Simulating Motor Impairments in ROS

EECS 399 – Professor Argall

Henry Spindell

14 June 2013

Abstract

There are many patients who need assisted transportation, but whose physical impairments impede their ability to smoothly operate a wheelchair. The goal of this project is to simulate different impairments through ROS, and blend the impaired user command with the automated planner's command to varying degrees, depending on the immediate local environment of the robot. This report focuses on bradykinesia and episodic tremor as the two main impairments, but the skeleton of the project allows for easy creation and insertion of any disability simulation. The blending function is intended to give the user some degree of freedom while ensuring avoidance of obstacles and guidance towards the goal position.

I. Introduction

There are several disabilities that can impair motor skills, but for the sake of time and accuracy I have focused on two here – bradykinesia and episodic tremor. The framework of the ROS packages (Figure 1) created certainly support plugging in any other impairment simulations to be developed in the future.

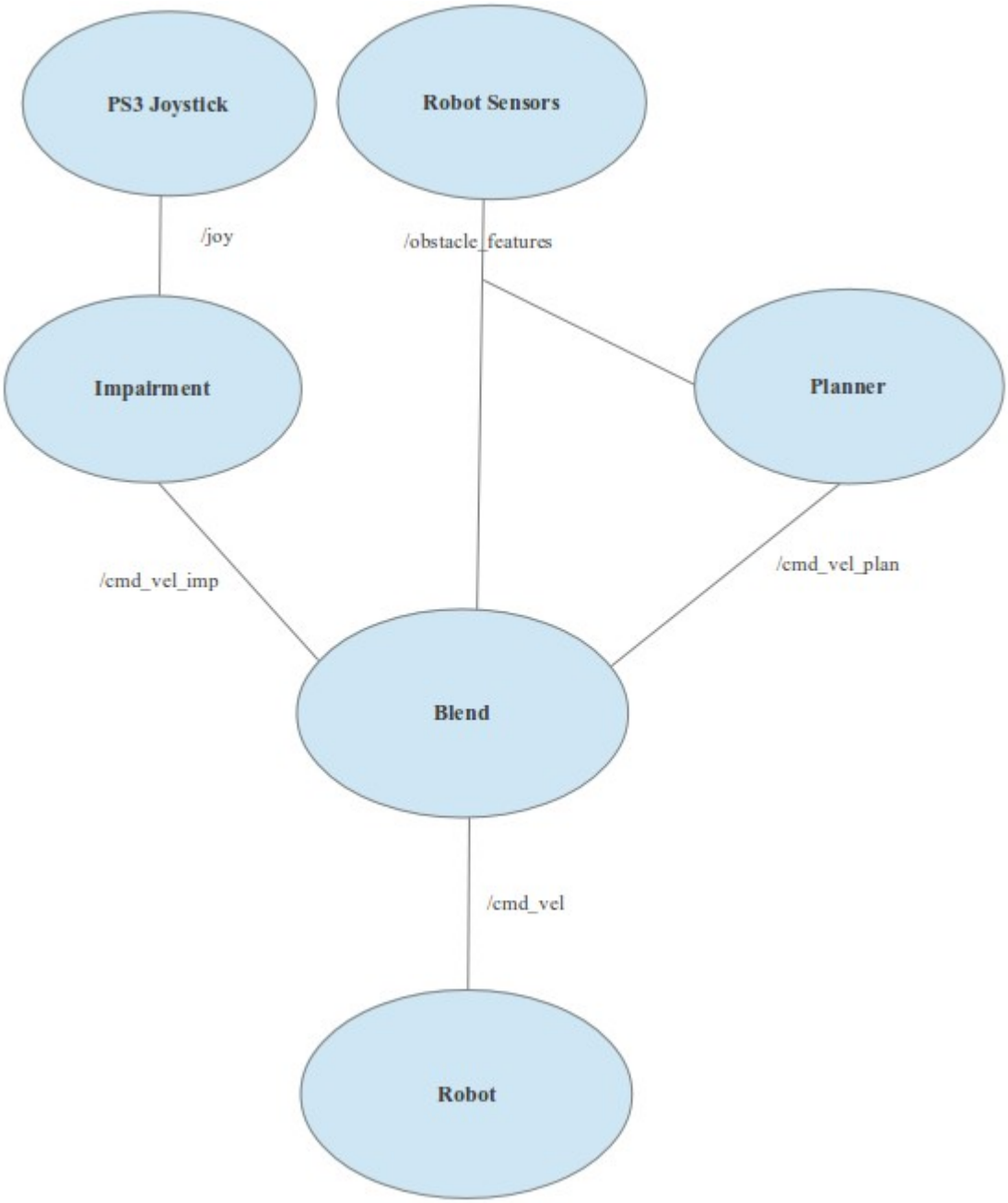
Bradykinesia, commonly linked with Parkinson's disease and other disorders of the basal ganglia, is characterized by extreme slowness in the execution of movement. The basal ganglia is an area of the brain which is responsible for voluntary motor control, and is believed to play a role in the "decision of which of several possible behaviors to execute at a given time"[i]. A patient with bradykinesia would hypothetically have trouble accelerating the wheelchair with the joystick.

Tremor is a voluntary semi-rhythmic contraction/relaxation of muscles resulting in twitching of one or more body parts. There are several variations of tremor. One of these is Parkinsonian tremor, a condition seen in at least 80 percent of PD patients[ii] and mostly characterized by a resting tremor, or inability to be still. The frequency of resting tremor is typically in the range of 3-7Hz. Most cases of Parkinsonian tremor are onset in a single limb after the age of 60, and progress to affect other parts of the body as well[iii]. Essential tremor (ET), on the other hand, is the most common movement disorder with a prevalence of 40-390 per 100,000 people.[iv]. ET involves both postural and kinetic action tremor in the range of 4-12Hz, and occurs only when then affected muscle is exerting effort. Postural tremor means it occurs when the affected limbs are voluntarily held up against gravity (e.g. a "pose"). Kinetic tremor applies to both goal-directed and non-goal-directed movements. ET mostly affects the arms, but can also affect other parts of the body such as the neck, head and legs. Among PD patients with resting tremor, 25 percent also have an associated action tremor[v]. Kinetic action tremor is what I have chosen to imitate for this project, because it is the most relevant and potentially most problematic type of tremor for operating a wheelchair.

Other impairments that I considered emulating include fatigue and rigidity. Fatigue can make navigating long distances difficult for a wheelchair user, and lead to collisions if the user cannot quickly respond to an obstacle. There are three types of rigidity identified with Parkinson's disease[vi]. 'Leadpipe' rigidity is when the muscle feels a constant sustained resistance to passive movement throughout the whole range of motion. 'Cogwheel' rigidity is a combination of leadpipe rigidity and tremor which results in a jerky resistance to passive movement as muscles tense and relax. A third type, spasticity, is present only at the start of passive movement,

and only elicited upon a high speed movement. It is easy to see how various types of rigidity could make operating a wheelchair difficult.

Figure 1 *A simplified graph representation of the relations between ROS nodes*



II. Methods

To develop the bradykinesia node, I used a command queue that intercepted commands from the PS3 joystick and split it into two commands, taking the average between the last command and the newest command to create the intermediate one. This creates the effect of doubling the number of commands published, and

therefore the robot/joystick takes twice as long to accelerate or complete a full movement. Not being able to find any hard data on the topic, choosing two as the dampening factor was a bit of an estimate on my part, but it would not be overly complicated to change this should some evidence to the contrary be found.

I modeled the tremor controller on a sine wave with an amplitude A of 0.50 – corresponding to one quarter of the joystick's range, purely as an estimate, for lack of solid evidence – and a period of 7Hz, so as to be in the middle of the range of essential tremor discussed above. However, I have parameterized amplitude, orientation and frequency in ROS to provide flexibility for different types and severities of tremor.

$$\text{tremor} = A(\sin(wt))$$

In this case, t is determined by when ROS receives the message, and the tremor is applied to the joystick message as a constant, not proportionally to the user's command. This is because essential tremor is not more or less severe based on the magnitude of the attempted action. I made sure to control for the edge cases, enforcing the minimum and maximum joystick command values whenever the applied tremor violates a boundary. From my research I could not find a definite answer on the direction of tremor, but either way it depends on how the user holds the joystick. For flexibility's sake, I applied lateral and vertical tremor in separate trials. From watching videos of people with essential tremor write with a pencil, my guess would be that the majority of tremor is lateral.

An appropriate blending function should utilize the planner's commands (v_u), but should not completely forgo the freedom of the user (v_p). For the purposes of this project and for the sake of simplicity, I chose to blend the two based on the robot's distance from the nearest obstacle.

$$v_b = \alpha v_p + (1-\alpha) v_u$$

In this function, the command that is executed, v_b, is the sum of two weighted components. This weight is represented by $0 \leq \alpha \leq 1$, where the user assumes full control of the robot when $\alpha = 1$, and only the input of the planner is considered when $\alpha = 0$. A sensor node publishes the metric distance of the nearest obstacle, and the value of α is computed by the blending node by normalizing this distance on a scale from touching an obstacle to having excessive free space. Additionally, the user is given complete control whenever there is no plan to be executed, or when the goal has been reached.

First, I tested each of the impairment controllers on an empty environment with the simple command to go forward at the maximum command speed for 12 seconds, and compared the results with a non-impaired control case. I created a node to issue the command automatically so as to have no variation, as might occur if I were operating the joystick myself.

To test the blending function, I added some obstacles to an environment to create a very narrow hallway along the path to the goal. To highlight the capabilities of the blending function, I used the most disruptive impairment possible in lateral tremor. To further complicate things, I set the amplitude parameter to 1.0, which is equivalent to the full range of the joystick. The idea is that for someone suffering from tremor in their hands or arms, this hallway would be virtually impossible to navigate through successfully. I exaggerated the conditions for the evaluation to highlight the effect of the blended controls. I also opted to control the joystick myself rather than publishing a command automatically through ROS, because it makes the user commands even less reliable.

III. Results (Part 1 – Impairment Controller Tests)

Figure 2 *Final position values from topic /odom*

Controller	**position.x (grid cells)**	**position.y (grid cells)**	**orientation.z (radians)**
No impairment*	3.74	0.00	0.00
Bradykinesia	3.63	0.00	0.00
Vertical tremor	3.09	0.00	0.00
Lateral tremor	3.61	0.28	0.10
Avg. Error (from *)	0.21	N/A	N/A

Figure 3.1 *The starting position and environment for the basic controller tests (below)*

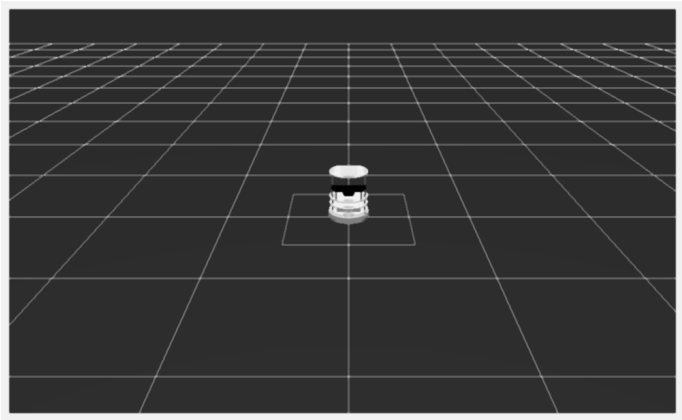
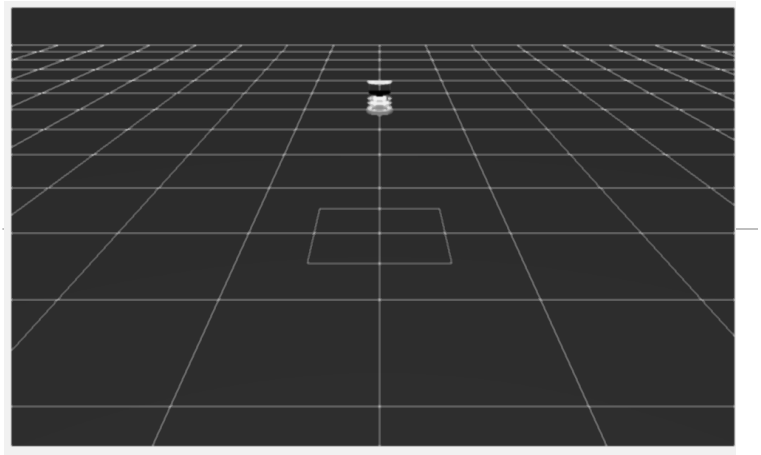
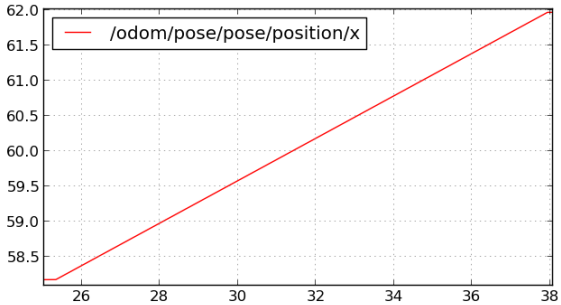


Figure 3.2 (a-c) *Unimpaired*





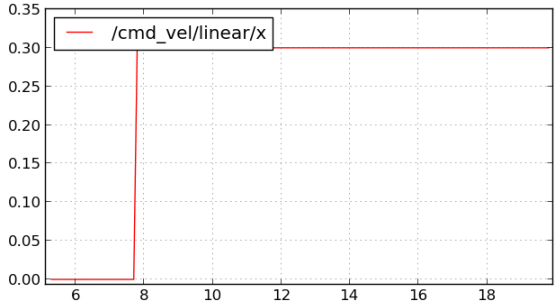
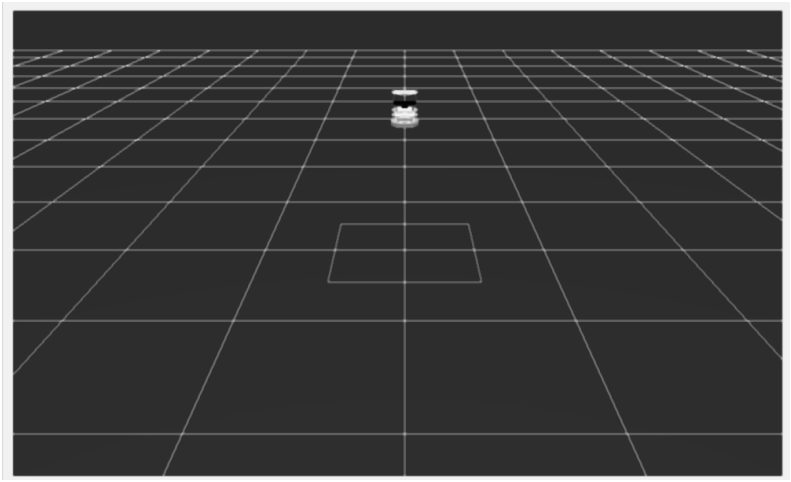
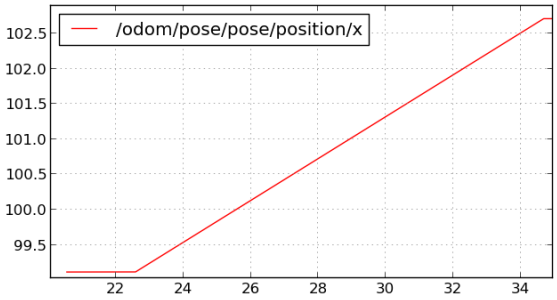


Figure 3.3 (a-c) *Bradykinesia*





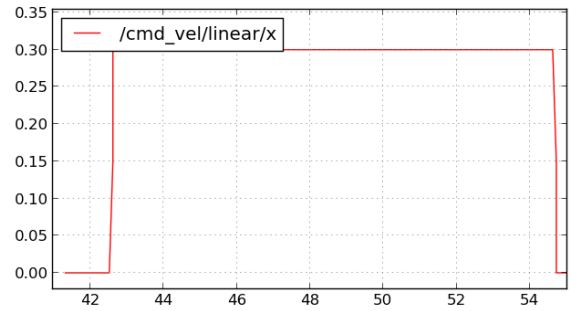
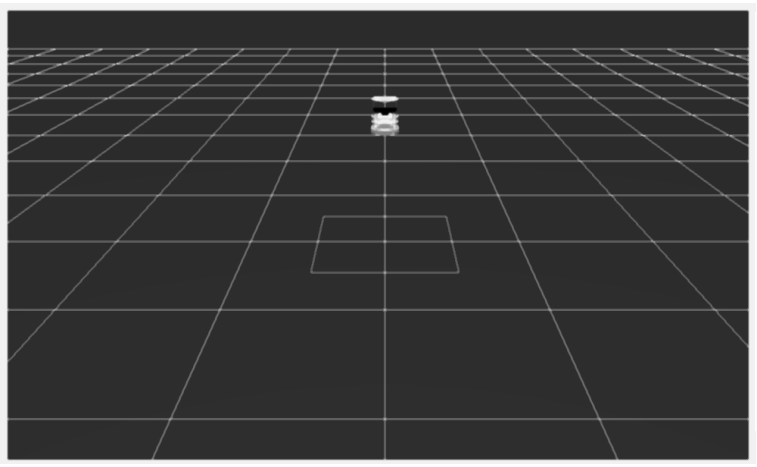
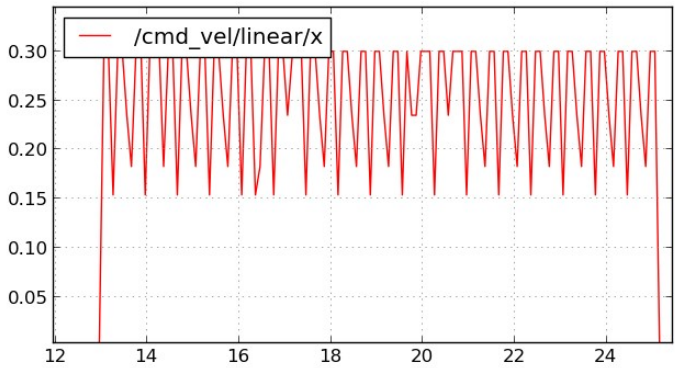


Figure 3.4 (a-c) *Vertical tremor*





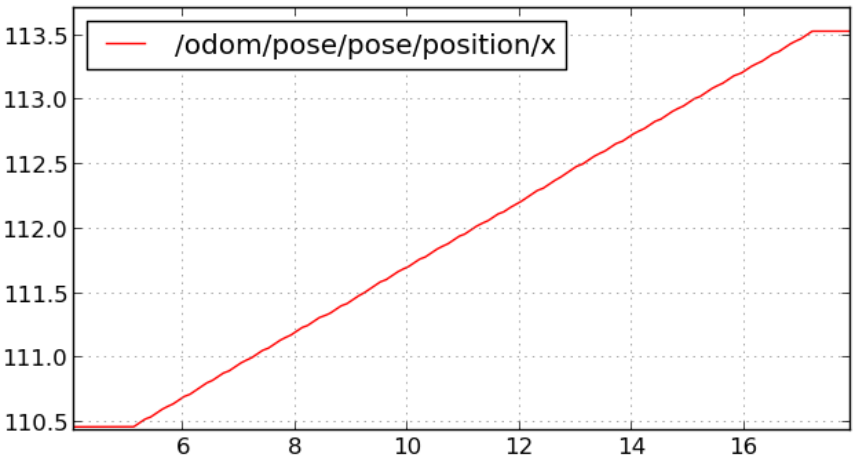
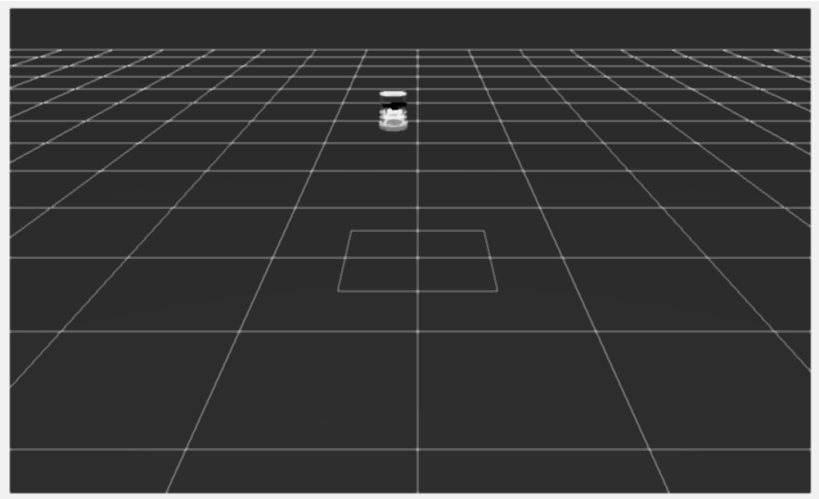
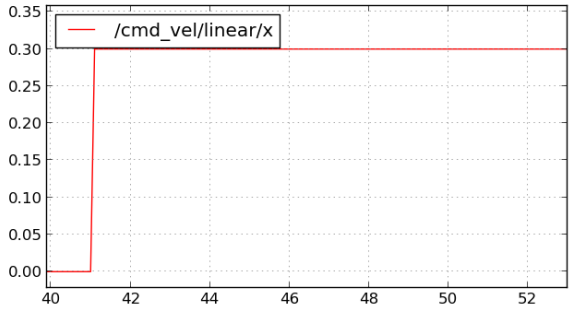
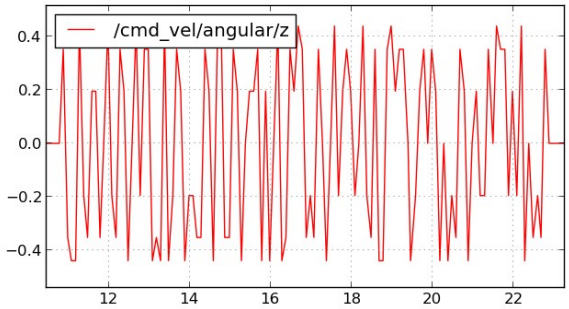
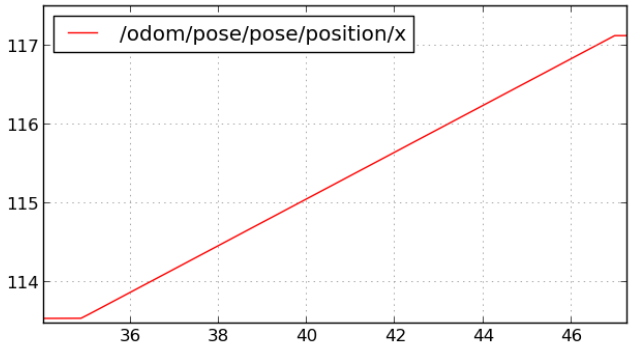


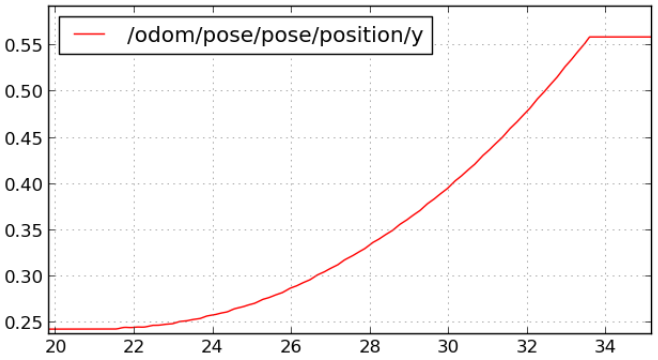
Figure 3.5 (a-f) *Lateral tremor*

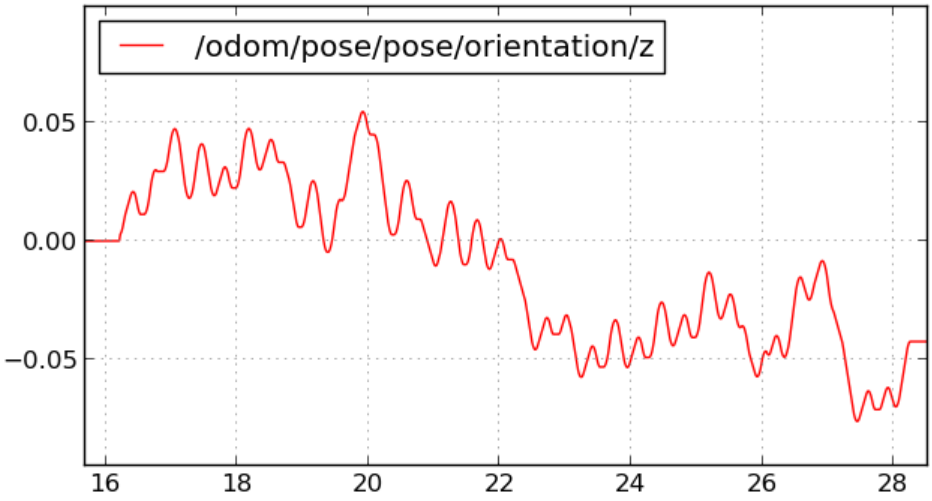












Some quick disclaimers for the figures above (but not below): each graph appears to run on its own time rather than the simulator time, and for this reason the x-axis figures don't always match. Each grouping of data did come from a single trial though, and each plot captures the activity over the same 12 seconds. Another thing you might notice is that the position coordinates from the /odom topic aren't consistent. This has something to do with it being the robot's local frame rather than the global frame, but what's important is the change from the starting position (more legible in Figure 2). Lastly, if you look closely at Figure 3.5(f), you can see that the ending z-orientation value is -0.05, which is inconsistent with the data in the Figure 2 table (0.10). This is because I plotted the wrong figure the first time, so I had to run it again to get the correct graph, but obviously the chances were very low for tremor that everything would come out the same way.

As you can see in the figures above, the impairments negatively impacted the user's ability to execute a simple command over the course of 12 seconds. This shows how many commands are wasted on horizontal movement in this case, and also the extent to which the robot may veer off course. Since tremor is somewhat rhythmic, it is unlikely that the robot would get extremely turned around, but it is an issue nonetheless. It would be impossible to get a completely consistent result with regards to the finishing orientation and x-position, but had I run the lateral tremor test multiple times, I estimate that the robot would be equally likely to finish anywhere between the middle two cell columns.

IV. Results (Part 2 – Blending)

Figure 4.1(a-b) *Starting position and attempted path for the blending test*

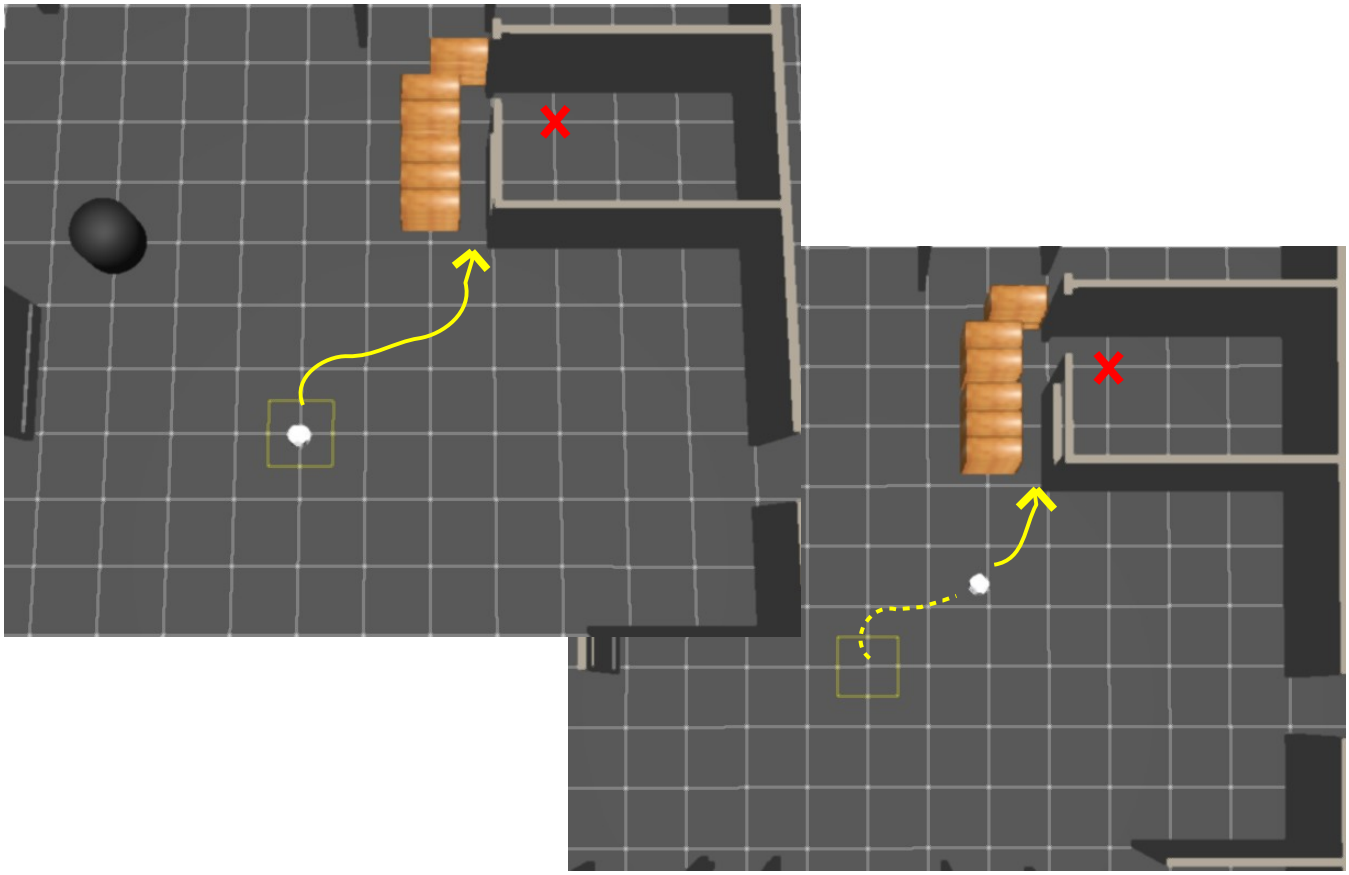


Figure 4.2(a-b) *The hallway leading to the goal (aerial, up close)*

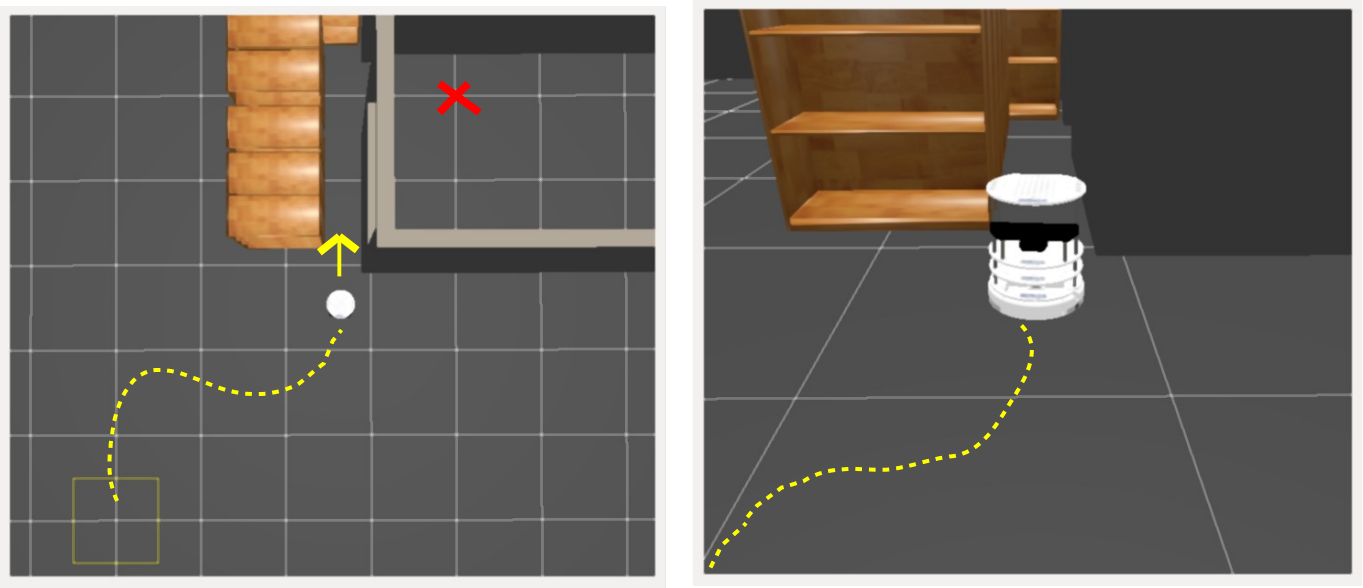


Figure 4.3 *The robot upon reaching the goal*

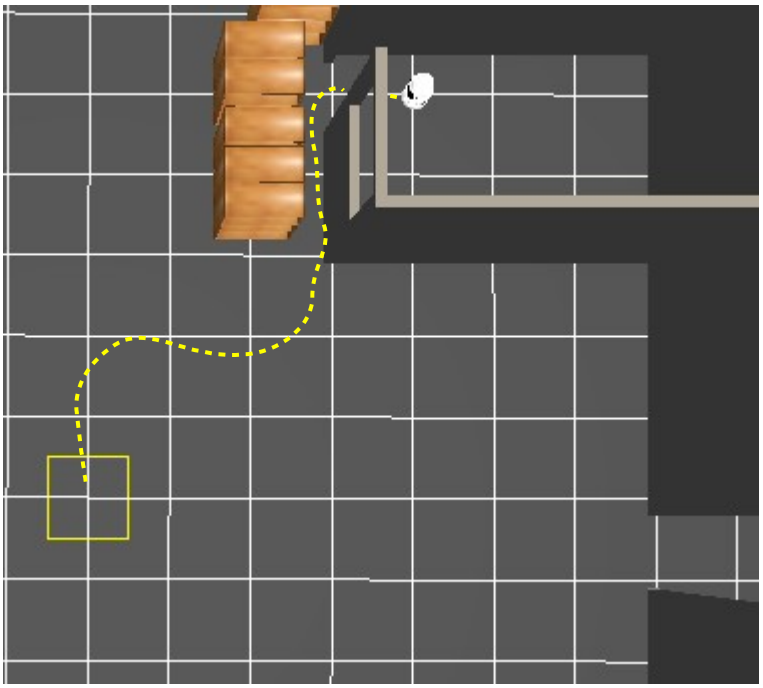
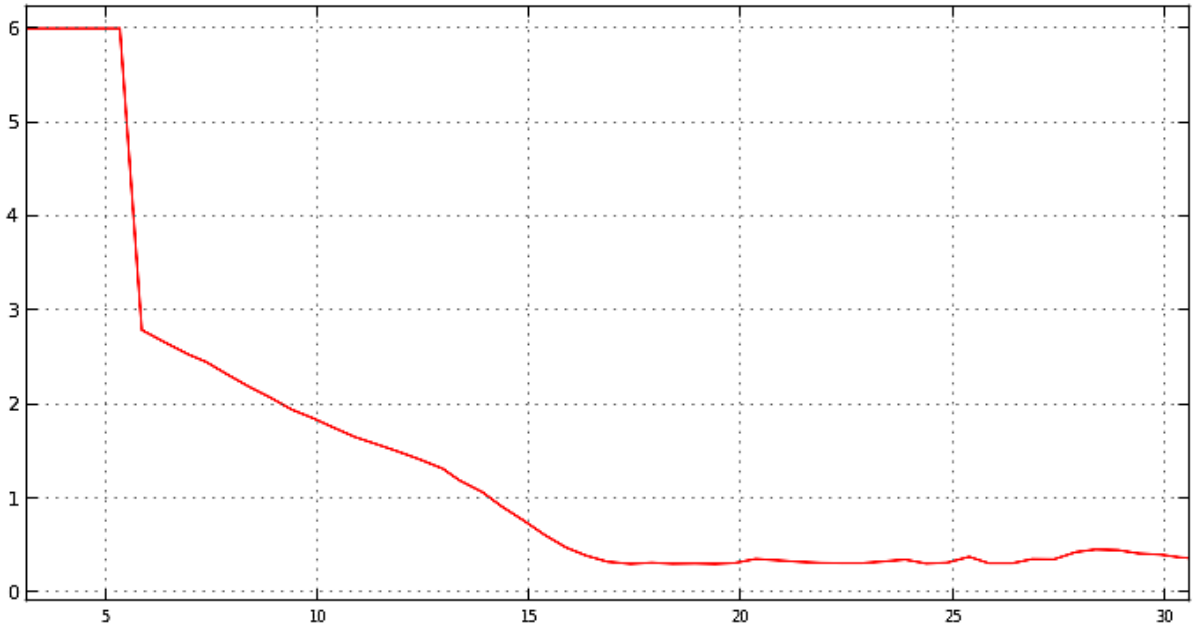
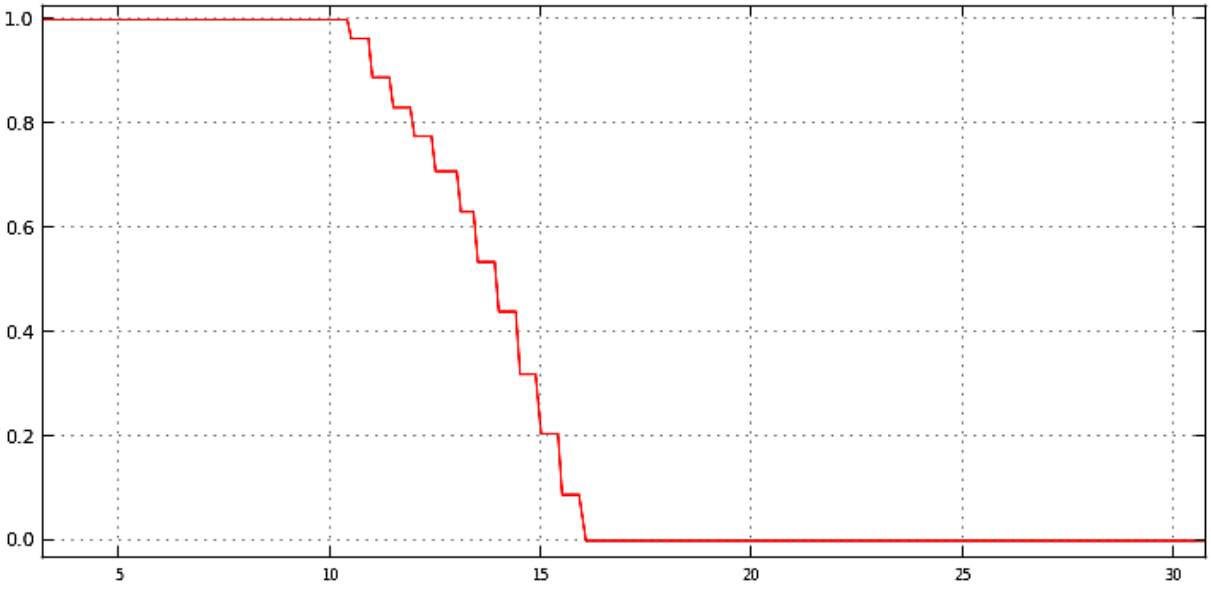


Figure 5(a-d) *Graphical representation of the effect of blending (x-axis represents time in seconds)*

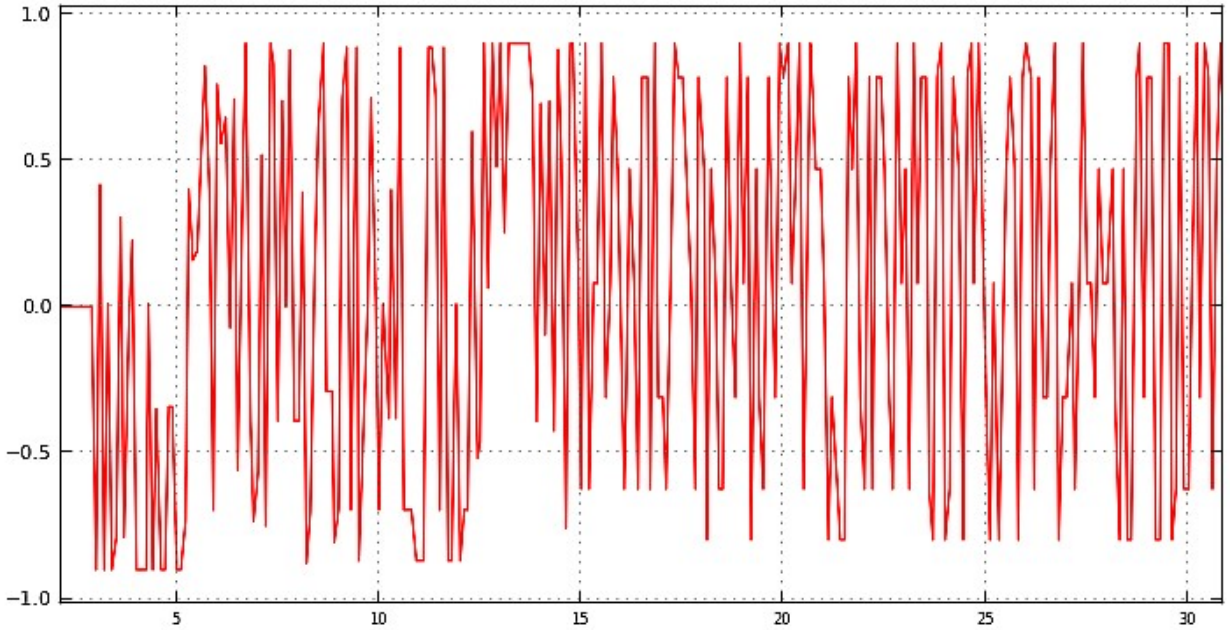
a) y-axis = minimum distance, in grid cells, of surrounding obstacles (/obstacle_features)



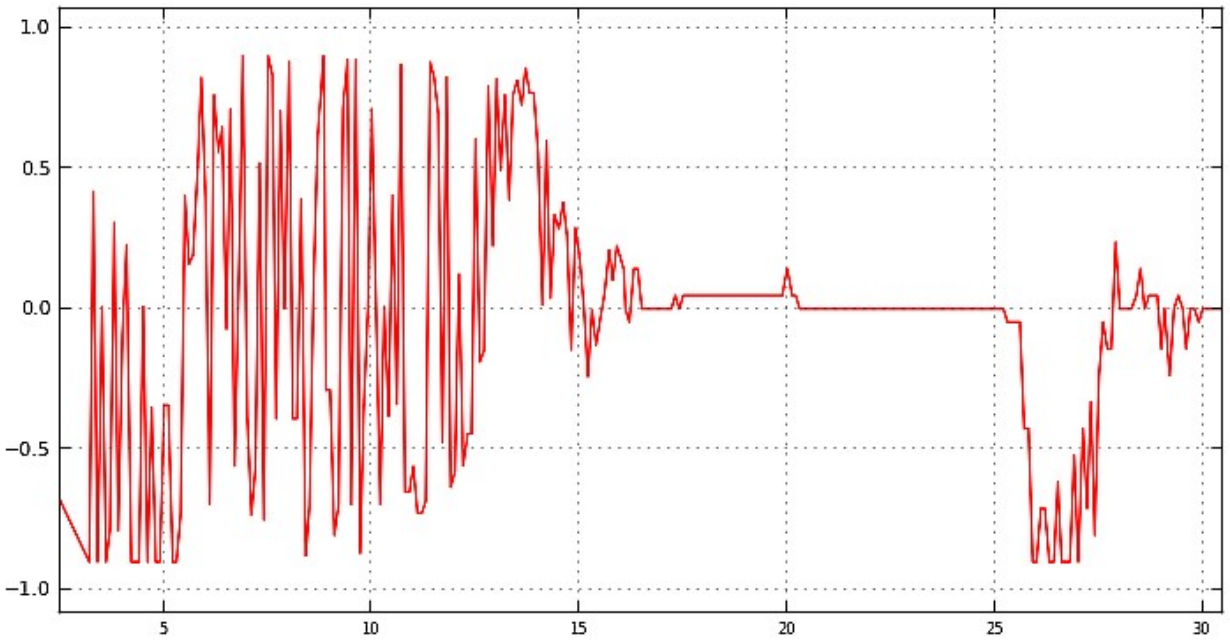
b) y-axis = coefficient of user control (/alpha)



c) y-axis = impaired angular velocity commands issued by the user (/cmd_vel_imp/angular/z)



d) y-axis = user/planner blended angular velocity commands (/cmd_vel/angular/z)



V. Discussion

At first thought, it seems as though bradykinesia would have the greatest inhibitory effect on executing a command, since it should theoretically take twice as long to complete each individual command. However in the case of operating a joystick, bradykinesia only lengthens the time taken to reach the target command, at which point it can be held, like it would be with an unimpaired patient. This is why the bradykinesia controller covered nearly the same amount of distance as the regular controller, because there was only one additional command of velocity 0.15 (Figure 3.3(c)). It is almost impossible to distinguish from the velocity plot in Figure 3.2(c), but the table in Figure 2 shows that there was a small difference (3.63 grid cells, rather than 3.74). In actual application, a user issuing a command to a joystick would also be issuing several intermediate commands in between, in which case the bradykinesia would have shown a larger effect and the slope of the graph would have been much more linear. This is something to consider for evaluating such controllers in the future, because an instantaneous leap in velocity is not realistic.

Another thing to note is that since the tremor node enforces the natural command ranges, and the command given was at maximum tilt throughout the duration, the vertical tremor in this case had a strictly negative impact on distance (only traversed 3.09 grid cells), whereas for any other range of commands, the tremor could both accelerate and decelerate from the original command. It is obvious, but also worth mention that greater amplitudes of tremor would result in even less predictability, but again, being rhythmic, it is hard to accurately estimate the net effect from trial to trial. As for lateral tremor, it can be assumed that the likelihood of straying far from the path would only increase with the duration of the command. In this case, however, the robot had already strayed almost a third of a grid cell off course (Figure 2), which is certainly enough to miss a doorway, and finishes with a diagonal heading of 0.1 radians (Figure 2, Figure 3.5(a)), which will only compound on itself in an extrapolated amount of time.

This error may only add up to a couple feet for a 12-second trial, but over the course of an extended hospital stay, fighting any of these conditions would add a significant extra workload. The proportional decrease in distance was only about 6% (0.21 / 3.74) on average, thanks to the flawed bradykinesia test, but was as high as 17.4% for vertical tremor. This added burden, along with lack of precision in tight doorways and corridors, is why it is necessary to develop an effective blending function.

▪ ▪ ▪

A quick look at Figure 5 reveals a pattern among the four graph plots. The goal was issued at t = 2.608 seconds (according to /move_base/status), corresponding to the robot's position in Figure 4.1(a). At this point there were no obstacles nearby, producing an α of 1.0, and giving the user (me) full control. As you can see, the planner did not start to impose on the output (Figures 5(c and d)) until around t = 10 seconds, as the robot approaches inside of two grid cells from the wall (Figures 5(a and b)). The hallway begins around t = 15 seconds, and here is where we see the greatest impact of the planner, as it takes over for the user. Notice that α hits zero as /obstacle_features dips below 0.5 grid cells from the wall. While the impaired angular commands continue to fluctuate wildly, the actual output smooths out (5(d)) to guide the robot straight through the hallway (t = 15 to t = 25). After that there is a clear plunge in /cmd_vel's angular velocity. This is when the robot turned through the door on the right, and the goal was reached at t = 30.642 seconds.

One possible improvement would be allowing the user to cancel a goal or plan at any time, because sometimes the planner publishes inaccurate commands, but does not give up on the goal, so the user cannot reclaim sole control. Another thing to look out for is the sensitivity of the sensors. In the simulator, the sensor

reading from directly facing the wall outputs a distance of 0.3 grid cells. In a real world application, this number may be more accurate (zero) or even less accurate (> 0.3). In this case the planner might take over too soon or too early, respectively. In the blend.cpp file there is a globally defined constant that determines these parameters to handle such situations. Lastly, some coordinates will confuse the planner and prevent any sort of plan from being sent out, but this is only a fault of the planner in itself. Overall I think the blending node performed very well. I encourage anybody to attempt to drive through the passage in this environment without blending to understand the (exaggerated, in this case) difficulty that motor-impaired patients face.

VI. Conclusion

I discussed multiple motor impairments that could affect a user's ability to operate a wheelchair, and talked about how I chose to imitate bradykinesia and essential tremor. I modeled my approach based on studies of the conditions, and parameterized components of the tremor function on the ROS parameter server to increase flexibility for all types of existing tremor. I described a function for blended controls that determines the amount of user control based inversely on the nearness of local obstacles. I evaluated the effect of these impairments by publishing identical commands to each, and showed the estimated short-distance effects with data and graphs of ROS topics throughout the trials. I tested the blending function by driving through a narrow corridor with a tremor impairment of magnified amplitude, and showed the assistance of the planner by plotting the /cmd_vel topic smoothing out relative to /cmd_vel_imp as the obstacle was approached. There are still several other conditions that would be useful to model for the purpose of further understanding the difficulties of impaired wheelchair control.

i Stocco, Andrea; Lebiere, Christian; Anderson, John R. (2010). "Conditional Routing of Information to the Cortex: A Model of the Basal Ganglia's Role in Cognitive Coordination". *Psychological Review* 117 (2): 541–74.

ii Gelb et al., 1999

iii Van Den Eden et al., 2003

iv Louis, 2005

v Hughes et al., 1992

vi O'Sullivan, Susan B.; Schmitz, Thomas J. (2007). "Parkinson's Disease". *Physical Rehabilitation* 5. Philadelphia: F.A Davis Company. pp. 856–857