Using Machine Learning to Blend Human and Robot Controls for Assisted Wheelchair Navigation

Aditya Goil*, Matthew Derry*, Brenna D. Argall*†
*Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208
Email: [aditya.goil,mderry]@u.northwestern.edu
†Rehabilitation Institute of Chicago, Chicago, IL, 60611
Email: brenna.argall@northwestern.edu

Abstract—**This work presents an algorithm for collaborative control of an assistive semi-autonomous wheelchair. Our approach is based on a statistical machine learning technique to learn task variability from demonstration examples. The algorithm has been developed in the context of shared-control powered wheelchairs that provide assistance to individuals with impairments that affect their control in challenging driving scenarios, like doorway navigation. We validate our algorithm within a simulation environment, and find that with relatively few demonstrations, our approach allows for safe traversal of the doorway while maintaining a high level of user control.**

I. Introduction

The powered wheelchair is an enabling, assistive technology that has been used to improve the quality of life for individuals with severe disabilities through enhanced independent mobility [1] [2] [3]. However, individuals with certain motor, cognitive, and visual impairments are excluded from using powered wheelchairs, thus restricting their independence and quality of life [4].

One approach to this problem is to provide a ”smart” wheelchair with a shared-control architecture. A common control split has robot autonomy handle path planning and the human user handle manual driving [5]. To maintain, or improve, an individual's quality of life, these systems aim to provide the right amount of control to the user, in the sense that the user is given as much control as they wish within the constraints of keeping them safe.

To date, numerous powered wheelchair systems have been developed to address some of the challenging situations that arise when operating a powered wheelchair. One example of a challenging scenario is the traversal of doorways. Intelligent systems such as Navchair [6], SYSIASS [7] [8], and the wheelchair system from Carlson and Demiris [5] implement doorway traversal. Our motivation with blending user and robot control is to provide assistance in this scenario, while also allowing the user to retain a maximal but safe amount of control over their mobility.

In this paper, we present an algorithm for blending user and robot control based on *learned* task variability. We extract this variance from a set of demonstrated executions of the task, based on the insight that *variance* in the demonstration data encodes *allowable flexibility* in the task execution (as noted in Argall *et al.* [9]). A Gaussian Mixture Model-Gaussian Mixture Regression formulation (GMM-GMR) [10] is used to extract the variance from the demonstrations. The result is a shared-control, assistive robot navigation framework. Of note is that our algorithm uses demonstration *only* to learn task variance for blending control—that is, only in order to decide how much control authority to cede to the user—and *not* to learn generalized motion trajectories or understand user intent.

The remainder of the paper is formatted as follows. Related work regarding automated wheelchair navigation systems and shared-control is overviewed in Section II. The core foundations of our blending approach are presented in detail in Section III. Section IV describes our system setup and simulation-specific details. Section V presents the experimental results of our evaluation. The paper is concluded in Section VI along with some directions for future work.

II. Related Work

In this section, various algorithms for collaborative wheelchair navigation are discussed, which mainly center around user performance estimation, user intent prediction, and trajectory learning based approaches for blending user and autonomous control.

Carmona *et al.* [11] presents a collaborative wheelchair navigation system using weighted blending by an efficiency function in a reactive, emergent way. In order to calculate the human contribution, they check the performance of the user, based on task metrics, namely, smoothness, directness and safety. The autonomous planner uses a pure Potential Field Approach presented by Khatib *et al.* [12], where every obstacle is modelled as a repulsive force and the goal as an attractive force. The robot's angular and translational speeds are the composition (weighted sum) of the user and robot's proposed velocities. Qinan *et al.* [13] use a similar approach with different performance indices, safety, comfort and obedience.

Another system that provides adaptive assistance through the prediction of user intent is presented by Carlson and Demiris [5]. In this work, a system is developed that evaluates the performance and attention of the user to provide shared control. They perform plan recognition using a multiple-hypothesis method where the user's known actions are represented by models. Comparing these models from multiple user actions, the required states are predicted in parallel to achieve the tasks. A confidence for each possible nearby task is calculated and when the system is confident a particular task is being undertaken, the system guides the wheelchair using waypoints along safe mini-trajectories. A similar concept of intent prediction is also seen in work of Urdiales *et al.* [14].

A behavior-based shared control system is presented by Philips *et al.* [15]. Here the system gives the user full control and blends in autonomous control based on the appropriateness of the assisting behavior. Appropriateness is calculated from a probability distribution over candidate controls, estimated from signals received from the user through a brain-computer interface. Given the environmental information, each behavior derives its appropriateness level. The shared control system then applies a winner-takes-all approach to determine which behavior it activates, where the assisting behavior with the highest appropriateness level is activated.

A common design pattern for these shared-control systems is to have a navigation planner that provides robot velocity control commands and, using a weighted sum, blend user commands with the planner's commands. In our work, we also use a navigation planner to provide autonomous control and blend the user control. However, our blending approach is novel, in that we augment user-based and task-based metrics with learned task variance when computing the blending factor.

III. Control Blending via Learned Task Variance

Our approach to blending user control with automated control considers the allowed variability in the control commands—as inferred from the variance seen between multiple demonstrations of the same task. Here the task variance encodes differences in execution speed seen during the demonstrations, and the automated control is a local obstacle avoidance navigation system (full details in Section IV-B).

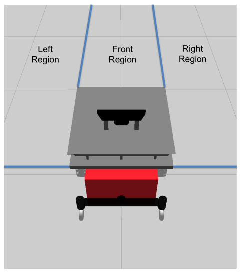
Our system is focusing on a very specific task environment: assisted doorway navigation. That is, we are trying to assist the users in tight spaces and more importantly blend their control during this assistive phase. Thus, detecting doorways is the first step of our system design that enables and starts the navigation system. Once a doorway is detected, we define a goal state. This goal state is a point 1 meter inside the door, pointing outwards from the door in the direction normal to the doorway. This goal is sent to the robot path planner when the user requests assistance via a button press on the joystick.

In Section III-A, we define our learning space. We then present our blending technique in Section III-B. In Section III-C, we briefly explain, the GMM-GMR technique. Finally in Section III-D, we discuss the doorway detection algorithm used.

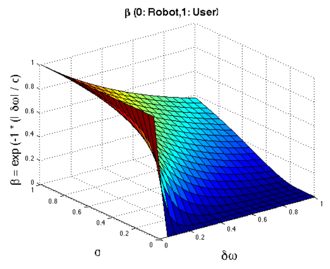
A. Learning Space

Several probabilistic approaches in robotics are used to generalize over a set of demonstrations [10], often learning means along with covariances and extracting generalized trajectories. Our work is concerned only with the variance encoded within such models. The specific formulation we will use is a Gaussian Mixture Model (full details in Section III-C). The input space for our Gaussian Model consists of sensor features and translational speed. The output space is the clockwise and counter-clockwise angular speed of the robot.

We define our dataset as a multivariate set of inputs and outputs. Sensor features $\{d_l, d_f, d_r\}$ are computed as follows. The local obstacle avoidance navigation (i.e. the automated controller) utilizes a local 2-D occupancy grid, which is a discretization of the ground plane into a grid and an *occupancy* value within [0,1] representing the probability of an obstacle within a given grid cell. We use this occupancy grid to generate our sensor features. The space surrounding the robot is divided into three regions defined by extending the distance to the front, left and right boundaries of the robot, as seen in Figure 1(a). A sensor feature is defined as the Euclidean distance from the robot to the closest occupied grid cell in a particular region.



(a)



(b)

Fig. 1. Left: Regions defined for computing sensor features. Right: The blending coefficient β is a function of learned covariance σ and $\triangle\omega$, the difference between user and planner angular speeds.

The input variables of our dataset then are $\xi^I = \{d_l, d_f, d_r, v_x\}$, where v_x is the robot's translational speed. We include the translational speed in our input space, because we expect the range of observed angular speeds to narrow, both in tight spaces and as the translational speed increases, leading to a smaller learned variance. The task variance thus is expected to also change as a function of translational speed, so it is modelled in our input space.

The robot's counter-clockwise angular speed ω_l and clockwise angular speed ω_r are the output variables $\xi^O = \{\omega_l, \omega_r\}$. We model the two speed directions separately because we suspect that the *variance* in counter/clockwise rotational speeds does differ. For example, in a scenario when exiting a doorway with obstacles to the right, but not to the left, we would expect to see a large variance in observed ω_l, and zero variance in ω_r (i.e. the demonstrations never turned right into walls, so all $\omega_r = 0$). This would not be captured by a single speed dimension however; instead, the large variance observed from turning left during demonstration would apply equally to angular speed, regardless of direction; effectively granting the user the flexibility to turn right, towards obstacles.[1]

B. The Blending

We define β as the blending coefficient that will weight the human and the planner's commands when computing a final angular speed. During the doorway navigation task, both the user and the robot planner are issuing translational speed and angular speed commands. The angular speeds need to be blended in a manner that constrains the user input. This is because in challenging situations, it is harder to control the heading of the robot, so the speed might need to be restricted.

[1]An alternative is to infer whether the variance applies to the counter- or clockwise directions by taking the sign of the mean, and presume the variance on the other direction to be zero. There are however cases for which this would over generalize (e.g. for ambiguous demonstrations).

```
1:  Given    Model $\Omega$
2:  while $DoorwayNavigationExecuting$ do
3:      $(v_H^t, \omega_H^t) \quad \leftarrow \ UserInput$
4:      $(v_P^t, \omega_P^t) \quad \leftarrow \ AutomationInput$
5:      $\{d_l^t, d_f^t, d_r^t\} \leftarrow ObstacleFeatures$
6:      $(\mu^t, \sigma^t) \ \leftarrow \ \Omega(d_l^t, d_f^t, d_r^t, v_{current}^t)$
7:      $\Delta\omega^t \quad \leftarrow |\omega_P^t - \omega_H^t|$
8:      $\beta^t \quad \leftarrow \ e^{\frac{-|\Delta\omega^t|}{\sigma^t}}$
9:      $v_{blend}^t \ \leftarrow \ \frac{v_P^t + v_H^t}{2}$
10:     $\omega_{blend}^t \ \leftarrow \ (1-\beta)\omega_P^t + \beta\omega_H^t$
11: end while
```

Fig. 2. Algorithm for blending user and automation inputs.

Our approach blends the angular speeds based on the amount of allowable flexibility in angular speed in either direction. Conversely, the translational speeds are directly blended by averaging the user's and the planner's translational speed input.

Our proposed approach *learns* this variation in angular speeds, from the variability observed during teacher demonstrations of the task. This learned variance is then used to calculate a coefficient that smoothly blends the user and planner's speed commands. Psuedocode for this approach is presented in Figure 2.

In particular, we blend the user's angular speed, ω_H^t, with the planner's angular speed, ω_P^t, according to

$$\omega_{blend}^t = (1-\beta^t)\omega_P^t + \beta^t\omega_H^t \tag{1}$$

where the blending coefficient, β^t, is calculated as

$$\beta^t = \exp(-\frac{\Delta\omega^t}{\sigma^t}), \quad \Delta\omega^t = |\omega_H^t - \omega_P^t| \tag{2}$$

The idea behind this blending coefficient formulation is the following: As the variance $\sigma^t \to 0$, all of the control should increasingly go to the planner, since there is no allowable flexibility in angular speed, and so $\beta^t \to 0$. As $\Delta\omega^t \to 0$, the user's speed commands are approaching planner's, and so $\beta^t \to 1$, giving the user the control. All other cases—including, importantly, cases like $\Delta\omega^t \neq 0, \Delta\omega^t < \sigma^t$ which allocate most control to the user—vary β^t smoothly (Fig. 1(b)).

In short, the larger the ratio $\frac{\Delta\omega^t}{\sigma^t}$, the less the user is able to pull the angular speed away from the planner's commanded speed.

Lastly, while two dimensions of allowable variance (σ_l and σ_r) are calculated from our learned model, only one is used at a given timestep in the calculation of the blending coefficient β^t: the variance related to whichever speed direction is commanded by the *planner*, since the planner is also performing obstacle avoidance and its commands are considered safe commands. Formally, the variance used in Equation 2 is selected according to

$$\sigma^t = \begin{cases} \sigma_r^t, & \omega_P^t < 0 \ or \ (\omega_P^t = 0 \ \& \ \omega_H^t < 0) \\ \sigma_l^t, & \omega_P^t > 0 \ or \ (\omega_P^t = 0 \ \& \ \omega_H^t > 0) \end{cases} \tag{3}$$

C. GMM-GMR

As explained in Section III-A our dataset has 6 dimensions: 4 input variables (ξ^I) and 2 output variables (ξ^O). We collect our set of 6-D data points through human demonstration, where the teacher/clinician drives the robot in different doorway scenarios. The dataset is encoded in a Gaussian Mixture Model (GMM) Ω with K Gaussian components (K is empirically set to 3 in our system), where each component $k \in [1..K]$ has prior probability π_k, mean μ_k, and covariance matrix Σ_k. Gaussian Mixture Regression (GMR) then considers the complete GMM when finding the expected probability of output ξ^O, given component k and input ξ^I,

$$\mathcal{P}\left(\xi^O|\xi^I, k\right) \sim \mathcal{N}(\hat{\xi_k}, \hat{\Sigma_k}) = \sum_{k=1}^{K} h_k \mathcal{N}(\hat{\xi_k}, \hat{\Sigma_k}) \tag{4}$$

where $\hat{\xi_k}$ is the expected mean and $\hat{\Sigma_k}$ the expected covariance matrix for component k,

$$\begin{aligned} \hat{\xi_k} &= \mu_k^O + \Sigma_k^{OI}(\Sigma_k^I)^{-1}(\xi^I - \mu_k^I) \\ \hat{\Sigma_k} &= \Sigma_k^O + \Sigma_k^{OI}(\Sigma_k^I)^{-1}\Sigma_k^{IO} \end{aligned} \tag{5}$$

and $h_k = \mathcal{P}(k|\xi^I)$ is the probability that the Gaussian distribution k is responsible for ξ^I

$$h_k = \frac{\mathcal{P}(k)\mathcal{P}(\xi^I|k)}{\sum_{l=1}^{K}\mathcal{P}(l)\mathcal{P}(\xi^I|l)} = \frac{\pi_k \mathcal{N}(\xi^I; \mu_k^I, \Sigma_k^I)}{\sum_{l=1}^{K}\pi_l \mathcal{N}(\xi^I; \mu_l^I, \Sigma_l^I)} \tag{6}$$

The conditional expectation of ξ^O given ξ^I can then be approximated with a single gaussian distribution $\mathcal{N}(\hat{\xi}, \hat{\Sigma})$,

$$\hat{\xi} = \sum_{k=1}^{K} h_k \hat{\xi}_k, \quad \hat{\Sigma} = \sum_{k=1}^{K} h_k^2 \hat{\Sigma}_k \tag{7}$$

We then obtain the variance $[\sigma_l, \sigma_r]$ for our output dimensions ω_l, ω_r by taking the square root of the associated diagonal elements from the covariance matrix $\hat{\Sigma}$.

D. Doorway Detection

We use the doorway detection algorithm presented by Derry *et al.* [16]. The algorithm uses 3D point cloud data generated by an RGB-D sensor to detect doorways. A 3D point cloud is a set of vertices in the three dimensional space, that stores depth information with respect to the camera frame. The algorithm uses only this depth data, thus making it robust to changes in illumination and lighting. The algorithm scans for walls in a scene; and for each wall in the scene, the wall is scanned for gaps within the range of door width. Once a gap is found, a count of points in the original point cloud is made, using the door height, ground plane, gap position, and wall location as boundaries. Open doorways are detected if this count is below a small threshold (to compensate for sensor noise). For each doorway detected, a position and orientation is calculated. We use this position and orientation to set the goal for the planner.

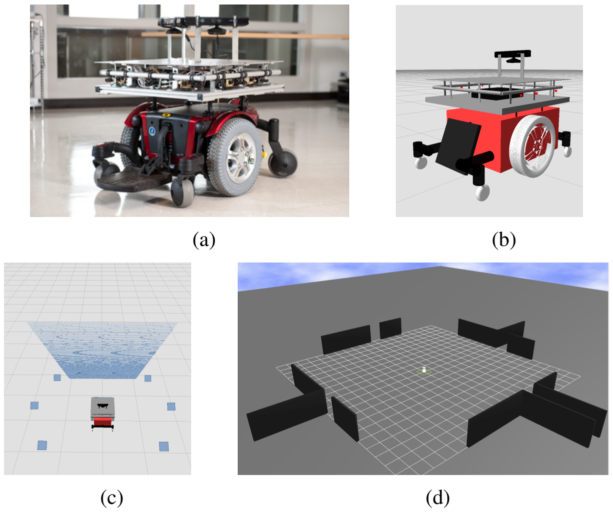


Fig. 3. Actual (a) and simulated (b) wheelchair robots. Simulated IR and Kinect sensor data (c) and validation doorway scenarios (d).

IV. System Implementation

In this section, we discuss our simulated experimental environment. We use a simulated approximation to our actual robot, Figure 3(a), that is built on a powered wheelchair base and outfitted with a ring of IR distance sensors and a Microsoft Kinect. The simulated model operates as a differential drive robot and simulates a front facing Microsoft Kinect Sensor, 6 outward facing IR distance sensors, 3 IR sensors on each of the left and right sides of the robot (Fig. 3(b)).

In order to validate our proposed approach, simulation is an important first step, before testing on real hardware and a target population that includes users with impairments. We use the Gazebo Simulator integrated with the Robot Operating System (ROS) to provide demonstrations, simulate, and test our algorithm. An instance of the environment is seen in Figure 3(c), where simulated sensor data is visible.

Our shared-control approach depends on an autonomous path planner to locally plan paths based on current sensor readings. We use the ROS move_base package, that implements local planning and navigation without the need of a global map. For the task of doorway navigation, we demonstrate a range of angular speeds in a variety of different doorway configurations, such that an estimate of the associated task variance (i.e. allowable flexibility in angular speed) can be learned.

Section IV-A presents the joystick controller for user input, and Section IV-B discusses the path planner. In Section IV-C we discuss our GMM learning.

A. User Control

The user drives the wheelchair model in the simulation using a Sony Playstation 3 (PS3) controller. The same interface is used when the teacher teleoperates the robot to provide demonstrations. The PS3 controller is a wireless bluetooth controller with 10-bit analog joysticks. We use the left analog joystick to control the wheelchair. We use the X button as an indicator for assistive mode, the pressing of which starts the autonomous navigation system, triggering the user and planner control blending, as discussed in Section III-B.

B. Local Path Planning

Our system targets indoor wheelchair navigation, but we do not require a global map for navigation assistance. Instead, we use the ROS move_base package which implements obstacle avoidance and local path planning using a rolling window local costmap. The costmap is a mapping of an occupancy grid to costs. These costs are directly related to the proximity of a cell to an obstacle. The costmap also inflates costs near obstacles to accommodate a margin of safety when the robot navigates close to an obstacle. The occupancy grid, from which the costmap is calculated, is generated from the Kinect sensor and the IR distance sensors which observe the local environment around the robot. In our system, navigation is used only to avoid obstacles and reach a defined goal as part of the doorway traversal assistance.

We assume that the user sends a binary flag (e.g. button press) asking for assistance. Once this indication is received, the doorway detection node scans the scene to autonomously detect the doorway location and orientation, and provides the desired goal to the safe path planner. The path planner then plans a path and starts issuing velocity commands to the robot base to reach the desired goal.

C. Learning the GMM

In order to capture task variability within the Gaussian Model, we need to first observe variations in angular speeds from a set of demonstrations. Our demonstrations are done in a simulated environment in different doorway scenarios where the robot approaches, enters and exits a doorway starting with several initial poses and reaching different final exit poses. There are four different scenarios: open doorway, left wall after doorway, right wall after doorway, and hallway, as seen in Figure 3(d).

During demonstration, the wheelchair platform is teleoperated by issuing user commands, received from the PS3 controller, to the robot base. The demonstrations are given at both moderate and slow translational speeds. At slower translational speeds, a broad range of angular speeds are possible. As translational speeds increase, this range narrows.

The task variance is learned offline[2] and the model is then used online to get the joint probability of the output given the input. As mentioned in Section III-C we model our GMM with 6 dimensions, 4 input variables and 2 output variables. We use 3 gaussian components to model our data. Empirically, σ^t of Equation 2 is set to 3σ, based on preferred performance ($\{\sigma, 2\sigma, 3\sigma\}$ were tested).

V. Results

Initial empirical validation has shown our approach to perform well, granting the user a large amount of control when possible (Sec. V-A). Moreover, the allowable flexibility in angular speed was learned from a relatively small number

[2]We thank Sylvain Calinon for the use of his GMM Matlab code, available at http://www.mathworks.us

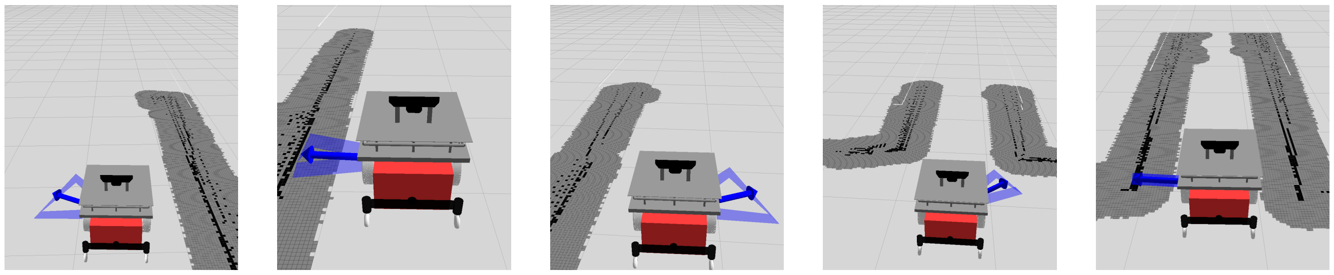


Fig. 4. Examples of learned variance. The value of the angular speed variance is directly proportional to the width of the blue triangle, which is centred on the user-commanded velocity vector (blue arrow). Narrower triangles mean angular speed is more restricted. The gray grid cells depict the obstacle costmap.

of demonstrations (6 per door configuration, assembled into a single dataset). This suggests that data will be very reasonable to collect on a real robot system, which is important for future transfer to general (i.e. non-laboratory) domains.

A. *Performance*

A sampling of example scenarios, and their associated variances extracted from the Gaussian model, are shown in Figure 4. Figure 5 shows three snapshots, in a single image, of the system being stress tested as the wheelchair is being assisted in a scenario where the doorway exits into a room with a wall on the right side.

From the graphs it can be seen that the user commands a hard left and right angular speed (i.e. stress tests). In both cases, throughout the execution, a certain amount of the user's commanded speed is included in the blended command. This can be seen prominently in the shaded regions of (a) and (b) (light gray depicts the user's commanded speed and dark gray the user's executed speed). In the final case (c), where the robot exits the door, the user commands a hard right towards a wall. Here, much less user control is executed (see dark/light gray shaded area after time t = 5), thus ensuring the wheelchair does not get too close to the wall.

We ran equivalent tests on all four types of doorways shown in Figure 3(d), with similar performance results. In particular, the user was never given so much control that a collision occurred, but was given more control than a comparative distance-based blending approach, discussed next.

B. *Comparison with Distance-Based Blending*

We compared our approach against a distance-based blending approach that takes as input the obstacle features and the translational speed to calculate β directly. This β formulation, β_{DIST}, is inversely proportional to the obstacle distance and translational speed.[3]

In Figure 5, the green line shows the blended control given by the distance based blending approach. One notable difference is during the doorway exit (c), when our approach allocates more control to the user control than the distance-based approach. Furthermore, the allocation made by our approach is quite responsive: when a hard right user command is issued toward a wall, the control immediately swings back to the planner as the user comes very close to the wall.

This highlights a strength of our approach, in that additional flexibility can be provided with appropriate demonstrations. By contrast, the distance based approach proved too rigid to allow for close obstacle navigation in some instances, yet more cautious obstacle navigation in others. In fact, our experience showed that tuning the distance-based blending proved to be rather difficult, as it generally either allowed the user to drive dangerously close to obstacles, or restricted too much control.

Another notable difference is during the doorway approach (a), when the wheelchair is far from any obstacles. In this case more control is allocated to the user by the distance-based method. This points to a limitation of our approach, and demonstration-based algorithms in general: the dependence on the contents of the dataset. Our demonstrations focused on showing variability in the exit conditions, and did not show the full range of controls appropriate for the open space before reaching the door—and accordingly had a smaller (than necessary) associated variance in that part of the state space. We note that this is a limitation easily surmounted by providing more demonstrations—that appropriately show the range[4] of desired controls—while acknowledging that such a solution will not scale well with large state spaces.

VI. Conclusion

This paper has presented a new approach to shared control for collaborative wheelchair navigation. The key idea is to learn task variance from demonstrations and extract allowable user command constraints from the variance. This variance is used to blend user and robot control in challenging navigation scenarios, like doorway navigation. First results in a simulation domain are promising and show good performance. Future work will transfer this approach to our real robot platform, and perform more extensive user studies.

References

[1] I. Pettersson, G. Ahlström, and K. Törnquist, "The value of an outdoor powered wheelchair with regard to the quality of life of persons with stroke: A follow-up study," *Assistive Technology*, vol. 19, no. 3, pp. 143–153, 2007, pMID: 17937056. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/10400435.2007.10131871

[2] R. A. Cooper, R. Cooper, and M. L. Boninger, "Trends and issues in wheelchair technologies," *Assistive Technology*, vol. 20, no. 2, pp. 61–72, 2008, pMID: 18646429. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/10400435.2008.10131933

[3] Specifically, replacing σ^t of Equation 2 with the distance to the nearest obstacle multiplied by the normalized (by the maximum possible) translational speed.

[4] All that actually is needed is to show the limits of the range.

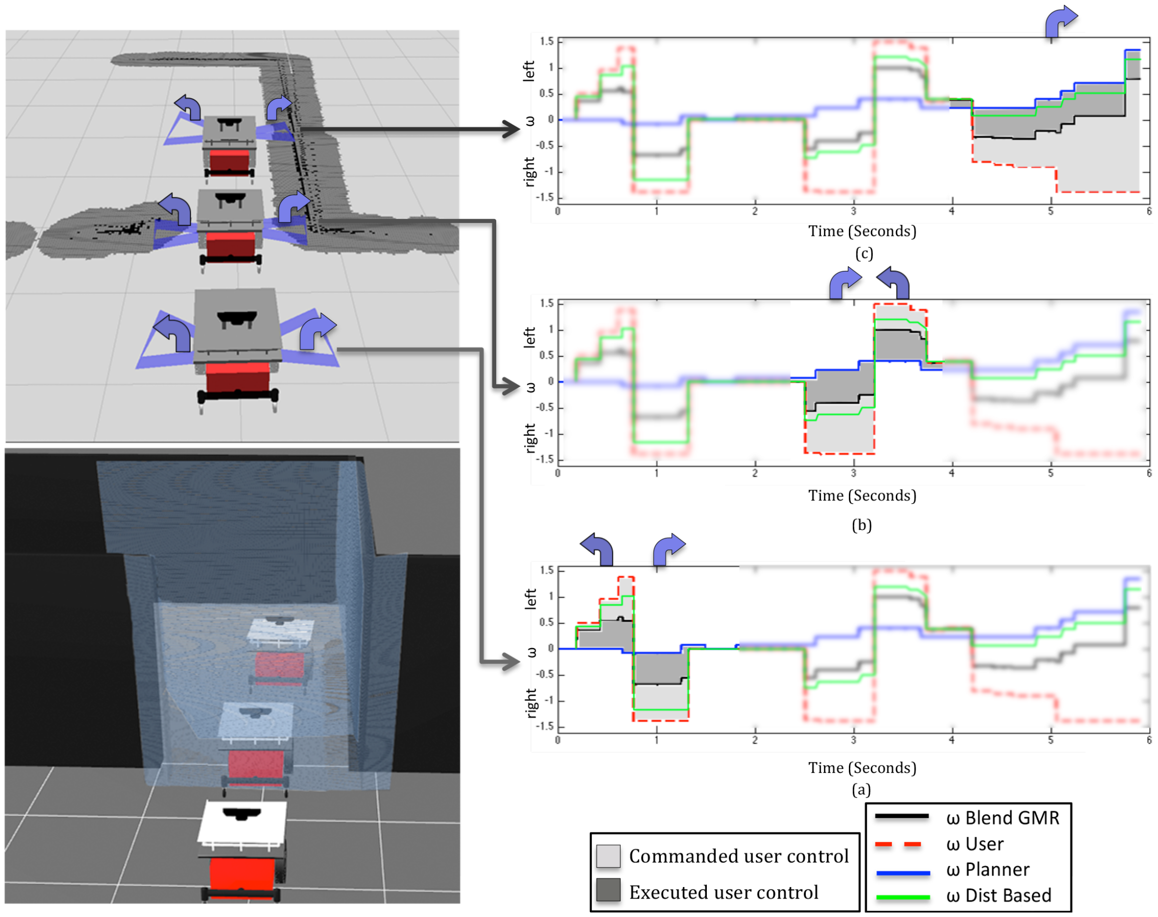


Fig. 5. Snapshots of movement of the wheelchair in simulation. The three snapshots are entering a doorway, inside the door, exiting a door. The top left image is a visualization in ROS rviz. The gray and black squares are inflated and actual obstacles respectively. The blue arrows depict the direction of user-commanded angular speed during stress testing. The image on the bottom left is the visualization in ROS Gazebo. The blue pattern is the point cloud generated by the simulated Kinect sensor. The graphs on the right, from bottom to top, correspond to the (a) doorway approach, (b) doorway entry and (c) doorway exit respectively. The blue arrows on the top of each graph show the direction in which the user commanded the angular speed. The dark gray region in the graphs depicts the amount of user control executed by the blending, whereas the light gray region depicts the remaining user control.

[3] A.-L. Salminen, Å. Brandt, K. Samuelsson, O. Töytäri, and A. Malmivaara, "Mobility devices to promote activity and participation: A systematic review," *Journal of Rehabilitation Medicine*, vol. 41, no. 9, pp. 697–706, 2009.

[4] R. C. Simpson, E. F. LoPresti, and R. A. Cooper, "How many people would benefit from a smart wheelchair?" *Journal of Rehabilitation Research and Development*, vol. 45, no. 1, pp. 53–71, 2008, pMID: 18566926.

[5] T. Carlson and Y. Demiris, "Collaborative control for a robotic wheelchair: Evaluation of performance, attention, and workload," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 3, pp. 876 –888, June 2012.

[6] S. Levine, D. Bell, L. Jaros, R. Simpson, Y. Koren, and J. Borenstein, "The navchair assistive wheelchair navigation system," *Rehabilitation Engineering, IEEE Transactions on*, vol. 7, no. 4, pp. 443 –451, December 1999.

[7] L. Chen, S. Wang, H. Hu, and K. McDonald-Maier, "Bzier curve based trajectory planning for an intelligent wheelchair to pass a doorway," in *International Conference on Control, 2012.*, September 2012.

[8] S. Wang, L. Chen, and K. McDonald-Maier, "Doorway passing of an intelligent wheelchair by dynamically generating bezier curve trajectory," in *Robotics and Biomimetics, 2012. ROBIO 2012. IEEE International Conference on*, December 2012, pp. 1464–1469.

[9] B. D. Argall, E. L. Sauser, and A. G. Billard, "Tactile guidance for policy adaptation," *Foundations and Trends in Robotics*, vol. 1, no. 2, 2010.

[10] S. Calinon and A. Billard, "Statistical learning by imitation of competing constraints in joint space and task space," *Advanced Robotics*, vol. 23, pp. 2059–2076, 2009.

[11] M. Fernandez-Carmona, B. Fernandez-Espejo, J. Peula, C. Urdiales, and F. Sandoval, "Efficiency based collaborative control modulated by biometrics for wheelchair assisted navigation," in *Rehabilitation Robotics, 2009. ICORR 2009. IEEE International Conference on*, June 2009, pp. 737 –742.

[12] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2, March 1985, pp. 500 – 505.

[13] Q. Li, W. Chen, and J. Wang, "Dynamic shared control for human-wheelchair cooperation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 4278 –4283.

[14] C. Urdiales, J. Peula, M. Fernandez-Carmona, R. Annicchiaricco, F. Sandoval, and C. Caltagirone, "Adaptive collaborative assistance for wheelchair driving via cbr learning," in *Rehabilitation Robotics, 2009. ICORR 2009. IEEE International Conference on*, June 2009, pp. 731 –736.

[15] J. Philips, J. del R. Millan, G. Vanacker, E. Lew, F. Galan, P. Ferrez, H. Van Brussel, and M. Nuttin, "Adaptive shared control of a brain-actuated simulated wheelchair," in *Rehabilitation Robotics, 2007. ICORR 2007. IEEE 10th International Conference on*, June 2007.

[16] M. Derry and B. Argall, "Automated doorway detection for assistive shared-control wheelchairs," in *Robotics and Automation, ICRA 2013, IEEE International Conference on*, May 2013.