on single words at a time, requiring a pause between each word. Continuous speech recognition systems operate on speech segments where words are connected together, i.e. not separated by pauses. Continuous systems are typically more difficult to implement because there is a bigger challenge in finding the start and end points of words, but

are ideal for interacting with robots in a natural manner. The rate at which the user is speaking also has a great effect on recognition accuracy for continuous systems

Milpet has an offline speech requirement to remove the internet connection dependency. There is currently a wide variety of open source options that tackle the problem of offline speech recognition [8] [4] [2] [12]. Pocketsphinx [2] is a continuous speech recognizer developed by Carnegie Mellon University. Pocketsphinx can interpret files with continuous voice audio as well as perform real-time recognition when using a microphone input. To operate, Pocketsphinx requires a language model and a dictionary which can be generated using CMUs Sphinx Knowledge Base Tool. The tool requires a corpus of all the words your robot should be able to interpret. Pocketsphinx is installed in the Milpet system and uses a custom made dictionary and language model to perform directional and navigation speech commands.

Goal Interpretation

There are three main types of speech commands that Milpet can understand and act on: directional, speed, and location.

1. Directional commands are used to perform simple immediate actions on the wheelchair. These commands consist of the words "move", "go", "turn", and "face" partnered with a corresponding direction.
2. Speed commands on Milpet are used to either stop, slow down, or speed up the wheelchair's movement speed. Users can either specify a speed to set the wheelchair too (fast, mid, or slow), or say "speed up", "slow down", and "stop" to perform each command.
3. Location commands require a known destination on the map for navigation purposes. By saying a location command such as "take me to the kitchen" or "go to the bedroom", Milpet extracts the goal location from the sentence, then proceeds to find the most efficient path to get from its current location to the end goal.

Robot Operating System

Robot Operating System (ROS) is an open source operating system for robots and is the means by which the different components on Milpet can communicate with each other. Each main system component has a ROS node, or executable program, associated with it as shown in Fig 12.

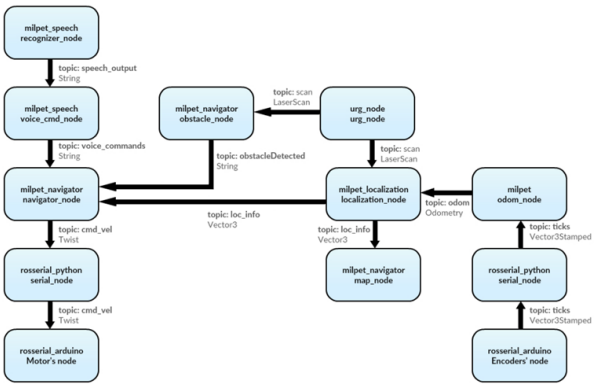

Figure 12. *Milpet ROS Architecture.*

These nodes can publish data, collect data that other nodes publish, and execute certain actions based on the state of the system. ROS topics are the channels for the data to be published or received between nodes. The data is transformed in the form of ROS messages that contain the relevant data. The major ROS nodes for the Milpet system are as follows:

- Recognizer_node - Starts Pocketsphinx and listens for incoming commands from the microphone to interpret. Once interpreted, the sentence is converted into a character array and is published as a string on the "speech_output" topic. The process is then repeated for new utterances in the microphone.
- Voice_cmd_node - Searches the command string for valid commands that Milpet can use to move. The node then creates a simplified set of commands that are published to the "voice_cmd" topic in an array of string commands.
- Encoder_node - Publishes "ticks" counted from motor encoders. There are 20,000 ticks per wheel revolution. An Arduino Due microcontroller is connected to both the encoders and the wheelchair's laptop system. It transmits tick count information via serial connection.
- Serial_node - Establishes the serial connection with the encoder Arduino Due and the laptop computer on board. This allows the tick information gathered from the encoders to be published on the "ticks" topic.
- Odom_node - Calculates the linear and angular displacement of the wheelchair based on the information published on the "ticks" topic. To calculate these values which are published on the "odom" topic (7)-(10) are used. Where Δs is the overall change in displacement of the wheelchair, Δs_r and Δs_l are the distance traveled by right and left wheels, $\Delta\theta$ is the change in wheelchair pose, b is the width of the wheelchair, and Δx and Δy are the position change in global coordinates.

$$\Delta s = \frac{\Delta s_r - \Delta s_r}{2} \tag{7}$$

$$\Delta\theta = \frac{\Delta s_r - \Delta s_r}{b} \tag{8}$$

$$\Delta x = \Delta s * cos(\theta + \frac{\Delta\theta}{2}) \tag{9}$$

$$\Delta y = \Delta s * sin(\theta + \frac{\Delta\theta}{2}) \tag{10}$$

- Urg_node - Interfaces with the LiDAR sensor. This node publishes LaserScan messages to the "scan" topic. The messages contain start and end angles of the LiDAR scan, the angular distance between measurements, the time increment between measurements, the minimum and maximum range values, and a vector of intensities.
- Localization_node - Uses the published LaserScan and Odometry messages along with the localization approach mentioned in the System Overview Localization section to determine the wheelchair's location. Once localized, this node publishes the x coordinate, y coordinate, and pose on the "loc_info" topic. The regions of detection for the LiDAR are shown in Fig. 13.
- Map_node - Subscribes to the "loc_topic" and visualizes the wheelchair's current location on a displayed map of the area.

- Obstacle_node - Reads the "scan" topic and determines if an obstacle enters a 2-foot radius surrounding the edge of the wheelchair. The corresponding direction, as shown in Fig. 13, is published as a string on the "obstacleDetected" topic.
- Navigator_node - The main node used to determine what the wheelchair should be dong at any given point in time. This node listens on the voice commands, localization info, and obstacle directions to successfully execute the voice commands without bumping into walls or other people. This node is in charge of sending the motor commands necessary to reach the users end destination. Linear and angular movement values are published using twist messages to the "cmd_vel" topic.
- Motor_node - Reads the "cmd_vel" commands sent from the navigator and interfaces with an Arduino Due microcontroller to move the motors on the wheelchair.

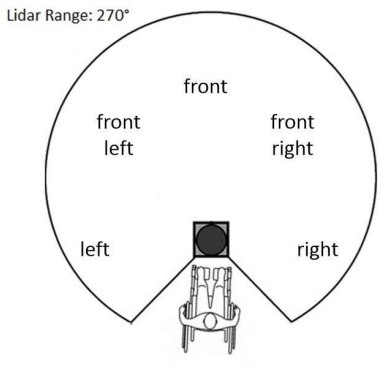

Figure 13. *Obstacle detection zones.*

Testing and Results

Test Setup

To verify the wheelchairs functionality, a mock home environment of 18' by 18' was set up and the wheelchair was placed randomly in the enclosed space. Fig. 14 shows a scaled down version of the test area with three labeled destinations. Destinations 1-3 were assigned the labels "Kitchen", "Bedroom", and "Bathroom" respectively. The white spaces on the test map are valid locations for the wheelchair to be in, while the black areas are invalid and blocked off by walls. Cardboard boxes were placed as obstacles in various locations around the mock home. This was done to ensure that each location on the map had a unique LiDAR signature and to observe the wheelchairs reaction to sensed obstacles while navigating. The red boxes in Fig. 15 correspond to the locations of the cardboard boxes in the mock home setup.

The multi-class SVM in the localizer node was trained on 42 location-pose combinations whose LiDAR scans were taken at various spots within the map. The trained model included up to four poses per class location depending on the most probable locations the wheelchair could face at the given spot. The particle filter algorithm was initialized to 500 particles in its initial pool and set to a 5% resampling rate. A series of directional and navigation speech commands were given to Milpet to verify its functionality.

Navigation Results		**Goal**		
		Kitchen	*Bedroom*	*Bathroom*
Start	*Kitchen*	3/3 = 100%	3/3 = 100%	3/3 = 100%
	Bedroom	2/3 = 67%	3/3 = 100%	3/3 = 100%
	Bathroom	2/3 = 67%	3/3 = 100%	2/3 = 67%
	Center	3/3 = 100%	3/3 = 100%	3/3 = 100%

Results

Overall, the system worked well when using the speech commands. When saying simple directional speech commands, the wheelchair acted accordingly. As the wheelchair moved forward, it steered clear of any obstacles blocking its path. If an obstacle was detected dead center of the front of the wheelchair, and there wasnt time to move in another direction to avoid it, the wheelchair stopped itself. This feature is very useful in case a moving object or person moves in front of the wheelchair.

Navigating from one destination to another was also successful. Milpet was able to compute the shortest path from its own location to the specified end goal. Once the path was determined, the wheelchair proceeded to move towards the goal while avoiding obstacles. Table 1 shows a subset of the navigation tests and results.

Although the system worked well overall, there were still a few navigation failures during test runs. One failure happened when going from the bedroom to the kitchen. The cardboard box representing obstacle 1 in Fig. 15 was hit when the wheelchair was attempting to avoid it. This was due to the wheelchair's turning radius being too small. Another problem occurred when trying to navigate from the bathroom to the kitchen. Obstacle 1 was also hit during this instance because the wheelchair started swerving to the left as it approached the end of the hall. Without some form of PID control, the wheelchair often finds itself off center of its path. Usually the wheelchair is still able to make it to it's goal destination while avoiding obstacles, but in some cases, the off

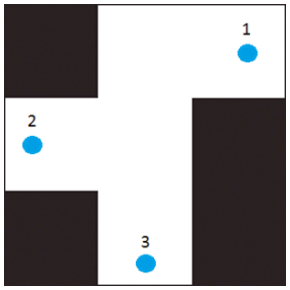

Figure 14. *Test map for Milpet navigation.*

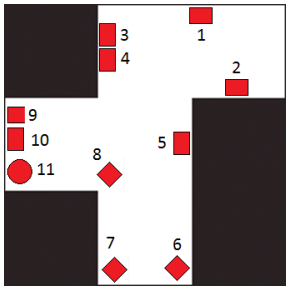

Figure 15. *Test map for Milpet navigation with detonated box locations.*

center wheelchair placement hurts the wheelchair's overall ability to navigate without hitting any obstacles.

Conclusions and Future Work

Autonomous wheelchairs are becoming more popular as research is done to improve the lives of those with disabilities. Autonomous wheelchairs are extremely beneficial for those confined to wheelchairs because they help to bring back independence to those who usually rely on caregivers to assist with mobility. The new framework implemented on Milpet ties multiple input sensors (encoders, LiDAR, and Omni-vision), localization (particle filters), path planning (A*), and output motor movement to demonstrate that autonomous wheelchair navigation is possible.

This paper is a small step towards giving independence and privacy back to wheelchair users. Future iterations of Milpet include increasing the set of commands that are used to control Milpet through its speech recognition interface, switching to an incremental speech model to account for broken speech patterns, tuning the speech model to handle a variety of accents, automating the process of acquiring maps for new areas, and updating existing maps if new obstacles or structures are found using the on board sensors.

References

[1] Huw Geddes. Voice ui market boosted by googles decision to go public ..., Apr 2016.

[2] David Huggins-Daines, Mohit Kumar, Arthur Chan, Alan W Black, Mosur Ravishankar, and Alexander I Rudnicky. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I. IEEE, 2006.

[3] Heramb Nandkishor Joshi and JP Shinde. An image based path planning using a–star algorithm.

[4] Akinobu Lee, Tatsuya Kawahara, and Kiyohiro Shikano. Julius—an open source real-time large vocabulary recognition engine. 2001.

[5] X Niyogi. Locality preserving projections. In *Neural information processing systems*, volume 16, page 153. MIT, 2004.

[6] Stanislav Ondas, Jozef Juhar, Matus Pleva, Anton Cizmar, and Roland Holcer. Service robot scorpio with robust speech interface. *International Journal of Advanced Robotic Systems*, 10, 2013.

[7] Tom Panzarella, Dylan Schwesinger, and John Spletzer. Copilot: Autonomous doorway detection and traversal for electric powered wheelchairs. In *Field and Service Robotics*, pages 233–248. Springer, 2016.

[8] Daniel Povey, Mirko Hannemann, Gilles Boulianne, Lukáš Burget, Arnab Ghoshal, Miloš Janda, Martin Karafiát, Stefan Kombrink, Petr Motlíček, Yanmin Qian, et al. Generating exact lattices in the wfst framework. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4213–4216. IEEE, 2012.

[9] Erwin Prassler, Jens Scholz, and Paolo Fiorini. A robotics wheelchair for crowded public environment. *IEEE Robotics & Automation Magazine*, 8(1):38–45, 2001.

[10] S Ramakrishnan and Ibrahiem MM El Emary. Speech emotion recognition approaches in human computer interaction. *Telecommunication Systems*, 52(3):1467–1478, 2013.

[11] Sherry Ruan, Jacob O Wobbrock, Kenny Liou, Andrew Ng, and James Landay. Speech is 3x faster than typing for english and mandarin text entry on mobile devices. *arXiv preprint arXiv:1608.07323*, 2016.

[12] David Schlangen and Gabriel Skantze. A general, abstract model of incremental dialogue processing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 710–718. Association for Computational Linguistics, 2009.

[13] Mohammad AM Abu Shariah, Raja N Ainon, Roziati Zainuddin, and Othman O Khalifa. Human computer interaction using isolated-words speech recognition technology. In *Intelligent and Advanced Systems, 2007. ICIAS 2007. International Conference on*, pages 1173–1178. IEEE, 2007.

[14] Richard C Simpson, Simon P Levine, David A Bell, Lincoln A Jaros, Yoram Koren, and Johann Borenstein. Navchair: an assistive wheelchair navigation system with automatic adaptation. In *Assistive Technology and Artificial Intelligence*, pages 235–255. Springer, 1998.

[15] Dmitry A Sinyukov, Ran Li, Nicholas W Otero, Runzi Gao, and Taşkın Padir. Augmenting a voice and facial expression control of a robotic wheelchair with assistive navigation. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1088–1094. IEEE, 2014.

[16] Matthew R Walter, Sachithra Hemachandra, Bianca Homberg, Stefanie Tellex, and Seth Teller. A framework for learning semantic maps from grounded natural language descriptions. *The International Journal of Robotics Research*, 33(9):1167–1190, 2014.

[17] Kyle Wieszchowski, Amar Bhatt, Michael Dushkoff, Samuel Echefu, Maria Shoaib, and Raymond Ptucha. *IEEE Western NY Image & Signal Processing Workshop*. 2015.

[18] Adelia Wong, Mohammed Yousefhussien, and Raymond Ptucha. Localization using omnivision-based manifold particle filters. In *SPIE/IS&T Electronic Imaging*, pages 940606–940606. International Society for Optics and Photonics, 2015.

Author Biography

Raymond Ptucha is an Assistant Professor in Computer Engineering and Director of the Machine Intelligence Laboratory at Rochester Institute of Technology. His research specializes in machine learning, computer vision, and robotics. Ray was a research scientist with Eastman Kodak Company where he worked on computational imaging algorithms and was awarded 30 U.S. patents with another 19 applications on file. He graduated from SUNY/Buffalo with a B.S. in Computer Science and a B.S. in Electrical Engineering. He earned a M.S. in Image Science from RIT. He earned a Ph.D. in Computer Science from RIT in 2013. Ray was awarded an NSF Graduate Research Fellowship in 2010 and his Ph.D. research earned the 2014 Best RIT Doctoral Dissertation Award. Ray is a passionate supporter of STEM education and is an active member of his local IEEE chapter and FIRST robotics organizations.

Samuel Echefu is a recent BS/MS Computer Engineering graduate from Rochester Institute of Technology. He recently completed his master's thesis on Speech Based Navigation Interface for Smart Wheelchairs, and has begun working for Microsoft.

Jacob Lauzon is a 6th year graduate student pursuing his BS and MS degree in Computer Engineering at Rochester Institute of Technology. His research focus is with indoor agent localization in combination with machine learning.

Suvam Bag is pursuing his MS in Computer Engineering from the Rochester Institute of Technology, with a special focus on machine learning and self-driving cars. He is doing his master's on finding a novel

method of deep learning localization for self-driving cars.

Rasika Kangutkar is currently pursuing a Computer Engineering Master's degree as a student at Rochester Institute of Technology. Obstacle perception, obstacle avoidance, and applying deep learning techniques for autonomous agents are her focus areas.

Amar Bhatt is in his 5th year pursuing his BS and MS in Computer Engineering from the Rochester Institute of Technology. He has a special focus on deep reinforcement learning, especially in the realm of robotics.