A Shared Autonomy Approach for Wheelchair Navigation Based on Learned User Preferences

Yizhe Chang, Mohammed Kutbi, Nikolaos Agadakos, Bo Sun, Philippos Mordohai
Stevens Institute of Technology, Hoboken, New Jersey, USA
{ychang1, mkutbi, nagadako, bsun7, pmordoha}stevens.edu

Abstract

Research on robotic wheelchairs covers a broad range from complete autonomy to shared autonomy to manual navigation by a joystick or other means. Shared autonomy is valuable because it allows the user and the robot to complement each other, to correct each other's mistakes and to avoid collisions. In this paper, we present an approach that can learn to replicate path selection according to the wheelchair user's individual, often subjective, criteria in order to reduce the number of times the user has to intervene during automatic navigation. This is achieved by learning to rank paths using a support vector machine trained on selections made by the user in a simulator. If the classifier's confidence in the top ranked path is high, it is executed without requesting confirmation from the user. Otherwise, the choice is deferred to the user. Simulations and laboratory experiments using two path generation strategies demonstrate the effectiveness of our approach.

1. Introduction

Powered wheelchairs are important transportation tools for people with certain motor impairments. It is estimated that on average 1% of the population requires a wheelchair, regardless of whether they have access to one. According to the 2010 census, there are 3.6 million wheelchair users in the US, while approximately 49% of older adults in Canadian institutional settings use a wheelchair [39]. Wheelchair users in Europe are estimated to be in the 5 million range. According to Ceres et al. [9], 2 million wheelchair users in the EU suffer from reduced upper-limb motor control and have to control their wheelchairs via alternative interfaces. Different studies have shown that 10% of wheelchair users require help while operating their manually-controlled wheelchairs and around 40% of users had difficulties in steering and maneuvering tasks using a powered wheelchair [12]. There is a need to develop technologies to assist these people. In order to relieve the burden of manual control, with the advancement of the automatic navigation technology, it is possible to achieve automatic navigation indoors for a powered wheelchair with a computer and a few additional sensors [9, 14, 20]. In most automatic applications, an expert system is developed: a wheelchair user only needs to specify a navigation goal, letting the computer to plan a path and decide how to reach the destination.

Beyond safety and efficiency, researchers have started considering subjective criteria, such as comfort, [15, 34, 40, 42] in path planning for robotic wheelchairs. Wheelchair users, however, do not share the same preferences and the standard of comfort varies from user to user. Among other factors, their preferences may differ in terms of speed, acceleration, curvature of the path, distance to obstacles or people. Therefore, there is a need for an approach that enables shared autonomy and allows robotic wheelchair control to be individually customized according to these preferences.

This paper presents such an approach that can learn user preferences through their commands during navigation or in simulation. When a user specifies a destination, the wheelchair plans several paths to it, instead of one as in an fully autonomous wheelchair system. If one among these paths is clearly better than the others according to the user's preferences, the wheelchair automatically executes it without asking her. Otherwise, if the provided paths do not show significant differences with respect to the user's preference, she is asked to make a choice. The objective is to achieve a trade-off between user satisfaction with the selected paths and user involvement in navigation. This is achieved via shared autonomy, with the user in the loop, but not engaged in mundane operations.

In order to generate several diverse paths for the system to rank or for the user to select from, two path planning methods are integrated into the platform. One method uses the Generalized Voronoi Diagram (GVD) and can generate homotopically distinct paths for places with loop paths (*e.g.* dinning room with a dining table in the center) [4, 25]; the other method iteratively uses an A* planner to generate both homotopically distinct and equivalent paths. Note that

comparing these strategies is outside the scope of this paper, since the criterion for the comparison is not straightforward to define. Instead, we show that our approach works well with both planners.

To score the paths during navigation, we define a set of features and train a Support Vector Machine (SVM) to rank the paths based on previous user choices. Two experiments, one in a simulator and one in our laboratory, show that our software can predict user preferences accurately. In the physical experiment, users control the wheelchair jointly with the autonomous navigation system in a studio via a speech recognition and synthesis interface.

The contributions of this work are:

- an approach for integrating user preferences into a shared-autonomy wheelchair that can achieve a desirable trade-off between adherence to user preferences and low user involvement in navigation,
- a user interface that facilitates shared autonomy by only involving the user in hard decisions,
- ease of training a ranking model in terms of annotation effort, and
- experimental and simulated demonstration of the proposed approach using two path planning strategies.

2. Related Work

In this section, we review the literature on shared autonomy for robotic wheelchairs. We are particularly interested in frameworks where decisions are made by both the user and the system, and not so much in fully autonomous navigation or interfaces that provide full control to the user. For an overview of robotic wheelchair systems we refer readers to [18, 41].

We begin with methods that combine multiple criteria for navigation in static scenes. Gulati et al. [15] introduce a measure of discomfort which is a weighted sum of travel time and time integrals of the squares of tangential jerk, normal jerk, angular velocity and angular acceleration. The method, however, does not consider obstacles, which is a focus of our paper. Shiomi et al. [40] adjust the speed to make their autonomous wheelchair match the behavior of caregivers towards each user. A robotic system following the user's preferred speed and uttering prespecified messages was more acceptable to elderly users.

As described in Section 3, we base one of our path planning strategies on the approach of Kuderer et al. [25] who generate a set of homotopically distinct paths on the generalized Voronoi diagram of the obstacles in the scene. Once paths have been generated, Kuderer et al. compute a cost for each of them as a weighted sum of features. Compared to [25], our approach requires a much weaker form of supervision (see Section 4), decides when to engage the user adaptively and can handle people in the path of the robot.

Research on shared autonomy for wheelchairs is closely related to ours. Parikh et al. [36, 37] propose a shared control framework for an intelligent wheelchair that considers deliberate notion plans, reactive behavior and human user inputs. Deliberate plans are generated given the current position and a destination by finding the minimum-length path (autonomous mode). The user can drive unassisted (manual mode) or aided by reactive controllers which are employed for obstacle avoidance in semi-autonomous mode. The PALMA project [9] led to the development of a robotic platform providing multiple levels of autonomy. It enabled users, primarily children, to navigate safely by performing obstacle avoidance automatically, when needed. Five different levels of user intervention are programmed and controlled by discrete commands via buttons. Zeng et al. [46] developed a collaborative wheelchair which relies on the user to specify the destination and speed, while the system is in charge of path planning under these constraints. The user remains involved and can alter the path to avoid obstacles or enforce her preferences.

Urdiales et al. [43] proposed a shared control approach for wheelchairs which combines commands generated by the robot and the user according to the relative efficiency of each in the task at hand. Efficiency is measured as the average of smoothness, directness and safety. A similar dynamic shared control system for wheelchairs was designed by Li et al. [29]. The level of assistance is adapted based on the user's capabilities and control is determined by optimizing an objective function that considers safety, comfort and obedience to user's commands. The work of Carlsson and Demiris [8] is another example of shared autonomy in which the user guides the wheelchair while the robot adjusts the control signals to ensure safety. Goil et al. [13] use machine learning to blend human and automatic commands in order to control the angular velocity of a wheelchair during simulations of assisted doorway navigation.

Shared control of a wheelchair using speech recognition as the input modality to the robot was addressed by Boucher et al. [6]. Multiple levels of commands ranging from turning to parking are supported. After the user selects a destination on the map, a cost function comprising motion, target reaching and obstacle avoidance components is minimized. Multiple ways of controlling the wheelchair, including a joystick in continuous mode, voice commands and discrete commands via a keyboard, were tested with users and non-users of powered wheelchairs. Mitchell et al. [33] adopted a Wizard of Oz design, in which a hidden experimenter, the wizard, controls the wheelchair as if it were autonomous, to study shared control policies for users with cognitive impairments. They observed that autonomy is not always desirable since it may give the impression of loss of control to the users. Three policies implementing different levels of autonomy were tested on scenarios, such as park-

ing the wheelchair and navigating in tight spaces, while the wizard ensures collision avoidance.

Next, we review methods that take into account people in the environment. We also refer readers to a survey for more information [24]. Sisbot et al. [42] presented a motion planner that integrates safety and comfort in its cost function. Safety is a function of distance from humans, while comfort is represented by a visibility criterion that keeps the robot in the field of view of people. Navigation considering people, as well as criteria for static scenes including distance traveled and distance from obstacles, are addressed by Kirby et al. [21]. Their COMPANION framework encodes human-robot social interaction, by respecting personal space and passing on the right, as constraints in an optimization framework, but the weights of the constrains are heuristically set. In recent work, Cosgun et al. [11] propose a path planning approach that anticipates people's reaction to the robot based on the social force model [16]. It comprises a static planner that computes the cost of a path as the weighted average of path length, distance from people and disturbance of groups of people, as well as a dynamic planner that refines parts of the path that are close to people. Morales et al. [35] address wheelchair navigation emphasizing comfort for both its passenger and pedestrians. Parameters for both passenger and pedestrian models are set by the researchers to produce the desirable behavior. The resulting planner was preferable to shortest path planning in a user study.

3. Planning Multiple Paths

Given a map in the form of an occupancy grid, the current position, and the destination, path planning strategies can generate paths that connect the two positions. There are several methods for generating paths for autonomous navigation [28]. Most of them, however, present only one path for execution. Here, we use two planning methods, separately, to generate a diverse set of paths aiming to include at least one that matches the user's preferences. One method uses the Generalized Voronoi Diagram (GVD) in order to generate homotopically distinct paths [25], while the other applies the A* algorithm iteratively after progressively placing "virtual obstacles" to block previously discovered paths.

In order to include the path a user would choose or the system would choose for her, we must generate a number of "different" paths. A useful definition of when two paths are different is based on homotopy [4, 25, 32]. For two paths to be homotopically distinct, there must be one or more obstacles between them, preventing a smooth transformation from one to the other. This, however, is not the only criterion. Considering Fig. 1(a) for example, the presence of a table in the center of the room gives rise to two homotopy classes of paths that pass on either side of the table. In the absence of the table in Fig. 1(b), paths cannot be distinguished based on homotopy, but a wheelchair user may prefer to navigate close to one of the wall to leave space for people to walk for example. We present path generation strategies that cover both cases.

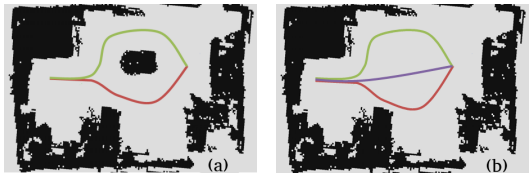

Figure 1. (a) Homotopically distinct paths. (b) Distinct paths in the same homotopy class.

3.1. Generating Homotopically Distinct Paths Using GVD Planning

Our strategy for generating homotopically distinct paths relies on the GVD of the obstacles in the map and follows the approach of Kuderer et al. [25]. It uses a property of the GVD, on which any different paths between two vertices are homotopically distinct. Therefore, the problem of finding homotopically distinct paths for navigation is converted into the problem of finding k-shortest paths on a graph.

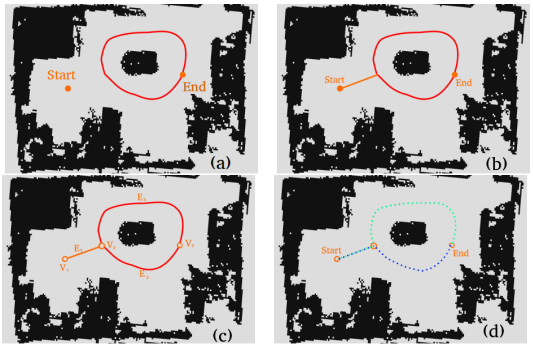

Figure 2. Planning homotopically distinct paths. (a) Create the GVD. (b) Add paths between start/end points to the GVD if they are not on it. (c) Convert the GVD to graph representation. (d) Find k-shortest paths (two in this example).

Figure 2 illustrates the process, which has four steps.

(a) Create the GVD of the obstacles in the map. The GVD of a map is a set of free points whose distances to the nearest two obstacles are equal [10]. It is called "generalized" because unlike the regular Voronoi diagram, in which the input sites are points, the sites of the GVD can be any continuous shape. The first step for computing the GVD is a Euclidean distance transform, for which we employ the implementation of Lau et al. [27]. The output of this transform is a binary occupancy grid Occ with the same size as the map, indicating whether a cell (x, y) is on the GVD or not (*i.e.* $Occ(x, y) = true, false$).

(b) Add the start and the end point to the GVD, since in general these points are not on the GVD. To make the

necessary connections, the closest cells to the start and end points on the GVD are found. Straight line paths are made between the start/end point and their respective closest cell.

(c) Convert the GVD into the graph representation $G = (V, E)$, which makes finding the k-shortest paths simpler. Therefore, the binary grid Occ is converted into a graph G. A breadth-first search is used for traversing the grid-represented GVD so that all vertices and edges can be found. Vertices are cells with more then two incoming edges. The cells containing the start and end points are considered vertices with one edge. The weight of each edge is the number of connected cells between its two vertices.

(d) Apply Dijisktra's algorithm to find k-shortest paths on the graph G. These k different paths are homotopically distinct because they correspond to different paths on the grid-represented GVD. For a map that contains n obstacles, the complexity of this algorithm is $O(k(nlogn))$ [25].

3.2. Path Planning using the Iterative A* Algorithm

In an empty room, or a corridor, homotopically distinct paths are unlikely to exist. However, wheelchair users may still have a preference among paths in the same homotopy class. Some may prefer to navigate near the wall, while others may prefer the center. Therefore, we propose a different method to generate multiple paths in such environments.

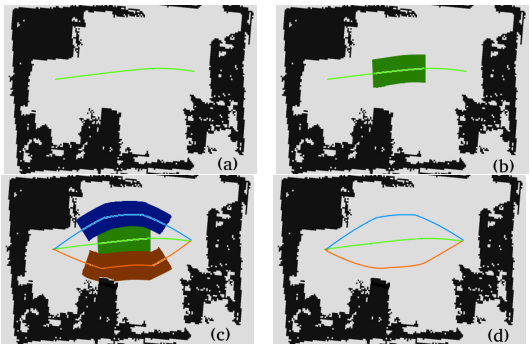

Figure 3. (a) Generate shortest-distance path. (b) Place virtual obstacle with pre-defined size. (c) Keep planning and placing obstacles until no new path can be planned. (d) Retrieve all generated paths.

Iterative A* is illustrated in Fig. 3 and works as follows.

(a) Use the A* algorithm to find the optimal path. The ROS move-base package is used for this purpose.

(b) Place a "virtual obstacle" blocking the most recently generated path. This is implemented by labeling as occupied the cells of the occupancy grid that are within a certain distance of the generated paths. The size of the virtual obstacle is pre-defined and depends on the size of the room and the grid cells. The first and last segments of the path are left free of virtual obstacles to allow the discovery of additional paths.

(c) Generate the next path using the A* algorithm on the modified map and return to step (b) until virtual obstacles block all possible paths.

4. Learning Users' Preferences

We provide a simple user interface (UI) that allows the user to click on a path in the simulator to indicate her preference, or to select a path by saying the name of a color that has been assigned to it in a display mounted on the wheelchair. In either case, when a user picks a path among multiple options, she performs a ranking action. In order to enable our software to mimic the user making a choice, a pairwise learning-to-rank model using a Support Vector Machine (SVM) is trained [17]. Training data are collected by recording a number of user selections and deriving ranking constraints from them. Specifically, given a user choice, we learn that the selected path should be ranked higher than any of the other available paths in that map. We learn nothing about the relative ranks of the other paths though.

We define a *scenario* as a map with a specified start point and destination. Annotation efforts are minimal since the user only needs to click once, to select a path, for every scenario in the simulator. (We are aware that clicking in the simulator is different than being in the wheelchair, but training can be augmented with real data. We evaluate the effectiveness of training on the simulator in Section 5.)

As a form of representing the paths, we define a set of features which are presented below, starting with static environments and continuing with environments that contain people who must be taken into account by the planner.

4.1. Feature Vector for Paths in Static Environments

A user may consider several criteria in order to select a path. We form a feature vector that captures relevant properties of a path. In a static environment, for a path $\boldsymbol{X}$ that consists of poses $\boldsymbol{x}$, the feature vector $\boldsymbol{f}(\boldsymbol{X})$ has four dimensions:

$$\boldsymbol{f}(\boldsymbol{X}) = [l, l_n, \bar{d}, d_{min}, \delta_{sum}]^T \quad (1)$$

(a) Path length: the shortest path is preferable among otherwise equivalent choices, while length and duration are always important features. The length is approximated as the sum of the lengths of the segments connecting successive poses.

$$l = \sum l(x_i, x_{i+1}) \quad (2)$$

(b) Narrow passage length: in each path, there are segments that are close to obstacles on both sides. These segments are called "narrow passages", in which the probability of wheelchair collision is larger than in open area. The narrow passage segments are segments between poses in $\boldsymbol{X_n}$ which is a subset of the path $\boldsymbol{X}$, such that $d(x_i) < d_n$

and $d(x_{i+1}) < d_n$ where d_n is a pre-set constant, with the default $d_n = 0.5m$.

$$l_n = \sum l(x_i, x_{i+1}), \quad \{x_i, x_{i+1}\} \in \boldsymbol{X_n} \tag{3}$$

This feature allows the system to learn the preferred trade-off between overall path length and tolerance of narrow passages. Users may prefer longer, wider paths up to a certain increase of overall path length.

(c) Average distance to obstacles: the distance from each pose x_i to its nearest obstacle can be estimated on the map. The average of these distances is a global feature representing comfort and likelihood of collisions. Similar features have been considered in the literature [6, 21, 43]

$$\bar{d} = \frac{\sum ||d(x_i)||}{N} \tag{4}$$

(d) Minimum distance to obstacle: the minimum among all distances from each pose to the respective closest obstacle.

$$d_{min} = min||d(x_i)|| \tag{5}$$

(e) Sum of turning angles: the angle between two consecutive poses reflects the angular velocity and acceleration of the wheelchair as the local plan is followed. Therefore, the sum of turning angles is a proxy for user comfort [15, 38, 43].

$$\delta_{sum} = \sum |\delta(x_{i+1}) - \delta(x_i)| \tag{6}$$

4.2. Feature Vector for Paths in Environments with People

The wheelchair should consider people as dynamic obstacles in the scene and treat them differently than static obstacles [11, 21, 24, 35, 42]. The SPENCER people detector is used for people detection[30, 31]. Wheelchair users may show different preferences if people are on or near a path. The feature vector here, in addition to the features for static scenes, contains two more features which capture the distances from a path to people. This feature vector has seven dimensions:

$$\boldsymbol{f(X)} = [l, l_n, \bar{d}, d_{min}, \delta_{sum}, \bar{d_p}, d_{p\_min}]^T \tag{7}$$

The two additional features are defined as follows.

(f) Average distance to people: like the sum of distance to obstacles, distances of all poses to their respective nearest people are averaged.

$$\bar{d_p} = \frac{\sum ||d_p(x_i)||}{N} \tag{8}$$

(g) Minimum distance to people: is defined similarly to the minimum distance to obstacles.

$$d_{p\_min} = min(d_p(x_i)) \tag{9}$$

4.3. Support Vector Pairwise Ranking

Given a set of paths from the starting point to the destination, we need to rank them in order of user preference. Our system does not classify paths as acceptable or unacceptable, but chooses one of the available options. Since during training the user only selects the best path in each scenario, we only have pairwise constraints between the selected path and all other paths, but we do not know which paths would have been the user's second or third choice. Therefore, we formulate the problem as ordinal regression using a Support Vector Machine (SVM) following the approach of Herbrich et al. [17] with pairwise constraints, instead of adopting a listwise approach [7]. The process is the same for static and dynamic environments.

Because the feature vector is heterogeneous, all features are normalized to have zero mean and variance equal to one. Normalized feature vectors are denoted by $\hat{\boldsymbol{f}}(\boldsymbol{X})$. To generate the training set, we form pairs of preferred and not-preferred paths (from the same scenario) and define *preference vectors* by subtracting the corresponding feature vectors. Half of the preference vectors are derived by subtracting a not-preferred path from the preferred one and the other half are derived by subtracting the feature vectors the other way. A preference vector $\boldsymbol{p_{ij}} = \hat{\boldsymbol{f}}(\boldsymbol{X_i}) - \hat{\boldsymbol{f}}(\boldsymbol{X_j})$ is labeled as follows:

$$y(\boldsymbol{p_{ij}}) = \begin{cases} +1 & \text{if user prefers path } X_i \text{ to } X_j \\ -1 & \text{if user prefers path } X_j \text{ to } X_i. \end{cases} \tag{10}$$

We then train a linear SVM on the preference vectors which learns to predict whether the minuend or the subtrahend is the preferred path, and produces a positive or negative prediction, respectively.

During testing, the linearity of the SVM allows us to compute the inner product of the SVM weight vector $\boldsymbol{w}$ and $\hat{\boldsymbol{f}}(\boldsymbol{X})$ before the subtraction. Therefore, we compute a score for each path

$$S(\boldsymbol{X}) = \boldsymbol{w^T}\hat{\boldsymbol{f}}(\boldsymbol{X}) \tag{11}$$

and select the one with the maximum score.

Selecting the path with the maximum score corresponds to fully autonomous navigation. Our goal, however, is to keep the user in the loop to make the harder choices. Our system decides to involve the user when the top two scores are close to each other, where closeness is defined as the difference of the scores. (User selections in these situations can be used to refine the SVM, but we have not pursued this yet.)

5. Experimental Results

In order to validate the accuracy of our approach in ranking paths according to user preferences we conducted sim-

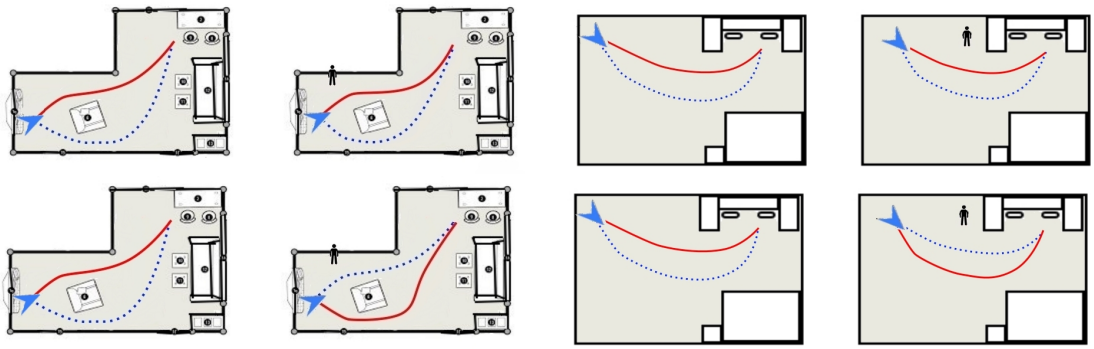

Figure 4. Examples of path selection by subjects. (The destination is on the right of the map in these examples.) The preferred paths are denoted by solid red curves while the ones not chosen by dotted blue ones. Top: selections by subject 1. Bottom: selections by subject 2. Notice the different choices in the presence of people.

ulations and experiments with two users who have different preferences. We chose this design to closely model the preferences of individual users as opposed to the "average" preference of the crowd [19]. To enable fast and diverse training, we developed a ROS-based simulator that integrates our path planner. We also performed experiments with a powered wheelchair and the same two users.

5.1. Data Collection in the Simulator

We generated four maps in the simulator that resemble the complexity of the space we use in our physical experiments. Some of the maps can be seen in Fig. 4. They contain furniture, such as a bed, a desk, a dining table, a sofa, etc. Some of these maps were intentionally made to have two or more homotopically distinct paths, while some maps only allow one homotopy class of paths. During training, each subject generates a number of scenarios by specifying a start and end point on the map. The system, then, invokes one of the planners to generate a set of plans and asks the user to select the one that he prefers.

In order to evaluate our approach in the presence of people, another dataset was collected and annotated in the simulator. These data demonstrate more clearly the differences between individual users, since one of our subjects had a strong preference for avoiding people.

5.2. Results in Simulated Maps

Two subjects were asked to select one path in scenarios generated in the maps of Fig. 4. Both planners were used; the GVD planner was only applied in maps with homotopically distinct paths, while the iterative A* method was applied in all maps.

The feature vectors of all generated paths, the start and end points and the user selections were recorded to form two datasets: one with and one without people. 36 scenarios were generated by each subject for each planner for static scenes and 36 more for each planner in the presence of people. Figure 4 shows examples of selected paths by both subjects in two of the maps. Each subset of the data (same subject, planner and static/dynamic condition) was randomly split into a training and test set, comprising two thirds and one third of the scenarios, respectively. A total of eight SVMs are trained as in Section 4.3.

To achieve the appropriate level of user involvement we define confidence as the difference of the top two path scores and threshold it to determine if the system should decide autonomously or not. Figure 5 shows the ROC of system accuracy and system decision rate (the fraction of times the system decides autonomously) as a function of this threshold. For clarity, we show ROCs for static and dynamic scenes, averaging over planners. As can be observed, accuracy is in the order of 85-90% when the system makes all decisions and quickly rises as the user is called upon to make the hardest decisions. We chose a threshold of 0.2 on the difference of SVM scores, defined in (11), as the operating point for the remainder of the paper.

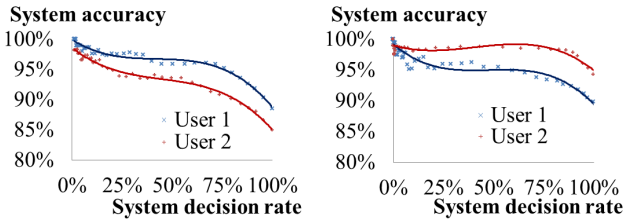

Figure 5. Relationship between system accuracy *vs.* system decision rate. Left: static scenes; right: dynamics scenes. (36 scenarios per planner per user)

Table 1 shows the average results for static scenes after randomly splitting the datasets 20 times, keeping the threshold at 0.2. Accuracy is never below 75%; around 10% of the decisions are deferred to the user; and at most 12.09% of paths were incorrectly selected by the system. Table 2 shows similar results on dynamic scenes under the same conditions. The overall accuracy and decision rate are similar to those in static scenes. It is worth pointing out that the consistent choices made by subject 2, who said "I always

	GVD Path Planner			Iterative A* Path Planner		
	Preferred Path Picked	**Not Preferred Path Picked**	**Ask subject to Decide**	**Preferred Path Picked**	**Not Preferred Path Picked**	**Ask subject to Decide**
subject 1	91.85%	2.46%	5.69%	76.39%	12.09%	11.52%
subject 2	79.61%	10.79%	9.60%	81.88%	11.12%	7.00%

Table 1. Accuracy in simulator for both subjects in static scenes. (36 scenarios per planner per user)

	GVD Path Planner			Iterative A* Path Planner		
	Preferred Path Picked	**Not Preferred Path Picked**	**Ask subject to Decide**	**Preferred Path Picked**	**Not Preferred Path Picked**	**Ask subject to Decide**
subject 1	93.90%	2.19%	3.92%	79.93%	14.40%	5.67%
subject 2	84.95%	6.71%	8.33%	97.33%	0.59%	2.08%

Table 2. Accuracy in simulator for both subjects in dynamic scenes. (36 scenarios per planner per user)

pick the path that was away from the person, especially for open spaces," make predictions for his preferences using the iterative A* method easy.

5.3. Results on Robotic Wheelchair

We also tested our approach in real scenes using a robotic wheelchair. We modified a commercially available powered wheelchair so that it can be controlled by a computer, by connecting the latter to the joystick interface, and added a Microsoft Kinect, which provides RGB-D images for localization and obstacle avoidance. A microphone is used to receive voice commands and a tablet, mounted in front of the subject, displays the map and messages to the user (see Fig. 6).

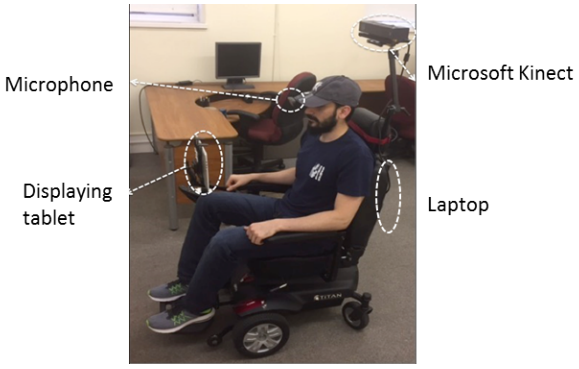

Figure 6. Setup of robotic wheelchair

The ROS move-base package is used for path planning and navigation. Given a, potentially partial, map, the move-base package can plan a path and execute it taking into account the robot's physical properties. For the experiments shown here, we replaced the global planning component of the ROS move-base package with the path-planning methods presented in Section 3. After a path is selected automatically or by the user, it is sent to the local planner for execution.

Other ROS packages used here include: RTAB-map which is used for mapping and localization [26] and the upper-body detector from the SPENCER project which is used to detect people [30, 31]. Both localization and people detection rely entirely on the Microsoft Kinect for input. The CMU Sphinx library [44] is used for voice recognition.

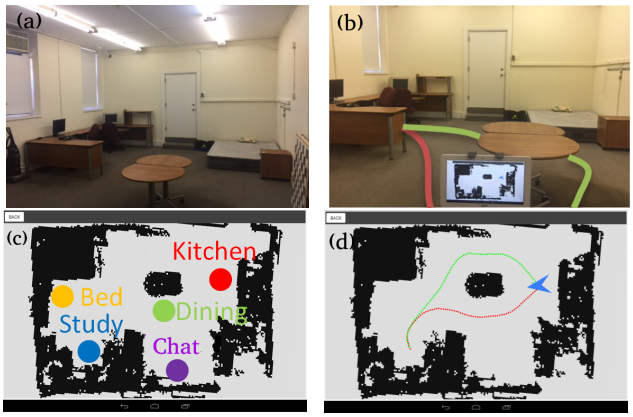

Figure 7. Experiment in studio. (a): Studio for navigation, (b): paths between kitchen and study, (c): map of studio, (d): paths displayed on tablet

A studio, shown in Fig.7, is used for testing. Inside the studio, there is a bed, a study area, a dining table, a chatting area, and a kitchen. The dining table is at the center of the room allowing the wheelchair to pass on either side. Therefore, there are two homotopically distinct paths between any two locations. In some cases, such as between the kitchen and bed, the top two paths have similar feature vectors, while in other cases, such as the bed and the study area, the feature vectors differ considerably.

A voice recognition interface is used for hands-free control of the wheelchair by defining words or phrases as commands. In addition to the commands, the subject only needs to say the color of a path to select it when needed. A typical interaction of the user with the wheelchair looks as follows:

1. The subject says "attention" to start.
2. The system responds with "ok, where do you want to go".

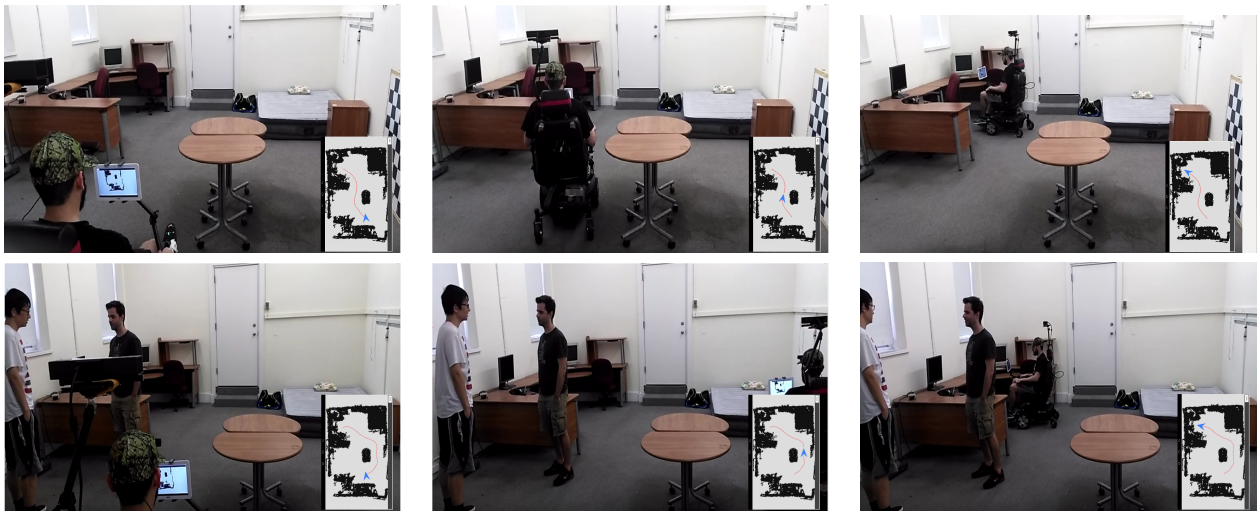

Figure 8. Photographs from experiments in studio with subject 2. Top: static scene. Bottom: dynamic scene. The system chose different paths due to the presence of people in the second scenario, even though they do not block the left path.

3. The subject says the name of one of the pre-specified locations (*e.g.* "kitchen", "study").

4. The system acknowledges receiving the location and searches for paths using one of the planners.

5. The system plans paths. If there is more than one path, the system evaluates the paths and decides whether to defer to the subject or not based on the difference of path scores.

6. If the system decides to engage the subject, it displays the paths on the tablet and asks the subject to choose the path he prefers. The subject responds by the color of the path (*e.g.* "red", "green").

7. The wheelchair executes the path selected by the subject or automatically.

8. The system goes back to idle state and the subject can say "attention" to restart the process.

One of the purposes of experimenting in a real environment is to test whether the learning-to-rank model trained on simulated data would transfer well to a robot. Preliminary observations support this hypothesis for both planners. Moreover, the difference in user preferences shown in the simulator has been maintained when the subjects operate the wheelchair and is manifested by the different paths chosen, especially in the vicinity of people. Photographs from one of the runs are shown in Fig. 8.

6. Conclusions and Future Work

This paper presents an approach for shared autonomy focusing on a robotic wheelchair that can learn user preferences and integrate them in path planning. Given a navigation goal, our approach generates multiple paths, using one of two available planners, and attempts to select the path that the user would have chosen. The selection is made by an SVM that has been trained to rank paths according to objective criteria, such as length, as well as more subjective criteria, such as distance to people and obstacles. Training the SVM does not require substantial annotation efforts since it can be done with one click per scenario in a simulator - much faster than actually navigating the wheelchair.

Unlike other research efforts that adjust the level of autonomy of robotic wheelchairs according to the user's capabilities [9, 29, 33, 40, 43], we target users without serious cognitive impairments. Our aim is to provide them with a user interface that alleviates the burden of navigation while adhering very closely to what the user would have done if she were driving. Our experiments in a simulator and using a powered wheelchair are encouraging that this goal can be attained. User studies at a much larger scale are our highest priority for future work.

Currently, our system only learns from a single click of the user who selects a path among a few options. We plan to record richer inputs while the user is navigating the wheelchair manually, using the joystick or another input mechanism. We can then apply learning from demonstration techniques [1, 3, 5] to capture user preferences and habits more faithfully. In the shorter term, it is straightforward to periodically retrain the system using data recorded during everyday operation when the user is engaged by the system. Finally, we plan to provide more sophisticated interaction capabilities with people in the scene by anticipating their behavior [2, 11, 22, 23, 45].

Acknowledgment Research reported in this publication was partly supported by the National Institute Of Nursing Research of the National Institutes of Health under Award Number R01NR015371 and the National Science Foundation under Award Number IIS-1637761.

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, 2004. 8

[2] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *CVPR*, pages 961–971, 2016. 8

[3] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009. 8

[4] S. Bhattacharya, M. Likhachev, and V. Kumar. Topological constraints in search-based robot path planning. *Autonomous Robots*, 33(3):273–290, 2012. 1, 3

[5] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In *Springer handbook of robotics*, pages 1371–1394. 2008. 8

[6] P. Boucher, A. Atrash, S. Kelouwani, W. Honoré, H. Nguyen, J. Villemure, F. Routhier, P. Cohen, L. Demers, R. Forget, and J. Pineau. Design and validation of an intelligent wheelchair towards a clinically-functional outcome. *Journal of neuroengineering and rehabilitation*, 10(1):1, 2013. 2, 5

[7] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *International Conference on Machine learning*, pages 129–136, 2007. 5

[8] T. Carlson and Y. Demiris. Collaborative control for a robotic wheelchair: evaluation of performance, attention, and workload. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(3):876–888, 2012. 2

[9] R. Ceres, J. L. Pons, L. Calderon, A. R. Jimenez, and L. Azevedo. A robotic vehicle for disabled children. *IEEE Engineering in Medicine and Biology magazine*, 24(6):55–63, 2005. 1, 2, 8

[10] H. Choset and J. Burdick. Sensor-based exploration: The hierarchical generalized voronoi graph. *The International Journal of Robotics Research*, 19(2):96–125, 2000. 3

[11] A. Cosgun, A. Sisbot, and H. I. Christensen. Anticipatory robot path planning in human environments. In *The 25th IEEE International Symposium on Robot and Human Interactive Communication*, pages 562–569, 2016. 3, 5, 8

[12] L. Fehr, W. E. Langbein, and S. B. Skaar. Adequacy of power wheelchair control interfaces for persons with severe disabilities: A clinical survey. *Journal of rehabilitation research and development*, 37(3):353, 2000. 1

[13] A. Goil, M. Derry, and B. D. Argall. Using machine learning to blend human and robot controls for assisted wheelchair navigation. In *IEEE International Conference on Rehabilitation Robotics (ICORR)*, 2013. 2

[14] H.-M. Gross, H. Boehme, C. Schroeter, S. Müller, A. König, E. Einhorn, C. Martin, M. Merten, and A. Bley. TOOMAS: interactive shopping guide robots in everyday use-final implementation and experiences from long-term field trials. In *IROS*, pages 2005–2012, 2009. 1

[15] S. Gulati, C. Jhurani, B. Kuipers, and R. Longoria. A framework for planning comfortable and customizable motion of an assistive mobile robot. In *IROS*, pages 4253–4260, 2009. 1, 2, 5

[16] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995. 3

[17] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. 1999. 4, 5

[18] T.-V. How, R. H. Wang, and A. Mihailidis. Evaluation of an intelligent wheelchair system for older adults with cognitive impairments. *Journal of neuroengineering and rehabilitation*, 10(1):90, 2013. 2

[19] A. Jain, D. Das, J. K. Gupta, and A. Saxena. Planit: A crowdsourcing approach for learning to plan paths from large scale preference feedback. In *ICRA*, pages 877–884, 2015. 6

[20] N. I. Katevas, N. M. Sgouros, S. G. Tzafestas, G. Papakonstantinou, P. Beattie, J. Bishop, P. Tsanakas, and D. Koutsouris. The autonomous mobile robot senario: a sensor aided intelligent navigation system for powered wheelchairs. *IEEE Robotics & Automation Magazine*, 4(4):60–70, 1997. 1

[21] R. Kirby, R. Simmons, and J. Forlizzi. Companion: A constraint-optimizing method for person-acceptable navigation. In *RO-MAN*, pages 607–612, 2009. 3, 5

[22] H. S. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. *PAMI*, 38(1):14–29, 2016. 8

[23] T. Kruse, A. Kirsch, E. A. Sisbot, and R. Alami. Exploiting human cooperation in human-centered robot navigation. In *RO-MAN*, pages 192–197, 2010. 8

[24] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013. 3, 5

[25] M. Kuderer, C. Sprunk, H. Kretzschmar, and W. Burgard. Online generation of homotopically distinct navigation paths. In *ICRA*, pages 6462–6467, 2014. 1, 2, 3, 4

[26] M. Labbe and F. Michaud. Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM. In *IROS*, pages 2661–2666, 2014. 7

[27] B. Lau, C. Sprunk, and W. Burgard. Efficient grid-based spatial representations for robot navigation in dynamic environments. *Robotics and Autonomous Systems*, 61(10):1116–1130, 2013. 3

[28] S. M. LaValle. *Planning algorithms*. Cambridge University Press, 2006. 3

[29] Q. Li, W. Chen, and J. Wang. Dynamic shared control for human-wheelchair cooperation. In *ICRA*, pages 4278–4283, 2011. 2, 8

[30] T. Linder and K. O. Arras. People detection, tracking and visualization using ros on a mobile service robot. In *Robot Operating System (ROS)*, pages 187–213. 2016. 5, 7

[31] T. Linder, S. Breuers, B. Leibe, and K. O. Arras. On multi-modal people tracking from mobile platforms in very crowded and dynamic environments. In *ICRA*, pages 5512–5519, 2016. 5, 7

[32] C. Mandel and U. Frese. Comparison of wheelchair user interfaces for the paralysed: Head-joystick vs. verbal path selection from an offered route-set. In *EMCR*, 2007. 3

[33] I. M. Mitchell, P. Viswanathan, B. Adhikari, E. Rothfels, and A. K. Mackworth. Shared control policies for safe wheelchair navigation of elderly adults with cognitive and mobility impairments: Designing a wizard of oz study. In *American Control Conference*, pages 4087–4094, 2014. 2, 8

[34] Y. Morales, N. Kallakuri, K. Shinozawa, T. Miyashita, and N. Hagita. Human-comfortable navigation for an autonomous robotic wheelchair. In *IROS*, pages 2737–2743, 2013. 1

[35] Y. Morales, T. Miyashita, and N. Hagita. Social robotic wheelchair centered on passenger and pedestrian comfort. *Robotics and Autonomous Systems*, 87:355–362, 2017. 3, 5

[36] S. P. Parikh, V. Grassi, V. Kumar, and J. Okamoto. Incorporating user inputs in motion planning for a smart wheelchair. In *ICRA*, volume 2, pages 2043–2048, 2004. 2

[37] S. P. Parikh, V. Grassi, V. Kumar, and J. J. Okamoto. Usability study of a control framework for an intelligent wheelchair. In *ICRA*, pages 4745–4750, 2005. 2

[38] J. J. Park and B. Kuipers. A smooth control law for graceful motion of differential wheeled mobile robots in 2d environment. In *ICRA*, pages 4896–4902, 2011. 5

[39] M. Shields. Use of wheelchairs and other mobility support devices. *Health reports*, 15(3):3741, May 2004. 1

[40] M. Shiomi, T. Iio, K. Kamei, C. Sharma, and N. Hagita. Effectiveness of social behaviors for autonomous wheelchair robot to support elderly people in japan. *PloS one*, 10(5), 2015. 1, 2, 8

[41] R. C. Simpson. Smart wheelchairs: A literature review. *Journal of rehabilitation research and development*, 42(4):423, 2005. 2

[42] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23(5):874–883, 2007. 1, 3, 5

[43] C. Urdiales, J. M. Peula, M. Fdez-Carmona, C. Barrué, E. J. Pérez, I. Sánchez-Tato, J. C. del Toro, F. Galluppi, U. Cortés, R. Annichiaricco, C. Caltagirone, and F. Sandoval. A new multi-criteria optimization strategy for shared control inwheelchair assisted navigation. *Autonomous Robots*, 30(2):179–197, 2011. 2, 5, 8

[44] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel. Sphinx-4: A flexible open source framework for speech recognition. 2004. 7

[45] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg. Who are you with and where are you going? In *CVPR*, pages 1345–1352, 2011. 8

[46] Q. Zeng, C. L. Teo, B. Rebsamen, and E. Burdet. A collaborative wheelchair system. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 16(2):161–170, 2008. 2