A New Hybrid Motion Planner: Applied in a Brain-Actuated Robotic Wheelchair

Article *in* IEEE Robotics & Automation Magazine · November 2016

DOI: 10.1109/MRA.2016.2605403

5 authors, including:

Ana Lopes
Polytechnic Institute of Tomar
17 PUBLICATIONS **103** CITATIONS
SEE PROFILE

Gabriel Pires
Polytechnic Institute of Tomar
35 PUBLICATIONS **447** CITATIONS
SEE PROFILE

Urbano Nunes
University of Coimbra
241 PUBLICATIONS **2,833** CITATIONS
SEE PROFILE

A New Hybrid Motion Planner applied in a Brain-actuated Robotic Wheelchair

Ana C. Lopes, João Rodrigues, Jorge Perdigão, Gabriel Pires, and Urbano J. Nunes

THIS paper presents a new Hybrid Motion (HM) planner, designed to allow robust indoor navigation in constrained environments of non-holonomic differential robots, such as RobChair, the ISR-UC brain-actuated robotic wheelchair [1]. Relying on this new planning algorithm, RobChair is now able to operate in real dynamic environments and to perform challenging maneuvers in narrow spaces. The HM planner integrates deliberative and reactive modules in a three-layer structure: fast 3D-global path planner; smoothing; and a new reactive local planner designated by Double-Dynamic Window Approach (D-DWA). The 3D-global path planner consists in the A* algorithm, which defines a path composed by a sequence of (x,y) points, with an interpolation module that has the purpose to assign orientation to each (x,y) point. The smoothing algorithm adjusts the (x,y) points to reduce accelerations and jerk associated to the trajectory. The D-DWA is in charge of dynamically adapt the robot motion, taking into account the robot geometry (non-circular robot) and local static/dynamic obstacles, unknown from the global planner perspective. Real time navigation is achieved since both the smoothing and the D-DWA algorithms are iteratively executed during navigation. The use of multiresolution local grid maps also contributes to fasten computation. To show the effectiveness of the proposed planning algorithm, results of real navigation experiments are reported herein. The experiments consist in steering RobChair in a real office-kind scenario by different participants, using a self-paced P300-based Brain-Computer Interface (BCI).

I. Introduction

RobChair is a brain-actuated robotic wheelchair that aims to help severe motor-impaired people to reach better levels of mobility, contributing to improve their life standards, and, ultimately, to increase their social inclusion. People suffering from severe motor disabilities are unable or may face great difficulty to control conventional interfaces. A self-paced P300-based BCI is herein applied to RobChair allowing users with severe motor disabilities to issue commands without using any muscular activity. However, BCI steering commands are discrete, sparse, and prone to errors. For this reason a brain-actuated robotic wheelchair must be self-reliant, holding a high level of navigation autonomy, since it cannot rely on the human to be always available or to provide accurate information. The proposed Hybrid Motion (HM) planner provides the required navigation autonomy to effectively use this Human Machine Interface (HMI) in non-holonomic differential robots, such as RobChair, with both circular and non-circular geometries, in indoor constrained environments. This planning approach, supported by an appropriate perception system, allows the robot to be self-reliant and able to cope with dynamic changes in the environment. In fact, given the sparse nature of the P300-based BCI commands, the robot must be able to safely execute user's navigation intents, with his/her minimum intervention, in constrained areas such as narrow door passages, as in the example shown in Fig. 1.

All authors are with the Institute for Systems and Robotics, University of Coimbra, Polo II, 3030-290 Portugal. Ana C. Lopes and Gabriel Pires are also with the Politechnic Institute of Tomar and Urbano Nunes is also with the Department of Elect. and Computer Engineering of Univ. of Coimbra. {anacris,gpires,urbano}@isr.uc.pt

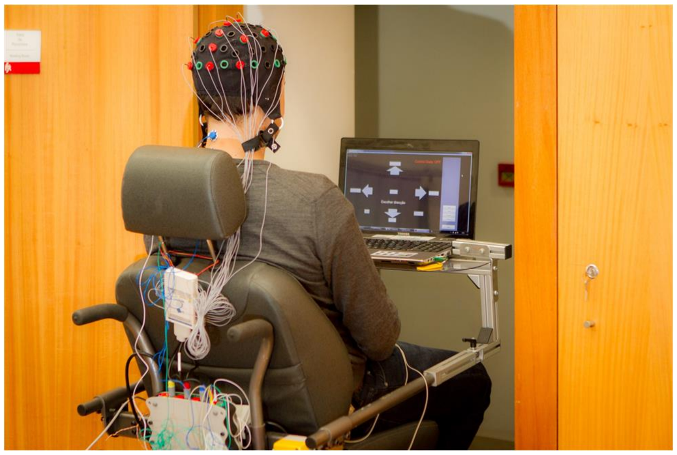
Fig. 1. Snapshot taken during an experiment with an able-bodied participant steering RobChair with the self-paced P300-based BCI.

Several brain-actuated robotic wheelchair prototypes were developed in recent years with the ultimate goal of improving mobility capabilities of severe disabled people. In [2] an asynchronous BCI based motor imagery (self-regulation of sensorimotor rhythms) is proposed to control a powered wheelchair, using a shared controller to couple user's intentions with data provided from a local planner. This strategy allows continuous control and does not require an *a priori* map of the environment, yet it is limited to two low-level commands (turn right and turn left) and moves forward when user is in an idle mental state. This kind of approach imposes a continuous mental-workload of the user and may lead to robot wandering behaviours, for example in open spaces such as corridors, since it does not allow selection of destination goals. On the contrary, P300-based brain-actuated wheelchairs [1], [3], [4] are more flexible as they can provide a large set of local and global commands which can be dynamically changed according to environment context. However, reported experimental tests have been primarily conducted in highly structured or open space environments. The proposed HM planner, which is one key module of CollabNAV - collaborative navigation architecture - depicted in Fig. 2, is capable of producing the appropriate set of linear and angular speed commands that allows RobChair to navigate autonomously, in real buildings with constrained areas, between global or local goals provided by the user using the self-paced P300-based BCI. Moreover, CollabNAV is also prepared to work with other types of discrete and sparse HMIs, such as voice control, or a simple switch or a set of switches that can be actuated by part(s) of the body better controlled by the user. In comparison to previous P300-based interfaces [1], [3], [4], the self-paced P300-BCI used here has a higher BCI speed, while maintaining a high reliability, which allows to reduce the user's workload and make the steering experience more natural. The wheelchair provides a symbiotic collaboration with the user with a "quasi-always in-movement capacity", which is achieved by reducing its speed when approaching decision areas to allow the user to provide a command on time. RobChair only stops if required by the user or if an ambiguous situation occurs, i.e. when the robot is not able to make a decision on its own, without an appropriate user command.

A. *Planning Approaches*

1) Global planning approaches: Planning approaches for mobile robot navigation can be roughly divided in three main categories: global, local, and hybrid planners. The global techniques assume that a complete model of the robot's environment is available. Two main classes of global motion planners are here outlined: sampling-based and search-based approaches. Sampling-based approaches satisfy weaker forms of completeness, i.e, they use a randomization of the configuration space. These techniques include, for instance, the probabilistic road maps [5], randomized path planners [6], and rapidly-exploring random trees [7]. Search-based approaches generate a graph representation of the planning problem. There are several algorithms to find the shorthest path in a graph. These methods include the A* [8] and D* [9] path planners. In recent years, 3D path planners (3D refers to (x,y,θ)) for indoor constrained environments based on graph search algorithms applied to state lattices have been proposed for non-holonomic robots [10], [11].

2) Local approaches: Local approaches, which rely on local environment information, must be able to quickly adapt the local plan in reaction to unforeseen and dynamic changes in the environment. Well known local planning methods include the potential field method [12], the Vector Field Histogram (VFH) [13], and the Dynamic Window Approach (DWA) proposed in [14]. These methods are extremely fast, and they typically consider only the small subset of obstacles close to the robot. Most local planning approaches generate motion commands for the robot in two separate stages: in the first stage, a desired motion direction is determined; in the second stage, the steering commands yielding a motion into the desired direction are generated. Unlike the potential field method and the VFH, the DWA method incorporates the dynamics of the robot and is particularly suited for robots navigating at high speeds.

3) Hybrid Approaches: Hybrid planning approaches combine global deliberative planning with local reactive planning in an integrated architecture. In [15] an hybrid method is proposed, where the deliberative part produces collision-free with shortest-distance path, while the reactive layer generates safe and time-minimal navigation paths. An hybrid planning method based on the integration of D* search algorithm with the dynamic window local obstacle avoidance algorithm is proposed in [16]. In [17], [18] a method designated as global DWA is proposed for holonomic mobile robots. It generates a navigation function over the C-space and combines it with the DWA, thereby avoiding the DWA's problem of local minima. In [19] an adaptation of the global DWA approach for non-holonomic robots, including the specific case of robotic wheelchairs, is proposed.

II. HM PLANNER

The navigation system is supported on an hybrid planning architecture designated as HM planner mainly composed by a global and a local planner, as depicted in Fig. 2. This planning strategy was established to allow robust indoor navigation of non-holonomic and non-circular mobile robots in constrained environments, such as RobChair. Figure 3 shows the HM planner architecture in detail. The global planner consists of a modified version of the well-known search-based A* algorithm, which was expanded to a 3D-path planner, $s_i = [x_i, y_i, \theta_i]$, using the interpolation module. Both the A* and the interpolation module are executed once a new navigation goal is provided to the navigation system, and taking into account the last updated version of the 2D local grid costmap. The global planner also includes a smoothing algorithm, based on elastic bands [20], to prevent sudden variations in the RobChair orientation. The local planner is required to solve unexpected situations in conflict with the global plan (e.g. dynamic obstacles). It consists of a Double-Dynamic Window Approach (D-DWA) to increase robustness during the execution of challenging maneuvers, taking into account the restrictions imposed by the robot geometry. The D-DWA algorithm is obtained by applying the DWA algorithm [14] at two different control points: (1) the robot center of mass; and (2) a point located at the front of the robot. Both the smoothing algorithm and the D-DWA run iterativelly in real time during global plan execution. The HM planner strategy will be explained in detail hereinafter.

A. *Costmaps*

The proposed planner uses three different costmaps, namely: a 2D global costmap, a low resolution local costmap, and a high resolution local costmap (see Fig. 3). The 2D global costmap is a low resolution costmap, with a resolution designated by mg_{res}, obtained from the *a priori* metric map, and

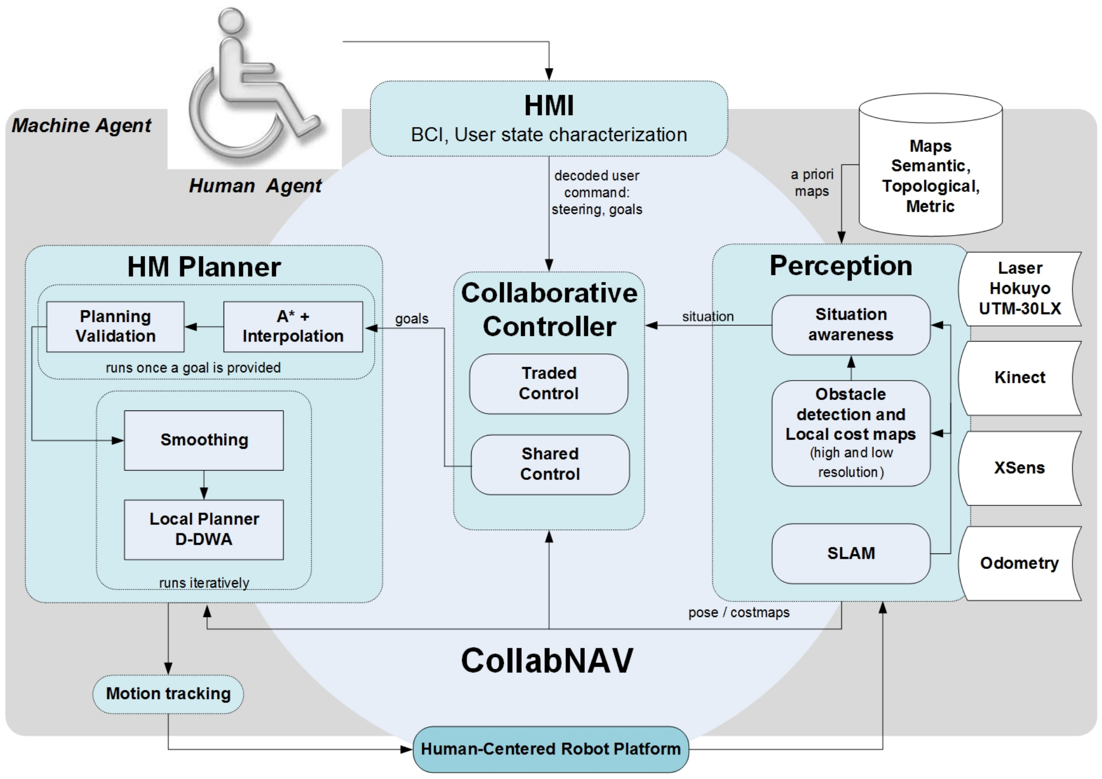

Fig. 2. Collaborative navigation architecture main modules and respective information flow: HMI, planning (HM planner), perception, and collaborative controller. The HM planner is a key module of the CollabNav architecture, providing the appropriate set of angular and linear speed commands to the motion tracking module. It requires pose data and costmaps provided by the perception module, and goals (local or global) determined by the collaborative controller taking into account user's intention and context variables.

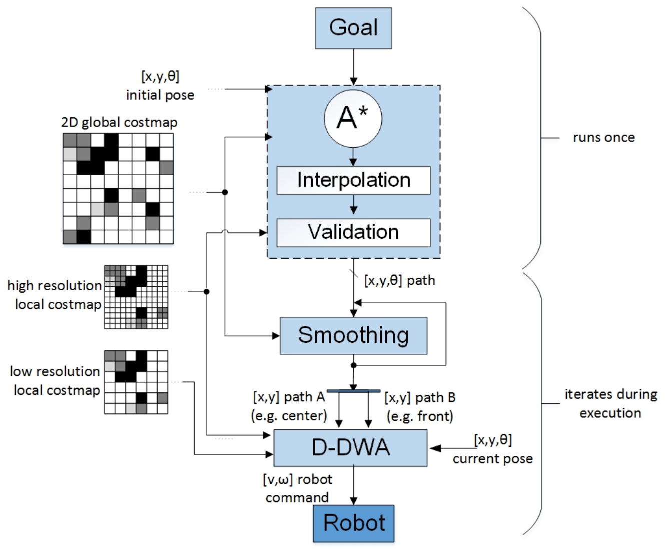

Fig. 3. HM planner: the hybrid planning architecture for non-holonomic differential robots. The HM planner includes a global planner based on an A* algorithm with interpolation to determine the 3D path (x, y, θ) to a final goal; the smoothing algorithm based on the elastic bands methodology and the local planner D-DWA are then applied iteratively to the global plan during its execution. The D-DWA is applied to two 2D global paths based on the 3D path originated previously, being one in relation to the robot center of mass, and the other one in relation to a middle point located at the front of the robot.

therefore includes all the static obstacles provided in that map. It also includes the obstacles detected by the laserscanner in a radius designated by o_{radius}. Obstacles are inflated in a distance of i_{radius} to mark all areas where the robot's position is not valid, regardless the robot's orientation. A cost is also assigned to areas where only a few robot orientations are allowed. Thus the robot will tend to move away from obstacles making easier the execution of maneuvers. The low resolution local costmap has a grid cell resolution of ml_{res}, and a dimension of ml_{dim}. This local costmap includes all obstacles detected by the laserscanner, and is used for most of D-DWA local plan validations. The high resolution local costmap has a grid cell resolution of mh_{res}, and a dimension of mh_{dim}. This local costmap also includes all obstacles detected by the laserscanner. It is used in a specific D-DWA local plan validation task, in particular to verify if any point of the local plan trajectory orginates a collision of the robot footprint. In the same manner, it is also used to validate the points generated by the 3D-global path planner.

B. *Fast 3D-global path planner*

The first global path to a specified navigation goal is obtained by applying the grid search algorithm A* to the inflated low resolution 2D global costmap. The calculation of the global path using the A* is very fast (on the order of tens of milliseconds), however the resulting path is very likely to be impossible to perform due to the geometric

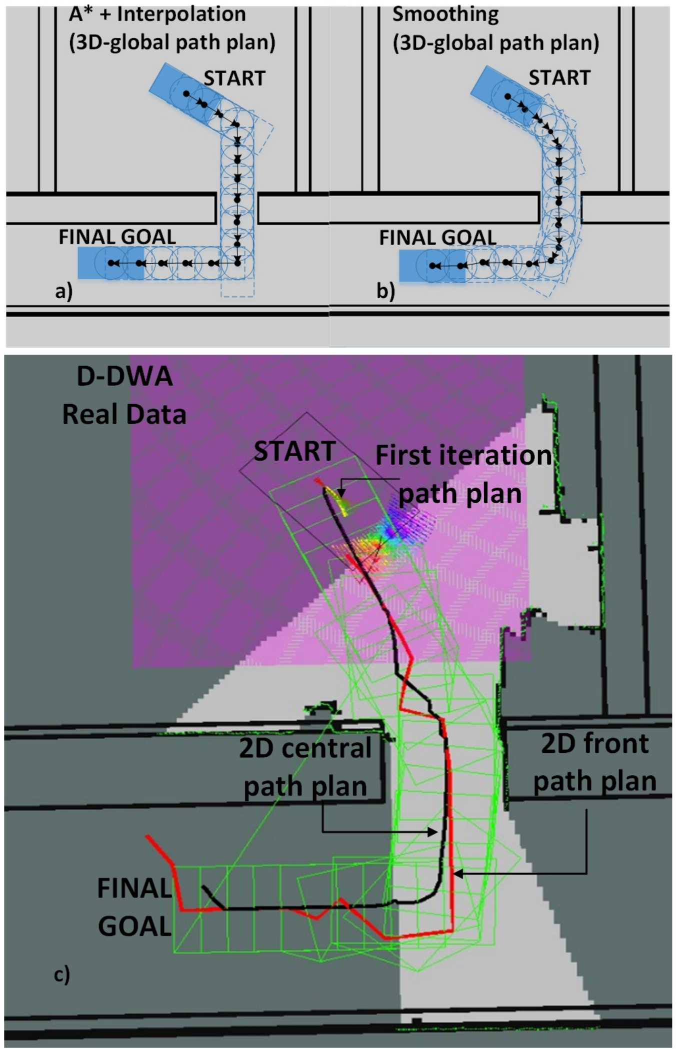

Fig. 4. Navigation planning steps: a) fast 3D-global path planner based on A* plus interpolation; b) exemplification of 3D-global path planner after apllication of the smoothing algorithm; c) on bottom figure a Rviz screenshot showing a real path plan being executed - 2D central (black) and front path plans (red), and first iteration path plan (yellow) after application of smoothing and D-DWA.

characteristics of RobChair (non-circular robot). The inflated costmap only guarantees the collision avoidance of a circle with a specified radius around the center of the robot. Thus using the A* algorithm on an inflated 2D costmap allows us to obtain a quick approximation of the feasible path, once the result is only a geometric path that does not take orientation into consideration. The interpolation module is in charge of transforming the 2D geometric path in a 3D-path defined by 3D waypoints $s_j = [x_j, y_j, \theta_j]$, by simply determining the orientation θ_j of each waypoint (x_j, y_j) as $\theta_j = arctan(\frac{y_j - y_{j-1}}{x_j - x_{j-1}})$ (see Fig. 4). Clearly, the latest strategy will most probably result in a non-feasible 3D-path with unsolved collision avoidance problems due to the robot geometry. Thus, a planning validation process (see Fig. 2) is applied, and whenever a point of the robot footprint collides with a high resolution obstacle area (in the high resolution 2D costmap), the respective robot pose is corrected to a valid one. A similar process is applied to narrow passages, such as doorways.

C. Smoothing algorithm

The fast 3D-global path obtained previously is most likely to present sharp transitions in the orientation, especially due to the corrections required to obtain a valid path plan. A smoothing algorithm, based on elastic bands methodology, is then applied to avoid sudden variations in the orientation profile of the valid 3D path. To obtain a smooth global path, all the N points $\mathbf{s}_j = [x_j, y_j]$ in the global path are considered as objects with mass m. Subsequent points are connected by an 1-DOF spring with equilibrium length $d_j^0 = |\mathbf{s}_{j+1} - \mathbf{s}_j|^0$ and elastic constant K_e. The smoothing algorithm is detailed in Algorithm 1, where $d_j = |\mathbf{s}_{j+1} - \mathbf{s}_j|$ and $\hat{\mathbf{n}}_j \perp \mathbf{s}_{j+1} - \mathbf{s}_j$.

Algorithm 1 $Smoothing(s_j = [x_j, y_j, \theta_j])$

```
1:for each iteration i do
2:    for j = 2 : N − 1 do
3:        $\mathbf{Fl}_j = K_e \left( -\left(d_{j-1} - d_{j-1}^0\right) \frac{\mathbf{s}_j - \mathbf{s}_{j-1}}{d_{j-1}} + \left(d_j - d_j^0\right) \frac{\mathbf{s}_{j+1} - \mathbf{s}_j}{d_j} \right)$
4:        $\tau_j = K_T \left( -\left(\theta_j - \theta_{j-1}\right) + \left(\theta_{j+1} - \theta_j\right) \right)$
5:        $\mathbf{F}_j = \mathbf{Fl}_j + \left( \frac{\tau_{j-1}}{\frac{1}{2} d_{j-1}} \hat{\mathbf{n}}_{j-1} - \frac{\tau j}{\frac{1}{2} d_j} \hat{\mathbf{n}}_j \right)$
6:        $\mathbf{a}_j = \frac{\mathbf{F}j}{m}$
7:    end for
8:    $\Delta s \leftarrow LeapFrog(\mathbf{a}_j)$
9:    $[x, y, \theta] \leftarrow Interpolation([x, y])$
10:   $[x, y, \theta] \leftarrow Validation([x, y, \theta])$
11:end for
```

For each iteration i a force $\mathbf{F}_j$ (lines 3 to 5) that is influenced by both spring $j-1$ and spring j (see Fig. 5) is determined for each waypoint $\mathbf{s}_j$, with the exception of the initial and final points that are fixed. $\mathbf{Fl}_j$ (line 3) is the force originated by a shift position of a waypoint j. To avoid sudden variations in the orientation, it is considered that each 1-DOF spring j connecting two points is subject to a rotation torque τ_j applied at its center, where θ_j is the orientation at point $\mathbf{s}_j$ (line 4). As depicted in Fig. 5, the effect of τ_j may be translated in two forces applied to points $\mathbf{s}_j$ and $\mathbf{s}_{j+1}$ in the perpendicular direction to vector that connects $\mathbf{s}_{j+1} - \mathbf{s}_j$. $\mathbf{F}_j$ (line 5) is in charge of generating an acceleration in each point of the path plan given by $\mathbf{a}_j$ (line 6). In each iteration the displacement Δs of each point is then determined applying a Leapfrog numerical integration, followed by intepolation and validation steps similar to those applied in the fast 3D-global path planner.

.

D. Double-Dynamic Window Approach

The new DWA-based algorithm was developed to execute the motion along the global path subject to an orientation profile. In our approach, designated as Double-Dynamic Window Approach (D-DWA), we determine two 2D global path plans based on the 3D path plan originated previously. The first one

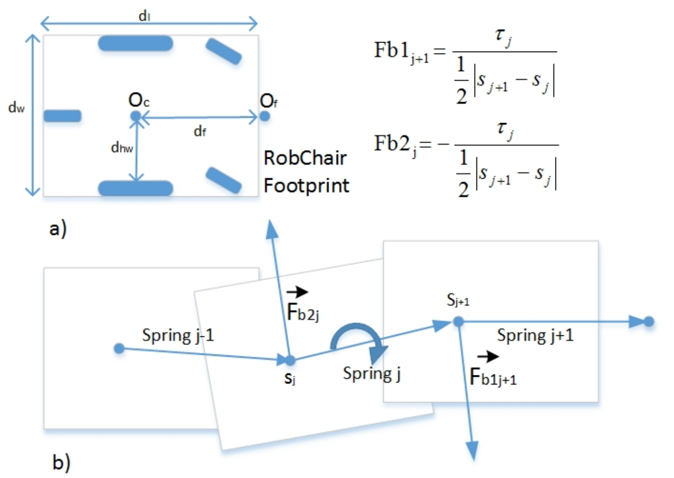
dl
dw
Oc
Of
df
dhw
RobChair Footprint
a)
$Fb1_{j+1} = \frac{\tau_j}{\frac{1}{2}|s_{j+1} - s_j|}$
$Fb2_j = -\frac{\tau_j}{\frac{1}{2}|s_{j+1} - s_j|}$
Fb2j
Sj+1
Spring j-1
Spring j
Spring j+1
sj
Fb1j+1
b)

Fig. 5. a) RobChair footprint; b) Effect of the rotation torque τ_j translated in two forces $\mathbf{F}b1_{j+1}$ and $\mathbf{F}b2_j$ applied to points $\mathbf{s}_{j+1}$ and $\mathbf{s}_j$, respectively.

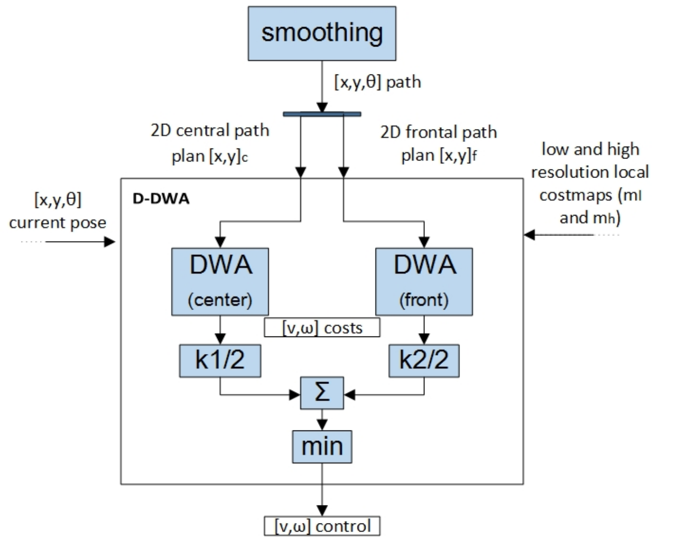

Fig. 6. Double-Dynamic Window Approach: two 2D global path plans are determined based on the 3D path plan originated previously. The first one $s_c = [x,y]_c$ is calculated in relation to the robot center of mass, and the second one, $s_f = [x,y]_f$, is calculated in relation to a middle point located at the front of the robot, O_f.The velocity dynamic window is determined for both the center,$[v,w]_c$, and the front path plans, $[v,w]_f$. The costs of center and front trajectories are weighed to determine the set of $[v,w]$ with lower weighted cost.

$s_c = [x,y]_c$ is calculated in relation to the robot center of mass, O_c, and the second one, $s_f = [x,y]_f$, is calculated in relation to a middle point located at the front of the robot, O_f, as depicted in Fig. 6. Since the linear and angular speed commands $[v,w]_c$ and $[v,w]_f$ are determined in respect to two different robot control points, O_c and O_f respectively, the expected position and orientation are obtained. Although this solution causes the robot to follow two different path plans, as if they were two different rails, the robot continues to have the ability to avoid unexpected obstacles that were not included in the global path plans. The D-DWA algorithm is detailed in Algorithm 2, where m_l and m_h stand for low and high resolution local costmaps respectively.

Algorithm 2 $D-DWA(O_c,O_f,s_c,s_f,m_h,m_l)$

```
1:for each iteration $i$ do
2:    $[v,w]_c \leftarrow DWA(O_c, m_l, s_{ci})$;
3:    $[v,w]_f \leftarrow DWA(O_f, m_l, s_{fi})$;
4:    for all $j$ do
5:        $t_{cj} \leftarrow trajectory([v_j,w_j])_c$
6:        $t_{fj} \leftarrow trajectory([v_j,w_j])_f$
7:        if($Collide(t_c,m_h)$) then
8:            reject $t_c$;
9:        else
10:           for all $k$ points in the trajectory $j$
11:               $Cost+=Cost(t_{cj},m_l,k)$;
12:           end for
13:       if($Collide(t_f,m_h)$) then
14:           reject $t_f$;
15:       else
16:           for all $k$ points in the trajectory $j$
17:               $Cost_f+=Cost_f(t_f,m_l)$;
18:           end for
19:   end for
20:   $[v,w] \leftarrow min(\frac{K_1}{2}Cost_c + \frac{K_2}{2}Cost_f)$
21:end for
```

According to Algorithm 2, in each iteration the next point of the center and front global path plans to be followed are provided to the D-DWA module. Then the velocity dynamic window is determined for both the center, $[v,w]_c$, and the front path plans, $[v,w]_f$ (lines 2 and 3). Two sets of trajectories, t_c and t_f, are determined for the set of velocities inside the respective dynamic window (lines 5 and 6). If any k point in the trajectory collides with an obstacle in the high resolution costmap then the trajectory is rejected (lines 7 and 13). A penalty is also applied to k points close to obstacles. The cost of each trajectory is determined by adding the cost of each k point in the low resolution costmap (lines 11 and 17). It is important to highlight that the cost of each k point decreases from the origin of the trajectory to the goal. Weighing the costs of center and front trajectories to determine the set of $[v,w]$ (line 20) leads the robot to perform the desired trajectories (center and front) with lower weighted cost.

III. CollabNAV: Collaborative Navigation Framework

The collaborative navigation framework (see Fig. 2) is composed by four main modules: HMI, Perception, HM-planner, and Collaborative Controller. It has been widely tested in ROS simulation environment, and in real experiments using RobChair [1], [21]. The commands sparsely issued by the user can be either global (e.g. WC, EXIT, etc.) or local (e.g. FORWARD, BACK, etc.). Both global and local commands of the P300-based paradigm could be dynamically set according to context awareness. However, a fixed set of 7 commands was used in the experiments reported here.

A. Self-paced P300 BCI

Users provide steering commands by using a P300-based BCI. The P300 signal is an event related potential (ERP)

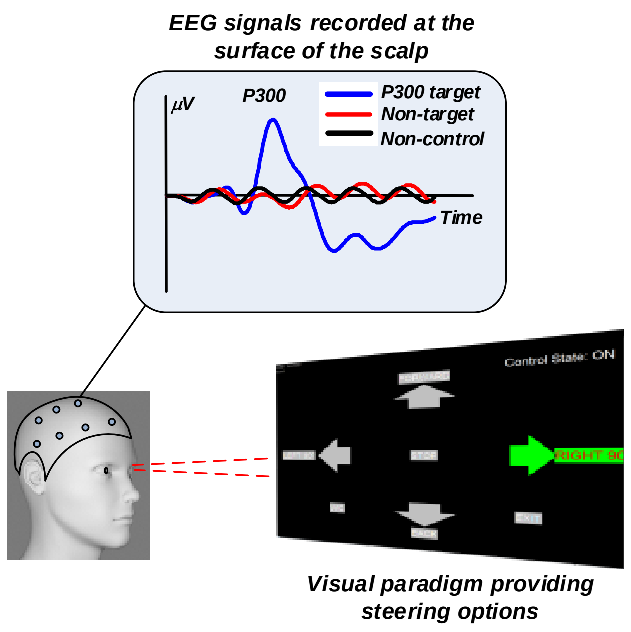

Fig. 7. Visual P300-based BCI providing 7 steering options. User is focused on the desired command, or inattentive if he/she does not want to convey any command (non-control state). On top are the EEG signals typically evoked by target and non-target events and non-control state.

elicited by a rare and relevant event (target) occurring randomly among non-target events, in a so-called oddball paradigm [22]. P300-based BCIs are especially suited to select items among possible choices. Our oddball paradigm consists of a set of 7 symbols flashing randomly, each one representing a possible choice for steering the wheelchair, namely FORWARD, BACK, LEFT90, RIGHT90, STOP, WC and EXIT (see Fig. 7). Each symbol is flashed during 100 ms and the interval between flashes is 75 ms. The relevant event is the symbol mentally selected by the user. The P300 ERP is characterized by a peak occurring around 300 ms after the onset of the target flash and it has typically a waveform similar to the one shown in Fig. 7. Responses to non-target events do not exhibit a P300 component and reflect mainly ongoing electroencephalography (EEG). The BCI system detects also a non-control state, in which the user does not pay attention to any particular event, because he/she does not want to convey any command (self-paced operation). The EEG signal during non-control state is similar to that of non-target events as depicted in Fig. 7. The poor signal-to-noise ratio (SNR) of EEG signals hampers the decoding of the relevant brain patterns. In particular, the P300 ERPs exhibit a SNR substantially lower than 0 dB [22]. Therefore, usually, the P300 response to a single relevant event is not enough to be detected, and it is necessary to combine the responses of several repetitions of the same relevant event. Thus, the detected symbol returned by the BCI occurs after one or more rounds of flashes, depending on the user's performance.

The online classification system of the P300-based BCI is schematically illustrated in Fig. 8. Before the online operation, the system is calibrated to obtain mathematical models of feature extractors and classifiers fitted to the user. Methods are described in detail in [22]. The P300-based system comprises two types of classification. A binary (two-class) classification and a C-class classification for symbol recognition (the number of classes is the same as the number of symbols). These two classifiers are applied sequentially:

- Binary classifier - discriminates two classes: target vs. 'non-target + non-control'. The classifier is applied to each event and returns a score;
- C-class classifier: the scores returned by the 7 binary classifications are combined to find the most likely target event. If the score of this event is lower than a pre-defined threshold it is assumed that the user is in a non-control state, not willing to transmit any command. Otherwise, the symbol associated with the most likely target is transmitted to the navigation system;

The BCI speed (number of symbols per minute) depends on the minimum number of repetitions that the system requires to reliably detect the user's mentally selected symbol. Three types of errors can occur: control state is detected instead of non-control state and vice versa, and control state is correctly detected but a wrong command is selected. Algorithms are tuned to maintain a low false positive rate, i.e. minimize the rate of false control states.

B. Perception

The perception system includes three main modules: situation awareness, obstacle detection and local cost mapping, and Simultaneous Localization and Mapping (SLAM). The latter is performed with Hector SLAM [23], which integrates data from a 2D laserscanner, Hokuyo UTM-30lx, placed at the front of RobChair. The CollabNav system also uses the information provided by three types of global maps constructed offline, in particular, a static metric map, a topological map and a semantic map. The semantic map includes the three main types of places being navigated (corridors, rooms, and doors), as well as a fixed area wrapping the nodes previously defined in the topological map. The static metric map is a grid map provided *a priori* to the SLAM module. A 2D global costmap is obtained from the a priori grid map and includes the laserscanner information in a 3 meter radius. The low and high resolution local costmaps are local grid maps, with dimensions of 3×3 m, that consist of samples of the global costmap constrained by the current robot pose. The situation awareness module is in charge of detecting all the ambiguous situations requiring user collaboration, changes in semantic (type of place being navigated), and environmental dynamics.

C. Collaborative controller

The collaborative controller consists of a decision-making layer that depends on the user's command, the type of place the robot is navigating, and the ambiguous situation it is dealing with. An ambiguous situation can be defined as the perception of a navigation event that has multiple solutions (e.g. intersection) or none (e.g. deadlock). The collaborative controller receives commands from two agents: the user, and the machine agent. The user sparsely issues commands with

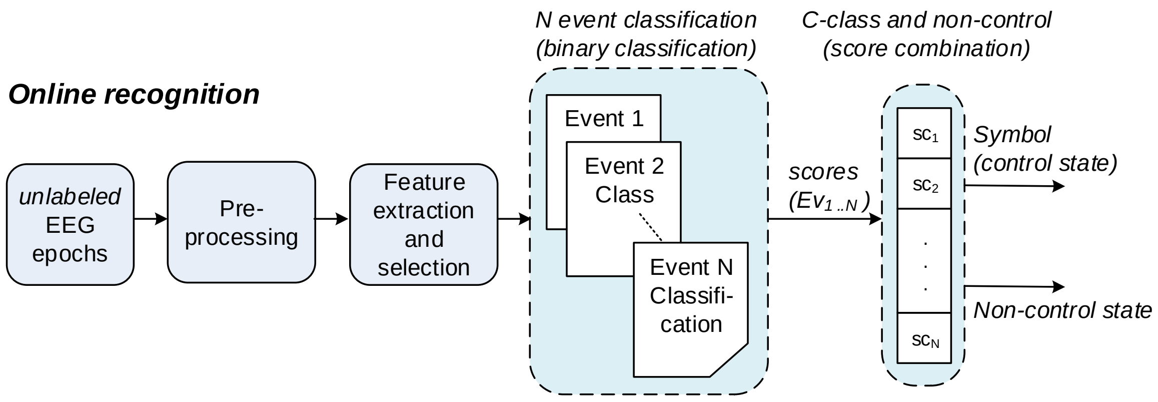

Fig. 8. Online classification architecture of a self-paced P300-based BCI system. EEG signal is segmented into epochs associated to each event, and preprocessed (filtering and normalization). Feature extraction and selection is applied and then a binary classifier is applied to each event. Then, the output scores are combined, identifying the symbol mentally selected by the user or a non-control state.

the self-paced P300-based BCI. Two different types of commands are possible, namely high-level commands consisting in a set of target goals belonging to the navigational space (e.g. WC, EXIT, etc), and low-level commands, consisting of a set of local steering commands, specifically FORWARD, LEFT90, RIGHT90 and BACK. The collaborative control architecture combines a traded control layer with a shared control layer (see Fig. 2). The former is responsible for enabling/disabling user commands, depending on situation awareness, and the latter is a shared controller that determines the appropriate navigation goal according to the validated user command, situation and place being navigated. The traded control layer is responsible for validating the commands provided by the user, subject to a set of perceived situations, and depending on the place being navigated. User commands are enabled by this layer according to the following set of perceived situations:

- $S1$: Multiple goals;
- $S2$: Multiple possible directions to avoid an obstacle;
- $S3$: Deadlock;
- $S4$: Occlusion caused by environmental dynamics;
- $S5$: Providing a global goal.

The perception module includes a situation awareness module that informs the traded controller on the occurred event $S1$ to $S4$. If a global goal is provided by the user, RobChair stops and expects a confirmation from the user. The traded controller module receives the user command, expressed either as a steering command or as a high-level global goal, the place being navigated (semantic) and the situation $S1$ to $S4$ (or none).

The shared-controller determines the final goal (either global or local), based on user commands properly filtered by the traded control layer, and taking into account a set of candidate goals proposed by the machine agent. If the user provides a global goal, the collaborative controller output is straightforward (situation $S5$), it simply asks for a confirmation and informs the global planner of the new goal. On the other hand, if a steering command (low-level command) is provided to the robot, the shared controller needs to determine the most appropriate local goal, taking into consideration the topological map provided *a priori* to the robot. The calculation of the local goal depends on the perceived situation and on the type of place the robot is navigating. In case the robot reached a node area (nodes of the topological map), all commands are admissible, and the calculated goal corresponds to the closest node in the direction provided by the user command. If the robot is passing a door, the local goal corresponds to the closest node that might be in front of the robot or in the backwards. For a perceived deadlock situation $S3$, the machine agent is not able to determine any free direction, and, in that case, the user is expected to provide a global goal to allow re-planning. If RobChair stops due to a temporary occlusion caused by environmental dynamics, situation $S4$, the user is required to provide a steering command.

IV. Experimental results: navigation using self-paced BCI

In this section we present an assessment of the current stage of RobChair navigation system, with particular emphasis on navigation in challenging places, to show the robustness of the proposed planning approach (see video with experiments in [21]). The navigation task consisted in steering RobChair in an office type environment using the self-paced BCI. As can be seen in Fig. 9, the user is prompted to drive RobChair from an office, denoted by START, to a LAB, denoted by END. To perform this navigation task, RobChair is required to go through three narrow doorways and to avoid several obstacles along the way. The navigation tasks were carried out by nine able-bodied users, without relevant experience using the P300-based BCI and so inexperienced in driving Robchair using BCI. All participants gave informed consent to participate in the study. Table I presents the parameters used in all experiments.

Experimental tests started with the BCI calibration. Participants were asked to select a set of predefined commands, in order to gather labeled data to train the BCI classifier. Subsequently, participants were asked to perform the real-time navigation task. They had to issue the following local steering commands: FORWARD, LEFT90, RIGHT90, and BACK to

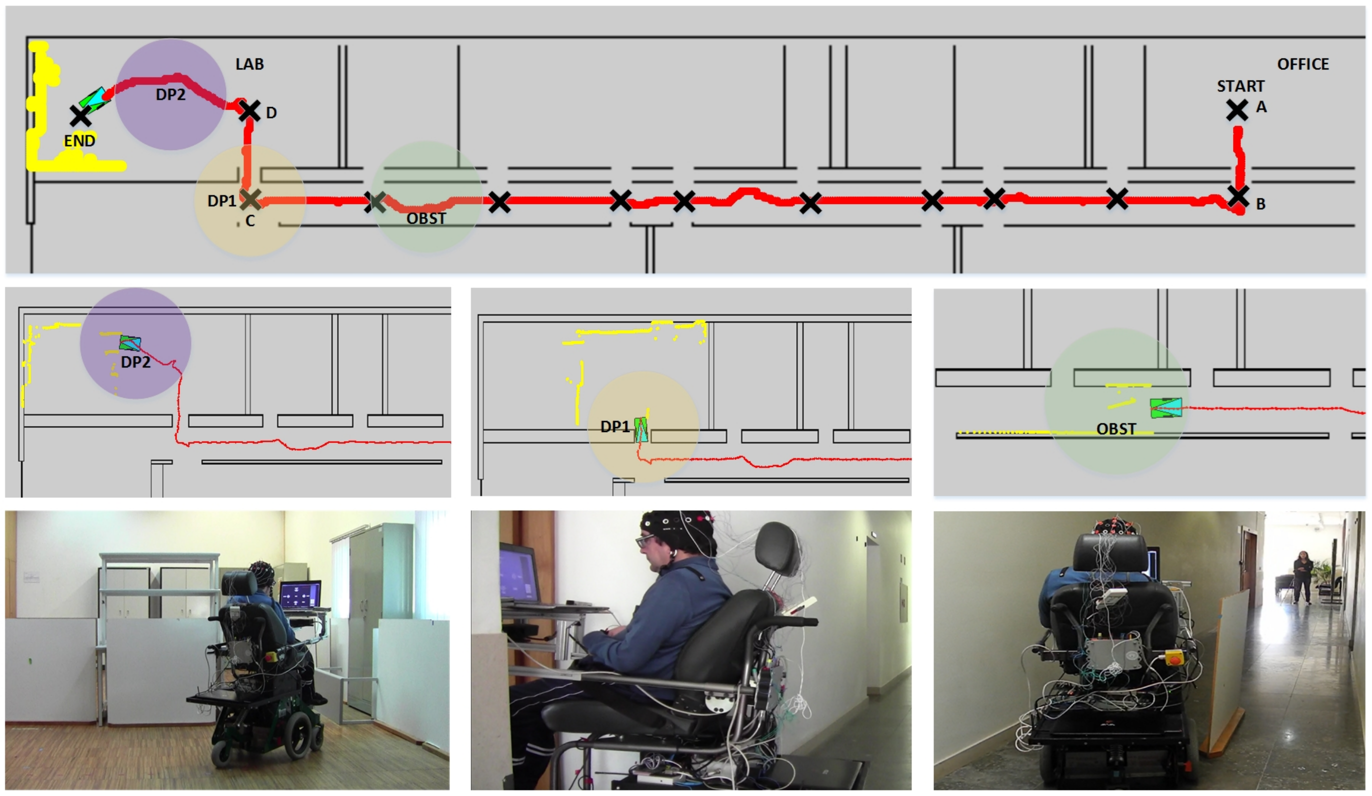

Fig. 9. Experimental results obtained for one user performing a navigation task in a real scenario, using self-paced P300-based BCI. The navigation task consists in going from the OFFICE to the LAB, only using local commands (FORWARD, LEFT90, RIGHT90, and BACK). Thirteen decision points are considered (represented by a cross) but the user only provides commands to start the navigation task (FORWARD) and in decision points B,C, and D. RobChair is able to successfully avoid new obstacles in the environment, and to go through narrow passages, such as doorways. The path performed by RobChair is represented in red, and laser scan data is represented in yellow.

Parameter	Value
$mg_{res}(cm \times cm)$	5×5
$ml_{res}(cm \times cm)$	3×3
$mh_{res}(cm \times cm)$	1×1
$o_{radius}(m)$	3
$ml_{dim}(m \times m)$	3×3
$mh_{dim}(m \times m)$	3×3
$i_{radius}(m)$	0.35
$d_l(m)$	1.1
$d_w(m)$	0.6
$d_{hw}(m)$	0.3
$d_f(m)$	0.6

TABLE I
PARAMETERS USED IN THE EXPERIMENTS.

reach the final destination. Since a self-paced P300-based BCI is being used, the user only needs to provide a command if he/she wants to change direction, otherwise RobChair will move according to the ongoing command. Therefore, users only have to provide local steering commands in decision points represented from A to D, as depicted in Fig. 10. They only had to issue the following local steering commands: FORWARD, LEFT90, RIGHT90, and BACK to perform the navigation task, showing the advantages of the self-paced P300-based BCI regarding user workload. If a non-self-paced BCI was used instead, as proposed in [1], thirteen local commands would be required to perform the same navigation task. At this moment, Robchair can be stopped under BCI actuation by selecting the “STOP” command. The time for command selection ranges typically between 6 to 8 seconds, depending on user’s performance. Given the low speed of the wheelchair, this time is fast enough for most situations in which users want to stop, unless there are sudden and unexpected changes in the environment. Situations like these should be perceived and solved by the navigation system.

The path to be followed in the navigation task is depicted in Figs. 9 and 10. It has a distance of about 59 m, and includes three narrow passages, and demanding rotation maneuvers to avoid obstacles, and to go through narrow spaces. The laserscan data obtained during the navigation task is also depicted in Fig. 10. Additionally, a screenshot associated to some challenging maneuvers are also presented in Fig. 9 to give a broader picture of the navigation task in analysis. Figure 11 shows the minimum clearance obtained along the navigation task executed by one of the 9 able-bodied users. The minimum clearance values were determined in relation to the laserscanner frame located at the front of the robot. The laserscanner field of view is 180º. For this user, the smallest value of minimum clearance of approximately 27 cm occurred around 476 s (identified by E in Fig. 11), which corresponds to the execution of a demanding maneuver needed to prepare the doorway passage coming from the corridor. In fact, local minima of less than 50 cm occurred in a small number of times along the navigation route. These situations

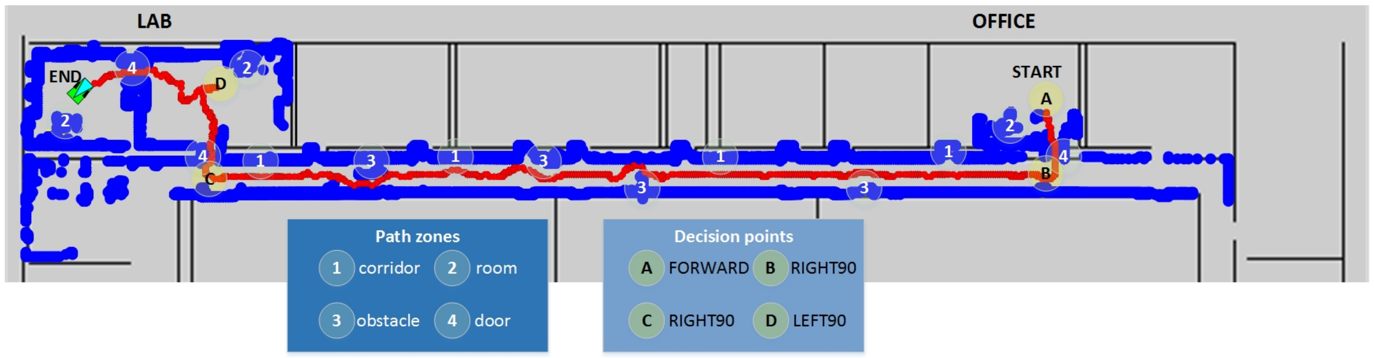

Fig. 10. Experimental results obtained for one user, according to path zone, for a navigation task in a real scenario using self-paced BCI. Path zone 1 corresponds to corridor, path zone 2 corresponds to room, path zone 3 denotes an obstacle avoidance, and path zone 4 denotes a door passage. The path performed by RobChair is represented in red, and laser scan data is represented in blue.

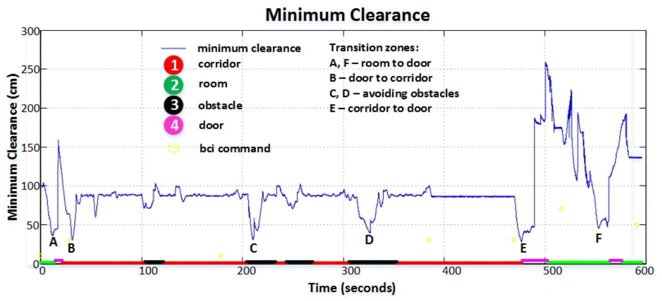

Fig. 11. Minimum-clearance values, per path zone, obtained for an experimental trial performed by one user. The local minima (A, B, C, D, E, F) were attained during the execution of demanding automatic maneuvers, in transition zones or in avoiding obstacles: A and F correspond the preparation for passage through a doorway coming from a room; B corresponds to a doorway passage to the corridor; C and D correspond to obstacle avoidance situations, and F corresponds to the preparation for a doorway passage from the corridor.

(A to F in Fig.11), are all related to transition zones that require the execution of several rotation maneuvers in narrow spaces. The smallest value of minimum clearance registered for all experiments was approximately 14 cm, and it also occurred during preparation maneuvers for a doorway passage from the corridor. Tables II and III summarize the results for the 9 users' experiments. In particular, Table II shows the results for linear and angular velocities (v and w respectively) and linear and angular accelerations ($\dot{v}$ and $\dot{w}$ respectively). On average RobChair took 530 s to travel the 59 m path, representing an average speed of about 0.11 m/s. This speed value is necessarily low once the navigation task requires the execution of several rotational and backward maneuvers in very narrow spaces. Additionally, it also included 13 decision points that required a reduction of the speed of the robotic wheelchair to give the user time to issue a BCI command. No collisions were registered during the experiments. Table II shows large maximum angular speed values, in comparison to mean angular speed values, corresponding to rotational maneuvers required in certain areas (e.g transition zones identified in Fig. 11). The mean angular speed was quite low for most of the time. Table III shows the BCI results, including the specificity (Spec) and sensitivity (Sen) of control vs. non-control detection, the command accuracy in control states (Acc), and the time required for selecting a command (ST). The very high values of specificity and accuracy show the reliability of the BCI. On average, users require 7.1 s to select a steering option, which is enough to provide commands timely, given the speed of the wheelchair and considering that user can anticipate the command before he/she reaches the decision point.

V. Conclusion and future work

A new planning algorithm was proposed to allow the execution of demanding maneuvers by non-holonomic and non-circular differential robots in real indoor constrained environment, in real time. The proposed HM planner consists of a hybrid planning architecture mainly composed by a global and a local planner. The fast 3D-global path planner consists of a modified version of the A* algorithm followed by an interpolation step providing a path composed by 3D waypoints (x_i, y_i, θ_i). The fast 3D-global path planner runs only once a goal is provided to the planning system. The global path planner also includes a smoothing algorithm, based on elastic bands [20] to reduce rotational jerk. The local planner is the D-DWA algorithm, which is obtained by applying the DWA algorithm at two different control points. Both the smoothing algorithm and the D-DWA are computed iteratively during global plan execution. Results show that the new planning algorithm allows the effective performance of very challenging maneuvers in constrained environments. A collaborative navigation framework was proposed to help reducing the impact of users' different skills on system performance. The collaborative controller allows the use of human perception and cognition skills without requiring continuous or time-critical response. If the human cannot respond because he/she is unavailable or busy performing other tasks, the system will still function, without posing the human in a dangerous situation. The proposed collaborative navigation framework was successfully tested, and the system showed a high level of performance in terms of navigation. An active safety module is currently being developed. The research and development

	$v(m/s)$			$w(rad/s)$			$\dot{v}(m/s^2)$			$\dot{w}(rad/s^2)$		
Path zone	max ↓	max ↑	mean	max ↷	max ↶	mean	max ↓	max ↑	mean	max ↷	max ↶	mean
Corridor	-0.1	0.2	0.1	-0.6	0.6	0.006	-2.7	2.0	0.0003	-8.9	6.7	0.002
Room	-0.06	0.2	0.08	-0.5	0.5	0.01	-1.7	1.4	-0.0008	-5.5	5.5	-0.001
Obstacles	-0.08	0.2	0.1	-0.4	0.4	0.002	-1.4	1.2	0.0004	-6.1	4.5	-0.0008
Door Passage	-0.06	0.2	0.08	-0.3	0.4	0.007	-1.0	0.8	-0.0003	-3.0	3.1	0.002

TABLE II

SUMMARY OF RESULTS FOR THE 9 USERS' EXPERIMENTS: LINEAR AND ANGULAR VELOCITIES AND LINEAR AND ANGULAR ACCELERATION, ACCORDING TO PATH ZONES.

	min	max	mean
Sens (%)	65.2	100	86.2
Spec (%)	96.1	100	99.1
Acc (%)	73.3	100	93.6
ST (s)	5.9	8.4	7.1

TABLE III

SUMMARY OF RESULTS FOR THE 9 USERS' EXPERIMENTS: BCI RESULTS INCLUDING SPECIFICITY (SPEC), SENSITIVITY (SEN) OF CONTROL VS. NON-CONTROL DETECTION, COMMAND ACCURACY IN CONTROL STATES (ACC), AND TIME REQUIRED FOR SELECTING A COMMAND (ST).

of a new BCI paradigm, with a dynamic set of commands that may vary according to navigation context, is also envisaged.

VI. ACKNOWLEDGMENTS

This work has been supported by Portuguese Foundation for Science and Technology (FCT) under the project ”AMS-HMI2012 - RECI/EEIAUT/0181/2012”, and project ”ProjB-Diagnosis and Assisted Mobility - Centro-07-ST24-FEDER-002028” with FEDER funding, programs QREN and COMPETE.

REFERENCES

[1] A. C. Lopes, G. Pires, and U. Nunes. Assisted navigation for a brain-actuated intelligent wheelchair. *Robotics and Autonomous Systems*, 61(3):245 – 258, 2013.
[2] T. Carlson and J. Millán. Brain-controlled wheelchairs: A robotic architecture. *IEEE Robotics and Automation Magazine*, 1(20):65–73, 2013.
[3] B. Rebsamen, C. Guan, H. Zhang, C.Wang, C. Teo, M. H. Ang, and E. Burdet. A brain controlled wheelchair to navigate in familiar environments. *IEEE Trans. Neural Syst. Rehab. Eng.*, 18(6):590–598, June 2010.
[4] I. Iturrate, J.M. Antelis, A. Kubler, and J. Minguez. A noninvasive brain-actuated wheelchair based on a P300 neurophysiological protocol and automated navigation. *IEEE Transactions on Robotics*, 25(3):614–627, June 2009.
[5] L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmar. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
[6] J. Bruce and M. Veloso. Real-time randomized path planning for robot navigation. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2383–2388 vol.3, 2002.
[7] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier. Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1056–1062, Sept 2008.
[8] Sven Koenig, Maxim Likhachev, and David Furcy. Lifelong planning A*. *Artificial Intelligence*, 155(1–2):93 – 146, 2004.
[9] S. Koenig and M. Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363, June 2005.
[10] M. Pivtoraiko, R.A. Knepper, and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.
[11] M. Rufli, D. Ferguson, and R. Siegwart. Smooth path planning in constrained environments. In *IEEE Int. Conf. on Robotics and Automation*, pages 3780–3785, May 2009.
[12] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *In Proceedings IEEE International Conference on Robotics and Automation (ICRA'85)*, 1985.
[13] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, June 1991.
[14] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *Robotics Automation Magazine, IEEE*, 4(1):23–33, Mar 1997.
[15] D. Nakhaeinia, P. Payeur, T. S. Hong, and B. Karasfi. A hybrid control architecture for autonomous mobile robot navigation in unknown dynamic environment. In *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*, pages 1274–1281, Aug 2015.
[16] M. Seder and I. Petrovic. Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In *IEEE Int. Conf. on Robotics and Automation*, pages 1986–1991, April 2007.
[17] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *IEEE Int. Conf. on Robotics and Automation*, volume 1, pages 341–346 vol.1, 1999.
[18] P. Ogren and N. E. Leonard. A convergent dynamic window approach to obstacle avoidance. *IEEE Transactions on Robotics*, 21(2):188–195, April 2005.
[19] D. Vanhooydonck, E. Demeester, A. Hüntemann, J. Philips, G. Vanacker, H. Van Brussel, and M. Nuttin. Adaptable navigational assistance for intelligent wheelchairs by means of an implicit personalized user model. *Robot. Auton. Syst.*, 58(8):963–977, August 2010.
[20] S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. In *Proc. IEEE International Conference on Robotics and Automation*, pages 802–807 vol.2, May 1993.
[21] ISR-UC 2016, RobChair 2.0: Brain-actuated wheelchair. https://sites.google.com/site/amshmi12/description/assisted-navigation. Accessed: 2016-05-05.
[22] G. Pires, U. Nunes, and M. Castelo-Branco. Statistical spatial filtering for a P300-based BCI: Tests in able-bodied, and patients with cerebral palsy and amyotrophic lateral sclerosis. *Journal of Neuroscience Methods*, 195(2):270–281, 2011.
[23] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf. A flexible and scalable SLAM system with full 3d motion estimation. In *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.