pre-print: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), workshop on Navigation and Manipulation Assistance for Robotic Wheelchairs*, 12 October 2012, Vilamoura, Portugal.

Wheelchair Navigation Assistance in the FP7 Project RADHAR: Objectives and Current State

Eric Demeester, Emmanuel Vander Poorten, Alexander Hüntemann and Joris De Schutter
Department of Mechanical Engineering, KU Leuven, Heverlee, Belgium
first.last@mech.kuleuven.be

***Abstract*—This paper presents the research objectives and current state of the semi-autonomous navigation components that are being developed in the FP7 project RADHAR (www.radhar.eu). RADHAR proposes a framework to fuse the inherently uncertain information from both environment perception and a wheelchair driver's steering signals by estimating the trajectory the wheelchair should execute in that environment, and to adopt this fused information for providing safe navigation assistance. In order to provide intuitive, bilateral navigation assistance a haptic joystick prototype was developed and control algorithms were developed and tested for this haptic joystick. The key components for this semi-autonomous navigation assistance in the RADHAR framework are explained and experimental results are shown.**

I. Introduction

Autopilots in airplanes greatly reduce the pilot's workload by taking over parts of the navigation. Their success in reducing navigational complexity and improving safety motivates the introduction of navigational assistance in other transportation means as well. However, implementing robotic navigation correction on a large scale also represents a potential safety risk for millions of users. For this reason, if navigation devices are to be correcting the driver's steering signals, a thorough understanding of driver behaviour is imperative, as well as pervasive environment perception and interpretation. The linking between environment perception, driver perception and modelling, and robot decision making has often been weak and ad hoc. The RADHAR project proposes a framework to seamlessly fuse the inherently uncertain information from both environment perception and the driver's steering signals by estimating the trajectory the robot should execute, and to adopt this fused information for safe navigation with a level of autonomy adjusted to the user's capabilities and desires. This requires lifelong, unsupervised but safe learning by the robot. As a consequence, a continuous interaction between two learning systems, the robot and the driver, will emerge, hence RADHAR: Robotic ADaptation to Humans Adapting to Robots.

This framework will be tested on a powered wheelchair as a concrete testbed, given that many wheelchair drivers would benefit from navigation assistance, given that they are driving in very challenging 3D dynamic environments, and given that this is typically a very heterogeneous user group with varying skills and abilities. RADHAR will provide a solution for both semi-autonomous and autonomous wheelchair navigation in everyday environments. The remainder of the paper focuses on the current state of RADHAR's semi-autonomous navigation components.

A. Expected contributions by RADHAR

The RADHAR project targets the following main project outcomes insofar semi-autonomous navigation is concerned:

1) A navigation assistance framework for fusing environment and user perception, and for taking safe robot navigation actions based on the estimated robot task from uncertain driver signals, with an ability to take navigation decisions at 5 Hz or higher.
2) A repeatable benchmark test to evaluate navigation assistance systems based on driver models.

B. Demonstration platform

Fig. 1 shows one of the 4 demonstrator platforms that will be built. A fully-fledged RADHAR demonstrator will contain one haptic joystick, a touch screen with GUI, 4 Kinects (1 pointed at the user), 2 bumblebees for outdoor navigation, 3 laserscanners, 2 magnetic encoders, an Xsens IMU, and a single PC. Besides these physical demonstrators, also a wheelchair simulator is built on which navigation assistance algorithms can be tested first in a safe manner, and with which users can be trained.

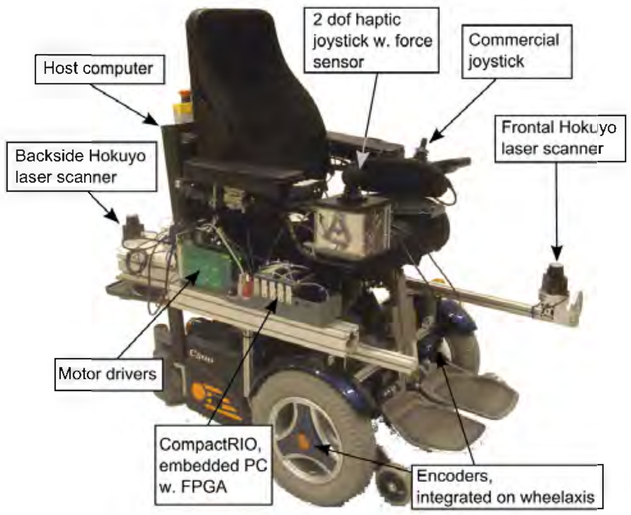

2 dof haptic joystick w. force sensor
Host computer
Commercial joystick
Frontal Hokuyo laser scanner
Backside Hokuyo laser scanner
Motor drivers
CompactRIO, embedded PC w. FPGA
Encoders, integrated on wheelaxis

Fig. 1: One of the 4 RADHAR demonstrator platforms.

C. Evaluation by user groups

The RADHAR system will be evaluated by wheelchair drivers from the school Windekind and from the Belgian national centre for Multiple Sclerosis. In particular, 3 groups of wheelchair drivers were selected: a group of starters who have not driven any powered wheelchair before, a group who easily collides when driving due to physical limitations, and a group with mainly cognitive problems, who disregard important aspects of their environment such as other people that are in their path.

D. Overview and system integration

This section provides an overview of the paper and describes at the same time how the different RADHAR components fit together.

Section II describes RADHAR's Bayesian navigation assistance framework in general. This framework estimates the driver's navigation intention from a set of executable robot trajectories and the driver's steering signals. For this, a statistical model of the driver's steering behaviour is learned.

In order to estimate the driver's intention and to quickly compute safe trajectories that can be executed, we need fast computation of a large number of collision-free robot trajectories, which is detailed in Section III.

The set of safely executable trajectories in the robot's local environment as well as the probability distribution over these trajectories will be adopted to provide navigation assistance to the driver. As described in Section IV, RADHAR aims to perform this using a haptic interface, as this establishes a fast bilateral communication channel between driver and computer.

Besides the navigation assistance components described in this paper, several other components are developed in the RADHAR project (more details and further references regarding these components can be found in [3]). They fit in the general framework as follows. To verify driver model assumptions such as focus-of-attention, the driver's posture, head orientation and eye (open or closed) will be continuously measured by a Kinect [5]. Besides driver perception and modelling, thorough environment perception and interpretation is required. Dynamic obstacles such as pedestrians and other wheelchairs are detected and labelled using cameras [11]. These objects are tracked, and they will be used as input to a socially compliant motion planner that generates feasible trajectories for the wheelchair taking the dynamic objects' motion into account [9]. Moreover, a local 3D map will be built and a traversability analysis will be performed [10], which is a necessary input to the motion planner as well. In order to compute safe navigation actions the driver's body posture is continuously observed and it is verified whether the driver is not outside a safety volume. This information will allow us to prevent the wheelchair from driving through a door when the driver's arms are pending outside of the wheelchair for example.

II. RADHAR'S NAVIGATION ASSISTANCE FRAMEWORK

To provide navigation assistance to wheelchair drivers with different abilities and needs, we've sought a general methodology that can adapt navigation assistance to each individual and to different types of interfaces. Furthermore, in order to avoid frustrating the driver and in order not to require the driver to provide too many input commands to the system or force him/her to learn new interfaces, the robot should recognise navigation plans in a way that considers the user's driving abilities such that the user can continue adopting his/her original interface.

We propose a *probabilistic* framework to recognise the user's navigation plans out of a set of *local trajectories*. In order to increase robustness, this framework considers the *uncertainty* when recognising user plans. It fuses past driving information with the user's specific driving style in order to estimate a posterior probability over user plans. Further, these estimated plans are the basis to share the control over the wheelchair according to the user's abilities and needs. The robot ensures the user's safety because it reasons about collision-free trajectories that are also physically executable, i.e. kinematically and dynamically feasible.

The robot first generates all possible user plans for a given situation (*plan generation*, see also Section III). These plans represent all navigation intents in an environment. Next, the robot reasons about the user's abilities to determine the probability of each plan. Calculating the probability of docking at a table (estimating the posterior) directly is more difficult than obtaining the probability of the input signals that are necessary to reach the table, assuming the user has a plan to reach the table in mind. We call the latter probability *user model*. However, the information available to the robot from the user model is quite limited. Often the robot is unable to disambiguate several intentions. Hence, it becomes essential to consider past driving behaviour.

Our probabilistic formulation employs Bayes' theorem to calculate the posterior probability of user plan $\boldsymbol{i}_k$, given the user's current input, $\boldsymbol{u}_k$, and past driving behaviour as follows:

$$\overbrace{p_{post}\left(\boldsymbol{i}_k|\boldsymbol{u}_k,\boldsymbol{\mathcal{H}}_{0:k}\right)}^{\text{posterior over user plans}} = \overbrace{p_{user}\left(\boldsymbol{u}_k|\boldsymbol{i}_k,\boldsymbol{\mathcal{H}}_{0:k}\right)}^{\text{user model}} \cdot \overbrace{p_{prior}\left(\boldsymbol{i}_k|\boldsymbol{\mathcal{H}}_{0:k}\right)}^{\text{prior over user plans}} \cdot \eta \qquad (1)$$

where η is a scale factor necessary to normalise the probability distribution. The history $\boldsymbol{\mathcal{H}}_{0:k}$ includes all previous user inputs up to time $k-1$, the sequence of robot actions $\boldsymbol{a}_{0:k}$, the sequence of robot poses, $\boldsymbol{x}_{0:k}$, and any external sensor readings, $\boldsymbol{z}_{0:k}$.

A. Learning the models of plan recognition

In order to recognise the user's navigation plans it is necessary to devise a model of the user's driving, as well as a way to relate past behaviour to the present. The first component is the user model of equation (1), which represents how a specific user transforms mental plans into inputs to the robot. The time evolution, linking the past to the present, concerns how to initialise the prior probability ($p_{prior}(\cdot)$) with the posterior of the previous time step.

We propose to learn the user model from direct observation of the user's driving. The user moves in the environment following a predefined sequence of destinations. Afterwards, during calibration, the complete trajectory of the user is known. Hence, it becomes possible to compare the actual trajectory with the set of local plans at each time step. The local path that resembles the actual trajectory the best is employed to predict the user's input with a probabilistic function estimation technique called Gaussian Process Regression [13]. Please refer to [8] for a complete description of the procedure to learn the user model.

The dataset of human-robot navigation also allows to learn the parameters of the prior in equation (1). Local plans, which resemble the actual trajectory of the user at subsequent times, should inherit probability. We formalise this insight as a series of soft probabilistic constraints in a Dynamic Bayesian Network [12] in order to learn the prior. We explain the complete procedure in [6].

B. Evolution of the framework

The framework to recognise the user's navigation plans was first proposed by Demeester et al. [1]. In this early work, navigation intentions are global destinations in the environment. Later work [2] expands the intention representation with the plan to reach the goal, as the same destination might be reached in different ways.

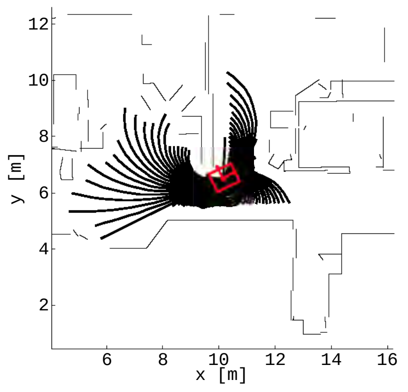
Another important aspect is the impact of steering disabilities on the user model. Some users might be unable to execute all trajectories perfectly. Hüntemann et al. study in [7] this effect for a user who is unable to turn left.

A further improvement is to learn the user model directly from driving data. In [8] Hüntemann et al. propose a learning framework based on Gaussian Process Regression [13] for user modelling and validate it on the driving pattern of a spastic user. The navigation plans in [8] are local trajectories instead of global plans. These local plans allow the robot to reason *locally* with rich collision information about a user who can drive anywhere. However, linking local plans across time without fixed global references is hard. The dissertation of Hüntemann [6] describes a learning framework to learn all parameters of local plan recognition.

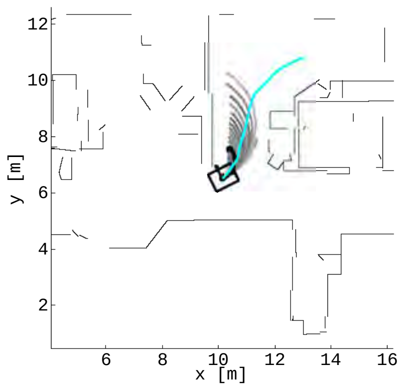
C. Analysing the plans of a user with spastic quadriplegia

In order to verify the validity of the approach, we have analysed the navigation plans of a user with spastic quadriplegia driving an electric wheelchair in an indoor environment. No navigation assistance was provided to understand how the user drives without external interference.

Fig. 2 shows a snapshot of the experiment, at which the user is locally turning on the spot after having driven forward. Fig. 2a, shows the set of considered local trajectories for plan recognition. Fig. 2b displays the same plans according to their posterior probability. Darker colours indicate higher probability. A cyan trajectory indicates the future path the user will follow, which allows to compare the estimated navigation plans with the actually executed trajectory.



(a) all local plans adapted to obstacles in the environment



(b) probability of each user plan, where darker colours indicate higher posterior probability

Fig. 2: Recognising the navigation plans of a user with spastic quadriplegia.

In this situation, plans of an intermediate curvature have the highest probability. Plans of high and low curvature still remain probable, whereas plans leading backwards are improbable. The most probable plan coincides locally with the future path of the user.

For an in-depth analysis of the user's driving style, please refer to the dissertation of Hüntemann [6].

III. Fast collision computation of a large set of possible paths

In order to maintain a probability distribution over a large set of possible driver plans, fast collision checking of the set of plan hypotheses is required. Section III-A explains the concept behind our approach to collision checking. Section III-B then applies this basic concept to a fixed set of local paths. Section III-C evaluates the performance of this algorithm.

A. General collision-checking concept

Our collision-checking algorithm for fixed sets of paths takes as input the following elements:

1) a map of the environment in the form of an uncertainty grid map consisting of a set $\mathcal{C}$ of grid cells. This grid map contains for each of its cells information about the presence of an obstacle in that cell, and possibly also other information such as whether or not the path is traversable. We model this information in our algorithm using the function $gridMap()$. The cells can be of any shape, depending on the needs of the application. Also 3D maps are possible.
2) a fixed set $\mathcal{P}$ of paths, each using a representation of choice, for example splines.
3) a representation of the robot's shape. As for the grid map, this can be of any type, for example a union of polygons.

The only requirement for the above elements is that for a given map cell $c \in \mathcal{C}$ and path $p \in \mathcal{P}$ it can be determined where along the path the robot's shape collides with the cell in case there is a collision, or, more generally speaking, what the minimum distance of the cell to the robot's shape is at each pose along the path. The robot shape needs to be known beforehand, though it may change along the path (e.g. due to castor wheels).

The algorithm then proceeds as follows:

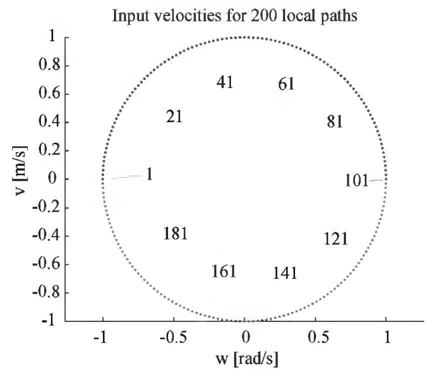
1) In the offline phase, two lookup tables are generated for the given map cells and the set of paths. The lookup table $affectedPaths$ stores for each cell c in the map which paths p cross this cell c. Lookup table $pathCost$ stores for each cell c its contribution to the cost of the paths that it affects.
2) In the online phase, it can then be efficiently checked for each of the occupied cells, which paths are affected by the cells and at which position along the path. This is explained in Algorithm 1.

Algorithm 1 Obstacle-based collision-checking.

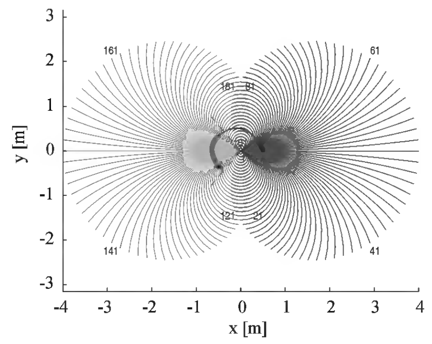
```
1:  for all $p \in \mathcal{P}$ do
2:      $cost(p) \leftarrow -\infty$
3:  end for
4:  for all $c \in \mathcal{C}$ do
5:      if $gridMap(c) == occupied$ then
6:          for all $p \in affectedPaths(c)$ do
7:              if $cost(p) < pathCost(c,p)$ then
8:                  $cost(p) \leftarrow pathCost(c,p)$
9:              end if
10:         end for
11:     end if
12: end for
```

B. Collision-checking concept applied to local paths

This section illustrates the cost-estimating approach for a fixed set of paths as explained in Section III-A by applying it to local paths, i.e. a fixed set of paths in the neighbourhood of the robot expressed in the robot's coordinate frame. Furthermore, the algorithm is described for the case where only collisions are computed. In this case, the $cost()$ function in Algorithm 1 only relates to the length of the path, and does not take other cost terms into account such as closeness to obstacles or terrain properties. Remark that a long path length corresponds to a low path cost. This paper does not focus on the design of the path shapes. Hence, only simple circular path shapes will be adopted. However, the algorithm is compatible with other path shapes as well.



(a) A set of 200 (v, ω) pairs, where v denotes the linear velocity in m/s, and ω the rotational velocity in rad/s.



(b) A set of 200 circular paths generated by integrating each of the (v, ω) pairs of the top figure during a time interval of $\Delta t = 4$s. Dark grey paths indicate forward motion, light grey paths indicate backward motion, and the two dashed black lines indicate turning on the spot over angle $\omega \cdot \Delta t$.

Fig. 3: Construction and representation of a set of circular local paths.

1) Input to construct the lookup table: Suppose we choose to check a set of circular paths. Fig. 3 shows a set of 200 such paths. The paths are obtained by integrating achievable robot velocities. Therefore, these paths are inherently kinematically feasible, i.e. they respect the robot's non-holonomic constraints. We represent a path as a $(v, \omega, \Delta t)$ tuple, where v denotes an arbitrary but constant linear velocity in m/s, ω a constant rotational velocity in rad/s, and Δt the time during which the velocities are integrated. For car-like vehicles, turning on the spot is not possible. Instead, a minimum turning radius should be respected. Obviously, paths for car-like vehicles can be constructed in a similar way, as well as paths for omni-directional vehicles.

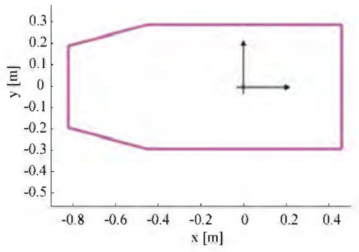


Fig. 4: Robot shape corresponding to the 2D footprint of the demonstrator in Fig. 1.

Furthermore the robot shape of Fig. 4 is chosen. This corresponds to the footprint of our robotic wheelchair, which is a front-wheel driven wheelchair.

Given the robot's shape and the set of paths, we can determine the area that is covered by the robot when moving over each of the paths. This determines in turn the required area of the local grid map (around 9.8 m $\times$ 7 m for the chosen parameters). Furthermore, we choose for a grid with square cells, and resolution Δx = 2 cm. Fig. 5 shows the dimensions of the grid map. It contains around 171.000 cells.

2) Offline computation of the lookup table: For each of the cells in the local grid map, it is now computed which of the paths pass through this cell, and at which position along the path. This can for example be performed by positioning the robot shape at a number of discrete poses along each of the paths, starting from the begin pose of the path, and to check which new cells are visited in each new step. In case new cells are visited, the path number is stored in that cell, as well as the pose along the path (modeled as a time in our representation). Fig. 6 illustrates this procedure for the circular paths. For the velocities (v_j, ω_j) of the j-th path it is determined which resolution δt will be adopted to go from the k-th pose $\mathbf{p_k} = [x_k \ y_k \ \theta_k]^T$ on this path to the $k+1$-th pose $\mathbf{p_{k+1}}$ by integrating $(v_j, \omega - j)$ starting from $\mathbf{p_k}$ during time interval δt. The time corresponding to $\mathbf{p_{k+1}}$ equals $t_{k+1} = t_k + \delta t$. In our implementation, we choose δt such that no point on the robot moves farther than half the grid resolution Δx. For all new cells that were visited during this time interval δt, the path number j is stored as well as the 'time' to collision, t_k. After this procedure, each cell knows the identity j of the paths that pass through it, as well as the time $t_{k,j}$ at which these paths cross that cell.

3) Online verification of collisions: Given the lookup table constructed in the previous section, and given an occupancy grid map of the surroundings of the robot, the set of paths represented with $(v, \omega, \Delta t)$ tuples is updated as follows. For each occupied cell, the Δt value of the paths j that go through it is set to the Δt stored in the lookup table for that cell and that path, unless the path's Δt was already smaller. Fig. 7 illustrates this procedure in case there are two grid cells occupied. Furthermore, the obstacles in the grid map are visited in a certain order: those cells that are encountered earlier on paths are checked first, as shown in Fig. 8.

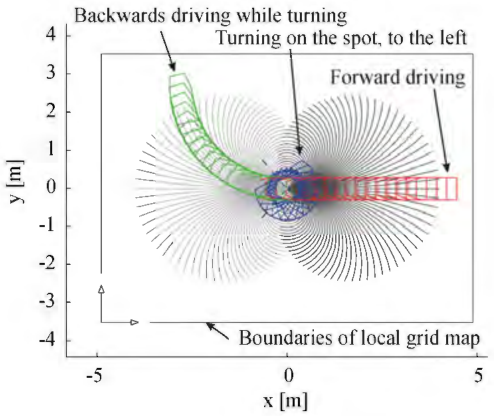
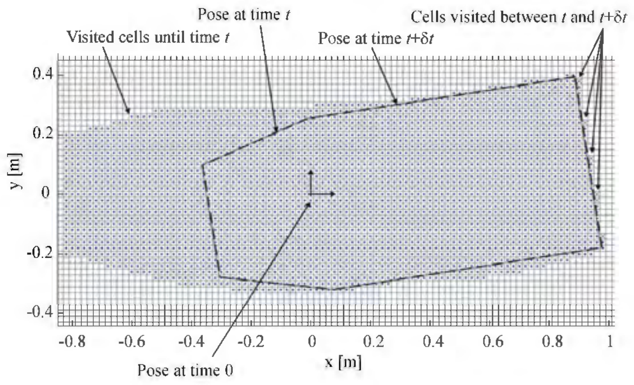


Fig. 5: Chosen local grid map with cell resolution Δx = 2 cm. Also shown are robot poses along 3 different paths: a pure rotation (v = 0 m/s, ω = 1 rad/s), a pure translation (v = 1 m/s, ω = 0 rad/s) and backwards driving while rotating (v = 0.94 m/s, ω = 0.34 rad/s).


Pose at time t
Cells visited between t and $t+\delta t$
Visited cells until time t
Pose at time $t+\delta t$
Pose at time 0
Fig. 6: This figure shows a part of path number 60 of Fig. 3. The cells visited between time 0 and time t_k are denoted with dots. The robot pose at time t_k is shown in grey full line, whereas the robot pose at time $t_k + \delta t$ is shown in dashed black lines. Since δt is chosen small (to accurately determine at which position new cells are visited), these two poses almost coincide. It is checked which new cells are visited (the red crosses in the figure, to the right) when integrating the path's velocities (v_j, ω_j) during time interval δt starting from pose $\mathbf{p_k}$. For these cells, the path number j and time to collision t_k are stored.

C. Evaluation and experiments

In [4], this approach is explained in more detail, and presents also some extensions such as for changing robot shapes that are user-initiated (e.g. with the driver's arms pending out of the wheelchair). We also compared our approach with a similar existing collision-checking approach and we show that our algorithm is both faster and less memory consuming. On a computer with Intel Core^{TM} dual core 2.80 GHz CPU with 4 GB RAM, it takes around 0.6 ms to compute collisions with 1000 paths in a home-like indoor environment. The computation time does not vary a lot with the number of occupied grid cells.

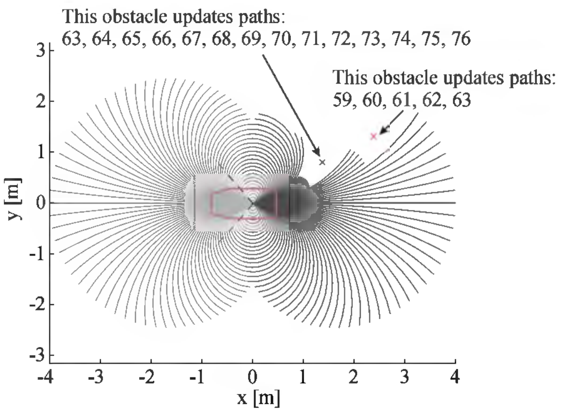


Fig. 7: This figure illustrates the online update of all paths based on two occupied cells, indicated by red crosses. The obstacles are occupied cells in the robot-centered local grid map shown in Fig. 5. In the cells, it is stored which paths should be adapted (indicated by the path numbers in the figure above), as well as their maximum length, which is in our representation determined by Δt. Both obstacles would update path 63, so path 63 is updated by the cell that gives it the smallest ‘length’ Δt.

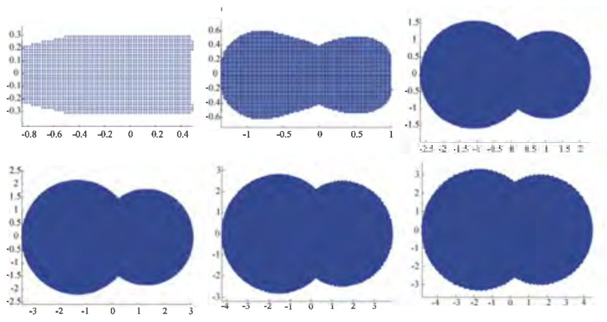


Fig. 8: Several snap shots showing the order in which cells in the occupancy grid are checked. Note that the scale of the axes changes. In total, almost 123.000 cells are checked for the paths in Fig. 3, the robot shape in Fig. 4, and a grid resolution of 2 cm (square cells).

However, this collision-checking approach does not take dynamic objects into account. In the near future, we intend to replace it with a a socially compliant motion planner [9].

IV. Haptic feedback and shared control

Given an estimation of the safe trajectories the driver most probably wants to execute, the RADHAR system is now able to provide actual navigation assistance to the driver. This will be performed by adopting a novel haptic joystick.

A. Haptic navigation assistance

Touch and kinesthesis (in short *haptics*) are subtle, effortless senses that are critically important for fast and accurate interaction with our environment. While picking up a pencil a complex interaction and exchange of forces takes place that allows us to grasp it in a stable manner without slipping nor thumbling. An interaction of that complexity takes place almost effortlessly, requiring hardly any mental effort. A *haptic display system* is a robotic joystick that can be programmed to exert well-controlled forces upon its user. Connected to a virtual reality environment with dynamics engine such joystick can replicate the forces that arise when e.g. picking up the pencil. In theory, and when well designed a user might find it troublesome to distinguish a virtual from a real pencil. Such system would thus feel natural and be intuitive to interact with.

Forces displayed by the haptic device should not necessarily replicate a purely physical phenomenon in order to be perceived as natural or intuitive. RADHAR looks at ways to *encode* navigation assistance schemes and present them via a haptic joystick to the wheelchair driver. If designed well, such *haptic environmental feedback* could feel intuitive and allow effortless navigation through complex environments. However, since there is often no physical equivalent for a navigation assistance scheme, finding an intuitive encoding is not straightforward at all. Furthermore, since wheelchair users often have limited power and get easily tired, haptic feedback should not increase the overall effort.

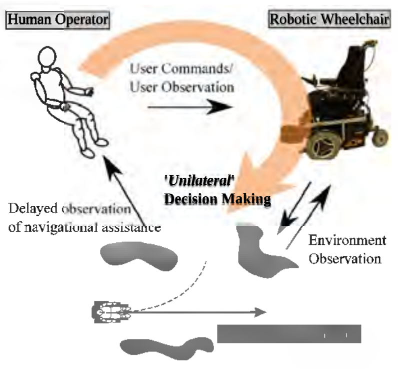
Compared to traditional navigation assistance or so-called shared control schemes [14], where users only perceive the decisions by the navigation system *after* wheelchair displacement, haptic guidance allows a more profound sharing of control. The user can negotiate directly with the navigation assistance system over this fast *bilateral* communication channel and should thus encounter less surprises on how the wheelchair will move (Fig. 9). Furthermore, by applying sufficient force upon the joystick, the user can always *overrule* the suggestion by the assistance system. So, in the end, the user can keep the control over the system.

B. Haptic obstacle avoidance along circular paths

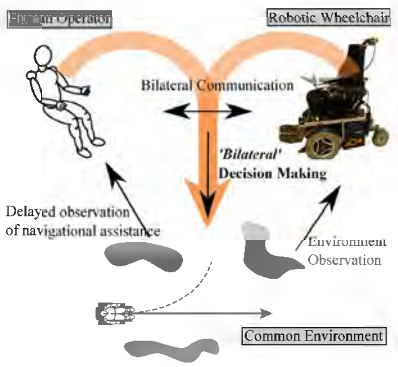
A novel haptic guidance scheme was developed to help steer the powered wheelchair users through narrow and complex environments [15]. The scheme encodes the local environment of the wheelchair as a set of collision-free circular paths, as explained in Section III. An adaptive impedance controller is then tuned based on this encoding so that it slows the wheelchair down in the direction of short circular paths (thus with imminent collisions) and bends the user towards longer circular paths (helping to avoid the collisions). Fig. 10 displays the resistance a user would feel at the joystick, for an environment and circular encoding displayed in 10a.

C. Experiments with backwards driving into an elevator

Preliminary experiments were conducted with this kind of haptic navigation assistance. With cardboard boxes an artificial environment was built up to represent an elevator. The user is asked to maneuver the wheelchair backwards inside this elevator beginning from a fixed starting position. The maneuvering capability with or without navigation assistance is measured during the execution of this task. Parameters that were recorded are time until completion and the number of collisions. At this stage of the research all experiments are conducted by one

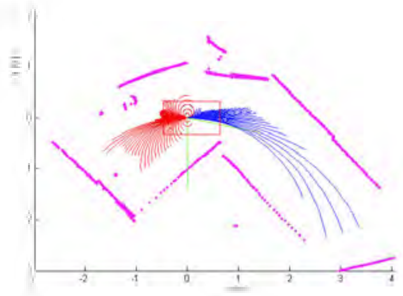


(a) Traditional shared control schemes display asymmetry in the decision making process. Users only understand decisions after actual displacement took place.

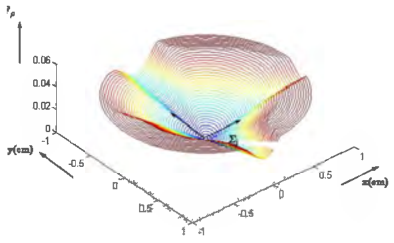


(b) Bilateral shared control schemes feature a deeper level of control sharing. The user can negotiate directly with the wheelchair over a haptic communication channel.

Fig. 9: Navigation assistance through unilateral or bilateral shared control.



(a) Collision-free paths.



(b) Associated force field.

Fig. 10: Collision-free paths and associated force field in haptic joystick.

TABLE I: Summary of results (time in s, average and standard deviation calculated for successful runs only).

run	1	2	3	4	5	6	7
type 1	13.23	13.15	10.11	14.31	11.94	9.45	9.69
type 2	11.00	12.09	11.47	14.01	9.6	10.34	17.51
type 3	25.4	10.43	21.86	15.23	19.48	13.37	26.65
run	8	9	10	**av.**	**stdev**	**coll.**	
type 1	10.46	12.07	10.14	11.14	1.47	1	
type 2	11.84	9.82	15.73	11.60	1.48	4	
type 3	13.02	40.83	17.94	18.23	5.89	4	

single able-bodied user (male, 33 years). Both experiments were conducted 10 times. Three types of experiments were conducted and executed in random order:

Type 1: navigation without guidance. The user was allowed to look backwards over the shoulder during these experiments. Note that this way of operation is not possible or very tiring for many typical wheelchair users.

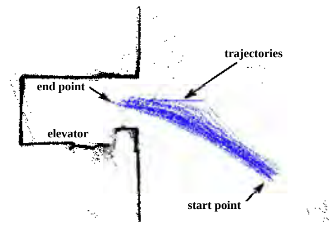
Type 2: navigation with visual guidance. The user was asked to maneuver the wheelchair while observing a GUI that displayed the environment encoded as a set of collision-free path lengths.

Type 3: navigation under haptic feedback. The user was asked to drive 'blindly' inside the elevator solely relying on his sense of touch and the haptic guidance.

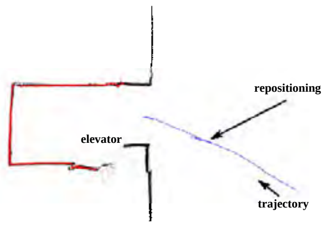
Table I and Fig. 11 summarize the results from the different experiments. At this moment, navigating the wheelchair while looking backwards over the shoulder is still superior, but, also here, collisions could not be avoided. Indeed, the task is quite challenging as the elevator is quite narrow, leaving only about 10 centimeters of space at both sides between wheelchair and door post. With only visual or haptic guidance, the amount of successful executions dropped to $6/10$. This score might seem quite low, but it must be stressed that the user did not have to look backwards over the shoulder. So such navigation strategy could come in handy to help especially those users that experience problems in looking over their shoulder while steering a wheelchair.

At this moment GUI-based navigation is still faster than navigating solely based upon haptic guidance (Table I). Under haptic guidance the user is somehow 'palpating' the environment to feel where the passage is, whereas the GUI immediately shows the user where the passage is. On the other hand the haptic guidance warns the user when a collision is near. Fig. 11b shows an exemplary trajectory where the user turns the wheelchair after such warning and successfully completes the task.

There is definitely still some room for improvement. But note that it is quite remarkable that already now in $6/10$ cases the user manages to drive into the elevator solely relying on haptic guidance. Without such guidance, without GUI or looking backwards over the shoulder such maneuver would be close to impossible.



(a) All trajectories of the experiments plotted. The clock is stopped when the wheelchair's frontal wheels are inside the elevator.



(b) Example trajectory under haptic guidance. Through haptic feedback the user turns wheelchair closeby elevator and makes a successful entry

Fig. 11: Navigation trajectories when driving backwards into an elevator.

V. Conclusion

This paper has described the main technological components of the RADHAR navigation assistance approach. Experimental results for each component separately have been shown. These components are now being integrated into one system, so that it can be tested by the user groups.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 - Challenge 2 - Cognitive Systems, Interaction, Robotics - under grant agreement No. 248873-RADHAR.

References

[1] E. Demeester, M. Nuttin, D. Vanhooydonck, and H. Van Brussel. A model-based, probabilistic framework for plan recognition in shared wheelchair control: Experiments and evaluation. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1456–1461, 2003.

[2] E. Demeester, A. Huntemann, D. Vanhooydonck, G. Vanacker, H. Van Brussel, and M. Nuttin. User-adapted plan recognition and user-adapted shared control: A bayesian approach to semi-autonomous wheelchair driving. *Journal of Autonomous Robots*, 24(2):193–211, February 2008.

[3] E. Demeester, E. Vander Poorten, A. Hüntemann, J. De Schutter, M. Hofmann, M. Rooker, G. Kronreif, B. Lau, M. Kuderer, W. Burgard, A. Gelin, K. Vanopdenbosch, P. Van der Beeten, M. Vereecken, S. Ilsbroukx, A. Fossati, G. Roig, X. Boix, M. Ristin, L. Van Gool, H. Fraeyman, L. Broucke, H. Goessaert, and J. Josten. Robotic ADaptation to Humans Adapting to Robots: Overview of the FP7 project RADHAR. In *1st International Conference on Systems and Computer Science*, Villeneuve d'Ascq, France, August 29-31 2012 2012.

[4] E. Demeester, E. Vander Poorten, J. Philips, and A. Hüntemann. Design and evaluation of a lookup-table based collision-checking approach for fixed sets of mobile robot paths. In *The 43rd International Symposium on Robotics (ISR2012)*, Taipei, Taiwan, 29-31 August 2012.

[5] G. Fanelli, T. Weise, J. Gall, and L. Van Gool. Real time head pose estimation from consumer depth cameras. In *33rd Annual Symposium of the German Association for Pattern Recognition (DAGM'11)*, 2011.

[6] A. Hüntemann. *Probabilistic Human-Robot Navigation – Plan recognition, user modelling and shared control for robotic wheelchairs*. PhD thesis, Katholieke Universiteit Leuven, March 2011.

[7] A. Huntemann, E. Demeester, G. Vanacker, D. Vanhooydonck, J. Philips, H. Van Brussel, and M. Nuttin. Bayesian plan recognition and shared control under uncertainty: Assisting wheelchair drivers by tracking fine motion paths. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3360–3366, 2007.

[8] A. Huntemann, E. Demeester, M. Nuttin, and H. Van Brussel. Online user modelling with gaussian processes for bayesian plan recognition during power-wheelchair steering. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 285–292, 2008.

[9] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, 2012.

[10] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *IEEE Intl. Conference on Robotics and Automation (ICRA)*, China, May 2011.

[11] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *Int. J. Comput. Vision*, 77(1-3):259–289, May 2008. ISSN 0920-5691.

[12] K.P. Murphy. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

[13] C.E. Rasmussen and C.K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press, second edition, 2006. ISBN 0-262-18253-X.

[14] R. C. Simpson. Smart wheelchairs: A literature review. *J Rehabil Res Dev*, 42(4):423–436, 2005. ISSN 0748-7711.

[15] E.B. Vander Poorten, E. Demeester, E. Reekmans, J. Philips, A. Huntemann, and J. De Schutter. Powered wheelchair navigation assistance through kinemally correct environmental haptic feedback. In *IEEE Conference on Robotics and Automation*, Minnesota, May 2012.