# A Biomimetical Dynamic Window Approach to navigation for collaborative control

Joaquin Ballesteros, Cristina Urdiales, Antonio B. Martinez and Gonzalo Ramos-Jiménez

***Abstract*—Shared control is a strategy used in assistive platforms to combine human and robot orders to achieve a goal. Collaborative control is a specific shared control approach in which user's and robot's commands are merged into an emergent one in a continuous way. Robot commands tend to improve efficiency and safety. However, sometimes assistance can be rejected by users when their commands are too altered. This provokes frustration and stress and, usually, decreases emergent efficiency. To improve acceptance, robot navigation algorithms can be adapted to mimic human behavior when possible. We propose a novel variation of the well known Dynamic Window Approach (DWA) that we call Biomimetical DWA (BDWA). BDWA relies on a reward function extracted from real traces from volunteers presenting different motor disabilities navigating in a hospital environment using a rollator for support. We have compared BDWA with other reactive algorithms in terms of similarity to paths completed by people with disabilities using a robotic rollator in a rehabilitation hospital unit. BDWA outperforms all tested algorithms in terms of likeness to human paths and success rate.**

## I. Introduction

Only in Europe, 8.3% of people aged 16 and over declared a severe disability in 2011 [1]. Assistive robotics enhance the autonomy of challenged people and their quality of life [2]. Assistive robots like wheelchairs, walkers or rollators typically rely on shared control techniques to manage interaction between user's and robot's navigation commands [3], [4], [5], [6], [7], [8], [9].

Depending on how much the navigation algorithm contributes to motion with respect to the user, we find different forms of shared control. In some approaches, the navigation algorithm only takes over when a dangerous situation is detected [3], [4]. In other approaches, it totally replaces users to perform a specific behavior (follow wall, doorway, etc) [5], [6]. In extreme, there are approaches in which the navigation algorithm has full control and the user only indicates his/her destination [7].

Some assistive devices like smart rollators require a highly collaborative profile because any struggle between user and robot may produce balance failure, which increases fall risk [10]. In this case, collaborative control (CC) [9], [8] presents clear advantages.

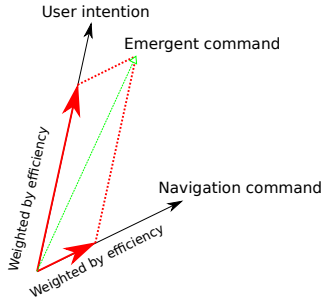



Figure 1. Emergent command in CC

Collaborative control is a type of shared control that analyzes user's and robot's commands to decide how much they should contribute to the emergent motion command at each time instant. CC increases the robot contribution when the user moves inefficiently. Besides, to minimize user's frustration and prevent loss of residual skills, it decreases the navigation algorithm contribution when the user is moving efficiently, even in hazardous areas. These efficiencies prioritize a contribution which: i) avoids oscillations; ii) avoids obstacles; and iii) maintains a consistent route towards the goal. Fig.1 shows how an emergent motion command can be calculated by adding human and robot motion commands, weighted by their respective efficiency, as proposed in [8]. It must be noted that collaborative control in [8], [11] was used on a wheelchair, so efficiency in a rollator would at least need to take balance into account as well.

CC algorithms tend to combine human input with reactive navigation algorithms. Deliberative algorithms are usually dismissed because prediction and optimization is hard when humans are included in the control loop [12] [13]. Some well known reactive navigation algorithms used in CC include Potential Field Approach (PFA) [14], Dynamic Window Approach (DWA) [15] and Vector Field Histogram (VFH) [16]. All these algorithms are in general more efficient than users, more so than people with disabilities, because humans do not necessarily optimize efficiency like navigation algorithms do [17]. In these cases, emergent commands may differ signifi-

cantly from human input, even when users' commands were valid. In previous experiments with collaborative control, we observed that some users tended to reject assistance when they realized that their commands were significantly changed [18]. This mostly happened when robot commands were much more efficient than their own even in areas where they did not need that much help. If reactive navigation commands were more similar to human input, users would find the corrections less distracting. Thus, we could enhance ergonomy and usability.

In order to mimic human trajectories, we could rely on biologically inspired navigation using Neural Networks [19], genetic algorithms [20] or Case Based Reasoning [21]. This approach requires intensive training, but the resulting system navigates like the training person. Alternatively, we could use an existing reactive navigation algorithm and simply adapt its optimization criteria to favor human-like trajectories. We are going to use this approach because it allows us to preserve all the algorithm benefits, e.g. continuity, navigation efficiency and to mimic humans as well. The goal of this paper is to adapt and validate a reactive navigation algorithm that mimics trajectories followed by human rollator users. Such an algorithm could be used in CC to improve assistance acceptance.

The next section describes our approach to develop a reactive biomimetic navigation algorithm. In order to feed the algorithm, we captured information from volunteers with different disabilities using a rollator to navigate in hospital environments. Section 2 presents our methodology to acquire and pre-process the required information. Our goal was to analyze how people navigated with a rollator. Section 3 presents our approach to extract such information from human trajectories. Section 4 shows several experiments to prove that trajectories provided by our navigation algorithm are very similar to human trajectories in our tests. Finally, section 5, presents conclusions and future work.

## II. Methodology

This section presents: i) our platform; ii) our test environments; iii) our volunteers; and iv) how acquired information is preprocessed to store data on human paths.

### A. The i-Walker rollator

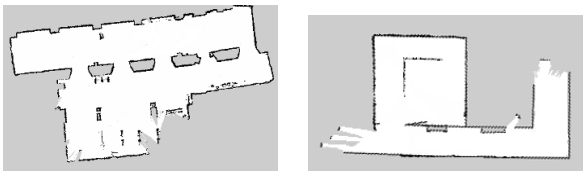
The i-Walker smart robotic rollator [22] relies on a standard MEYRA® rollator frame. It includes encoders in both wheels and it has 3 force components in each handlebar sensor. In addition, it also includes a tilt sensor, 2 forces sensor to measure the normal and a 2D laser. We work with Robot Operating System (ROS) framework [23] and we have used the Simultaneous Localization and Mapping (SLAM) algorithm proposed in [24] to obtain environment maps (Fig. 3). The i-Walker in these tests does not provide active help: trajectories provided by users uniquely depend on their condition and skills with a rollator. Sensors and encoders are only used to obtain information about these trajectories and about obstacles in the environment that might influence paths. Hence, we could work with any other wheeled rollator frame and BDWA could be implemented in any robot with odometry and a range sensor. To implement shared control in a rollator using BDWA, the rollator should also have the means to estimate the user intention, e.g. force sensors.

Figure 2. Volunteer during a test at Hospital Regional Universitario

### B. Volunteers in our tests

All volunteers in the presented work were patients from Hospital Regional Universitario of Malaga (HRU) or inpatients from Fondazione Santa Lucia (FSL) in Rome.

It is a common mistake to test assistive devices with healthy people. Users in need of support distribute their weight and move very differently from healthy ones when using a rollator. Some authors rely on asking therapists to train healthy volunteers to emulate rollator users [25]. Rather than extracting a navigation model from healthy individuals that actually do not need assistance to walk, we will extract our model from people in real need of an ambulatory device. Van Hook et al describe in [26] which gait disorders can benefit from ambulatory devices. Our volunteers had to present one of these disorders. They were also required to have previous experience with rollators to avoid *cold start* related issues. In this work we collected data from 41 volunteers (30 from HRU and 11 from FLS) presenting physical/cognitive disabilities: 20 females and 21 males. Users were in average $66.58 \pm 12.71$ years old (range $31-86$ years). Twenty four volunteer presented physical disabilities: fractures (intertrochanteric hip, prosthetic



(a) Hospital Regional Universitario of Malaga (b) Fondazione Santa Lucia in Rome

Figure 3. Maps of test areas

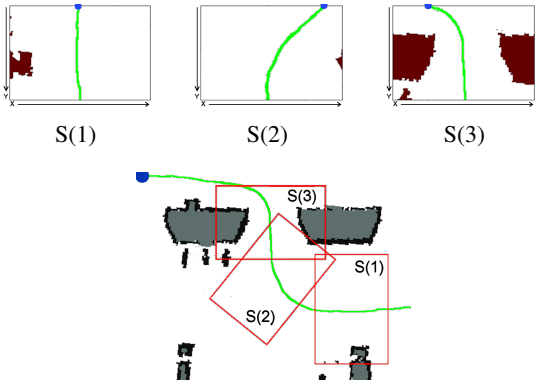



Figure 4. Example of local situation extraction: Input trajectory (Line) and destination (Dot), plus situation grids at S(1), S(2) and S(3)

femur (x2), intertrochanteric fracture femur, knee, tibial (x2), ankle), lower limb amputation (x5), polytraumatism (x3), meniscus tear (x2), tetraparesis (x2), total hip replacement, hip arthroplasty, spinal fusion and rotated left leg. Seventeen volunteers presented cognitive disabilities: Parkinson (mild or severe, x7), vestibular diseases (x2), dementia (mild or severe , x4), ischemia, stroke (x2) and multiple sclerosis.

### C. Tests

Tests were performed at hospital corridors at HRU[1] and FSL during several weeks (Fig. 3). These locations are not the same from a global point of view. Nevertheless, from a local point of view, both topologies allow the user to turn (left or right, smooth or sharp) and move forward in a corridor. No special constraint was requested, so equipment and furniture could be found in both places. Volunteers followed different routes to fixed goals selected dynamically by the therapist, while other patients and staff moved freely around during 3 minutes (Fig. 2). According to clinicians, these environments are representative for indoor assisted navigation.

The user's trajectories only depended on their target, nearby obstacles location and their way of walking. Therefore, for each local situation (per user and path) we stored: i) location of nearby obstacles; ii) local goal; and iii) trajectory followed by the user. All situations were stored in 60x40 grids, that we represented as images (e.g. Fig. 4). We stored a new local situation each time that: a) the trajectory curvature changed more than 10% of the maximum curvature variation allowed by out platform, as extracted from our odometry tests.; or b) the nearby obstacle configuration changed significantly, i.e. more than 25% of the cells in the current grid had changed with respect to the previous one. Thus, any local situation presents a path with a different curvature and/or obstacle distribution than the previous one. In Fig. 4 three local situations, corresponding to two major curvature changes in the path, are extracted (S(1-3)). All situations are relative to their departure point (30,0). Our tests returned 19096 different local situations.

User's trajectories are sampled at a fixed rate, so they may be represented by a sequence of points in a 2-D space. These trajectories could be represented as graphs or tensors [27], but we have chosen a matrix representation rather than a graph because: a) matrix indices implicitly provide space information, so trajectory matching is computationally cheap; and b) we can use the same matrix representation to combine all trajectories within a cluster and obtain prototypes (section IV-D).

Tensors could have been used as an extension of a matrix to include additional information [27], but handling them increases complexity. A matrix representation can store all the information we need in our method, so we chose this option.

## III. A NEW REACTIVE NAVIGATION ALGORITHM: BDWA

Any reactive navigation algorithm typically tries to generate trajectories to arrive to a goal in a safe, efficient and non necessarily optimal way using only local information. Fig. 5 shows three examples of reactive trajectories for PFA, VFH and DWA. In this example, none of them is too similar to the trajectory returned by one of our volunteers. As commented, PFA based solutions are simple and easy to adapt to new restrictions. However, PFA may return unstable commands in local situations with a disperse obstacle distribution. Also, they are reportedly affected by oscillations when paths are close to obstacles [28], as seen in Fig. 5. Furthermore, like VFH based methods, in their simplest formulation, they assume that robots are holonomic and, hence, may return oversimplistic trajectories. These solutions are unacceptable for rollators because sharp steering may produce balance failure and increases fall risk. Additionally, oscillations and unstable commands in narrow areas may affect stability as well. Therefore, PFA and VFH solutions can not be adapted to mimic user trajectories.

[1]Collaboration under the framework *New technologies in rehabilitation: walking aids: a pilot study with robotic walker*

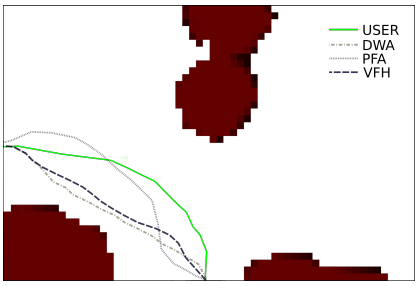



Figure 5. Reactive navigation solutions and human solution for a local situation

DWA solves these problems, at least partially, because it explicitly takes the dynamics of the platform into account to generate a response within the agent physical framework. First, DWA sets boundaries to the problem solution space to limit platform movements and avoid dangerous maneuvers. Then, DWA selects the linear and angular velocity $(v, \omega)$ within the window of allowed velocities imposed by the mobile dynamics that maximizes the following reward function:

$$
\begin{aligned}
&R(v,\omega,o,g) = \\
&\sigma(-\alpha{\cdot}angle(v,\omega,g) + \beta{\cdot}dist(v,\omega,o) + \gamma{\cdot}vel(v,\omega)) \\
&v \in \left\{ v_{ini} + t * \Delta v \mid 0 \leq t \leq \frac{v_{end} - v_{ini}}{\Delta v} \right\} \\
&w \in \left\{ w_{ini} + t * \Delta w \mid 0 \leq t \leq \frac{w_{end} - w_{ini}}{\Delta w} \right\}
\end{aligned}
\tag{1}
$$

being:

- $v_{ini}$, $v_{end}$, $\Delta v$: Minimum, maximum and step linear velocities.
- $w_{ini}$, $w_{end}$, $\Delta w$: Minimum, maximum and step angular velocities.
- $angle(v, \omega, g)$: Function that returns the angle to goal $g$ if we applied $v$ and $\omega$.
- $dist(v, \omega, o)$: Function that determines how far obstacles $o$ would be if we applied $v$ and $\omega$. Its value is set either to the distance to the nearest obstacle or to $-\infty$ in case of obstacle collision (invalid trajectories).
- $vel(v, \omega) = v$: This function returns the linear velocity and ensures that the platform is moving and not rotating. Its value is set either to the $v$ linear velocity or to $-\infty$ in case of obstacle collision (invalid trajectories).
- $\alpha, \beta$ and $\gamma$: Constants to adjust the relevance of each function. In this work $\alpha = 0.2$, $\beta = 2$ and $\gamma = 0.2$ as proposed in[15].
- $\sigma$: Smoothing constant.

The trajectory produced by DWA for goal $g$ and obstacles $o$ only depends on reward function $R(v, \omega, o, g)$ and on the window of possible velocities imposed by the dynamics of the platform ($v_{ini}$, $v_{end}$, $\Delta v$, $w_{ini}$, $w_{end}$, $\Delta w$). Hence, any DWA solution needs to define a reward function and the velocities window to operate. If reward functions are based on navigation optimization factors, paths become efficient, but not necessarily similar to human trajectories(Fig. 5).

### A. Defining a Window of Velocities

Since our platform is physically controlled by a person, the combined system dynamics also depend on the user. Therefore, we measured the dynamic of users and selected their boundaries ($v_{min}^{users}$, $v_{max}^{users}$,$w_{min}^{users}$, $w_{max}^{users}$). These boundaries are the Window of Velocities of the new biomimetical DWA (BDWA) reward function. The new Window of Velocities reduce balance failure problems because it limits steering and forward velocities to the user's limits according to real experience.

### B. Defining a Reward Function

After boundaries have been properly fixed, DWA uses reward function $R$ (Eq. 1) to choose the best velocities within the allowed solution space. However, people do not necessarily rely on optimization to navigate. Furthermore, optimization parameters may differ significantly among people with disabilities: depending on their physical condition, some paths may be preferable to apparently more efficient ones from a traditional point of view (e.g. shorter ones).

Analytically defining all potential effects of disability on walking with rollators for a general population is almost impossible and it is out of the scope of this work. We gather data from users with a variety of physical and cognitive disabilities to generalize how they cope with the different situations they face. Using this information, we can generate a new DWA reward function $R_b(v, \omega, o, g)$ that favors trajectories similar to human ones.

In order to extract and represent how rollator users walk and, hence, define $R_b$, we have gathered data from several volunteers presenting different disabilities. Extracted data is clustered into groups of similar situations. Cluster prototypes are used to generalize how people deal with each situation. Next section covers the clustering method we employed.

## IV. An Evidence Based Human Navigation Model

In this work, we rely on a k-centroids algorithm [29] to cluster our local situations K-centroids algorithms split

data ($x \in X$) into $k$ groups choosing a set of centroids $\{c_1, ..., c_k\}$ which minimizes the average distance $d(x, c) : X \times C \to R^+$ to each element from its group:

$$\underset{\{c_1,...,c_k\}}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} d\left(x_n, centroid(x_n)\right)) \tag{2}$$

We have specifically chosen this method because resulting clusters have a representative element (centroid), as desired. New situations can be easily classified into a cluster depending on their distance to its centroid.

### A. Choosing a distance

Clustering results depend on the employed distance, the initial set of centroids and how they are calculated [30]. If centroids are equal to the average of all element within a group and we use the Euclidean distance, the resulting algorithm is known as **k-means** [31]. Another option is **k-median**, where centroids are the median of each element within a group and we use the Manhattan distance instead. In general, we observe that $k-median$ preserves better the original data nature, whereas $k-means$ provides more compact groups. However, it is often advisable to test both distances and check which one is more adequate for the data to be clustered.

### B. Choosing the correct k

The major drawback of k-centroids algorithms is that the number of groups, $k$, needs to be established a priori. The quality of clustering depends heavily on $k$.

There are several indexes to measure clustering quality and, hence, to suggest which $k$ to select [32]. After some tests, we choose the well known Davies-Bouldin index [33], because it provided the best clustering results for our data set. This index basically defines a measure of separation and compactness for clusters and tests several values of $k$ to choose which one optimizes those values.

### C. A 2-Steps Clustering

Since we are working with a large amount of data, we have designed a 2-steps clustering process for computational efficiency. First, we pre-cluster all data according to target similarity. This process is fast because it involves only 2 variables from each sample. Resulting groups are clustered according to obstacle configurations (2400 obstacles grid). We use Davies-Bouldin index in each clustering step in order to select the optimal number of groups.

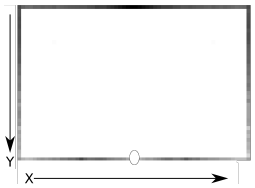



Figure 6. Users' goals in our dataset (Greyscale values represent frequency, white color is the low frequency).

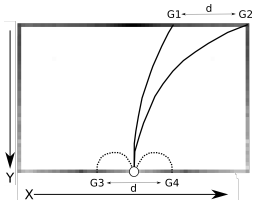



Figure 7. Clustering problem: Both the Euclidean and the Manhattan distance $d$ are the same for $(G1, G2)$ and $(G3, G4)$

*1) Goal based clustering:* Any local situation grid has 196 (60+60+38+38) potential goals, i.e all boundary cells of the grid. Figure 6 shows every potential goal in a grid, where the departure point is marked with a circle. Most frequent destinations in our tests are represented in black, and least frequent ones are printed in white. Locally, users tend to favor goals ahead of their position, because any significant curvature change leads to create a new grid.

Unfortunately, paths to reach a goal in a grid may change significantly, specially when complex maneuvers are performed. Therefore, our clustering algorithm can not directly work with the distance between goal coordinates to cluster trajectories. Fig. 7 shows how very different goals may present the same distance, both using the Euclidean and the Manhattan distance.

Function $genTraj_{goal}(g_1)$ generates a DWA trajectory to arrive to goal $g_1$. It uses the windows of velocities defined in section III-A in a grid without obstacles. We define the goal based cluster distance between goal $g_1$ and goal $g_2$, as the mean squared error (MSE) between $genTraj_{goal}(g_1)$ and $genTraj_{goal}(g_2)$ (Eq. 3).

$$\begin{aligned} &MSE_g(g_1, g_2) = \\ &\quad MSE(genTraj_{goal}(g_1), genTraj_{goal}(g_2)) \end{aligned} \tag{3}$$

Using $MSE_g$ as distance function, we can precluster our data according to goal parameters. In this precluster stage we use a k-median algorithm because the distance defined in Eq. 3 requires real goals to work with and prototypes in a k-means algorithm are averages of real elements.

In this stage, we chose the lowest Davies-Bouldin index from 100 repetitions for each $k$. Fig. 8 shows the results. We observe how Davies-Bouldin index value

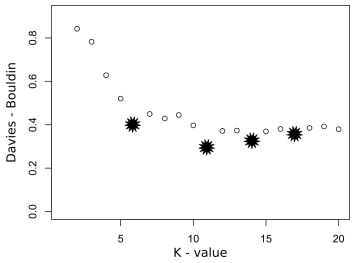



Figure 8. Goal based clustering: Davies-Bouldin for $2 \leq k \leq 20$

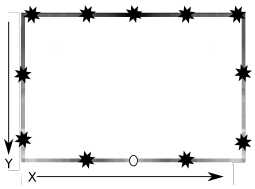



Figure 9. Clusters centroids for $k = 11$.

stabilizes for values over eleven. Other inflection points are $k = \{6, 14, 17\}$. We discarded $k = \{6, 14\}$ because they did not return a cluster for the most frequent trajectories (move ahead). Also, we discarded $k = \{17\}$ because resulting centroids were too close (less than $0.3m$). Therefore, we chose $k = 11$.

Fig. 9 shows resulting centroids for $k = 11$. These 11 centroids roughly correspond to 1 move ahead path, 3 turn right/left maneuvers and 2 right/left U-turn maneuvers.

After this stage, each cluster includes paths that share a similar destination. These paths may be very different depending on the obstacle configuration in the grid (fig. 10). Our next clustering stage splits the situations in each bin into classes depending on the obstacle configuration.

*2) Obstacle based clustering:* Again, we use a k-median algorithm in this stage because we want centroids to reliably represent real obstacle configurations. Otherwise, even a minimal obstacle dispersion in class elements would return wide diffuse obstacle areas. Any two obstacle grids are simply compared cell by cell using a Manhattan distance.

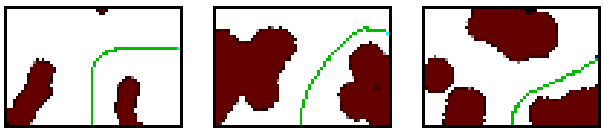


Figure 10. Different obstacle layouts in the same bin after goal-based clustering

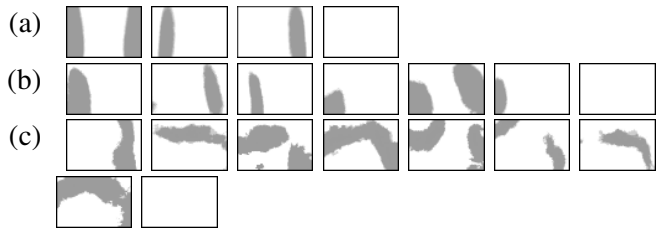



Figure 11. Group centroids: 11.(a) Move ahead, 11.(b) SMT Left 1, 11.(c) SHT Left 1. Gray scale represents obstacles's location probability (white is lack of obstacles).

Table I
OBSTACLE BASED CLUSTERING: DAVIES BOULDIN INDEXES PER BIN

| | $k$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| *Group* | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Move ahead | 2.55 | 2.52 | 2.91 | 2.67 | 2.82 | 2.74 | 3.07 | 2.89 | 3.00 | 3.06 |
| SMT left 1 | 2.89 | 2.83 | 2.62 | 3.62 | 2.57 | 2.69 | 2.64 | 2.80 | 2.89 | 2.61 |
| SMT right 1 | 3.14 | 2.83 | 3.20 | 2.96 | 2.46 | 2.76 | 3.41 | 2.57 | 2.54 | 2.58 |
| SMT left 2 | 3.51 | 3.07 | 2.45 | 2.54 | 2.31 | 2.59 | 3.63 | 3.09 | 2.94 | 2.77 |
| SMT right 2 | 2.59 | 3.49 | 3.39 | 3.60 | 2.52 | 2.59 | 3.43 | 3.27 | 2.71 | 3.52 |
| Turn Left | 2.10 | 2.47 | 2.88 | 4.26 | 2.38 | 2.00 | 3.68 | 2.811 | 2.24 | 2.29 |
| Turn Right | 3.83 | 2.80 | 3.05 | 3.33 | 3.18 | 2.32 | 3.96 | 2.72 | 2.94 | 2.89 |
| SHT left 1 | 2.41 | 2.21 | 2.43 | 2.22 | 2.59 | 1.83 | 1.68 | 2.24 | 2.05 | 2.25 |
| SHT right 1 | 3.56 | 2.64 | 3.00 | 2.78 | 2.79 | 1.97 | 1.54 | 2.09 | 2.05 | 2.25 |
| SHT left 2 | 3.02 | 2.46 | 2.58 | 2.14 | 2.80 | 2.47 | 1.90 | 2.65 | 2.23 | 2.31 |
| SHT right 2 | 3.03 | 2.63 | 2.60 | 2.38 | 2.39 | 2.23 | 1.87 | 2.31 | 2.62 | 2.75 |

This clustering process is applied to each of the eleven bins resulting from our previous goal based clustering with $k$ ranging from 3 to 12. In this case we simply selected the lowest local minima of the Davies-Boulding index in the defined $k$ range (Table I).

Bins like Move Ahead present just a few obstacle configurations (Fig. 11.(a)). More complex maneuvers typically result in a larger number of classes. Sharp turns (SHT) in general require more groups ($k = \{9\}$) than smooth turns (SMT) ($k = \{7\}$) because goals are closer to the origin and then, there are more potential obstacle distributions in the grid (Figs. 11.(c), 11.(b)).

*3) Scalability of the proposed methodology:* Scalability can be tested by running our method for an increasing number of volunteers and checking if resulting classes tend to stabilize. We have performed our 2-steps clustering for $\{5, 10, 15, 20, 25, 30, 35, 41\}$ volunteers, randomly selected from our full population. Figure 13 shows the best $k$ values according to the Davies Boulding index for our goal-based clustering step. As commented, if $k$ is low no cluster represents the most frequent trajectories (move ahead) and if $k$ is large, centroids are often too close. We can observe that k=11 is consistently the best choice in most cases, except for 5 and 15 volunteers. The same effect can be observed in the obstacle-based clustering step (Figure 12). In general, when the number of volunteers is too low, the best $k$ values may change significantly, but they tend to stabilize around a fixed value when the number of volunteers grows.

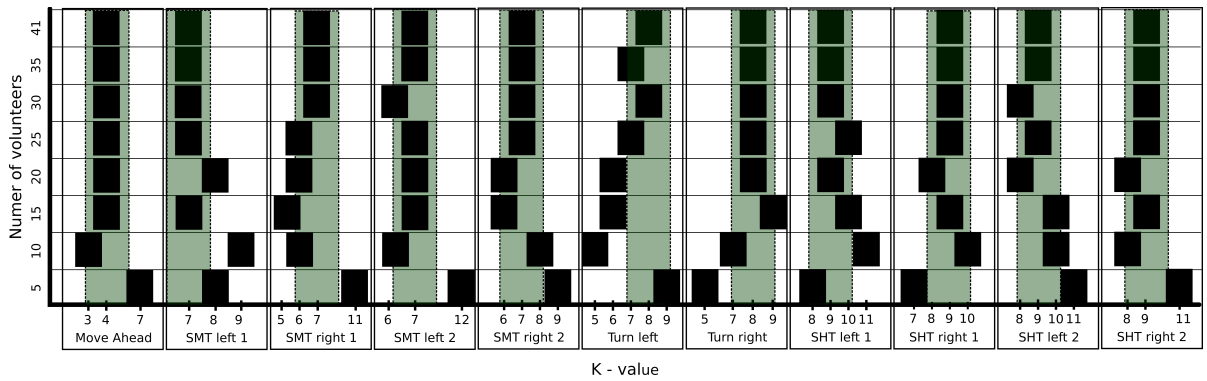



Figure 12. Best $k$ per class in the obstacle-based clustering step.

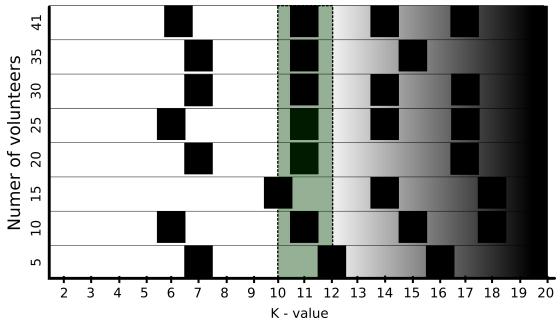



Figure 13. Best $k$ in goal-based clustering step.

### *D. Information representation and Reward Function calculation*

After our 2-step clustering process was finished, we obtained 84 clusters. Each cluster includes all (partial) paths completed by our volunteers to reach a similar goal $g$ and a similar obstacle distributions $o$ (Fig. 14(a)). We can represent human trajectory information for each cluster by averaging all trajectories in that cluster, i.e. we increase by 1 the values of the cells included in each trajectory. This average can be stored into a matrix $MP_{g,o}$ (Fig. 14(c)), where $g$ is the centroid of the goal cluster and $o$ is the centroid of the obstacle cluster. It must be noted that obstacle averaging does not return wide obstacle areas anymore because obstacles in the same cluster tend to be in similar locations at this point.

The highest elements of $MP$ correspond to locations included in most users' paths for each each given goal and obstacle configuration. Originally, we noticed that all maxima were close to the starting area. This happened because all grids have the same departure point, so every path in our tests started there. To solve this issue, every $MP_{g,o}$ element is weighted by its distance to the origin. After this factorization, we obtained weighted matrices $MP^f_{g,o}$. Fig. 14(d) shows how peaks are not shifted towards the departure point anymore after weighting.

Finally, we noticed that discretizing local situations into 60x40 grids affected generated trajectories negatively: averaging sometimes created valleys in crowded obstacle areas. In order to reduce these valleys, we applied a gaussian blur to $MP^n_{g,o}$ to obtain the final matrices (14(e)).

At this point, element $< x, y >$ in $MP^n_{g,o}[x, y]$ is a metric for the frequency of users crossing $< x, y >$ while traveling to a goal close to $g$ with an obstacle configuration similar to $o$. Hence, $MP^n_{g,o}$ represents how people in average navigate towards $g$ given the obstacle configuration $o$. Our BDWA reward function (Eq. 4) uses $MP^n_{g,o}$ to calculate how similar a given trajectory is to this average user solution. The more similar the trajectories are the higher its value.

$$R_b(v, w, o, g) = \frac{\sum_{s=0}^{LEN(TRJ(v,w))} MP^n_{g,o}[TRJ^s(v, w)]}{LEN(TRJ(v, w))} \tag{4}$$

being

$$v \in \left\{ v_{ini} + t * \Delta v \mid 0 \leq t \leq \frac{v_{end} - v_{ini}}{\Delta v} \right\}$$

$$w \in \left\{ w_{ini} + t * \Delta w \mid 0 \leq t \leq \frac{w_{end} - w_{ini}}{\Delta w} \right\}$$

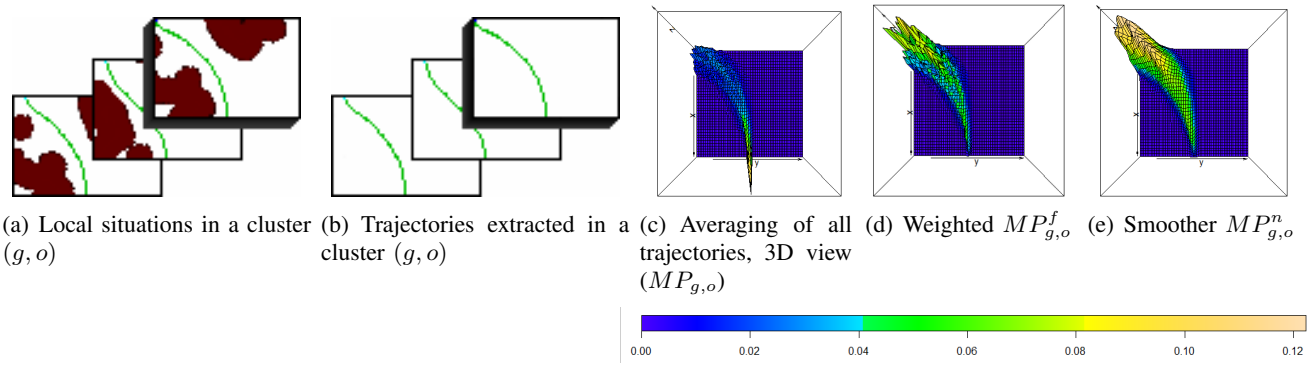
$TRJ(v, w)$ Trajectory generated with $v$ and $w$.

$LEN(TRJ(v, w))$ Length of trajectory TRJ(v,w) or number of step inside de windows (60x40).

$TRJ^s(v, w)$ Robot position (x,y) in trajectory $TRJ(v, w)$ at $s$ step.

In order to use equation 4, we need to calculate which matrix $MP^n_{g,o}$ is more similar to trajectory $TRJ(v, w)$, generated with $v$ and $w$, in terms of goal and obstacle distribution. This process increases BDWA computational cost with respect to DWA. Next section shows how to optimize the process to implement our BDWA algorithm in a computationally efficient way.

*1) BDWA implementation:* Given a local situation with obstacles $o$ and goal $g$, DWA returns a pair of




(a) Local situations in a cluster $(g, o)$ (b) Trajectories extracted in a cluster $(g, o)$ (c) Averaging of all trajectories, 3D view $(MP_{g,o})$ (d) Weighted $MP^f_{g,o}$ (e) Smoother $MP^n_{g,o}$

Figure 14. $MP_{g,o}$ Matrix extraction steps using the whole set of local situations in cluster $(g, o)$.

lineal and angular velocities $(v_{best}, w_{best})$ which maximizes the DWA reward function in the interval imposed by the dynamic of the robots ($v \in [v_{ini}, v_{end}]$ and $w \in [w_{ini}, w_{end}]$) (see algorithm 1).

**Data:** Obstacle $o$, Goal $g$
**Result:** Best linear $v_{best}$ and angular $w_{best}$ velocities

1 *//Initialization*;
2 $value_{best} \leftarrow 0$;
3 $v_{best} \leftarrow v_{ini}$;
4 $w_{best} \leftarrow w_{ini}$;
5 *//Iteration within the robot dynamics*;
6 $v \leftarrow v_{ini}$;
7 **while** $v \leq v_{end}$ **do**
8  $w \leftarrow w_{ini}$;
9  **while** $w \leq w_{end}$ **do**
10  $value \leftarrow R(v, w, o, g)$;
11  **if** $value_{best} \leq value$ **then**
12  $value_{best} \leftarrow value$;
13  $v_{best} \leftarrow v$;
14  $w_{best} \leftarrow w$;
15  **end**
16  $w \leftarrow w + \Delta w$;
17  **end**
18  $v \leftarrow v + \Delta v$;
19 **end**
20 **return** $(< v_{best}, w_{best} >)$;

**Algorithm 1:** DWA pseudo code

DWA iterates $n$ times, being $n$ equal to $\frac{v_{end}-v_{ini}}{\Delta v} * \frac{w_{end}-w_{ini}}{\Delta w}$. As commented, we could implement our BDWA replacing $R(v, w, o, g)$ by $R_b(v, w, o, g)$ in algorithm 1 (line 10), but this approach is too computationally expensive.

In order to spare online processing in BDWA (algorithm 2), we rely on a 2 stage offline process that needs to be performed only once:

- In the first stage, we store for each of 197 possible goals in a local situation, which centroid belongs to them (see section IV-C1). $goalList(g)$ function (line 2) implements it, it returns which centroid belongs to goal $g$.
- In the second stage, we calculate and store a sorted list of $< v, w >$ in descending order of $R_b(v, w, o, g)$ for each of our 84 $MP^n_{g,o}$. $userSolution$ function (line 3) returns a ordered list of velocities of matrix $MP^n_{goalCluster,o}$ which is more similar to the obstacle centroid cluster $o$ if goal is $goalCluster$.

At worst, $userSolution$ performs 9 matrices comparisons (see table I). Once we have selected matrix $MP^n_{g,o}$, $getBest(MP^n_{g,o}, cont)$ function (line 7) returns the best solution in the evaluated $cont$ group. These additional costs are decreased by swapping the double loop in algorithm 1 (lines 7-18) by a simple search in a ordered list in algorithm 2 (lines 6-9).

Our experiment section proves that BDWA trajectories are close to human ones, as expected.

**Data:** Obstacle $o$, Goal $g$
**Result:** Best linear $v$ and angular $w$ velocities

1 *//First, we get* $MP^n_{g,o}$;
2 $goalCluster \leftarrow goalList(g)$;
3 $MP^n_{g,o} \leftarrow userSolution(goalCluster, o)$;
4 *//Second, we looking for best solution*;
5 $cont \leftarrow 1$;
6 **repeat**
7  $< v, w > \leftarrow getBest(MP^n_{g,o}, cont)$;
8  $cont \leftarrow cont + 1$;
9 **until** $not(isValid(< v, w >, o))$;
10 **return** $(< v, w >)$;

**Algorithm 2:** BDWA pseudo code

Table II
SUCCESS RATE BY GOAL GROUPS. PAIRED WILCOXON TESTS. BDWA GREATER THAN: DWA ($p = 0.00049$). PFA ($p = 0.00049$) AND VFH ($p = 0.01611$).

| | Algorithm | | | |
|---|---|---|---|---|
| Group | BDWA | DWA | PFA | VFH |
| Move ahead | 99.87 | 99.05 | 99.27 | 99.35 |
| SMT left 1 | 99.89 | 83.97 | 96.80 | 98.86 |
| SMT right 1 | 99.69 | 85.41 | 97.85 | 99.08 |
| SMT left 2 | 99.53 | 62.31 | 66.10 | 95.90 |
| SMT right 2 | 99.41 | 70.25 | 72.66 | 98.06 |
| Turn left | 98.87 | 63.25 | 69.79 | 96.73 |
| Turn right | 97.76 | 64.01 | 76.46 | 98.43 |
| SHT left 1 | 99.17 | 52.89 | 83.47 | 88.43 |
| SHT right 1 | 98.49 | 73.37 | 96.48 | 94.47 |
| SHT left 2 | 95.84 | 89.54 | 71.29 | 97.76 |
| SHT right 2 | 98.35 | 58.70 | 82.75 | 96.22 |
| Average | 98.81 | 72.98 | 82.99 | 96.66 |

## V. EXPERIMENTS AND RESULTS

We have completed a k-fold cross validation [34] with $k = 10$ for assessing the comparison with the whole data set. First, we randomly split all samples gathered in our experiments into $k$ groups of equal size. Then, the cross-validation process is repeated $k$ times. In each iteration, $k-1$ data groups are used to feed all reactive navigation algorithms.

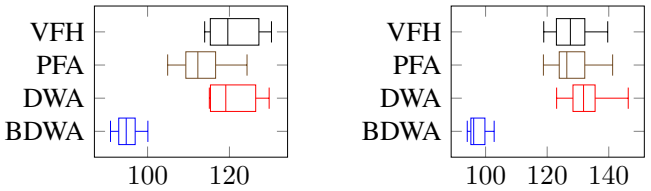
We have implemented four algorithms in order to check which trajectories are actually closer to human ones: PFA, VFH, DWA (algorithm 1) and our BDWA (algorithm 2). All four algorithms are provided the same target and they have to autonomously reach it. The last data group in k-fold is used as reference for all implemented reactive navigation algorithms, so we can compare the trajectories they return with real user solutions, obtained from our volunteer group described in subsection II-B.

Equation 3 compared two trajectories point by point using a Euclidean distance between goal coordinates. This approach is valid for simple trajectories, but MSE is sensitive to noise and disturbances [35]. Hence, in these tests we have used an implementation of the Dynamic Time Warping (DTW) distance [36]. DTW results depend more on the shape of the paths than in punctual differences. We have compared human paths in all our 19096 situations with results provided by PFA, VFH, DWA and BDWA for each situation conditions (departure and arrival points and obstacle configuration). Low DTW values for a given algorithm indicate that its trajectories are similar to human ones. We have represented the results in a box-and-whisker format, including the median, the first and third quartile, and the maximum and minimum values of our data (Figure 15).

BDWA consistently returns more human-like trajectories than the other algorithms for all SMT goals (Figs. 15(a)-15(d)) and right/left turns (Figs. 15(e)-15(f)): median DTW values decrease between 15.58% and 49.18% in these situations. Unlike other algorithms, BDWA does not return the shortest path. Instead, it searches for a path that mimics what our volunteers did in each specific situation, taking into account the physical constraints of the rollator. Fig. 16 shows an example of this behavior for a SMT left 2 goal. It presents the paths returned for a specific situation by every tested algorithm compared to the path followed by a female volunteer (77 years old) with a prosthetic femur fracture. VFH return the shortest paths, since PFA presents oscillations near obstacles. BDWA does not reproduce the specific behavior of this particular user; it is based on the average behavior of all our volunteers to cope with that specific situation. Still, BDWA returns the most human-like path: DTW is equal to 130.71, 165.23 and 220.13 for BDWA, PFA and VFH respectively.

Improvements are not so significant in SHT situations: some users may require more space to cope with SHT than others from the same obstacle cluster (Fig. 19). Hence, variability within a cluster is larger: some users are well represented by its prototype, whereas others are not. This variability provokes an increase in DTW. Still, median DTW values for BDWA in these situations are still 7.71% to 39.7% lower than the rest (figures 15(g)-15(j)).

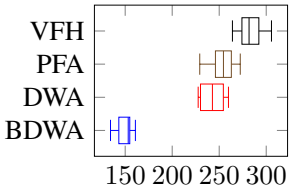
Other algorithms outperform BDWA only in 1 out of 11 classes: Go forward (Fig. 15(k)). Differences in DTW medians in this case are small (18.77 at most, VFH vs DWA), but VFH and PFA outperform BDWA in terms of human similarity. This happens for two reasons. First, human trajectories within this class have a very large variability, because these situations are less constrained than the rest. Hence, BDWA paths are similar to some human ones, but not to others. Also, in these cases many volunteers tended to go straight to the goal, even if they had to force the rollator sometimes. VFH and PFA do not operate under constraints, so they outperform DWA and BDWA in Go Forward situations. DWA follows on rollator constraints, whereas BDWA combines human/rollator constraints. Hence, DWA returns the worst results in terms of human similarity whereas BDWA balances human similarity and safety constraints. Fig. 17 presents an example of this behavior for a female volunteer (45 years old) with polytraumatism. All paths are very similar in these situations, as commented. Still, it can be noted that VFH returns the most human like path, followed by PFA, which is affected by oscillations. The DWA path is shifted towards the left obstacle mass,



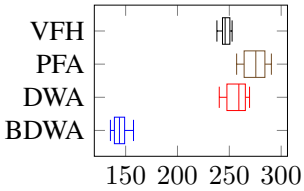

(a) SMT right 1



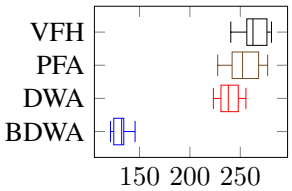

(b) SMT left 1



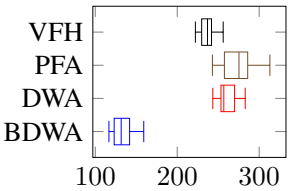

(c) SMT right 2



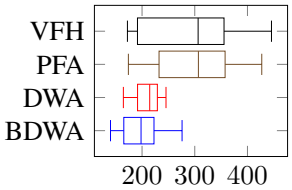

(d) SMT left 2



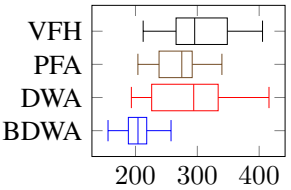

(e) Turn right



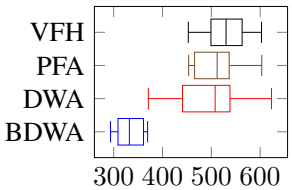

(f) Turn left



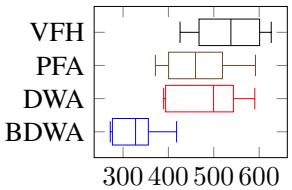

(g) SHT right 1



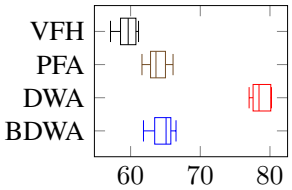

(h) SHT left 1




(i) SHT Right 2




(j) SHT left 2




(k) Go forward

Figure 15. DTW distances (X axis) for each local situations between the reactive navigation solutions (Y axis) and the human one.

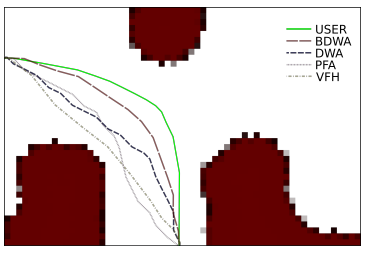



Figure 16. Comparative results obtained for a *SMT left 2* goal for a random user.

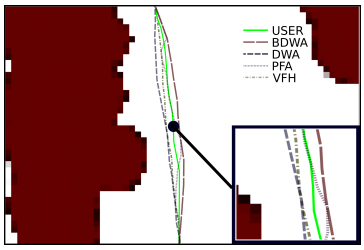
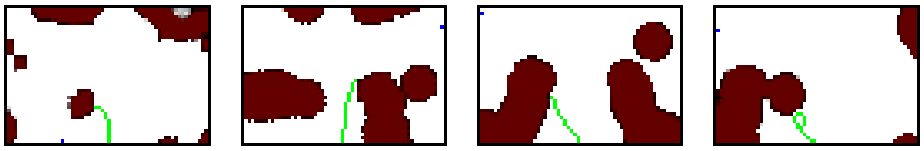



Figure 17. Comparative results obtained for a *Go forward* goal for a random user.

probably because it is safe and preserves curvature. However, humans usually tend to keep away from obstacles if possible. Hence, BDWA is shifted towards the right, even though this choice implies some minor steering. DTW distances to the human path are 67.94, 73.64, 63.49 and 58.28 for BDWA, DWA, PFA and VFH, respectively.

Given the reactive nature of all tested algorithms, we need to measure not only likeness to human paths, but also success rate. Given a situation, a path is successful if it safely connects the departure and arrival points. Unsuccessful paths converge to local minima instead. Fig. 20 shows some examples of unsucessful paths. BDWA failed due to the commented cluster variability in sharp maneuvers. DWA failed because its last window did not detect the obstacle and the robot got too close to it. PFA failed because forces produced by obstacles on both sides of the robot almost cancelled each other and made the robot drift towards a local minimum. VFH failed because it reached a local minimum and fell into a loop. Our groups include similar obstacle configurations and goals (Fig. 11), but there are still different situations in a group. A navigation algorithm might solve some of these situations and fail to solve others. If the success rate of a given algorithm for a group is not high enough, sometimes it can not return a path to the goal. In collaborative control, that means that the robot might not be able to help the user to solve that situation.

Table II shows the success rate per algorithm to reach the 11 goals in Fig. 9. BDWA achieves a higher success rate than any other solutions in average. Besides, paired Wilcoxon tests [37] validate a significant success rate improvement using BDWA. BDWA has a higher success rate than DWA ($p = 0.00049$), PFA ($p = 0.00049$) and VFH ($p = 0.01611$). Although VFH is second best in terms of success rate, VFH paths are often not similar to human ones (Fig. 15). BDWA outperforms VFH and DWA in terms of success rate, plus they do not necessarily return trajectories similar to human ones.



(a) BDWA (b) DWA (c) PFA (d) VFH

Figure 18. Examples of unsuccessful paths for all tested algorithms.

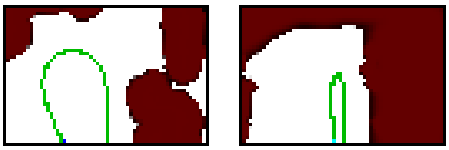


Figure 19. Different solutions to the same situation (U-turn)

## VI. Conclusion and Future Work

This paper has presented a biomimetic variation of DWA that we have called BDWA. In a shared control, the differences between the navigation commands and the users command may produce rejection and stress in the users. Hence, integrating BDWA in a shared control system may help to decrease them.

BDWA is based on defining a new reward function to generate trajectories that are significantly closer to what a human in need of a rollator would do. This reward function has been extracted via clustering from real navigation traces from volunteers with different disabilities using a passive rollator in a hospital environment. Given the nature of our input data, this version of BDWA can work in any rollator. However, it is not extendable to other support devices like walkers or canes because they affect trajectories in a different way and their target population walks differently. The methodology could be extended to these devices, but it would be necessary to extract and process data from the correct population group.

Since BDWA is based on conventional DWA, the risk of fall is reduced because dynamics are always taken into account. Besides, we have implemented BDWA in a fast, efficient way by adding a pre-calculation offline stage. This implementation is less efficient than DWA in terms of memory storage, but it works at frequencies greater than 10 times per second in a embedded system (Raspberry Pi).

In order to prove that BDWA returns more human like trajectories than other reactive algorithms, we have compared trajectories from four different algorithms (PFA, VFH, DWA and BDWA) with human rollator trajectories gathered from volunteers with disabilities during real navigation tests with rollators.

BDWA offers the best results in terms of human likeness in 10 out of 11 types of maneuvers, plus its success rate is the best among all tested algorithms. Differences between BDWA and other navigation algorithms are particularly noticeable during steering maneuvers, as expected.

We have tested how much our methodology depends on the processed number of volunteers to check its scalability. If this number is larger than 10, we have observed that our model tends to stabilize. Although results are promising, we plan to capture and analyze more experimental traces to validate our model further and obtain even more similarity to human trajectories.

Future work will focus on implementing BDWA in a collaborative control framework and exhaustively testing it for different target groups.

## Acknowledgements

This work has been partially supported by the Spanish Ministerio de Educacion y Ciencia (MEC), Project. TEC2011-29106, Project n. TEC2014-56256-C2-1-P, Hospital Regional Universitario of Malaga and Fondazione Santa Lucia of Rome.

## References

[1] S. Grammenos *et al.*, "European comparative data on europe 2020 & people with disabilities," 2013.

[2] S. W. Brose, D. J. Weber, B. A. Salatin, G. G. Grindle, H. Wang, J. J. Vazquez, and R. A. Cooper, "The role of assistive robotics in the lives of persons with disability," *American Journal of Physical Medicine & Rehabilitation*, vol. 89, no. 6, pp. 509–521, 2010.

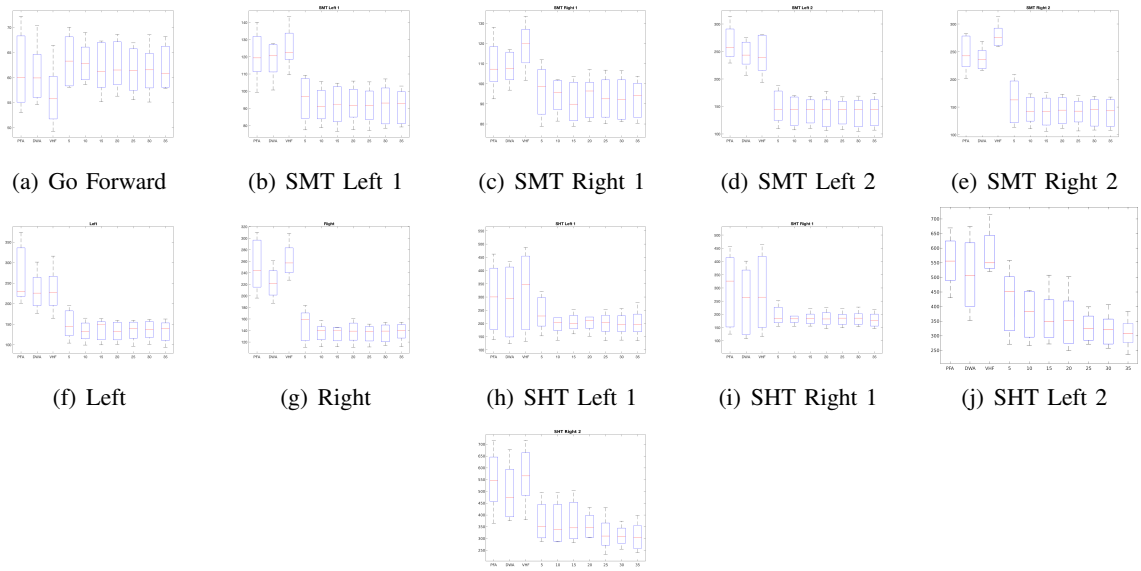
[3] S. Parikh, V. Grassi, V. Kumar, and J. Okamoto, "Usability study of a control framework for an intelligent wheelchair," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 4745–4750.

[4] S. McLachlan, J. Arblaster, D. Liu, J. Miro, and L. Chenoweth, "A multi-stage shared control method for an intelligent mobility assistant," in *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*. IEEE, 2005, pp. 426–429.

[5] D. Bruemmer, D. Few, R. Boring, J. Marble, M. Walton, and C. Nielsen, "Shared understanding for collaborative control," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 35, no. 4, pp. 494–504, 2005.

[6] R. Rao, K. Conn, S. Jung, J. Katupitiya, T. Kientz, V. Kumar, J. Ostrowski, S. Patel, and C. Taylor, "Human robot interaction: application to smart wheelchairs," *Departmental Papers (MEAM)*, p. 29, 2002.

[7] T. Carlson and J. Millán, "Brain-controlled wheelchairs: a robotic architecture," *IEEE Robotics and Automation Magazine*, vol. 20, no. EPFL-ARTICLE-181698, pp. 65–73, 2013.



(a) Go Forward (b) SMT Left 1 (c) SMT Right 1 (d) SMT Left 2 (e) SMT Right 2

(f) Left (g) Right (h) SHT Left 1 (i) SHT Right 1 (j) SHT Left 2

(k) SHT Right 2

Figure 20. Learning curve. Number of volunteers (5,10,15,20,25,30,35) vs error (DTW distances).


[8] C. Urdiales, M. Fernández-Carmona, J. Peula, R. Annicchiaricco, F. Sandoval, and C. Caltagirone, "Efficiency based modulation for wheelchair driving collaborative control," in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 199–204.

[9] C. Urdiales, J. Peula, C. Barrué, E. Pérez, I. Sánchez-Tato, J. del Toro, U. Cortés, F. Sandoval, R. Annicchiarico, and C. Caltagirone, "A new collaborative-shared control strategy for continuous elder/robot assisted navigation," *Gerontechnology*, vol. 7, no. 2, p. 229, 2008.

[10] J. Rowe, "The management of falls in older people: from research to practice," *Reviews in Clinical Gerontology*, vol. 10, no. 04, pp. 397–406, 2000.

[11] Q. Zeng, C. L. Teo, B. Rebsamen, and E. Burdet, "A collaborative wheelchair system," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 16, no. 2, pp. 161–170, 2008.

[12] L. Zeng and G. M. Bone, "Mobile robot collision avoidance in human environments," *International Journal of Advanced Robotic Systems*, vol. 10, no. 41, 2013.

[13] R. Chipalkatty, G. Droge, and M. B. Egerstedt, "Less is more: Mixed-initiative model-predictive control with human inputs," *Robotics, IEEE Transactions on*, vol. 29, no. 3, pp. 695–703, 2013.

[14] Y. Hwang and N. Ahuja, "A potential field approach to path planning," *Robotics and Automation, IEEE Transactions on*, vol. 8, no. 1, pp. 23–32, 1992.

[15] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[16] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 3, pp. 278–288, 1991.

[17] S. Zhu and D. Levinson, "Do people use the shortest path? an empirical test of wardrop's first principle," in *91th annual meeting of the Transportation Research Board, Washington*, vol. 8. Citeseer, 2010.

[18] C. Urdiales, J. Peula, M. Fdez-Carmona, C. Barrue, E. Perez, I. Sanchez-Tato, J. del Toro, F. Galluppi, U. Cortes, R. Annichiaricco, C. Caltagirone, and F. Sandoval, "A new multi-criteria optimization strategy for shared control in wheelchair assisted navigation," *Autonomous Robots*, vol. 30, no. 2, pp. 179–197, 2011.

[19] L. Wang, S. X. Yang, and M. Biglarbegian, "Bio-inspired navigation of mobile robots," in *Autonomous and Intelligent Systems*. Springer, 2012, pp. 59–68.

[20] T. W. Manikas, K. Ashenayi, and R. Wainwright, "Genetic algorithms for autonomous robot navigation," *Instrumentation & Measurement Magazine, IEEE*, vol. 10, no. 6, pp. 26–31, 2007.

[21] C. Urdiales, J. Vázquez-Salceda, E. Perez, M. Sànchez-Marrè, and F. Sandoval, "A cbr based pure reactive layer for autonomous robot navigation," in *Proceedings of the 7th IASTED International Conference on Artificial Intelligence and Soft Computing*, 2003, pp. 99–104.

[22] R. Annicchiarico, C. Barrué, T. Benedico, F. Campana, U. Cortés, and A. Martínez-Velasco, "The i-walker: an intelligent pedestrian mobility aid." in *ECAI*, 2008, pp. 708–712.

[23] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009.

[24] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 2432–2437.

[25] J. W. Youdas, B. J. Kotajarvi, D. J. Padgett, and K. R. Kaufman, "Partial weight-bearing gait using conventional assistive devices," *Archives of physical medicine and rehabilitation*, vol. 86, no. 3, pp. 394–398, 2005.

[26] F. W. Van Hook, D. Demonbreun, and B. D. Weiss, "Ambulatory devices for chronic gait disorders in the elderly." *American family physician*, vol. 67, no. 8, pp. 1717–1724, 2003.

[27] Y. Zheng, "Trajectory data mining: an overview," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 3, p. 29, 2015.

[28] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Robotics*

*and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 1398–1404.

[29] A. Jain, M. Murty, and P. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[30] F. Leisch, "A toolbox for k-centroids cluster analysis," *Computational statistics & data analysis*, vol. 51, no. 2, pp. 526–544, 2006.

[31] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, no. 14. California, USA, 1967, pp. 281–297.

[32] B. Desgraupes, "Clustering indices," *University Paris Ouest Lab Modal'X*, 2013.

[33] D. Davies and D. Bouldin, "A cluster separation measure," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 2, pp. 224–227, 1979.

[34] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2, 1995, pp. 1137–1145.

[35] H. Wang, H. Su, K. Zheng, S. Sadiq, and X. Zhou, "An effectiveness study on trajectory similarity measures," in *Proceedings of the Twenty-Fourth Australasian Database Conference-Volume 137*. Australian Computer Society, Inc., 2013, pp. 13–22.

[36] T. Giorgino, "Computing and visualizing dynamic time warping alignments in r: the dtw package," *Journal of statistical Software*, vol. 31, no. 7, pp. 1–24, 2009.

[37] D. F. Bauer, "Constructing confidence sets using rank statistics," *Journal of the American Statistical Association*, vol. 67, no. 339, pp. 687–690, 1972.