Learning to Navigate Robotic Wheelchairs from Demonstration: Is Training in Simulation Viable?

Mohammed Kutbi*
Saudi Electronic University
m.kutbi@seu.edu.sa

Bo Sun*
Stevens Institute of Technology
bsun7@stevens.edu

Yizhe Chang
California State Polytechnic University, Pomona
chang@cpp.edu

Philippos Mordohai
Stevens Institute of Technology
Philippos.Mordohai@stevens.edu

Abstract

Learning from demonstration (LfD) enables robots to learn complex relationships between their state, perception and actions that are hard to express in an optimization framework. While people intuitively know what they would like to do in a given situation, they often have difficulty representing their decision process precisely enough to enable an implementation. Here, we are interested in robots that carry passengers, such as robotic wheelchairs, where user preferences, comfort and the feeling of safety are important for autonomous navigation. Balancing these requirements is not straightforward. While robots can be trained in an LfD framework in which users drive the robot according to their preferences, performing these demonstrations can be time-consuming, expensive, and possibly dangerous. Inspired by recent efforts for generating synthetic data for training autonomous driving systems, we investigate whether it is possible to train a robot based on simulations to reduce the time requirements, cost and potential risk. A key characteristic of our approach is that the input is not images, but the locations of people and obstacles relative to the robot. We argue that this allows us to transfer the classifier from the simulator to the physical world and to previously unseen environments that do not match the appearance of the training set. Experiments with 14 subjects providing physical and simulated demonstrations validate our claim.

1. Introduction

Machine learning has had a profound effect on robotics over a long period of time. In the past few years, learning methods have become more data intensive and have achieved even higher accuracy, as long as sufficient training data are available. This requirement, however, is not always easy to satisfy because in many cases it is hard and expensive to acquire and annotate large datasets. Very recently, scientists were able to make breakthroughs in a number of perception problems by training, or pre-training, on synthetic datasets, which can be generated and annotated automatically. Examples of these success stories include progress achieved in object pose estimation [28], disparity and optical flow estimation [25], object detection for assisted and autonomous driving [1, 17], action recognition from video [11], and RGB-D image segmentation for grasping [10].

A number of frameworks for generating synthetic data have been created to facilitate these efforts: the MPI Sintel Flow Dataset [6] has aided progress in optical flow estimation; SYNTHIA [30] generates training data for semantic segmentation of urban scenes; the Virtual KITTI dataset [13] is a collection of synthetic videos for object detection, tracking, scene and instance segmentation, depth, and optical flow estimation; AI2-THOR [39] trains a visual navigation system in synthetic environments via deep reinforcement learning; Sim4CV [27] is a generator of simulated videos for multi-object tracking and autonomous driving as well as a number of other applications. These frameworks can generate large amounts of annotated training data without requiring any interaction with users. Ground truth for autonomous driving in Sim4CV is automatically generated by a waypoint prediction network, for example.

A different class of simulators generate data that are used for learning from demonstration by observing human users perform the tasks of interest. The most similar work to ours is CARLA [12], an open-source framework for training navigation systems for autonomous driving. Among many other functionalities, CARLA provides an imitation learning capability based on observations, commands and

*Equal contribution

actions recorded by users driving through the simulator.

The success of approaches based on simulated data in perception tasks is a very encouraging indicator of potential success in other domains. We advocate that the capability to train software for robotics in simulation can be transformative by reducing the cost and risk of developing such systems. It is also much more scalable, as previously argued by Zhu et al. [39], since copies of the simulator can be distributed to large numbers of "trainers" who do not need access to the physical robot or to appropriate space for training it.

In this paper, we present an approach for learning from demonstration using real and simulated data. Our approach has been implemented for a human-robot interaction scenario, the navigation of a powered wheelchair, but is applicable in other settings with appropriate modifications. The task we have targeted is learning the navigation preferences of wheelchair users by observing the choices they make when multiple paths are available between their current location and their destination. The challenge lies in that, while users can easily choose their preferred option, they have difficulty specifying an objective function that encapsulates all the criteria they consider. This makes automatic annotation of the data hard. To overcome this challenge, we train our classifiers using data that can be generated more easily by the users. Specifically, we use the ranking of the available paths that is generated every time a user makes a navigation choice. Each choice provides a signal that the chosen path is superior to those that were not chosen.

An important difference between our approach and those mentioned above is that photo-realism is not critical for the success of our simulator. The robotic wheelchair is equipped with a consumer depth camera (Microsoft Kinect) that allows it to estimate a partial occupancy map of the environment, as well as distances to obstacles and the locations of people. The planner reasons based on these data and not on input images. Our simulator enables users to navigate a virtual robot in maps, which is a natural task for most people. It is also able to generate the same type of data, namely occupancy maps and distances to obstacles and people.

Regardless of whether the data and user commands are captured on the physical wheelchair or in the simulator, the back end of our system comprises the same path planner and classifier. The planner must generate a number of diverse paths so that the options *not* selected by the user are also represented. To this end, among other options, we employ a planner based on the Generalized Voronoi Diagram (GVD) [5, 20, 9] that generates multiple homotopically distinct paths. During training, we map the user's trajectory to its homotopically equivalent path, while the other paths are labeled as less desirable than the one chosen, as shown in Fig. 1. A Support Vector Machine (SVM) is trained to rank

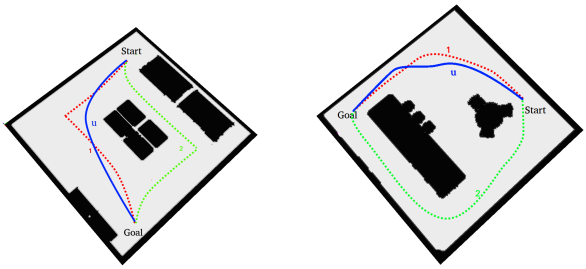

Figure 1. Examples of paths followed by a human subject (in solid line, labeled with a 'u') and generated by the GVD planner (in dashed lines, labeled with numbers). The subject's paths are homotopically equivalent to path '1' in both cases. Note that, even though the planner generated four homotopically distinct paths in the right map, only two are shown to reduce clutter.

paths according to user preferences.

We believe that our framework can be used in a variety of applications where subjective user preferences need to be captured to train a robotic system. Here, we develop an instance of the framework for training an assistive robot to select paths autonomously based on high-level user commands. We selected powered wheelchairs due to their importance as assistive devices to an estimated 1% of the world's population according to the 2010 US census and other sources [31, 8]. Our framework is well-suited for this problem since operating the wheelchair requires balancing criteria such as efficiency, comfort and safety.

To validate our approach, we recruited 14 subjects who navigated the wheelchair in 10 simulated and 2 actual environments. We first validated that the classifier can be trained by demonstration and can achieve high accuracy on real data with either real or simulated training data. We then deployed the trained classifier on the wheelchair and show that, given the destination by a voice command, the robot can choose the appropriate path to reach it. We feel that this last step is absolutely necessary since classifiers that achieve high accuracy on instantaneous inputs are not guaranteed to perform well in continuous mode in the world. The supplementary video shows examples of the deployed system.

2. Related Work

In this section, we review literature on path selection for passenger-carrying robots that take into account comfort and user preferences, as well as on training machine learning based systems on simulated data.

For an overview of robotic wheelchair systems we refer readers to [16] and [32]. The following publications are particularly relevant to the human-robot interactions aspects of our work. The approach of Gulati et al. [14] characterizes comfortable motion taking into travel time and measures such as tangential jerk, normal jerk, angular velocity and angular acceleration, but does not consider obstacles.

Zeng et al. [38] developed a collaborative wheelchair which automatically plans the path based on user-specified destination and speed. The user can take over to alter the path at any time. Urdiales et al. [35] proposed a shared control approach that combines commands generated by the robot and the user according to their relative efficiency of each task. A similar dynamic shared control system for wheelchairs was designed by Li et al. [23]. 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. In the work of Carlson et al. [7], the user guides the wheelchair while the robot adjusts the control signals to ensure safety.

The above methods do not address the presence of people in the environment. In addition to the following publications, we also refer readers to a relevant survey [19] for more information. Sisbot et al. [33] presented a motion planner that integrates safety and comfort in its cost function. Kirby et al. [18] proposed a navigation approach that takes into account people in the environment, travel distance and distance from obstacles. The resulting system is aware of expectations in human-robot social interaction; it respects personal space and passes on the right. Morales et al. [26] address wheelchair navigation emphasizing comfort for both its passenger and nearby pedestrians. Vanhooydonck et al. [36] presented a framework that determines whether a wheelchair user requires assistance based on estimates of the user's intentions and user-specific model of skills and behaviors. In this paper, we have adopted and modified aspects of the work of Chang et al. [9] who presented a method for learning personalized models for path selection and applied it in a shared autonomy framework where the user and the robot jointly navigate in the scene. All of the above methods required training and tuning the robot in the field. In this paper, we show that training in simulation is viable.

Soh and Demiris [34] proposed the Learning Assistance by Demonstration (LAD) approach that focuses on assistive systems and, like our approach, is implemented on wheelchairs. The key idea is that LAD does not attempt to learn how to drive, but rather it learns when and how to help the user. LAD aims to train a shared control system by demonstration using paired haptic controllers. A simulated and a physical robot are trained separately. The main differences with our approach are that we focus on autonomous operation by the smart wheelchair and that we transfer learning from a simulated to a real robot.

We now turn our attention to research on using synthetic or simulated data for training. We focus on continuous activities such as navigation where training is based on demonstration, as opposed to instantaneous tasks, such as object recognition. We assign to the latter category publications that address single-frame vehicle detection or semantic segmentation for autonomous and assisted driving [1, 13, 17, 29, 30]. As mentioned in the introduction, Sim4CV [27] can be used to generate synthetic data for training autonomous driving systems, but it does not support learning from demonstration. The proposed navigation system learns to pass through the generated waypoints, but not the driving preferences of the user. AutonoVi-Sim [4] is a similar framework that can rapidly simulate complex traffic scenarios, but also does not allow learning from demonstration.

CARLA [12] is an open source framework that provides a simulator with many functionalities including conditional imitation learning from user demonstrations, which is closely related to this paper. In contrast to our work, the observations of the environment are in the form of images, while we reason on a more abstract representation. Zhu et al. [39] presented an approach dubbed The House Of inteRactions (AI2-THOR) that uses deep reinforcement learning to train a visual navigation system. AI2-THOR relies on an actor-critic model that aids generalization, but its action space is limited to four actions (forward, left, right and back).

Our approach attempts to learn relatively complex relationships between the environment and the actions that the robot should perform. Recently, Bajcsy et al. [2] proposed an approach for learning objective functions from human guidance focusing on the reduction of unintended learning. All training, however, requires the physical robot. Basu et al. [3] addressed a similar problem using comparisons between trajectories, instead of demonstrations, as input to train the system. To enrich the information content of the comparisons, they automatically generate queries to the user on the importance of features to their choices.

3. Physical and Simulated Platforms

In this section, we present the platforms used to collect demonstration data for learning user preferences in path planning. In both physical and simulated settings, we record the environment and the trajectories as a user drives wheelchair.

3.1. Robotic Wheelchair

To collect data in real environments, we designed a robotic wheelchair based on a commercially available powered wheelchair controlled by a computer connected to the joystick interface. A Microsoft Kinect provides RGB-D images for mapping and localization (Fig. 2). The GUI has three screens, two of which are shown in Fig 3: (i) the main screen that allows the user to select the navigation mode (level of autonomy), (ii) the user navigation screen which shows a virtual joystick and controls to start and stop trajectory recording, and (iii) the autonomous navigation screen

which shows the map of the environment and all possible paths.

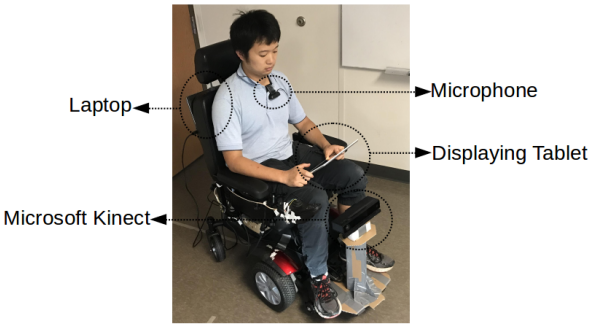

Figure 2. A picture of the robotic wheelchair

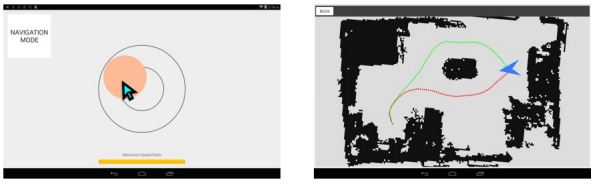

Figure 3. The user (left) and autonomous (right) navigation screens of the interface.

3.2. Simulation

Our simulator is built using the Robot Operating System (ROS) and the rviz package. The physical wheelchair has two differential drive wheels and two passive casters allowing it to rotate in place. It is not holonomic, since it cannot move sideways without rotating first. We implemented this motion model in the simulator by modeling the wheelchair as a Husky, an unmanned ground vehicle which is also non-holonomic and has four differential drive wheels allowing it to rotate in place.

We designed ten indoor environments using a free design tool, namely Homebyme (https://home.by.me/en/). We chose daily life environments, such as dining rooms, bedrooms, etc. The simulator interface shows the map, with the locations of the robot, obstacles and people, and the destination. Users can navigate the robot with realistic direction and speed control in the map using the arrow keys. Figure 6 shows two examples of simulated environments with and without people other than the wheelchair user. The other environments have more furniture and therefore have more homotopically distinct paths from one place to another.

4. The Generalized Voronoi Diagram (GVD) Planner

Given a map from the simulator or an RGB-D scan from the Kinect, we first convert them to an occupancy grid. Path planners [22] take as input the occupancy grid, the current location of the robot, its motion model and the goal location and generate paths connecting the current and the goal location. Most planners, however, generate only one path.

In order to learn the preferences of the robot's users, we must generate a number of different candidate paths. We distinguish paths based on the concept of homotopy [5, 20]. 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, as shown in Fig. 5. Considering Fig. 4(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 actual navigation, a wheelchair user can clearly identify these two paths. In the absence of the table in Fig. 4(b), the same paths cannot be distinguished based on homotopy. A wheelchair user is very likely to be indifferent to the choice between these two paths.

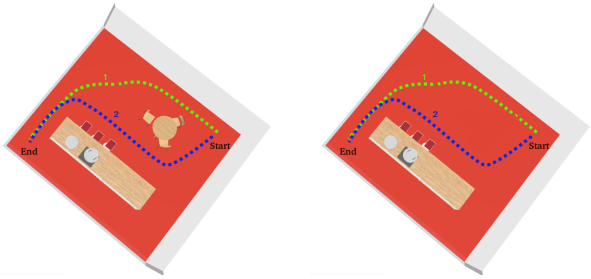

Figure 4. Left: Homotopically distinct paths. Right: Homotopically equivalent paths.

Our strategy for generating homotopically distinct paths relies on the GVD of the obstacles in the map, as presented initially by [20] and for semi-autonomous wheelchair navigation by [9]. It uses a property of the GVD, on which any different paths between two vertices are homotopically distinct. Therefore, the problem of finding k homotopically distinct paths for navigation is converted into the problem of finding the k shortest paths in a graph. Chang et al. [9] relied on this property to generate discrete plan hypotheses that were presented to the user to choose from. Here, we learn from demonstration and map the actual trajectories of the users to homotopically equivalent paths generated by the planner.

Figure 5 illustrates the path generation process. The GVD of a map is a set of unoccupied points whose distances to the nearest two obstacles are equal. It is called "generalized" because the input sites can be of any shape,

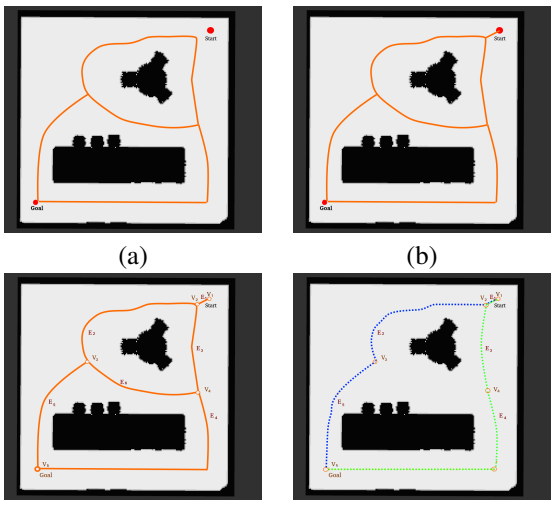

Figure 5. Illustration of GVD path planner. (a) Points on the GVD are extracted from the map. (b) Paths connecting the start and end points to the GVD are added. (c) The graph representation is extracted. (d) The k-shortest paths are found.

not just points. We start by computing a Euclidean distance transform to obtain a binary grid M that indicates whether a cell (x, y) is on the GVD or not. We then connect the start and end point to the nearest cells already on the GVD with straight paths.

The binary grid M is then converted to a graph representation $G = (V, E)$. A breadth-first search is used to traverse M and detect the vertices and edges of G. Vertices are cells with more than two incoming edges, while the cells containing the start and end points are vertices with one edge. We apply Dijisktra's algorithm to find the 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. Figure 5(d) shows the two shortest paths in the graph of Fig. 5(c).

After we have computed the GVD of the scene, we can map any path, complete or partial, the user has followed to its corresponding homotopy class. We can then attach features to the paths, as shown below, and use the classifier to compare two or more paths.

5. Path Representation and Ranking

In this section, we discuss how paths generated by the GVD planner or by users operating the wheelchair in a physical space or the simulator can be encoded in a representation that enables ranking. User-generated paths, real and simulated, are recorded during navigation as sequences of consecutive robot poses. They are first mapped to the GVD-generated path in the same homotopy class using the vector of winding angles as the descriptor of a path [20]. After this step, all paths can be treated in the same way regardless of how they were generated.

Every time we record a path, we assume that the user has selected the current path over a number of other paths generated by the planner. Therefore, the input to our path classifier comes in the form of ranking actions implicitly made by the user. Following [9], we model the system using a pairwise learning-to-rank model implemented as a Support Vector Machine (SVM) [15].

We use the term *scenario* to refer to the navigation from a specific start position to a specific goal position on a specific map. *Hence, there can be multiple scenarios in the same map.* Training data are collected by deriving ranking constraints from the data recorded during physical or simulated navigation. Specifically, given a user's chosen path, we infer that this path should be ranked higher than any of the other available paths in the current scenario. Candidate paths not chosen by the user cannot be ranked relative to each other and as a result they provide no additional constraints. Each scenario, therefore, provides k-1 constraints between the chosen path and each of the k-1 other paths.

5.1. Extracting Features from Paths

In this paper, unlike [9], we do not distinguish between scenarios with and without people, since the absence of people would set all relevant features to default values and would not affect ranking. The same classifiers work well with and without people. Our representation of a path comprises all the features in [9]. These features are: (i) path length, (ii) average distance to obstacles, (iii) minimum distance to obstacles over the entire path, (iv) narrow passage length, (v) sum of turning angles, (vi) average distance to people, and (vii) minimum distance to people over the entire path.

Note we do not directly compute the feature vectors of the paths that are demonstrated by the users. Instead, we first identify their homotopically equivalent paths, using the vector of winding angles mentioned above. We then compute the feature vector of these equivalent paths.

5.2. Support Vector Pairwise Ranking

Given a set of paths from the same scenario, we need to rank them in order of user preference. The input to this stage is pairwise constraints gathered at regular intervals during navigation and the feature vector of each path. We formulate the problem as ordinal regression using a Support Vector Machine (SVM) with pairwise constraints [15] .

All features are normalized to have zero mean and variance one. 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 $\hat{\boldsymbol{f}}(\boldsymbol{X})$. A preference vector $\boldsymbol{p_{ij}} = \hat{\boldsymbol{f}}(\boldsymbol{X_i}) - \hat{\boldsymbol{f}}(\boldsymbol{X_j})$ is associated with a label to indicate the preferred path.

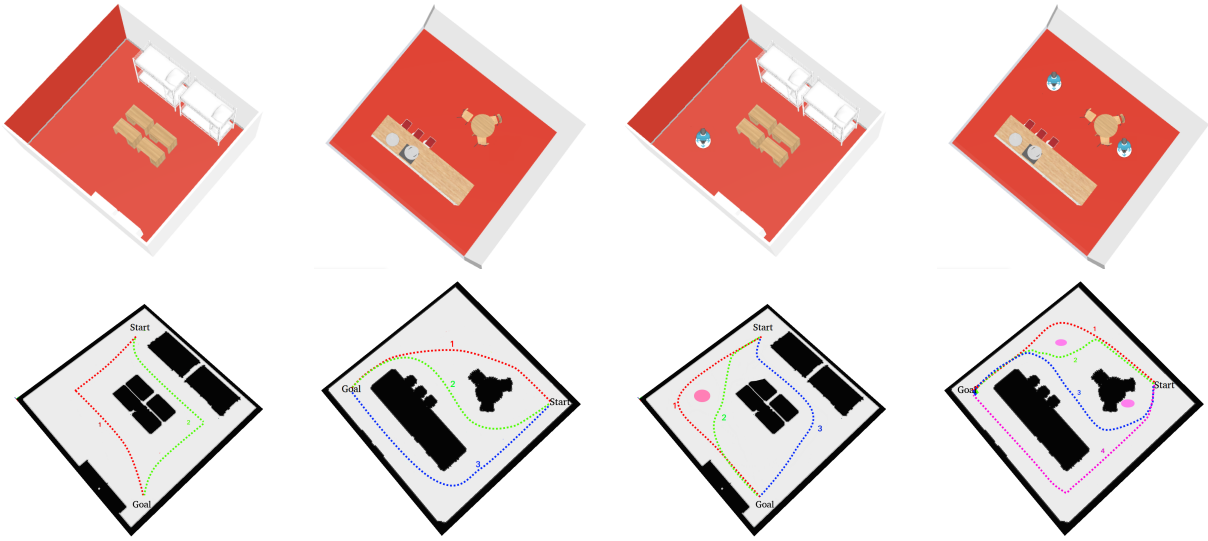

Figure 6. Sample scenarios in the two simplest simulated environments. Top: maps generated by the home design software, with and without people in the scene. Bottom: homotopically distinct paths for the above maps. People (shown as pink dots) may increase or decrease (block) the number of paths. For visualization purposes, not all generated paths are shown.

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

We train a linear SVM on the preference vectors to predict the preferred path.

Since the SVM is linear, we can compute the inner product of the SVM weight vector $\boldsymbol{w}$ and $\hat{\boldsymbol{f}}(\boldsymbol{X})$ before the subtraction to obtain a score for each path and select the one with the maximum score.

$$S(\boldsymbol{X}) = \boldsymbol{w}^{\boldsymbol{T}} \hat{\boldsymbol{f}}(\boldsymbol{X}) \quad (2)$$

6. Experimental Results

Fourteen able-bodied subjects were recruited and asked to navigate the robot indoors and in the simulator. The rooms where the physical experiments took place were mapped a priori using the Kinect and the RTAB-Map software [21]. The robot was operated in localization mode during the experiments and its trajectory was recorded. In both real and simulated runs, the GVD planner was applied without any modification to generate homotopically distinct paths. Unlike [9], in which subjects selected one of the k paths generated by the planner, we adopted an LfD approach. In our protocol, a researcher specifies the goal location verbally and the subject drives the robot following the most intuitive path, which is recorded.

Training examples are generated by executing the GVD planner at regular intervals, 0.5 m here, to produce up to eight paths each time. Among the generated paths, the one that is in the same homotopy class as the path taken by the user is selected as the positive example and ranking constraints are introduced between the positive and each of the remaining paths. Then, features are computed for all paths and classifiers are trained as described in the remainder of this section.

6.1. Data Collection in the Simulator

In the simulator, each subject was asked to navigate in 10 environments, with approximately 10 scenarios in each environment.[1] There are no people present in half of the scenarios, while one or two simulated people were added to the other half. We do not expect our system to accurately predict the future locations of the simulated people, since this would be a very error-prone strategy. Instead, our approach is to re-plan every 0.5 m to take into account new observations of the static and dynamic elements of the scene. Subjects were asked to navigate based on their preferences, taking into account the presence of people.

Figure 6 shows maps for simulated environments. Scenarios were designed to have 2–8 homotopically distinct paths. In static environments, five scenarios have two paths, five scenarios have four paths, and all others have eight paths; in environments with people, five scenarios have four paths and all others have eight paths. We set $k = 8$ for all experiments and drop paths that are not among the k shortest. No subject chose a path that was not in the top-8 in any experiment. In total, we collected 5227 samples in the simulator.

[1] A *scenario* is defined as a navigation between specific start and goal positions in a specific environment.

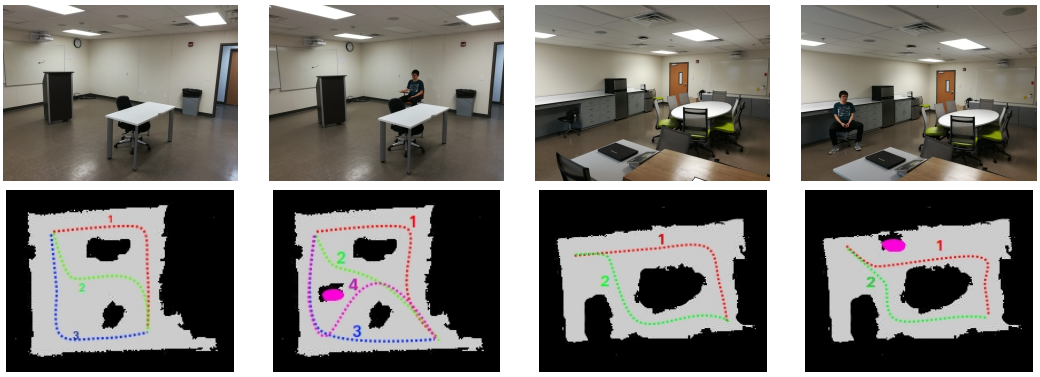

Figure 7. Scenarios in actual environments. Top: a classroom and a conference room with and without people. Bottom: homotopically distinct paths for the above scenarios. People (shown as pink dots) may increase or decrease (block) the number of paths. For visualization purposes, not all generated paths are shown.

Model/Scene	Static			Dynamic			Both		
(training) to (testing)	sim to sim	real to real	sim to real	sim to sim	real to real	sim to real	sim to sim	real to real	sim to real
Per-user mean	92.71	90.63	74.77	88.35	91.20	76.89	91.12	89.28	77.85
Per-user median	93.08	90.37	71.42	87.80	92.33	81.93	91.10	89.18	77.62
All users	92.60	84.84	73.91	89.32	84.75	81.88	91.01	86.20	78.44

Table 1. Average classifier accuracy using two-fold cross-validation. Left: on static scenes without people. Middle: on scenes with people. Right: on all scenes. We trained both per-user and across-user classifiers.

6.2. Data Collection using the Robotic Wheelchair

We performed physical experiments in a conference room and a classroom after re-arranging the furniture to create paths wide enough for the wheelchair. The two rooms are shown in Fig. 7. A large table is at the center of the conference room allowing the wheelchair to pass on either side. Therefore, there are two homotopically distinct paths between any two locations. In some cases, for example when the start point and the destination are on opposite sides of the table, the top two paths have similar feature vectors, while in other cases the top paths have considerably different feature vectors. In the classroom, a table and a podium introduce four distinct paths in general. Each subject was asked to complete ten navigation scenarios in each room, five with and five without people present. We used the SPENCER upper-body detector [24] to detect people. As mentioned above, we do not attempt to predict people's motion due to the difficulty of the task. Re-planning is a more effective strategy. The people in the scene were instructed not to behave adversarially, for example by trying to block all paths. We expect that the wheelchair will be deployed in environments in which people try to help its users. In total, we collected 1249 samples in real scenes.

6.3. Training and Validation Results

Using the collected data, several SVM classifiers were trained using two-fold cross-validation, according to Section 5.2. The classifiers are different in terms of the type of scenes they were trained on (with or without people) and in terms of the type of training and test data (simulated or real). We trained classifiers on: (i) data from scenes without people, (ii) data from scenes with people, and (iii) all data. We also trained classifiers to make predictions on: (i) simulated data based on simulated data, (ii) real data based on real data, and (iii) real data based on simulated data. (We do not see much value in classifying simulated data based on physical demonstrations.) We also trained classifiers combining data from all users and per-user classifiers.

When training a classifier on data from all subjects, all data from a single subject were placed in the same fold. In other words, each fold comprises data collected from seven subjects and it is used as the training or the test set. No classifier was tested on data from a subject that was included in the training set. Moreover, the maps of the actual rooms are not used in the simulator.

We also trained per-user classifiers. In this case, the data are split into folds according to environment. Data from the same map cannot belong to different folds.

Table 1 shows two-fold cross-validation results on all the combinations described above. In addition to the accuracy of classifiers trained on all data, we also report the mean and median accuracy of the 14 per-user classifiers for each setting. Overall, classifiers trained and tested on simulated data have the highest accuracy, most likely because there are fewer unmodeled sources of error in the simulator. Training and testing on real data achieves the second highest accuracy, while transferring a classifier trained with simulated data to the real domain leads to a loss of accuracy between 3 and 11% depending on the scene type. It is

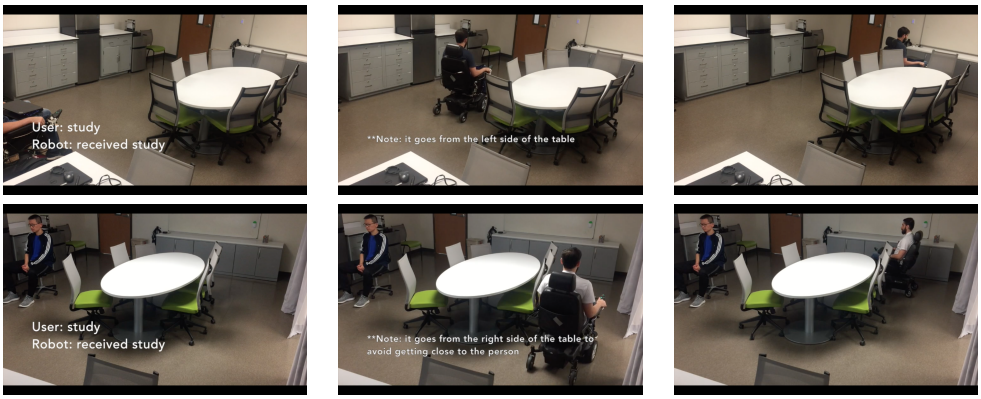

Figure 8. Photographs of the system operating in autonomous mode. Top: static scene. Bottom: dynamic scene. The system chose different paths due to the presence of a person in the second scenario, even though the person does not block the left path. Please see the supplementary video.

worth noting that the presence of people has stronger effects on navigation in real scenes, compared to navigation in the simulator.

6.4. Autonomous Navigation

As a final validation of our system, we deployed classifiers trained on simulated data on the physical robot, which runs ROS. The ROS move-base package is used for navigation after a modification that replaces the global planner by our GVD planner and the classifier. We use the dynamic window approach as the local planner. Given a, potentially partial, map, the move-base package can execute a path taking into account the robot's physical properties. We also use RTAB-Map [21] for mapping and localization and SPENCER [24] for upper-body detection. Localization, mapping and person detection rely entirely on a Microsoft Kinect for input. Finally, the CMU Sphinx library [37] is used for voice recognition.

In autonomous mode, the user gives high-level verbal commands to specify one of the labeled areas of the environment as the destination. The planner then invokes the classifier to select the path that the user would have taken under the same conditions. Figure 8 includes frames of the supplementary video showing autonomous navigation using a user-specific classifier. Navigation is smooth, even though the classifier has been trained using solely simulated data in different maps than the test scene.

7. Conclusions

We have presented an approach for learning path planning preferences from demonstration, which can take place either using a physical robot or in a simulator. After the data have been recorded on either platform, scenarios are generated and processed in the same way. A scenario consists of a set of paths between two points in a given map. Each path is assigned to a homotopy class and represented by a feature vector. This representation that does not depend on image appearance is what enables us to apply the same training process on real and synthetic data. To achieve this capability we built on the classifier and one of the planners presented by Chang et al. [9], but pursue different research objectives, specifically full instead of shared autonomy. We demonstrate that: (i) classifiers trained on synthetic data can be effectively deployed on the physical robot, and (ii) learning by sampling multiple decisions (every 0.5 m) from the recorded paths leads to richer training data than observing a single path selection per map.

Extensive experiments were performed with the aid of 14 subjects, who provide approximately 20 physical and 100 simulated demonstrations each. While there is a gap in accuracy between path planners trained on real and simulated data due to errors in localization and people detection, the latter can be deployed successfully on the robot. We believe that the answer to the question we posed is affirmative and that training in simulation is viable. The implications of this finding is that training can now be faster, safer and cheaper compared to conventional, physical LfD approaches.

Our future work will focus on closing that gap. One promising direction is better modeling of people in the simulator, since there are discrepancies between their influence and that of real people on path planning. Informally, simulated people appear to be easier to ignore, while real people have a more imposing presence and stronger influence on the subjects controlling the robot during training.

Acknowledgement

This work was supported in part by the National Institute of Nursing Research of the National Institutes of Health under Award R01NR015371 and the National Science Foundation under Awards IIS-1527294 and IIS-1637761.

References

[1] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *IJCV*, 126(9):961–972, 2018.

[2] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan. Learning from physical human corrections, one feature at a time. In *ACM/IEEE International Conference on Human-Robot Interaction*, pages 141–149, 2018.

[3] C. Basu, M. Singhal, and A. D. Dragan. Learning from richer human guidance: Augmenting comparison-based learning with feature queries. In *ACM/IEEE International Conference on Human-Robot Interaction*, pages 132–140, 2018.

[4] A. Best, S. Narang, L. Pasqualin, D. Barber, and D. Manocha. Autonovi-sim: Autonomous vehicle simulation platform with weather, sensing, and traffic control. In *CVPR Workshop on Autonomous Driving*, 2018.

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

[6] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.

[7] 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.

[8] 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.

[9] Y. Chang, M. Kutbi, N. Agadakos, B. Sun, and P. Mordohai. A shared autonomy approach for wheelchair navigation based on learned user preferences. In *Workshop on Assistive Computer Vision and Robotics*, 2017.

[10] M. Danielczuk, M. Matl, S. Gupta, A. Li, A. Lee, J. Mahler, and K. Goldberg. Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic point clouds. *arXiv preprint arXiv:1809.05825*, 2018.

[11] C. R. De Souza, A. Gaidon, Y. Cabon, and A. M. L. Peña. Procedural generation of videos to train deep action recognition networks. In *CVPR*, 2017.

[12] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Conference on Robot Learning (CoRL),*, 2017.

[13] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016.

[14] 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.

[15] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *International Conference on Artificial Neural Networks (ICANN)*, 1999.

[16] 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.

[17] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? *ICRA*, 2017.

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

[19] 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.

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

[21] M. Labbe and F. Michaud. Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM. In *IROS*, 2014.

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

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

[24] 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.

[25] N. Mayer, E. Ilg, P. Fischer, C. Hazirbas, D. Cremers, A. Dosovitskiy, and T. Brox. What makes good synthetic training data for learning disparity and optical flow estimation? *IJCV*, pages 1–19, 2018.

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

[27] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem. Sim4cv: A photo-realistic simulator for computer vision applications. *IJCV*, 2018.

[28] J. Papon and M. Schoeler. Semantic pose using deep networks trained on synthetic RGB-D. In *ICCV*, pages 774–782, 2015.

[29] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *ECCV*, pages 102–118, 2016.

[30] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, pages 3234–3243, 2016.

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

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

[33] 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.

[34] H. Soh and Y. Demiris. Learning assistance by demonstration: Smart mobility with shared control and paired haptic controllers. *Journal of Human-Robot Interaction*, 4(3):76–100, 2015.

[35] 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.

[36] D. Vanhooydonck, E. Demeester, A. Hüntemann, J. Philips, G. Vanacker, H. Van Brussel, and M. Nuttin. Adaptable navigational assistance for intelligent wheelchairs by means of an implicit personalized user model. *Robotics and Autonomous Systems*, 58(8):963–977, 2010.

[37] 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. Technical report, Sun Microsystems, Inc., Mountain View, CA, USA, 2004.

[38] 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.

[39] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017.