Noname manuscript No.
(will be inserted by the editor)

A decision-theoretic approach for the collaborative control of a smart wheelchair

Mahmoud Ghorbel · Joelle Pineau · Richard Gourdeau · Shervin Javdani · Siddhartha Srinivasa

Received: date / Accepted: date

Abstract While assistive robot technology is quickly progressing, several challenges remain to make this technology truly usable and useful for humans. One of the aspects that is particularly important is in defining control protocols that allow both the human and the robot technology to contribute to the best of their abilities. In this paper we propose a framework for the collaborative control of a smart wheelchair designed for individuals with mobility impairments. Our approach is based on a decision-theoretic model of control, and accepts commands from both the human user and robot controller. We use a Partially Observable Markov Decision Process to optimize the collaborative action choice, which allows the system to take into account uncertainty in the user intent, in the command and in the environment. The system is deployed and validated on the SmartWheeler platform, and experiments with 8 users show the improvement in usability and navigation efficiency that are achieved with this form of collaborative control.

Keywords First keyword · Second keyword · More

1 Introduction

Assistive technologies are essential for the well-being of many of us. The elderly and disabled communities in particular stand to benefit from new developments in technology that can help with activities of daily living, transportation, socialization, and management of illness or impairment. Yet this promise will only be realized if we can develop technologies that are easy and enjoyable to use.

Fig. 1: The intelligent robot: *SmartWheeler*

In this line, we have been developing a smart robotic wheelchair, the SmartWheeler (Figure 1), designed to assist individuals with mobility disorders with movement throughout their environment, including home, outdoors, shopping, etc. The robot is fully equipped with sensors, onboard computing, wifi, and autonomous navigation functionalities. Until now, users could only operate this wheelchair in one of two control modes: direct teleoperation (human) and full autonomy (machine). However, several experiments have shown that users are not satisfied with either mode of control. For several users, direct teleoperation can be difficult or impossible due to reduced muscle strength, tremors, perceptual impairments or cognitive deficits [6, 17]. In contrast, full au-

✉ Mahmoud Ghorbel · Richard Gourdeau
Department of Electrical Engineering, Polytechnique Montréal,Canada
E-mail: mahmoud.ghorbel@polymtl.ca

✉ Joelle Pineau
School of Computer Science, McGill University, Canada
E-mail: jpineau@cs.mcgill.ca

Shervin Javdani · Siddhartha Srinivasa
Carnegie Mellon University, USA

tonomy leaves many users feeling a loss of control and in some case frustration or confusion as to the machine's intent [11].

Our aim is to enable a middle ground through *collaborative control*, where the user and autonomy work together to achieve a goal. We present a decision-theoretic model that incorporates two components: *Prediction* and *Assistance*. The Prediction module utilizes both sensor data of the environment and user inputs to predict the user's intended goal. The Assistance module takes in this predicted distribution, and selects an action to minimize the expected user cost-to-go. To do so, we utilize a Partially Observable Markov Decision Process (POMDP) [9], and the model is based on the work of Javdani et al. [8], which focused on shared autonomy model for manipulator robots. To handle the particular challenges of navigation, in particular longer-horizon planning and dynamic obstacles, our model is designed to handle multiple hypotheses about user intent, command given, and environmental events. The proposed collaborative control method has been deployed onboard a smart wheelchair, and evaluated in a user study. In particular, we study the impact of the control model on navigation performance, cognitive workload and user preferences. Our results show that users had significantly less cognitive workload using our method than direct teleoperation, freeing them to focus on other tasks while safely navigating their environment.

The reminder of this paper is organized as follows: Section 2 presents a general description of the *SmartWheeler* robot platform and software architecture. Section 3 describes the proposed collaborative control architecture, including the Prediction and Assistance modules. Section 4 presents results of a user study comparing the usability and effectiveness of the manual, collaborative and autonomous navigation modes. Section 5 summarizes related work in this area and we finish with conclusions and ideas for future works.

2 The SmartWheeler robot platform

The *SmartWheeler* (Figure 1) was designed to provide users with driving assistance to enhance their mobility and ability to carry out their activities of daily living. Standard driving assistance features include driving forward/backward along a given path, avoiding static and dynamic obstacles, navigating narrow passageways (doors, corridors), docking near tables. Until recently, the user was restricted to choosing between two levels of control: manual mode, in which the user fully controls the robot without any intervention of the machine, and autonomous driving mode, where the onboard computer has full control of the navigation.
The *SmartWheeler* is built on top of a commercial powered wheelchair that has been equipped with multimodal communication interfaces including a usb touch-monitor, a microphone and two joysticks. It has an onboard laptop, wifi and GPS communication, custom boards (Raspberry Pi, Arduino) and several sensors including: sonars, Hokuyo UHG-08LX and SICK laser range finders, a 3D camera, a backup camera, an Inertial Measurement Unit (Gyroscope, Magnetometer, Accelerometer), and custom wheel encoders. This robot is also equipped with high quality customizable seating and cushion units, to ensure comfortable testing of users with a variety of health conditions.
The onboard laptop records all sensor and motion data. The Robot Operating System (ROS) is used to implement and execute all navigation functionalities. We also make significant use of virtual environment simulation of the SmartWheeler platform with *Gazebo*. Such simulation work is crucial as a first step to evaluate the robot in difficult or dangerous scenarios without any harm.

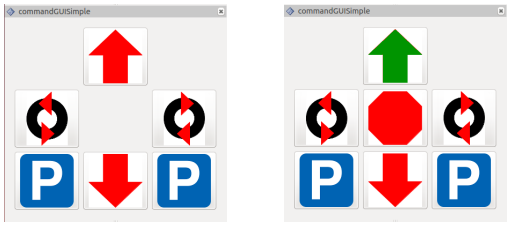


Fig. 2: Rviz Guide User Interface: The user choose a straight line move

A schema of the architecture operating onboard the *SmartWheeler* is shown in the Figure 3. As shown, the system incorporates data from the sensors into the two main navigation components. The Simultaneous Localization and Mapping (SLAM) provides real-time localization and contextual information. The Planner autonomously calculates a path to the goal. The supervisor incorporates messages from different controllers and sends velocity commands to the *SmartWheeler* motors.

To operate the robot in the autonomous driving mode, we developed a simple user interface through which the user can choose one of several primitive actions. S/he can select one action at a time from six choices (left and right in-place rotation, forward and reverse travel and left and right parallel parking). When the user selects an action (e.g forward travel) from the interface, the arrow turns green and a stop option appears (see Figure 2).

The *SmartWheeler* navigates autonomously while avoiding obstacles in the environment and stops automatically if there is no path or when the pilot hits the stop button or chooses another action. We can also operate the machine manually without any assistance or autonomy. To do that the controller applies directly the joystick inputs without any modification.

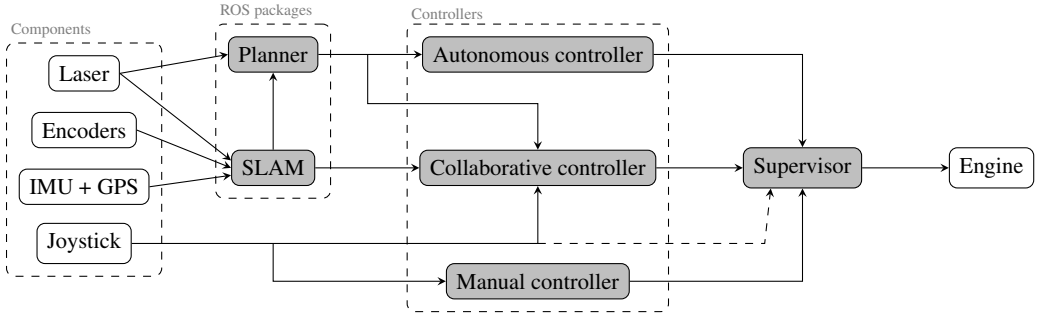


Fig. 3: Global Architecture of the three controllers

3 Modeling the collaborative control

We now outline the theoretical formulation of the semi-autonomous module's program. The collaborative control task is modelled as a Partially Observable Markov Decision Process *(POMDP)* with uncertainty over the destination. The system considers simultaneously the action provided by the robot, and the input data of the controlling joystick (user choice), to select an action that minimizes a cost function incorporating features of both control objectives.
Optimizing the *POMDP* policy poses some computational challenges, therefore we opt for an approximation based on *Hindsight Optimization*. This is a hybrid approach between the *MDP* (Totally Observable Markov Decision Process) and the *POMDP* (Partially Observable Markov Decision Process) solutions. It generalizes the value function defined over state in the *(MPD)* to a value function over beliefs for the *(POMDP)*. This approach is computationally efficient, with the same complexity as regular *MPD* solving. Throughout our work, we assume that the set of possible goal locations that the user might want to reach is known. The following Figure 4 shows the architecture of our collaborative controller.

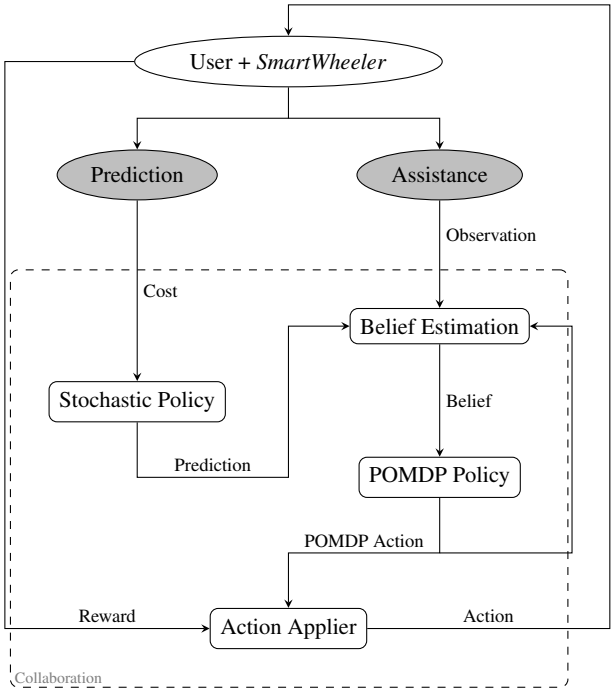


Fig. 4: Architecture of the collaborative controller

3.1 Wheelchair Motion Model

The *SmartWheeler* is a wheeled robot with two independently-controlled wheels and four casters (front and rear) for stability. Figure 5 describes the nomenclature of the kinematics of our model.

The motion model can be described as follows:

$$\begin{cases} \dot{x} = v\ \cos(\theta) \\ \dot{y} = v\ \sin(\theta) \\ \dot{\theta} = \omega \end{cases}$$

We can infer the transition-state of the robot for each couple of velocity command (v, ω) given by the user. We assume that this couple is constant due to the short estimation period of time. The coordinates in the map frame can be expressed with the following equations: (x_0, y_0, θ_0) represents the present configuration of the machine; $(x_\tau, y_\tau, \theta_\tau)$ describes the transition configuration at time τ in the future. If the user applies only a translation movement i.e $\omega = 0$,

$$x_\tau = x_0 + v\ \cos(\theta_0)\,\tau \tag{1}$$
$$y_\tau = y_0 + v\ \sin(\theta_0)\,\tau \tag{2}$$
$$\theta_\tau = \theta_0. \tag{3}$$

When the robot movement is composed of a translation and a rotation the transition configuration is computed as follows:

$$x_\tau = x_0 + \frac{v}{\omega}\ (\sin\theta_\tau - \sin\theta_0) \tag{4}$$
$$y_\tau = y_0 + \frac{v}{\omega}\ (\cos\theta_0 - \cos\theta_\tau) \tag{5}$$
$$\theta_\tau = \theta_0 + \omega\,\tau. \tag{6}$$

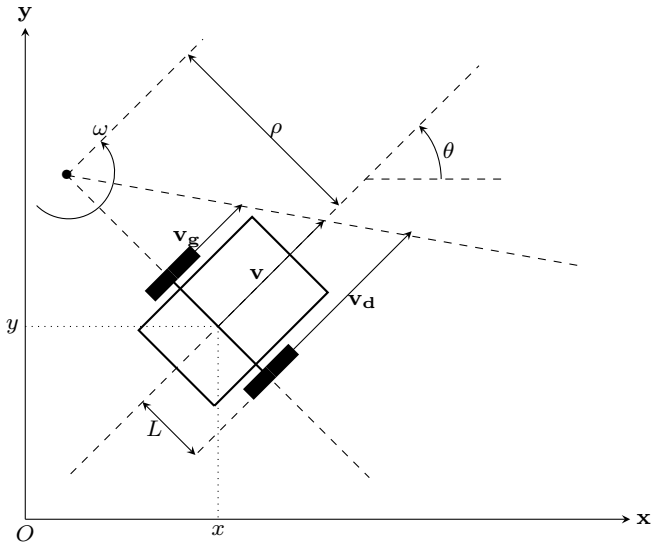


Fig. 5: *SmartWheeler* - Instant Center of Rotation •

3.2 Goal Prediction

Let $x \in X$ be the robot state (position and orientation), $a \in A$ the action (angular and linear velocity), $T : X \times A \rightarrow X$ the transition function (see 3.1), $u \in U$ the user's data input via the joystick, and $D : U \rightarrow A$ the Direct Manual mode function.

In this model the user optimizes a cost function $C_g^{usr} : X \times U \rightarrow \mathbb{R}$ for each goal $g \in G$. The goal prediction is defined by the tuple (X, U, T, C_g^{usr}), which captures our proposed model of the user choice. We consider a stochastic policy based on the maximum entropy inverse optimal control (MaxEnt IOC) [8, 24, 25]. This is slightly different from the standard Markov Decision Process *MDP* as we only use the cost from each input to predict the position of the intended goal and thus do not need to solve the policy decision completely to get the optimal action from the *MDP* (see the architecture in Figure 4). This model summarizes the *prediction* step of our collaborative control. It is generalizable to a sequence of states and inputs $\xi = \{x_0, u_0, \cdots, x_T, u_T\}$; for this, we define the cost as the sum of costs over time $C_g^{usr}(\xi) = \sum_t C_g^{usr}(x_t, u_t)$. The rest of the computational details are provided in a next subsection.

We note that $x_{t+1} \in \xi$ is not necessarily the result of applying u_t to the state $x_t \in \xi$. In other words, ξ is not a trajectory, but rather represents a set of states and inputs over time. In addition, while the *MPD* is generally defined with the action space, in our work, the user's Direct Manual command directly links the action space with the user's inputs space U. Also, this user policy is used only to predict the intended goal at each time step.

3.3 POMDP for Collaborative Control

As the system does not know in general which goal the user wants to reach, the Partially Observable Markov Decision Process *(POMDP)* model provides a mechanism to infer a distribution over possible goals [15, 18]. This offers a general framework to make decisions even if the states remains uncertain and the actions have stochastic effects.

In the *prediction phase* we compute a probabilistic distribution over goals, which is then used as an observation model in the *POMDP*. Similar decision models were recently used in several assistive algorithms [19, 20, 10, 8, 12]. In our work, we extend the state space X by augmenting the robot position and orientation with the intended goal g, $s = (x, g)$ and $S = X \times G$. With this modification, the transition function is analogous to the previous function but with a fixed goal $T : S \times A \rightarrow S$. The observation model of the *POMDP* is defined over the user's input space U and the resolution of the prediction policy. We compute $\pi_{t,g}^{\approx}(u_t|x_t)$ (user policy) with the user cost function for each goal to get a stochastic observation.

This model also uses a reward or cost function to evaluate the action chosen by the robot when the system is in the s state and the user gives the input u. This is denoted by $R^{robot} : S \times A \times U \rightarrow \mathbb{R}$. The fact that this function depends on the user input can lead to penalization of the robot final action if it is too different from the Direct Manual action $D(u)$. This formulation is adapted from [8], which defines a similar model for the case of a manipulator robot collaborating with a user. To summarize, our *POMDP* model is defined by the full tuple $(S, A, T, R^{robot}, U, \Omega)$

3.4 Observation Model of the POMDP

We assume that the human gives commands indicating the intent to move the robot to a specific goal. As mentioned above, we can generalize the user policy to a set of inputs and states ξ. For that, we consider the model of learning a cost function from demonstration [24], then we finally got the stochastic expression (8) for a specific goal. Note that the user policy for a sequence ξ depends only from the user inputs. Then, with Bayes's rule we could extract the probability of a goal given a sequence ξ for a period of time T.

$$p(\xi|g) \propto \exp(-C_g^{usr}(\xi)) \tag{7}$$

$$p(\xi|g) = \prod_t \pi_{t,g}^{\approx}(u_t|x_t) \tag{8}$$

$$p(g|\xi^{0\rightarrow T}) = \frac{p(\xi^{0\rightarrow T}|g)p(g)}{\sum_{g'} p(\xi^{0\rightarrow T}|g')p(g')} \tag{9}$$

This probability (9) represents then our *POMDP* observation model Ω.

The intelligent system policy, $\pi_g^{\approx}(x) = p(u|x,g)$, is based on the Maximum Entropy Inverse Optimal Control (MaxEnt IOC) model [8, 24, 25] . In order to determine the MaxEnt IOC model, we must compute the probability of the sequence ξ for a known goal. This requires integrating the exponential expression given in the equation (7) over all possible trajectories to determine the partition function of the probabilistic model. Unfortunately, this requires enumerating all the sequence and calculating all the costs for each step of time, which is computational intractable for all but the most trivial cases. Instead, we turn to a *dynamic programming* solution to determine the Value and Q-value functions. We adopt the system of Bellman equations described in Ziebart et al. [25].
Let $x^{'}$ be the transition state of the robot after applying a user command at time t, $x^{'} = T(x, D(u))$. We define the *softmin* function as follows :

$$\underset{u}{softmin}\, f(u) = -\log \int_u \exp\left(-f(x)\right) dx$$

These Bellman equations explain how to calculate the value function $V^{\approx}$, and Q-value $Q^{\approx}$, dynamically:

$$Q_{g,t}^{\approx}(x,u) = C_g^{usr}(x,u) + V_{g,t+1}^{\approx}(x^{'}) \tag{10}$$

$$V_{g,t}^{\approx}(x) = \underset{u}{softmin}\, Q_{g,t}^{\approx}(x,u). \tag{11}$$

This dynamic programming allows us then to compute the user policy as follows:

$$\pi_{t,g}^{\approx}(u|x) = \exp(V_{g,t}^{\approx}(x) - Q_{g,t}^{\approx}(x,u)). \tag{12}$$

This equation is in line with Eqns 7 & 8, under the assumption that the global cost is the sum of all costs over time, where Eqn 7 specifies that $p(\xi|g) = \prod_t \exp(-C_g^{usr}(x_t,u_t)$, and from Eqns 10 & 12 we infer Eqn 8.
Many previous works use a similar dynamic programming approach, with a slight difference in the choice of the optimization function [25]. We opted for a "softer" version of the *MDP* in order to get trajectories probabilistically distributed according to their values, rather than having a single optimal trajectory for the solution as is the case when using a *min* or *max* function. Let $V_{g,t}^{\approx}$ be expressed as follows:

$$V_{g,t}^{\approx}(x) = -\log \int \exp\left(-C_g^{usr}(\xi_x^{t\to T})\right),$$

where the integration is over the full set ξ, starting at time t from state x.
In the end, we obtain an estimate of the probability defining the observation model of the *POMDP*. The resulting probability is denoted by b and represents the *belief* function. This function depends on a single source of uncertainty, namely the intended destination g, $b(g) = p(g|\xi^{0\to T})$.

3.5 Decision-Theoretic Model for the Assistance Phase

To get the optimal assistive action for the collaboration, it is necessary to solve the *POMDP*. This may be computationally expensive, depending on the complexity of the policy, especially in continuous spaces. We opt for the *Hindsight Optimization* or *QMDP* which has similar level of complexity as an *MDP* solution and has been extensively used in other robots [8, 23, 18].

Using the observation model defined above, the collaborative action can be computed as follows:

$$\mathcal{Q}(b,a,u) = \sum_g b(g)\mathcal{Q}_g(x,a,u) \tag{13}$$

$$\pi(b,u) = \underset{a}{argmin}\, \mathcal{Q}(b,a,u) \tag{14}$$

These functions represent the value function $\mathcal{Q}$ and the policy π of the *POMDP* . When the robot begins to interact with environment the belief function is updated by solving the *MDP* corresponding to the possible robot goals using dynamic value iteration. The set of possible beliefs are approximated by stochastic sampling [21, 18].
Next we approximate the value function for a specific goal $\mathcal{Q}_g(x,a,u)$ with the *POMDP* cost function $R^{robot}(s,a,u)$, then running an iteration to compute the *POMDP* value function for each goal. This formalism relies on the specification of the two cost functions. We consider functions based on angular and linear distances to goal, as described further below.

3.6 Multi-Trajectory Model

In most environments, there may be many safe ways to reach a specific goal, for example alternative paths, or alternative ways to get around an obstacle. This choice can introduce further uncertainty in the collaboration between the human and the machine. To alleviate this problem, we define *intermediate goals*, also called *targets* and denoted by κ, that help the robot rapidly update and follow the user intention.

Consider the extension of robot and user cost functions, C_κ^{usr}, R_κ^{robot}, for targets, from which we can compute the κ-*value* and κ-*Qvalue* functions for prediction ($V_\kappa^{\approx}$, $Q_\kappa^{\approx}$) and the κ-*functions* for assistance (cost-to-go $\mathcal{V}_\kappa$, action-value $\mathcal{Q}_\kappa$). These last two functions follow the same *dynamic programming* as above but with a hard *min* function.

$$\mathcal{Q}_{\kappa,t}(x,a) = R_\kappa^{robot}(x,a) + \mathcal{V}_{\kappa,t+1}(x^{'}) \tag{15}$$

$$\mathcal{V}_{\kappa,t}(x) = \underset{a}{min}\, \mathcal{Q}_{\kappa,t}(x,a). \tag{16}$$

We now approximate the action-value function of the *POMDP* for each goal $\mathcal{Q}_g(x,a,u)$ using targets. Formally, let $\mathcal{V}_\kappa(x)$ be the cost-to-go for a specific target. We assume that the *POMDP* cost of a state action pair is the cost for the target

with minimum cost-to-go. We modify the notation of the robot cost to simplify the equations, so that $R^{robot}(x,g,a) = R^{robot}_g(x,a)$ The following equations describe the solution for the assistance step. The details of the demonstration are given in Appendix A.

$$R^{robot}_g(x,a) = R^{robot}_{\kappa^*}(x,a)\,,\ \kappa^* = \underset{\kappa}{argmin}\,\mathcal{V}_\kappa(x^{'}) \tag{17}$$

$$\mathcal{Q}_g(x,a) = R^{robot}_{\kappa^*}(x,a) + \mathcal{V}_{\kappa^*}(x^{'}) \tag{18}$$

$$\mathcal{V}_g(x) = \min_{\kappa}\,\mathcal{V}_\kappa(x) \tag{19}$$

Regarding the prediction step, it is worth noting that with a planned trajectory for assisting the user in the avoidance of dangerous scenarios, we can compute the partial probability for each target (7) $p(\xi|\kappa) \propto \exp(-C^{usr}_\kappa(\xi))$, then compute the different value functions by applying a *softmin* over the targets of the values distribution.

$$V^{\approx}_g(x) = \underset{\kappa}{softmin}\,V^{\approx}_\kappa(x) \tag{20}$$

$$Q^{\approx}_g(x) = \underset{\kappa}{softmin}\,Q^{\approx}_\kappa(x) \tag{21}$$

This choice is in line with the user policy mentioned before (see (12)). The proof is based on marginalizing the partial probability over the user input and over all the targets.

$$\begin{aligned}\pi^{\approx}_{t,g}(u_t,\kappa|x_t) &= \frac{\exp(-C^{usr}_\kappa(x_t,u_t))\int\exp(-C^{usr}_\kappa(\xi^{t+1\to T}_{x_{t+1}}))}{\sum_\kappa\int\exp(-C^{usr}_\kappa(\xi^{t\to T}_{x_t}))}\\ &= \frac{\exp(-C^{usr}_\kappa(x_t,u_t))\exp(-V^{\approx}_{\kappa,t+1}(x_{t+1}))}{\sum_\kappa\exp(-V^{\approx}_{\kappa,t}(x_t))}\\ \pi^{\approx}_{t,g}(u_t,\kappa|x_t) &= \frac{\exp(-Q^{\approx}_{\kappa,t}(x_t,u_t))}{\sum_\kappa\exp(-V^{\approx}_{\kappa,t}(x_t))}\end{aligned}$$

Finally we marginalize over the targets κ:

$$\begin{aligned}\pi^{\approx}_{t,g}(u_t|x_t) &= \frac{\sum_\kappa\exp(-Q^{\approx}_{\kappa,t}(x_t,u_t))}{\sum_\kappa\exp(-V^{\approx}_{\kappa,t}(x_t))}\\ &= \exp\left(\log\left(\frac{\sum_\kappa\exp(-Q^{\approx}_{\kappa,t}(x_t,u_t))}{\sum_\kappa\exp(-V^{\approx}_{\kappa,t}(x_t))}\right)\right)\\ \pi^{\approx}_{t,g}(u_t|x_t) &= \exp\left(\underset{\kappa}{softmin}\,V^{\approx}_{\kappa,t}(x_t) - \underset{\kappa}{softmin}\,Q^{\approx}_{\kappa,t}(x_t,u_t)\right).\end{aligned}$$

Throughout our empirical studies, we leverage algorithms for trajectory planning to enable static and dynamic obstacles avoidance. The trajectory in that case is computed over the set of targets considered. We primarily used the [1]*move_base ROS* open source package, both for planning to targets and for autonomous navigation, though other packages could be substituted.

[1] http://wiki.ros.org/move_base

3.7 Cost and Reward Functions

We conclude this section by exploring the choice of cost function. We primarily consider costs based on linear and angular distances to targets. As a result, all the *value* $V^{\approx}$ and *Qvalue* $Q^{\approx}$ function are defined for the translation mode and for the rotation mode. We choose the same model for these two modes but with different parameters. Also, depending on how far from the destination point or the final orientation the robot is, the prediction values vary linearly or quadratically in relation to the distance.
The user cost function is defined as follows:

$$C^{usr}_\kappa(x,u) = \begin{cases} c_1 & \text{if } d_\kappa(x,u) > \delta\,,\\ c_2 + c_3\,d_\kappa(x,u) & \text{if } d_\kappa(x,u) \leqslant \delta\end{cases}$$

where d represents the Euclidean distance between the robot position (state x) and the goal, or the angular distance between the orientation of the robot and the desired orientation of the goal/target depending on the mode. Here c_1, c_2, c_3 and δ are parameters to specify based on user preferences, needs, and driving behavior, as well as possibly basedon the robot's dynamics.
We further simplified the expression of the prediction function so it is represented as the integral of the user cost, hence the definition of both linear and quadratic modes. At each step, we compute $V^{\approx}_{trans}$ for translation and $V^{\approx}_{rot}$ for the rotation and the global prediction value $V^{\approx}$ is defined as a linear combination of these two values.

$$V^{\approx} = \alpha\,V^{\approx}_{trans} + \beta\,V^{\approx}_{rot}$$

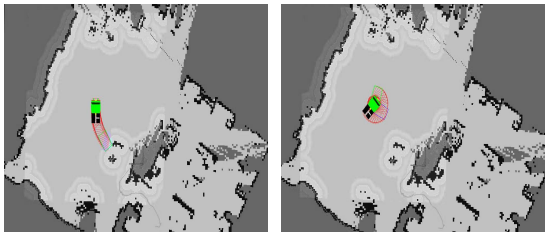
For the robot reward we consider the following model:

$$R^{robot}_\kappa(x,u,a) = C^{usr}_\kappa(x,u) + \|D(u) - a\|^2$$

So we can penalize the action of the robot if it's too different from the Direct Manual command.

3.8 Emergency assistance

Our approach also incorporates velocity assistance when the robot is near an obstacle. This assistance is triggered when the virtual footprint of the robot gives a non-zero cost. Using the transition function, the collaborative controller performs forward simulation from the robot's current state in order to predict what would happen if the given velocity command was applied for some period of time depending on the actual velocity (double of the stopping distance) and how the global cost would change. In those dangerous scenarios, the global costs increase when the user is approaching an obstacle. We compute the angular and the linear distance for the collision, if any, to identify the safest spot where the wheelchair should stop. Knowing this actual velocity, and the distance needed to



(a) (b)

Fig. 6: Simulation and visualization on *RviZ*
These figures show the trajectory that *SmartWheeler* will follow if the user gives a constant velocity command. The left figure 6a describes a potentially dangerous scenario with a composed command. The right figure 6b describes a circular trajectory without any collision. The green footprint represents the safest location to stop and the red ones represent every simulation step.

get the robot stopped, we can determine the new velocity for the next step and warn the user of an impending danger. To visualize the simulation of each command, we modeled a 3cm spaced set of footprints in *Ros Visualization* (see Figure 6).

4 Experimental Results

We performed several experiments with the real wheelchair to compare the Collaborative control described above with a direct Manual (joystick) control mode. The procedure as outlined below was approved by the Research Ethics Board at the participating universities.

4.1 The Wheelchair Skills Test

The list of tasks chosen to compare the controllers is extracted from the *WST*, version 4.2.1 [2], which was developed as part of the *Wheelchair Skills Program (WSP)*. The effectiveness of this test procedure to evaluate interface and control for smart wheelchairs was established in previous work [16]. A copy of the 10 tasks targeted in the experiments below is provided in the Appendix B.

4.2 Subjects

We recruited 8 participants (4 male and 4 female) from the university community. None of them had prior experience with either the *SmartWheeler* platform, or with the Wheelchair Skills Test, to avoid any training bias.

[2] *http://www.wheelchairskillsprogram.ca/eng/documents/FORM_WST-P-WCU_4.2.1_approved.pdf*

4.3 Experiment Setup

The setup of the experiment is inspired by prior work [3]. It requires subjects to not only complete the navigation tasks extracted from the WST, but also to perform some mental arithmetic. This increases the cognitive load on the subject, thus giving a richer measure of the efficiency of the collaborative control model.

For each participant, the steps are as follows:

1. The participant begins with answering a two-digit mathematical addition quiz for 2 minutes in order to evaluate individual math proficiency baseline level. The participant is given 4 possibilities for each equation and 5 seconds to choose an answer. At the end of the quiz, if the participant has more than 50% of success, s/he continues the experiment with a two-digits quiz, otherwise the difficulty level is reduced to a one-digit addition problem for the rest of the experiment.
2. Training Phase: The participant is asked to guide the robot around the robotics laboratory (Figure 7) using one of the two control schemes being tested (Collaborative control vs Manual Control). Each participant is trained with only one of the two modes. Randomizing between training mode allows us to measure the effect of training on the experimental results.
3. Testing Condition 1: The participant is now asked to do the WST tasks using the mode that he was trained with, but this time s/he is also asked to simultaneously answer the mathematical addition quiz of his/her level (without chronometer). The participant is instructed to aim to answer as many equations as possible, while ensuring that the robot does not make contacted with any obstacle.
4. Testing Condition 2: The participant repeats the same procedure (WST tasks + arithmetic) using the other control system. If s/he started with the Collaborative mode, then s/he moves to using the Manual mode, and vice versa.

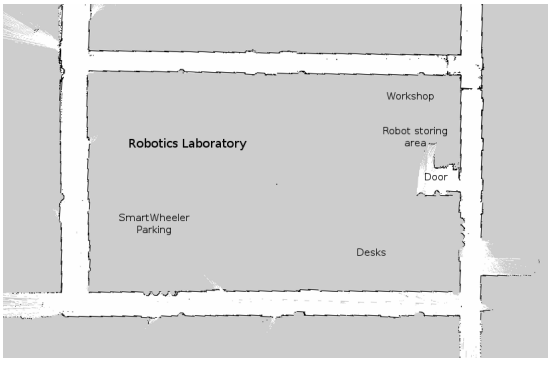


Fig. 7: The environment used for the training phase.

4.4 Data Acquisition and Evaluation Criteria

We consider four main criteria to evaluate the efficiency of the human-machine interaction: *Driving Performance*, *Human Performance*, *Cognitive Workload* and *Neglect* rate. These criteria and the method of evaluation for each was inspired by previous work [3, 13, 8].

Driving Task Performance (T_{ratio}) The first criteria measures the driving performance. It is a time-ratio measure, calculated as: $T_{ratio} = \frac{T_{optimal}}{T_{spent}}$, where the optimal time, $T_{optimal}$ is computed assuming the maximum velocity to travel the full distance for the trajectory of each task.

Secondary Task Performance (Q) The second criteria measures the numbers of arithmetic questions answered per minute. This performance term is not used to compare between two individuals, but rather we compare Q for a given participate under the two control schemes (manual vs collaborative).

Cognitive Workload (H) The task of operating a power wheelchair (intelligent or not) requires significant concentration, which can be affected by a disability or by the external environment, with adverse effect on the driver performance. The Cognitive Workload criteria characterizes the behavioral entropy of each participant, as measured by lack of smoothness in the control operation. Following the work of [7], we use collected data (joystick, sensors, commands) to build a predictive model of the joystick magnitude and angle at any given time. Their are several choices for modeling the joystick data; we apply a simple second-order model, whereby the joystick command at a time t is determined using the following formula:

$$\begin{aligned}\mathbf{J}_{p,t} &= \mathbf{J}_{t-1} + (\mathbf{J}_{t-1} - \mathbf{J}_{t-2}) \\ &\quad + \frac{1}{2}\left((\mathbf{J}_{t-1} - \mathbf{J}_{t-2}) - (\mathbf{J}_{t-2} - \mathbf{J}_{t-3})\right), \\ \mathbf{J}_{p,t} &= \frac{5}{2}\,\mathbf{J}_{t-1} - 2\,\mathbf{J}_{t-2} + \frac{1}{2}\mathbf{J}_{t-3},\end{aligned}$$

where

$$\mathbf{J} = \begin{bmatrix} v \\ \omega \end{bmatrix}.$$

If the participant gives the same control at times $t-3$ through $t-1$, the Taylor expansion predicts the same command at time t.
The error $\mathbf{e}$ is then computed as the difference between the actual command of the joystick $\mathbf{J}$ and the prediction one $\mathbf{J}_p$:

$$\mathbf{e}_t = \mathbf{J}_t - \mathbf{J}_{p,t}.$$

To evaluate the behavior entropy, it is important to determine a probability mass function to estimate the prediction error density function $p(\mathbf{e})$. The entropy is defined as follows:

$$\mathbf{H} = -\sum_{\mathbf{e}\in\mathbf{E}} p(\mathbf{e})\, Log\left(p(\mathbf{e})\right). \tag{22}$$

We create a normalized histogram from the error values, then discretize it into $2N+1$ unequally spaced bins. For this, we evaluate the 90 percentile value α of the error distribution $p(-\alpha \leqslant \mathbf{e} \leqslant \alpha) = 0.9$ in order to classify each error into intervals or bins.

$$[-\infty; -N\alpha],\ [-N\alpha; (-N{+}1)\alpha],\ ...,\ [(N{-}1)\alpha; N\alpha],\ [N\alpha; \infty].$$

Here α indicates the fundamental steering behavior for each individual and is used as the reference to measure the workload when performing different activities at the same time [13]. Additionally, $\mathbf{E} = \{\mathbf{e}_{-N}, \mathbf{e}_{-N+1}, ..., \mathbf{e}_{N-1}, \mathbf{e}_N\}$ represents the prediction error sequence that has been observed. The entropy, Eq.22, is calculated from this distribution, assuming a log base of $2N+1$.

Neglect Rates (n) This measure captures the percentage of time that the operator spent doing arithmetic problems. In order to evaluate the neglect rate, we compute the time spent solving the addition questions, divided by the total time spent doing the navigation task:

$$n = \frac{T_{quiz}}{T_{spent}}.$$

Questionnaire In addition to above criteria, we also collected additional feedback through a short questionnaire. Each participant was asked to answer the following questions, for each method of control, by giving a score from 0 to 5:

1) "I have the total *control* of the robot"
2) "The robot executed my *commands*"
3) "I am able to accomplish the WST tasks *quickly*"
4) "The robot reached the destination goal with *precision*"
5) "I feel *safe* while performing tasks"
6) "I *prefer* this control method"

Finally, participants were given the opportunity to give free-form comments on the collaborative navigation system, to be used for further improvement of the system.

4.5 Results

In addition to the measures taken during the user study, the robot also recorded all sensing and control data throughout the experiments. We start the analysis of the experiment by reporting some results based on this data collected through ROS.

Table 1: Smoothness of the driving behavior

	Collaborative Controller		**Manual Controller**	
Participant	P1	P2	P1	P2
$\alpha_{lin}\,(m.s^{-1})$	0.1832	0.1480	0.1326	0.1404
$\alpha_{ang}\,(rad.s^{-1})$	0.2230	0.1345	0.1494	0.0636

Goal Prediction Accuracy As part of the testing procedure, we considered four possible navigation goals and asked the user to guide the robot to one goal at a time. Our collaborative controller tracks the probability of each goal in real-time, and selects collaborative actions based on available path information for the inferred goal. Three cases are presented in the Figure 8. We observe that the system can correctly predict the goal based on available information. This is meant as a simple demonstration of the behavior of the system.

Emergency Assistance Next, we look at the behavior of the emergency assistance module by contrasting the behavior of two particular participants. Consider the linear commands given by individuals P1 and P2. We observe (comparing Figure 9 (a) and (c)) that the first participant seems very confident when operating with the collision avoidance mode ON: he engages near maximum linear velocity for a significant portion of time. Both of the participants completed the collaborative scenario without any collision; P1 experienced a collision when the collision avoidance module was OFF. This suggests that our module is reliable and helps the user navigate freely with less effort.

Consider now the overall driving style for P1 and P2 in Table 1. Here α is defined as in Section 4.4 (just below Equation 22). A lower α corresponds to smoother control. We observe that P2 seems to be more efficient and accurate in his joystick commands. Figure 10 shows the prediction error histogram for participant P1 used to evaluate the entropy.

User Study All participants successfully completed all the tasks with the two different control modes. The results for each participant are presented in Table 2. For most measurements, subjects tended to do better using the collaborative controller than the manual controller. Behavioral Entropy, which is correlated with the total input (less command) and with the neglect rate (more time to do math problem), shows that all users had better performances in the collaborative navigation. We did find that the collaborative module resulted in decreasing the global speed in some participants (lower T_{ratio}), but not all. But the incidence of Collisions was reduced to zero with collaborative control. The Secondary Task Performance (number of arithmetic problems solved) increased for all participants, as did the neglect rate. Participant *G* presents a slight exception, with lower T_{ratio} for the manual mode; this participant slowed down significantly after experiencing several collisions at the beginning of the Manual mode phase of the test.

Figure 11 illustrates the effect of training on the results. We observe that participants that trained with the Collaborative model require fewer inputs in the experiment with the Collaborative mode than those who trained with the manual mode. The number of commands for the Manual mode testing remained similar, regardless of training mode (though sightly lower for those trained in Manual mode).

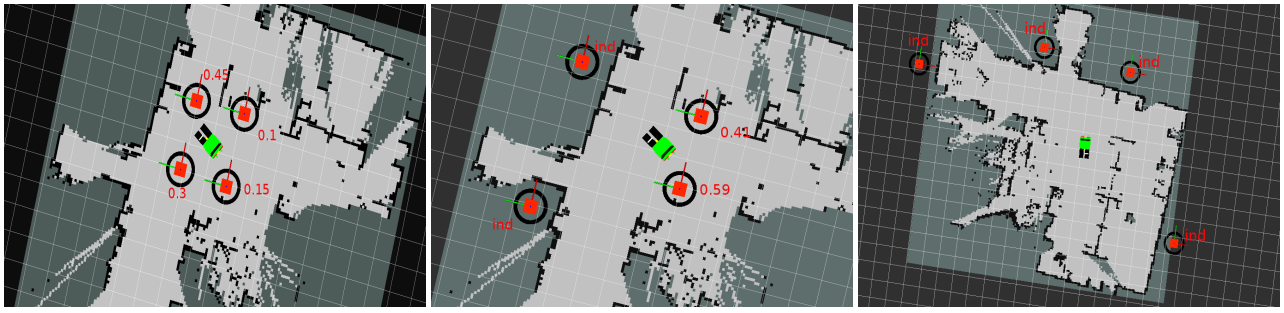
Figure 12 presents the results of the questionnaire given to each participant. We performed a Wilcoxon rank-sum test on the preference question [8]. The results suggest that the users perceive the collaborative control module to be safer, and potentially more precise. As expected, participants felt they had more control and the robot followed their commands more under the Manual mode. Participants also noticed the speed decrease. On average, participants seemed to have a preference for the Manual mode, primarily on the basis of the higher speed.

We do observe from the plot on the right side of Figure 12 that those users that had the highest preference for the Collaborative control (top right dots) also had larger difference between their preference for Collaborative vs Manual mode.

Autonomous Navigation with Collaboration Finally, it is worth pointing out that the Collaborative control module can work even when the user is not giving any joystick command which means that in the first mode when the module does not detect any input of user, no assistance is provided and the *SmartWheeler* stops.

To support this new mode, we created a semi-autonomous controller in which the robot operates primarily in autonomous mode, but the operator can take control at any time by putting a goal in the map and letting the *SmartWheeler* handle the navigation from that point on. For this mode, we updated some reward values of the module in order to increase the correction of the speed when the user is stopping.

This configuration was evaluated by some of the participants in a narrow corridor. We asked three subjects to navigate manually around our laboratory (Figure 7) for 50 meters, put a goal inside the workshop and then let the robot navigate by itself to the goal. They could give some commands if the wheelchair got stuck due to the emergency assistance.
All subjects were able to navigate autonomously in the hallway with other people wandering around, no collision was detected even when the wheelchair passed through the narrow door into the laboratory. The mean number of inputs per minute was 19.97 commands per minute for the three subjects, which is much lower than the results in Figure 11. These exploratory results suggest that the collaborative architecture described in this paper can serve for a wide range of different levels of navigation autonomy.



(a) (b) (c)

Fig. 8: Estimation of the observation for the POMDP

The estimated probability of each state goal object (orange cube), during collaborative control. (a) All the goal are accessible from the initial position (pointing to the spot on the top right), the user is giving a rotation command to the right (clockwise), the value function is changing for all the goals. The collaborative module is assisting the operator to reach the goal on the top left. (b) We change two of the goals (those on the left) to new locations with no accessible path, at which point their value functions become indefinite and the robot now assists the user to reach the nearest goal (at bottom left). (c) All the goals are changed to inaccessible locations, the user can guide the robot freely without any assistance, only the collision avoidance remains on.

Table 2: Evaluation criteria results. Each result represents an average value over the 10 WST tasks with a specific controller.

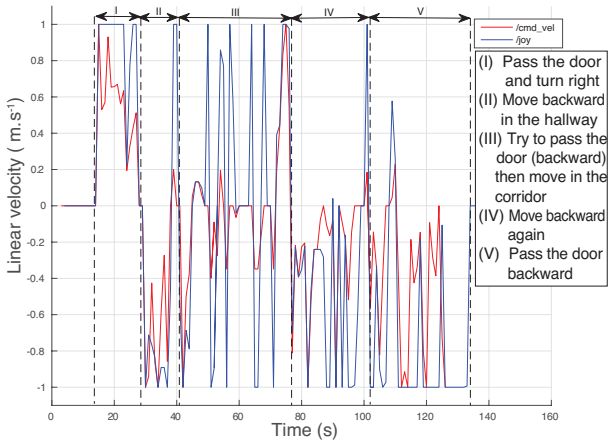
Participant	Mode	A	B	C	D	E	F	G	H	Ave.
Entropy $H\%$	**Manual**	58.99	52.12	49.56	51.9	48.45	52.03	51.4	55.19	52.46
	Collaborative	48.25	46.02	46.86	46.5	47.37	49.22	49.66	52.1	48.25
Driving Task Performance T_{ratio}	**Manual**	50.83	50.03	41.24	43.15	45.53	46.82	25.16	25.96	41.09
	Collaborative	45.1	35.5	38.01	34.42	44.30	36.02	35.18	25.86	36.8
Collision c	**Manual**	0	2	0	4	0	0	7	5	2.25
	Collaborative	0	0	0	0	0	0	0	0	0
Secondary Task Performance Q	**Manual**	14.13	13.29	15.16	11.13	13.15	12.71	8.7	13.41	12.71
	Collaborative	20.11	13.97	17.67	12.57	13.19	13.24	11.57	16.21	14.82
Neglect Rate $n\%$	**Manual**	97.2	75.42	93.16	76.67	91.88	85.72	88.04	94.66	87.84
	Collaborative	99.46	84.5	97.11	86.61	96.72	98.74	94.44	99.52	94.64

5 Related works

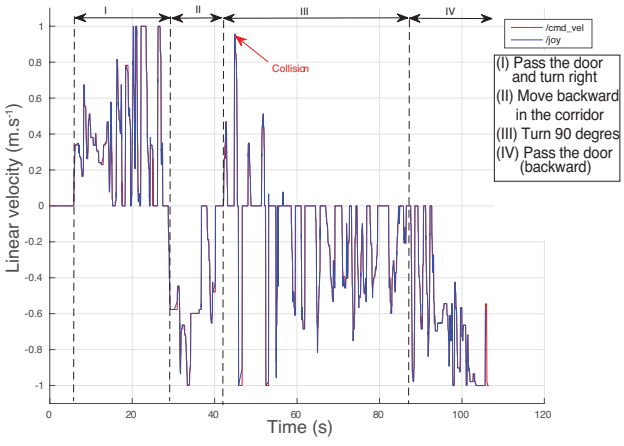
Several other works have investigated the problem of navigation for smart wheelchairs. A few examples are: *LURCH*, *Sharioto*, *NavChair* and *SHARE-it* (see [2, 1, 22, 4, 14]). Most of the algorithms proposed for these systems deal with very specific navigation scenarios, such as avoiding an obstacle, docking near a table, following a wall, driving through a door, etc. In many cases, the wheelchair can be controlled in only one of two modes: direct teleoperation, where the burden of control is entirely on the user, or full autonomy, where the robot takes on all required work but the user does not get to affect how the task is accomplished. Furthermore, many of the platforms require the user to explicitly select the control mode using a variety of interfaces (joystick, screen, buttons) [1]. This can be difficult for some users, due to their limited mobility and control abilities.

Two main categories of shared control have been explored. In the first case, the wheelchair chooses a specific control mode according to a user command. For example, if the operator is directing the chair towards at an obstacle, the collision avoidance module will be automatically activated without asking the user for permission. This approach may create confusion in some cases between the user and the wheelchair that can lead to collisions due to lack of communication between the two entities. The main disadvantage of this selection approach is that it does not take into account of the user's driving behavior. It supposes that every operator acts and reacts according to a general pattern which is not true especially for people with different pathologies. The *SHARE-it* project proposes an alternative solution whereby the robot adapts its autonomous commands to the user's driving style [2]. To achieve this, the controller must go through a learning phase involving different scenarios in order to create a case-base model, tuned to the user's preferences.

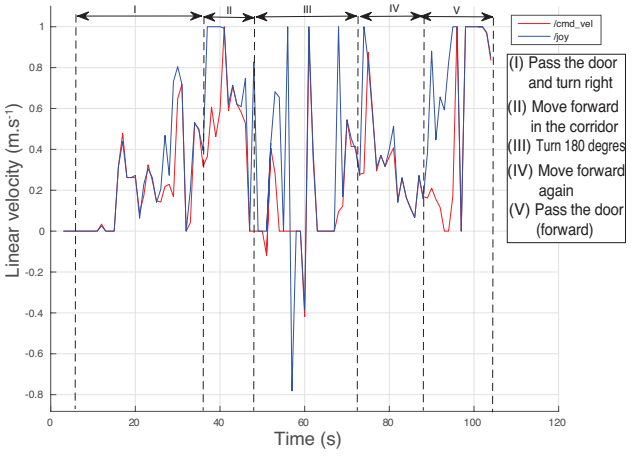
A second class of approaches uses information about the environment in order to implicitly change the mode of control. *NavChair* for example used sensor data to detect the type of environment and to select an assistance algorithm. However, this approach does not incorporate information about the user velocity commands, which can be problematic in some scenarios. For example, if we imagine a pedestrian (dynamic obstacle) moving toward the wheelchair, there is no way for the subject to communicate whether s/he wishes the chair to approach this pedestrian (e.g. to engage in conversation), or if s/he prefers to avoid the pedestrian by activating



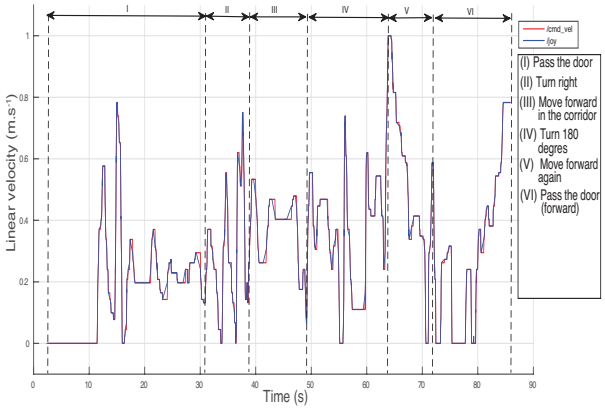
(a) P1: Collision Avoidance is ON



(b) P1: Collision Avoidance is OFF



(c) P2: Collision Avoidance is ON



(d) P2: Collision Avoidance is OFF

Fig. 9: Linear velocities

The blue graphs represent the velocity commands given by the user via the joystick and the red ones represent the commands sent to the motors, after mediation by the collaborative controller. We discretized those graphs into several period of times that include all the steps of every scenario described above.

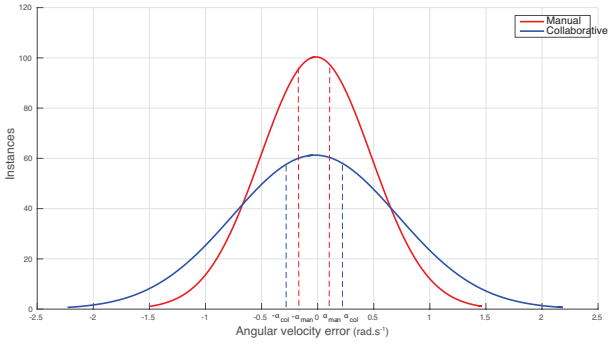


Fig. 10: Prediction linear error histograms of participant P1 for both control schemes. A Gaussian distribution is fitted to the histogram to find the coefficient α.

an obstacle avoidance module. There are many similar scenarios, where a subtle exchange of information between the robot and human can lead to much more fluid navigation behaviors.

The method we present generalizes the work of Javdani et al. [8], who present a shared autonomy framework for robotic manipulators. We adapt their framework based on the challenges of navigation, including handling longer-horizon planning and dynamic obstacles, and incorporating multi-trajectory capabilities. We also modify the space of states, actions, transitions and rewards to accommodate the navigation setting.

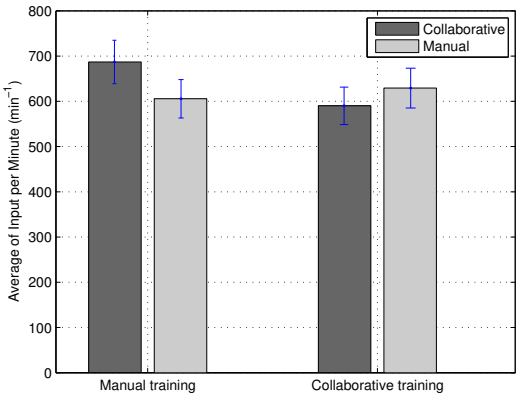


Fig. 11: Average of inputs per minute for all trials
On the left side, the number of input per minute for the subjects ($N = 4$) who trained with the Manual control mode. On the right side, the same measure for the subjects ($N = 4$) who trained with the Collaborative mode. Here, lower is assumed to be better (i.e. fewer input commands needed to achieve the same goal.)

6 Discussion and Future Work

This paper presents an approach for the collaborative control of smart wheelchairs. It uses a decision-theoretic strategy based on the Partially Observable Markov Decision Process to predict the intention of the user and assist in navigation, even in the presence of multiple goals. We performed an extensive user study to characterise the performance of our collaborative control module, compared to standard manual joystick control. The results show that our collaborative control gives more freedom to the participate to focus on other tasks and navigate securely.

Despite positive results for several of our metrics, overall the subjects still seem to have a mild preference for the manual control mode. For example, some users showed a preference for the manual controller when performing straight forward tasks because they liked having control of the wheelchair on a such easy task and reducing the speed for their safety with the collaborative module was not well accepted. However, when they were performing a backward tasks, reducing the velocity of the robot was liked by the users because it gives them an idea about the distance that separates them from some obstacles. The mild preference for manual control may be explained by the lack of practice and the fact that sometimes the policy makes choices that do not match perfectly the participant behaviour. But it could also be explained by the fact that the manual mode was faster; previous results have shown that user preference is correlated to the timing results [5].

During free-form comments, some participants indicated that we should give more explanation of how the collaborative module works and provide a method for the robot to explicitly communicate its policy to the user, especially when choosing goals or directions that are different from the user's command. This finding will lead to interesting new directions for the further development of the *SmartWheeler* platform.

The collaborative control strategy presented here for smart wheelchairs could be adapted to other robot platforms. However there are several considerations that may arise when transferring to other robots. First, we assume throughout that the user commands are provided via a joystick. Second, the user providing the manual commands has a first-person perspective on the environment, which will often not be the case for other robot platforms. In that case, it will be necessary to incorporate the difference in information available to each the robot and human operator. Finally, the type of environment challenges may be different in other domains. Handling different environments may require revisiting the definition of the cost and reward functions used in the collaborative controller.

Acknowledgements Funding for this project was provided through the NSERC Canadian Field Robotics Network (NCFRN), the NSERC Discovery program and the AGE-WELL NCE. Many thanks to Martin Gerdzhev, Alan Do-Omri, Anne-Marie Hébert and Dahlia Kairy for helpful suggestions throughout the experimental phase of the work.

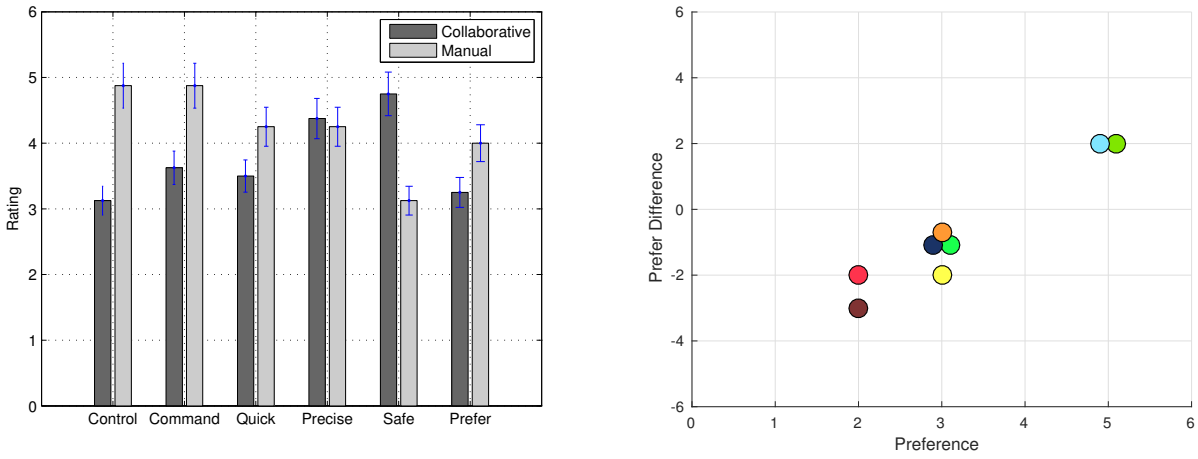


Fig. 12: Questionnaire results

Results for all questions (left graph). On the right, the difference in preference between modes (Collaborative preference rating - Manual preference rating), as a function of the preference level for the collaborative mode. Each dot characterize one participant (a specific color). When some users had the same result, the dots were spread slightly to show individuals.

Appendix A: Theorem demonstration

We detail in this paragraph the mathematical proof (by reverse induction) of the equation (15, 16):

Proof At the final time T,

$$\begin{aligned}\mathcal{Q}_g^T(x,a) &= R_g^{robot}(x,a)\\ &= R_{\kappa^*}^{robot}(x,a)\ \checkmark\end{aligned}$$

$$\begin{aligned}\mathcal{V}_g^T(x) &= \min_a \mathcal{Q}_g^T(x,a) \text{ (dynamic programming)}\\ &= \min_a R_g^{robot}(x,a) \text{ (no transition state at } T)\\ &= \min_a R_{\kappa^*}^{robot}(x,a)\\ &\geqslant \min_\kappa \min_a R_\kappa^{robot}(x,a)\\ &\geqslant \min_\kappa \min_a \mathcal{Q}_\kappa^T(x,a) \text{ (no transition state at } T)\\ \mathcal{V}_g^T(x) &\geqslant \min_\kappa \mathcal{V}_\kappa^T(x) \text{ (dynamic programming)}\end{aligned}$$

We seek to minimize our cost for the final destination at all time, so $\forall t \ \ \forall \kappa\, \mathcal{V}_g^t(x) \leqslant \mathcal{V}_\kappa^t(x)$, then $\mathcal{V}_g^t(x) \leqslant \min_\kappa \mathcal{V}_\kappa^t(x)$
Finally

$$\mathcal{V}_g^T(x) = \min_\kappa \mathcal{V}_\kappa^T(x)\ \checkmark$$

We assume that our equations are true for t,

$$\begin{aligned}\mathcal{Q}_g^{t-1}(x,a) &= R_g^{robot}(x,a) + \mathcal{V}_g^t(x') \text{ (dynamic programming)}\\ &= R_{\kappa^*}^{robot}(x,a) + \min_\kappa \mathcal{V}_\kappa^t(x') \text{ (Supposition } t)\\ \mathcal{Q}_g^{t-1}(x,a) &= R_{\kappa^*}^{robot}(x,a) + \mathcal{V}_{\kappa^*}^t(x') \text{ (definition of } \kappa^*)\ \checkmark\end{aligned}$$

$$\begin{aligned}\mathcal{V}_g^{t-1}(x) &= \min_a \mathcal{Q}_g^{t-1}(x,a) \text{ (dynamic programming)}\\ &= \min_a \left(R_{\kappa^*}^{robot}(x,a) + \mathcal{V}_{\kappa^*}^t(x')\right)\\ &\geqslant \min_\kappa \min_a \left(R_\kappa^{robot}(x,a) + \mathcal{V}_\kappa^t(x')\right)\\ &\geqslant \min_\kappa \min_a \mathcal{Q}_\kappa^{t-1}(x,a) \text{ (dynamic programming)}\\ &\geqslant \min_\kappa \mathcal{V}_\kappa^{t-1}(x,a) \text{ (dynamic programming)}\\ \mathcal{V}_g^{t-1}(x) &= \min_\kappa \mathcal{V}_\kappa^{t-1}(x,a) \text{ (previous reason)}\ \checkmark\end{aligned}$$

Our proof is complete. □

Appendix B: Wheelchaire Skills Test

Wheelchair Skills Test (WST) 4.2.1
Power Wheelchair-Wheelchair User
Participant Number :_____
Tester : _____
Date : _____
Time start : _____ Time finish : _____

#	Individual Skills	Score (0-2)	Comments
1	The user is placed in a huge space, then he moves in straight line for 10 meters and avoids pedestrians		
2	Move backward for 3 meters with avoiding pedestrians		
3	Go straight then turn 90° (Left or right)		
4	Turn 90° when moving backward		
5	Turn 180°		
6	Go through a narrow hallway (1.8 meter wide)		
7	Reach a destination point placed behind a ramp		
8	Reach a destination point placed behind a big obstacle		
9	Park next to a table or a wall (20 cm)		
10	Pass through a 1m width door		

Additional comments : _____
_____
_____
_____
_____
_____

Scoring options for individual skills

	Score	What this means
Pass	2	Task independently and safely accomplished without any difficulty.
Pass with difficulty	1	The evaluation criteria are met, but the subject experienced some difficulty worthy of note.
Fail	0	Task incomplete or unsafe.
Not possible	NP	The wheelchair does not have the parts to allow this skill.
Testing error	TE	Testing of the skill was not sufficiently well observed to provide a score.

References

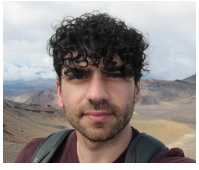
1. Bonarini, A., Ceriani, S., Fontana, G., Matteucci, M.: Introducing lurch: A shared autonomy robotic wheelchair with multimodal interfaces. In: Proceedings of IROS 2012 Workshop on Progress, Challenges and Future Perspectives in Navigation and Manipulation Assistance for Robotic Wheelchairs (2012)
2. Cortes, U., Barrue, C., Martinez, A.B., Urdiales, C., Campana, F., Annicchiarico, R., Caltagirone, C.: Assistive technologies for the new generation of senior citizens: the share-it approach. International Journal of Computers in Healthcare **1**(1), 35–65 (2010)
3. Crandall, J.W., Goodrich, M.A.: Characterizing efficiency of human robot interaction: A case study of shared-control teleoperation. In: Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on, vol. 2, pp. 1290–1295. IEEE (2002)
4. Demeester, E., Nuttin, M., Vanhooydonck, D., Van Brussel, H.: Assessing the user's intent using bayes' rule: application to wheelchair control. In: Proceedings of the 1st International Workshop on Advanced in Service Robotics, pp. 117–124 (2003)
5. Dragan, A.D., Srinivasa, S.S.: A policy-blending formalism for shared control. The International Journal of Robotics Research **32**(7), 790–805 (2013)
6. Fehr, L., Langbein, W.E., Skaar, S.B.: Adequacy of power wheelchair control interfaces for persons with severe disabilities: A clinical survey. Journal of Rehabilitation Research and Development **37** (2000)
7. Goodrich, M.A., Boer, E.R., Crandall, J.W., Ricks, R.W., Quigley, M.L.: Behavioral entropy in human-robot interaction. Tech. rep., DTIC Document (2004)
8. Javdani, S., Bagnell, J.A., Srinivasa, S.: Shared autonomy via hindsight optimization. arXiv preprint arXiv:1503.07619 (2015)
9. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence **101** (1998)
10. Karami, A.B., Jeanpierre, L., Mouaddib, A.I.: Partially observable markov decision process for managing robot collaboration with human. In: Tools with Artificial Intelligence, 2009. ICTAI'09. 21st International Conference on, pp. 518–521. IEEE (2009)
11. Kim, D.J., Hazlett-Knudsen, R., Culver-Godfrey, H., Rucks, G., Cunningham, T., Portee, D., Bricout, J., Wang, Z., Behal, A.: How autonomy impacts performance and satisfaction: Results from a study with spinal cord injured subjects using an assistive robot. IEEE Transactions on Systems, Man, and Cybernetics, Part A **42** (2012)
12. Matignon, L., Karami, A.B., Mouaddib, A.I.: A model for verbal and non-verbal human-robot collaboration. In: 2010 AAAI Fall Symposium Series (2010)
13. Nakayama, O., Futami, T., Nakamura, T., Boer, E.R.: Development of a steering entropy method for evaluating driver workload. SAE transactions **108**(6; PART 1), 1686–1695 (1999)
14. Nuttin, M., Vanhooydonck, D., Demeester, E., Van Brussel, H.: Selection of suitable human-robot interaction techniques for intelligent wheelchairs. In: Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on, pp. 146–151. IEEE (2002)
15. Pineau, J., Gordon, G., Thrun, S., et al.: Point-based value iteration: An anytime algorithm for pomdps. In: IJCAI, vol. 3, pp. 1025–1032 (2003)
16. Pineau, J., West, R., Atrash, A., Villemure, J., Routhier, F.: On the feasibility of using a standardized test for evaluating a speech-controlled smart wheelchair. International Journal of Intelligent Control and Systems **16**(2), 124–131 (2011)
17. Simpson, R.C.: Smart wheelchairs: A literature review. Journal of rehabilitation research and development **42**(4), 423 (2005)
18. Spaan, M.T., Spaan, M.T.: A point-based pomdp algorithm for robot planning. In: Robotics and Automation, 2004. Proceedings.

ICRA'04. 2004 IEEE International Conference on, vol. 3, pp. 2399–2404. IEEE (2004)
19. Taha, T., Miró, J.V., Dissanayake, G.: Pomdp-based long-term user intention prediction for wheelchair navigation. In: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pp. 3920–3925. IEEE (2008)
20. Taha, T., Miró, J.V., Dissanayake, G.: A pomdp framework for modelling human interaction with assistive robots. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 544–549. IEEE (2011)
21. Thrun, S.: Probabilistic algorithms in robotics. Ai Magazine **21**(4), 93 (2000)
22. Vanhooydonck, D., Demeester, E., Nuttin, M., Van Brussel, H.: Shared control for intelligent wheelchairs: an implicit estimation of the user intention. In: Proceedings of the 1st international workshop on advances in service robotics (ASERâŁ™03), pp. 176–182. Citeseer (2003)
23. Yoon, S.W., Fern, A., Givan, R., Kambhampati, S.: Probabilistic planning via determinization in hindsight. In: AAAI, pp. 1010–1016 (2008)
24. Ziebart, B.D., Maas, A.L., Bagnell, J.A., Dey, A.K.: Maximum entropy inverse reinforcement learning. In: AAAI, pp. 1433–1438 (2008)
25. Ziebart, B.D., Ratliff, N., Gallagher, G., Mertz, C., Peterson, K., Bagnell, J.A., Hebert, M., Dey, A.K., Srinivasa, S.: Planning-based prediction for pedestrians. In: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, pp. 3931–3936. IEEE (2009)

Mahmoud GHORBEL
is a developer and an engineer. He received his M.Sc degree from Polytechnique Montréal, Canada and his BSc in Electrical Engineering and Computer Science from Supélec, France. His research interests focus on robotics, artificial intelligence and shared autonomy control.

Joelle PINEAU
is a William Dawson Scholar and Associate Professor at the School of Computer Science, McGill University, in Montreal, Canada. She is the co-director of the Reasoning and Learning and a member of the Center for Intelligent Machines. She is also a Senior Fellow of the CIFAR program on Learning in Machines & Brains. Her research interests are in artificial intelligence, probabilistic machine learning and robotics.

Richard GOURDEAU
is a Professor of Electrical Engineering in Polytechnique Montréal. He received his Ph.D from Carleton University in 1991. His research interests focus on robotics manipulators control, control of mechanical systems, digital control and mobile robots control.



Shervin JAVDANI
is a PhD student at The Robotics Institute at Carnegie Mellon University. His goal is to enable people to use teleoperation systems efficiently through shared autonomy. To do so, his research focuses on utilizing machine learning and decision making under uncertainty to provide assistance. He received a BSc in Electrical Engineering and Computer Science from the University of California, Berkeley in 2010, and an MS in Robotics from Carnegie Mellon University in 2014.

Siddhartha SRINIVASA
is the Finmeccanica Associate Professor at The Robotics Institute at Carnegie Mellon University. He works on robotic manipulation, with the goal of enabling robots to perform complex manipulation tasks under uncertainty and clutter, with and around people. To this end, he founded and directs the Personal Robotics Lab, and co-directs the Manipulation Lab. He has been a PI on the Quality of Life Technologies NSF ERC, DARPA ARM-S and the CMU CHIMP team on the DARPA DRC.