



Reinforcement Learning Aided Robot-Assisted Navigation: A Utility and RRT Two-Stage Approach

Luís Garrote¹ · João Paulo¹ · Urbano J. Nunes¹

Accepted: 29 August 2019 / Published online: 13 September 2019
© Springer Nature B.V. 2019

Abstract

This work proposes a robot-assisted navigation approach based on user intent adjustment, in the context of robotic walkers. Walkers are prescribed to users with gait disorders so that they can support their body weight on the upper limbs, however, the manipulation of such devices can be cumbersome for some users. Common problems for the users are lack of dexterous upper limb control and visual impairments. These problems can render walkers' users helpless, making them unable to operate these devices effectively and efficiently. We present a new approach to robot-assisted navigation using a utility decision and safety analysis procedure with user intent adjustments learned by reinforcement learning (RL) and supported on a rapidly-exploring random tree inspired algorithm. The proposed approach offers full control of the assistive platform to the user until obstacles are detected. In dangerous scenarios, corrections are computed in order that the assistive platform avoids collisions and follows social norms, effectively guiding the user through the environment while enforcing safer routes. The experimental validation was carried out in a virtual environment and in a real world scenario using a robotic walker built in our lab (ISR-AIWALKER). Experimental results have shown that the proposed approach provides a reliable solution to the robot-assisted navigation of a robotic walker, in particular the use of utility theory to evaluate candidate motions together with a RL model increases the safety of the user's navigation.

Keywords Assistive robotics · Reinforcement learning · Robot-assisted navigation · Robotic walker · Rapidly-exploring random tree · Robot operating system

1 Introduction

The use of assistive platforms such as robotic walkers and wheelchairs has increased in the last years. The increase tendency is directly influenced by demographic statistics which show an accelerated ageing tendency, resulting in increasingly older societies [19]. Adding to this tendency, with the progress of age, several health complications arise. Mobility decay is for instance, one of the most recurrent disorders, mainly due to muscular, neurological, and osteoarticular

decay. New technological approaches that focus on helping sufferers of such disorders have been emerging. Such is the case of Assistive Robotics which aims to provide the means to offer greater degrees of independence and less need for dedicated caregivers. A mobility assistance widely used to provide body weight support while walking is the walker. This type of platform provides a reliable solution in assistive and rehabilitative contexts. The ability to support the body weight contributes to the user's safety while walking, which is of great importance to mitigate a phenomenon exhibited by users of these devices, Fear Of Falling (FOF) [1]. However, these devices require dexterous manipulation, for the user to be able to navigate safely without collisions [2], which, for instance, is difficult for Parkinson's disease patients [5]. Another drawback is that a significant number of elderly users have visual impairments [27], adding an additional degree of stress to handle the walker in order to avoid obstacles, and to constantly correct the trajectory to be followed in order to reach the goal lying further ahead. In this paper, we propose an assisted navigation

✉ Luís Garrote
garrote@isr.uc.pt
João Paulo
jpaulo@isr.uc.pt
Urbano J. Nunes
urbano@isr.uc.pt

¹ Department of Electrical and Computer Engineering, Institute of Systems and Robotics, University of Coimbra, Coimbra, Portugal

approach fusing both user intent and environment perception, to offer a reliable driving experience to the user, providing a good performance navigation through the environment. This approach, which combines the user intent derived from the HMI and a local environment model, was deployed in the ISR-AIWALKER [25]. The approach weights the user intent, taking into consideration the environmental constraints, and induces a safe and smooth navigational behavior. Commands that lead to collisions are iteratively learned and penalized, leading to an accurate user intent translation over time. The main contribution of this work is the robot-assisted navigation method (henceforth named two stage utility theory aided RL and RRT-UURL-RRT) incorporating the user's intent, a local environment model and the RL model's decisions using utility functions and utility theory (Sects. 4.2–4.4). In particular the following contributions are highlighted:

- An RL model that considers a multi-state RL representation in order to fuse three inputs (user's intent, pose and local environment model). The RL model provides an output action representing a correction to the user intent based on the user's navigational behavior (Sect. 4.3). The decision computed by the utility theory is used to update the RL model when the computed control command does not match the selected RL action (Sect. 4.3.4).
- An RRT inspired approach searches for driveable space by exploring and expanding nodes in a tree over the local environment representation, biased towards a given control command hypothesis. An explorability index that aids in the selection of the control command hypothesis is computed from the explored tree nodes (Sect. 4.4).

The experimental validation of the proposed method was carried out with the deployment in the ISR-AIWALKER (with a force-based HMI) in a real-world scenario and in a virtual environment. For the evaluation in the virtual environment the ISR-AIWALKER HMI provides the user's intent that moves a realistic virtual model of the walker. In this paper, Sect. 2 refers to the related work concerning robot-assisted navigation. Section 3 provides a general overview of the ISR-AIWALKER. Section 4 details the development of the robot-assisted navigation approach. The following section presents experimental results, and respective discussion, obtained using the test platform. Final conclusions are drawn in Sect. 6.

2 Related Work

Looking at the aspects of motion planning or obstacle avoidance in the literature for assistive platforms, three main categories concerning the type of navigation employed can be listed (see examples at [21]): direct user control, autonomous or destination based control and shared control. In direct user

control the user keeps all the control of the platform with the navigation approach activated only when a collision scenario is detected. With target/destination based control the user provides a goal/destination and the navigation approach deals with all aspects of control. Finally, in shared control, the navigation approach computes which control commands should be applied at each iteration while the user keeps partial control of the platform.

Through the years, several works can be found which address navigation for assistive platforms, particularly robotic walkers, following the aforementioned motion planning categories. This related-work section focuses on works addressing robot-assisted navigation relying on learning schemes to take into account the user's driving behavior. To provide some historic context, some works relying on simple navigation approaches will be mentioned. The PAM-AID walker is the pioneer work in this field [15], designed to increase the independence of persons with visual impairment and/or reduced mobility. It is a passive walker, providing solely steering control of the front wheel. It boasts a frontal ultrasonic array, which serves for detection and avoidance of obstacles. Another example is the PAMM smart walker [7], which employs motorized locomotion, as well as, a sonar array and a camera. Taking into account a map, the walker follows visual landmarks and avoids obstacles using the sonar. User's commands/intention, on both works, is provided through a force/torque sensor on the handles. Assistance is limited to an obstacle avoidance perspective or to a navigation on a structured and mapped environment. Following this initial paradigm, there are, as well, the robotic walkers described in [4,17,22,23]. More complex navigation approaches have been proposed focusing on context-awareness, integrating simultaneous localization and mapping (SLAM) techniques. The mobile walkers Guido [29], ARSO [34], and the one described in [32] are examples of devices integrating that type of context-aware navigational modules.

However, all the previous navigation approaches still consider the user's intention as merely a destination goal. They navigate, knowing the goal *a priori*. The work described in [33] combines two sources of user intent input, direct force-based input, and estimation of direction and orientation through a laser-range finder. In [11], a model for moderating user intention and user fatigue is proposed that takes human input into account in three different decision-making algorithms, related to path planning, collision avoidance, and the degradation of user-based commands due to physical fatigue. An approach relying on drift–diffusion model is used, combining user intention and the robot's perception. A system of rewards is given to each action ruling the decision-making. It showed better performance than simpler models. The drift–diffusion model was tested in [24], where two different approaches are tested for mobile assistive robots. The other method is based on the Dynamic Window Approach

Table 1 Summary of robot-assisted navigation approaches related to walker platforms in the literature

References	Navigation approach	Sensors	Environment perception
[7]	Shared control approach	Sonars; camera	A priori map
[22]	Policy search	Sonars	A priori map
[4]	Walker dynamics	Laser rangefinder	Obstacles
[23]	Context-aware	Laser rangefinder	Road context
[32]	Lookahead controller	Tilting laser rangefinder	A priori map
[34]	A*, Nearness diagram	Camera; Sonars; Laser rangefinder	A priori map
[29]	Clean sweep	Sonars; Laser rangefinder	Obstacle projection
[12]	Q-Learning	Sonars	Obstacles
[24]	Drift–diffusion model, DWA	Laser rangefinder	Obstacles
[18]	A*, Hybrid DWA	Laser rangefinder	Grid map (SLAM)
[9]	RLNA	Microsoft Kinect	Local map [10]

(DWA), and a motion command for the platform is computed upon the user intent's input. The results of the tests showed an improvement in navigation efficiency of both methods when compared to unassisted mobile platforms. Still in an assistive context, a Double DWA (D-DWA) approach was proposed [18] that provides smooth movements and effective traversal of tight spaces. In [8], an approach for assisted navigation of visually impaired individuals through a Simulated Passivity approach of the FriWalk is presented. It provides a graphical user interface that provides indications to the user about the path and only activates the walker's motors when a compensation is required, hence offering a perception of a passive walker.

On the scope of social behavior, related to assistive platforms, interesting works in that particular domain can be found [26,31]. In [31] the robot avoids interfering with humans by considering a dynamic social zone integrated in the motion planner that models social interactions. On the other hand, the work in [26] uses inverse RL to teach social behaviors to a robot, by inputting the learning outputs to an optimal rapidly-expanding random tree planner. The inverse RL model is achieved through demonstrations.

An important aspect for assistive platforms is learning from experience, more particularly, learning from user's actions. To this end, RL has been applied in multiple scenarios [14], and one of its growing fields of application lies in the process of aiding the navigation of robotic systems. In [12] RL is used by a robotic platform to learn the best navigation behaviors when confronted with obstacles in a closed course. The learned Q-values are used to moderate the actions of the fuzzy-logic controller that initially defines the platform's behavior based on the presence and location of obstacles. RL can also be used to assist a robotic platform without learning precedent conditions, based solely on rewards attributed recursively for each state-action pair. An example of this type of approaches can be seen in [6], where

a navigation algorithm employs RL to learn a model using a Q-matrix that acts upon a Fuzzy Neural Network-based Action-State space. In [9], a walker's motion planning is carried out by a modified DWA where the DWA's cost function is aided by an RL model in order to incorporate the learned user's behavior [reinforcement learning robot-assisted navigation approach (RLNA)].

Considering the aforementioned works, a summary of robot-assisted motion planning approaches, in the context of walker platforms, is provided in Table 1.

Our proposed robot-assisted navigation approach, belongs simultaneously to the direct user control and the shared control categories of motion planning approaches. The method provides assistance without requiring a global map or a global plan of the user's actions. Using a local map of the environment, the user can make local decisions and influence the walker's motion, but when a possible collision in the user's pathway is detected, the motion of the walker is adapted using the proposed method. Given its two-stage nature, the method helps to avoid local minimum scenarios, complies with social behavior, learns from user's behaviors and moves the assistive platform smoothly in the environment.

3 Robotic Assistance Architecture

In this section, we present an overview of the proposed robotic assistance architecture (shown in Fig. 1), with particular focus on the robotic walker and its user intent and environment models. The robot-assisted navigation approach is the main contribution of this work and is detailed in the next section.

3.1 Robotic Walker

The ISR-AIWALKER is a differential motorized robotic platform with a force-based HMI (see Fig. 2). The platform was

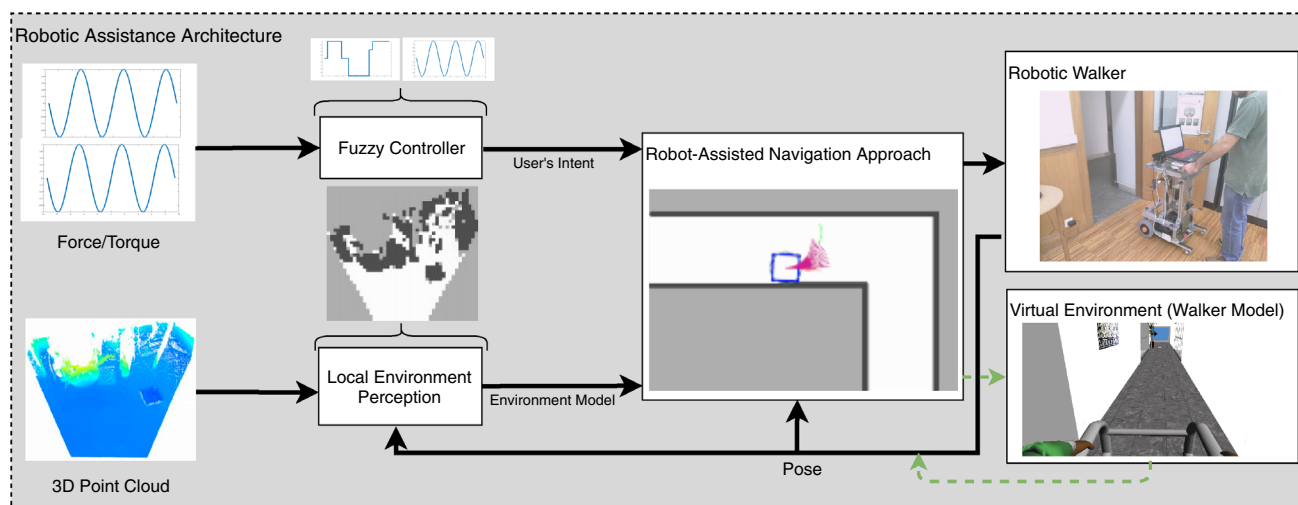


Fig. 1 Block diagram of the overall robotic assistance architecture, composed by: the proposed robot-assisted navigation approach; a local environment perception module; a Fuzzy module and interfaces with the walker platform and a virtual walker. The inputs of the robot-assisted

navigation approach are the user's intent and an environment model, the output is the control command for the walker platform or for the walker model in the virtual environment

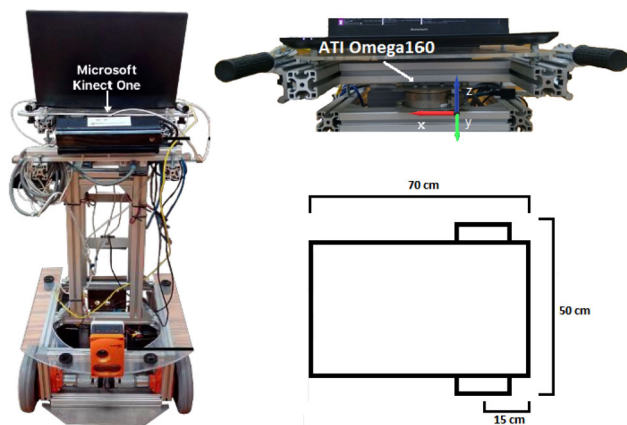


Fig. 2 ISR-AIWALKER robotic walker platform

designed and built at the ISR-UC and is able to sustain a nominal vertical load of up to 80 kg (a more in-depth description of the platform is presented in [9,25]). The walker is also equipped, for tasks related to environment perception and motion planning, with a Microsoft's Kinect One and a Leddartech Leddar sensor. Addressing the HMI's design, which is of importance to this work, the HMI's mechanical structure follows the typical frame of the walkers, used in the healthcare field, i.e. a configuration of parallel handles. It employs an ATI Omega160 NetBox Force/Torque sensor with a fuzzy controller that converts the forces exerted by the user on the handles into linear and angular speed commands.

3.2 Virtual Environment

For this work a simulation platform was designed to test the proposed robot-assisted navigation approach. Using the

ISR-AIWALKER's HMI feedback, a realistic model of the walker is guided in a virtual environment. The walker's virtual model was obtained by modeling the step response of the ISR-AIWALKER motors (first order transfer functions) with the encoders and dimensions accurately represented in the virtual model. This simulation environment helps to broadly assess the proposed pipeline (i.e., with a large number of volunteers) guaranteeing the safety of all volunteers. The simulation environment architecture is presented in Fig. 3. The virtual environment simulation software was developed in OpenGL/Qt/C++ with a TCP/IP connection to the ISR-AIWALKER's HMI (see Fig. 3). With the use of such a hybrid setup, we are able to ensure that volunteers can operate the physical HMI and, at the same time, remain safe, without the need to expose them to real collisions during tests (see Fig. 4). The user's input (through the HMI) serves as input for the robot-assisted navigation approach. Using a virtual representation of the scenario, a virtual 3D point cloud sensor generates 3D point cloud data. The generated data is used to compute the local map and detect collisions. The ISR-AIWALKER motion is computed by a differential drive model.

3.3 User's Intent

The desired walker's motion is defined by a control command u_{ui} (i.e., user's intent) composed by linear v and angular w speed components ($u_{ui} = (v, w)$) issued by the HMI. Considering that the typical users of mobility assistances are elderly and/or impaired users, we assume that the user intent provided by the HMIs of these assistive devices is coupled with uncertainty due to inherent constraints (i.e.,

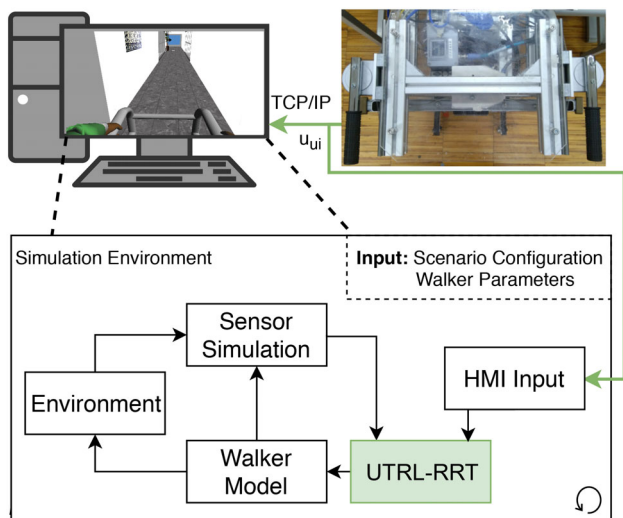


Fig. 3 Diagram representation of the simulation environment with the HMI user intent (u_{ui}) provided by the ISR-AIWALKER



Fig. 4 A volunteer performing a run in the virtual environment using the ISR-AIWALKER’s HMI as input

users with impaired vision, uncoordinated motion and loss of arm strength). This uncertainty on the user intent increases as the user capabilities decrease [2]. The user intent uncertainty is here defined as a region R_u comprised between $[[v_{ui} - \Delta v_{ui}, v_{ui} + \Delta v_{ui}], [w_{ui} - \Delta w_{ui}, w_{ui} + \Delta w_{ui}]]$ in a continuous velocity space. The user intent provided by the HMI is constrained in a continuous space limited by the upper and lower system boundaries ($R_r = [[v_{min}, v_{max}], [w_{min}, w_{max}]]$ and $R_u \in R_r$).

3.4 Local Environment Perception

The local environment model M_{local} is defined as a two-dimensional (2D) environment representation ($n_{rows} \times n_{cols}$), composed by a set of cells $M_{local} = c_{ij}$ with indexes i and j corresponding to row and column, respectively. Each cell c_{ij} is defined with constant dimensions and contains a probabilistic occupancy that represents to some degree the existence of some obstacle on the environment and is updated from sensor measurements. The environment is represented

on a local view centered on the walker’s base frame and is used to evaluate possible collisions and detection of avoidable obstacles. Each cell c_{ij} in M_{local} is converted to a binary notation (free or occupied) to aid the collision detection and planning. On the context of this work, the approach proposed in [10] is used (see Fig. 5). The input is a 3D point cloud provided by the Microsoft Kinect One sensor. For each 3D point cloud, an intermediary 2.5D representation is generated and a ground plane is extracted using an RRT inspired Ground-Plane Detection approach (RRT-GPD). Using an inverse sensor model and the obtained ground plane, each 2.5D voxel is converted to an occupancy probability and updated in the local environment model (M_{local}). The local environment model is centered on the walker’s base frame with constant grid size. The walker’s displacement and orientation are required to update the cells in the local representation, and orientation is required for collision detection. The walker’s pose in the local environment model will be $(0, 0, \theta)$ at any instant, where θ is the orientation of the platform.

4 Robot-Assisted Navigation Approach

In this section, we introduce our assisted navigation approach, which includes a utility based decision step with RL and a safety stage inspired by a RRT algorithm (UTRL-RRT). The proposed approach is the result of the combination of several modules summarized in Fig. 6. At every iteration, the user’s intent is derived from the HMI present on the ISR-AIWALKER. Using the local environment model, the method analyses the environment and the hypothesis (candidates) for safe navigation. If the user’s intent represents a safe command, no assistance is provided (green arrow in Fig. 6). On the other hand, if the user’s intent represents an unsafe command, the proposed approach will provide an adequate linear and angular speed to preserve safety, resolve the detected conflict or immobilize the platform if no valid solution is found. The proposed approach is presented in Algorithm 1.

4.1 Scenario Assistance Awareness

The platform’s user maintains most of the motion control duties and is aided when a critical situation is detected based on the provided user intent u_{ui} . The proposed method has three main input variables: the user intent; a local environment model and the walker’s pose. We consider the motion model of the robotic walker platform as being based on the differential drive geometry kinematic model, given by

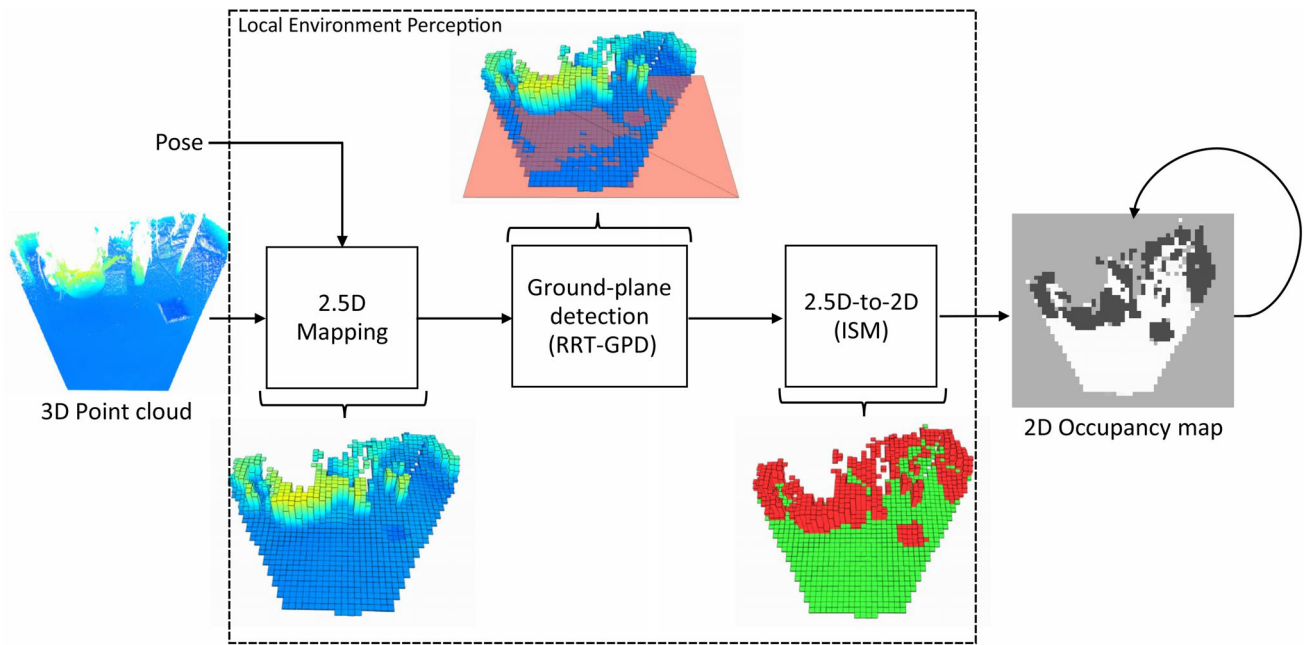


Fig. 5 Local environment perception approach from [10]. The approach expects as input a 3D point cloud and outputs a local 2D Occupancy gridmap

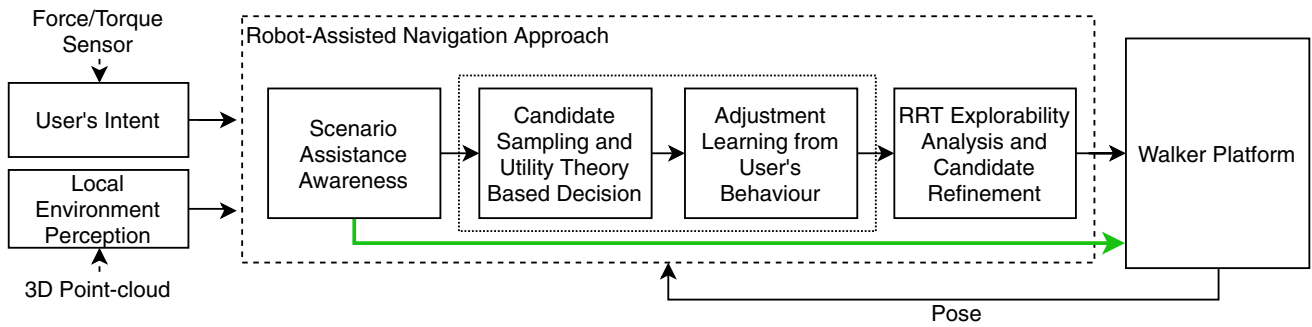


Fig. 6 Robot-assisted navigation’s functional diagram. (Color figure online)

$$x_{k+1}(u(v, w)) = \begin{cases} x_{k+1} = x_k + v\Delta t \cos \theta_k \\ y_{k+1} = y_k + v\Delta t \sin \theta_k \\ \theta_{k+1} = \theta_k + w\Delta t \end{cases} \quad (1)$$

where (x, y) is the 2D position of the walker’s center of mass, Δt the time step and θ is the walker’s orientation. For a given user intent u_{ui} a set of velocity samples is taken uniformly from R_u and paths for each sample are generated by the computation of Eq. 1 with a predefined number of lookahead iterations. If all of those samples produce a valid path (**ValidRequest** in Algorithm 1), the velocity command is sent to the walker platform. Otherwise, if one or more paths fail either because a collision was detected or a generated path provided a solution that was too close to an occupied cell in M_{local} than a safety threshold, the method tries to find a solution that respects the user intent but drives the walker along the surrounding environment. The procedure

ValidRequest, using the user’s intent uncertainty region R_u , computes two borderline linear and angular commands and assesses possible collisions in an iterative procedure using Eq. 1, with a predefined lookahead and a local representation for collision computation.

4.2 Candidate Sampling and Utility Theory Based Decision

Given the user intent u_{ui} , a set of velocity samples S_v with size (N) is uniformly sampled within the region R_r (**CandidateSampling** in Algorithm 1). For each velocity $u_c \in S_v$, using Eq. 1, we determine a set of poses (S_p) in the 2D-Cartesian space (here defined as motion candidate) with each pose defined as $p_i = (x, y, \theta)$ (for $i = 0, \dots, n$), poses generated from $t = 0$, to $t = T_{LA}$ with T_{LA} , the lookahead step of the candidate. The T_{LA} represents how far

Algorithm 1: Proposed approach for robot-assisted navigation

```

Input: User intention ( $u_{ui}$ )
         Local environment model ( $M_{local}$ )
         Candidate iteration time ( $T_{LA}$ )
         Auxiliary parameters/variables
         ( $t_r, K, A_{k-1}, S_{k-1}$ )
1 if ValidRequest( $M_{local}, u_{ui}, T_{LA}$ ) then
2    $u' \leftarrow u_{ui}$ ;
3 else
4    $S_v \leftarrow$  CandidateSampling( $u_{ui}$ );
5    $S_c \leftarrow$  CandidateGeneration( $M_{local}, S_v, T_{LA}$ );
6    $S_c \leftarrow$  ComputeUMetrics( $S_c$ );
7    $S_c \leftarrow \emptyset$ ;
8    $a \leftarrow$  MultiStateRL( $M_{local}, u_{ui}, A_{k-1}, S_{k-1}$ );
9   foreach  $x$  in  $S_c$  do
10     $S_c \leftarrow S_c \cup x \wedge$  ActionUtility( $a, x$ );
11   $S_c \leftarrow$  SelectBestUtilitySet( $S_c$ );
12   $S_e \leftarrow \emptyset$ ;
13  foreach  $x$  in  $S_c$  do
14     $i_e \leftarrow$  RRT-E( $x, u_{ui}, t_r, K, M_{local}$ );
15     $S_e \leftarrow S_e \cup i_e$ ;
16   $u' \leftarrow S_v \cap S_c \cap \arg \max_{i_e \in S_e}$ ;
17  RLActionUpdate( $\|u_{ui} - u'\|$ );
Output:  $u'$ 

```

the system should iterate the equation in order to provide a reliable candidate. Defining S_c as the set of candidates, we use the local environment model M_{local} to perform collision checking and exclude invalid nodes (**CandidateGeneration** in Algorithm 1). Given a valid set of candidates S_c , we start by extracting characteristics for each candidate:

1. Candidate traveled distance:

$$d_t = \sum_{i=1}^{|S_p|-1} \|(S_p(i) - S_p(i + 1))\| \tag{2}$$

2. User intent to candidate velocity (u_c) norm:

$$d_{ui} = \|(u_c - u_{ui})\| \tag{3}$$

3. Minimum distance to obstacles on the left (d_l) and right (d_r) side of the candidate c in the local map M_{local} :

$$(d_l, d_r) = \text{LocalDistance}(c, M_{local}) \tag{4}$$

Equations 2–4, provide a description in terms of length, similarity to the requested user intent, and distance to near obstacles considering the left (d_l) and right (d_r) side of the platform. From these equations, and based on utility theory [3,28], we define three utility functions that best describe how useful each characteristic is:

1. Obstacle avoidance utility function: Obstacle avoidance is an important part of any motion problem, but as robots enter in human populated environments they are required to take into account social conventions [13] (e.g., tend to one side of hallways, respect personal space, do not interrupt conversations). In this case, we incorporate the social behavior of approaching to the nearest walls, in either direction in order to leave a free pathway. Equation 5 provides such behavior as utility is modeled through two Gaussian functions with the center of each peak the higher utility value, conditioned by the nearest obstacle position

$$U_O(d_l, d_r) = \begin{cases} e^{-\left(\frac{(d_l - \mu_o)^2}{2\sigma_o^2}\right)}, & \text{if } d_l \leq d_r \wedge d_l < d_o^{max} \\ e^{-\left(\frac{(d_r - \mu_o)^2}{2\sigma_o^2}\right)}, & \text{if } d_r < d_l \wedge d_r < d_o^{max} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

with μ_o the defined useful distance, σ_o the useful width and d_o^{max} a threshold value for the distance to obstacles.

2. User intent-candidate utility function: The user intent u_{ui} given by the HMI represents the user’s intent coupled with uncertainty. Although this uncertainty could be troublesome, in cases where the user is correct in the scenario assessment, this utility function guarantees that the initial intent is taken into account in the decision of the utility of a candidate, given by:

$$U_{ui}(d_{ui}) = e^{-\left(\frac{(d_{ui})^2}{2\sigma_{ui}^2}\right)} \tag{6}$$

with σ_{ui} the Gaussian width of the user intent utility.

3. Traveled distance utility function: On the set S_c , and since we are employing a constant lookahead step T_{LA} , depending on the input u_c for each candidate, the traveled distance for each candidate will be different, so in order to reward candidates that maximize the exploration of wide areas of the environment model, the utility function increases as the traveled distance increases, so that

$$U_t(d_t) = \frac{d_t}{d_{max}} \tag{7}$$

with d_{max} the maximum distance threshold.

From [20] the total utility (**ComputeUMetrics** in Algorithm 1) of a candidate c is now

$$U_c = W_O U_O + W_{ui} U_{ui} + W_t U_t \tag{8}$$

with W_O , W_{ui} and W_I being the behavior dependent weights. The candidate with maximum utility c^* is then given by

$$c^* = \operatorname{argmax}_{c \in S_c} (U(c)) \quad (9)$$

with $U(c)$ the set of utility values for each candidate ($U(c) = [U_{c0}, \dots, U_{cN}]$). From Eq. 9, a set S_C of size $|S_C| = N_L$ candidates that maximize the utility of being a solution, is selected (**SelectBestUtilitySet** in Algorithm 1). The aim in choosing more than one candidate when possible (i.e., depending on the scenario), is to provide a second layer of safety. This enables the prevention of possible complex situations that may arise, such as local minimum outside of the initial system's lookahead that are not modeled in the utility function $U(c)$, as well as to avoid scenarios where the platform would face inevitable collisions, on the other hand, reducing the number of candidates also reduces the computational burden introduced by further exploring each candidate.

4.3 Adjustment Learning from User's Behavior

Although a local motion solution can be found using the aforementioned approach, learning from the actions performed by the user may aid in providing timely corrections. Users of devices such as walkers or wheelchairs can suffer from multiple overlapping symptoms which may require a modified behavior of the local strategy in order to comply with the user's true intent [30]. Following the work proposed in [9], a solution inspired by Q-Learning is proposed. In the classic Q-Learning model, the inputs are the states (S) and actions (A), and the output is the action that maximizes the utility/reward of taking an action given an observed state. Our approach follows a different procedure where, it tries to model multiple observations of states in order to maximize the utility of a chosen action. The states are modeled as obstacles surrounding the assistive platform, and the actions represent corrections that should be applied to the user's intent.

4.3.1 Multi-State Q-Learning Approach

The proposed RL approach is presented in Algorithm 2. For this particular problem the Q matrix is modeled as a 3D matrix where rows and columns represent X and Y coordinates in the assistive platform's local workspace and height cells represent the Q-values for each action. The Q matrix is initialized with constant bias towards a non correcting action, with dimensions $|S| \times |S| \times |A|$.

For each update of the RL model, the first step consists in getting the current observed states by iterating Eq. 5 (**CurrentStates**), given the user intent, pose and the envi-

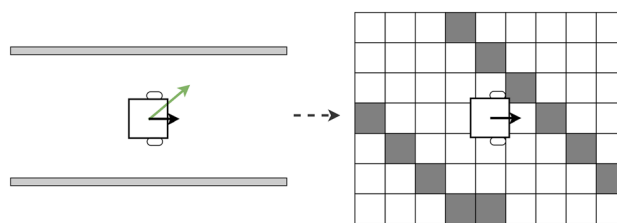


Fig. 7 Representation of the definition of states from an environment snapshot, considering the user's intent (green). (Color figure online)

ronment representation (see Fig. 7). Modelling all possible combinations of obstacles is not a tractable problem in this context, and in this work we propose an approximation to model states using the obstacles in the local environment model. For the current set of observations S_k a set of unique possible actions is obtained (**UniqueActionsForStates**). This set of actions represent all possible actions that the platform is allowed to perform in the current state. The reward R is computed by minimizing the reward value for each state in S_k (**RewardForState**) considering Eq. 11. For each state s in S_{k-1} , using a nearest neighbour approach (**NearestState**), we find the nearest state in S_k , and retrieve its maximum Q-value (Q_{S_x}). Using s and the previous action A_{k-1} , the Q-value is updated ($Q(s, A_{k-1})$). A set containing all possible output actions (A_k^*) is updated with the action that maximizes the Q-value for state s . A value corresponding to the weight of the observation s is also computed. This weight represents the contribution of a given state to the final output action of the RL model (**StateWeight**). The weight, considering the observed reward (r) for s , is given by

$$w(r) = \begin{cases} K_w ||r||, & \text{if } r < 0 \\ 1, & \text{otherwise} \end{cases} \quad (10)$$

where K_w is a weight gain. The output action is computed using a weighted histogram (**WeightedHistogram**). The histogram is computed considering all the actions in A_k^* where in each action's bin (in a histogram sense), instead of adding one unit to the bin, the bin is incremented with the weight for each action. The output is then computed by finding the bin/action that maximizes the weight.

4.3.2 States

The input of the proposed robot-assisted navigation approach is a local environment model which means that the robot is centered in the representation, and the representation contains data from the nearest obstacles. In order to apply the RL approach a view of the representation must be considered when taking into account the desired intent of the user. From Eq. 1 a lookahead simulation is performed by considering

Algorithm 2: MultiStateRL - Q-Learning algorithm adapted for user intent adjustment

Input: Environment Model (M)
 User Intent (u_{ui})
 Last Action (A_{k-1})
 Last set of States (S_{k-1})

```

1 Initialization:
2  $Q \leftarrow A_{(|S| \times |S|)}; A \leftarrow [0 \dots 1 \dots 0]$ ;
3 Update Loop:
4  $S_k \leftarrow \text{CurrentStates}(M)$ ;
5  $A' \leftarrow \text{UniqueActionsForStates}(S_k)$ ;
6  $A_k^* \leftarrow \emptyset$ ;
7  $w_k \leftarrow \emptyset$ ;
8  $R \leftarrow \underset{s \in S_k}{\text{argmin}} \text{RewardForState}(s)$ ;
9 foreach  $s \in S_{k-1}$  do
10    $S_x \leftarrow \text{NearestState}(s, S_k)$ ;
11    $Q_{S_x} \leftarrow \max_{A'}(Q(:, S_x))$ ;
12    $Q(s, A_{k-1}) \leftarrow Q(s, A_{k-1}) + \alpha(R + Q_{S_x} - Q(s, A_{k-1}))$ ;
13    $A_k^* \leftarrow A_k^* \cup \underset{A'}{\text{argmax}}(Q(A', S_{k-1}) \wedge Q(A', S_{k-1}) > 0)$ ;
14    $w \leftarrow w \cup \text{StateWeight}(R, s)$ ;
15  $A_k \leftarrow \text{WeightedHistogram}(w, A_k^*)$ ;
16  $S_{k-1} \leftarrow S_k$ ;
17  $A_{k-1} \leftarrow A_k$ ;
Output:  $S_k, A_k$ 
    
```

the user’s intent (u_{ui}). From the final position of the simulation, the nearest obstacles are extracted using an occupancy threshold and rotated considering the final orientation caused by the user’s intent. The states represent a lookahead view of the representation provided the platform executed the issued user’s intent. A depiction of this procedure is shown in Fig. 7.

4.3.3 Rewards

The rewards were designed in order to reward motions that do not collide with the observed states. In particular, states close to the platform contribute with a critical negative reward in order to signal an unwanted maneuver. Rewards are decreased from those critical negative rewards to positive rewards considering the type of maneuvers executed by the assistive platform (see Fig. 8). In the lateral regions the rewards increase faster in order to not penalize motions that follow walls or traverse doors. In the frontal area it increases slowly and spans a wide area to avoid frontal collisions, the same idea is applied to the rear region in order to protect the walker’s user. The span of the longitudinal region increases and decreases according to the walker’s linear speed. The reward is given by

$$R(x, y, b, v) = K_b - K_r e^{\left(-\frac{x^2}{2(K_x v)^2} + \frac{y^2}{2(K_y b)^2}\right)} \tag{11}$$

where b is the baseline distance of the walker platform, v is the walker’s speed, K_x, K_r, K_b and K_y scaling factors and

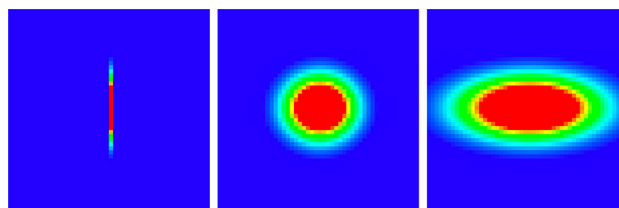


Fig. 8 Three examples of the rewards based on a likelihood of collision. From red to blue, the rewards are represented where red represents a negative reward and blue the maximum positive reward. From left to right: reward for a stationary platform; reward for a platform moving at half the maximum speed; and reward for a platform moving at maximum speed. (Color figure online)

(x, y) the positions of the states considering that this reward representation is centered in the walker’s referential.

The Algorithm 2 considers the worst case scenario for the learning approach which means that all Q-values are updated with the lowest reward. It is also possible to compute a reward for each state s in S_k and individually update the Q-values. Due to the nature of the application, a decision was made to force the learning in the worst case (i.e., to guarantee that more maneuvers are avoided) but a more relaxed approach can be obtained by computing a reward for each state s , by removing the minimum computation of R and adding $R \leftarrow \text{RewardForState}(S_x)$ after the computation of the nearest state.

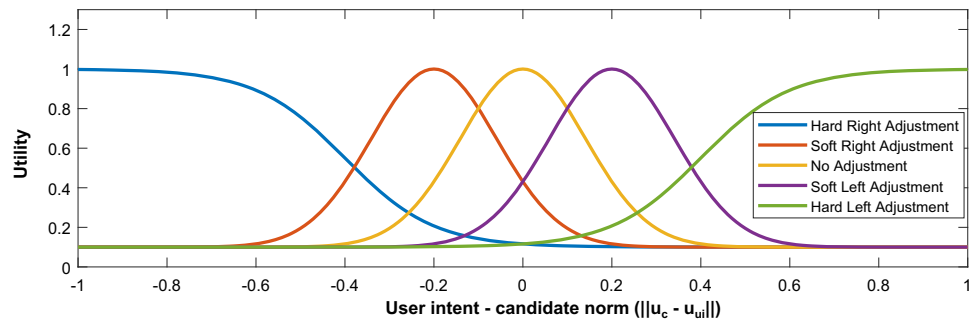
4.3.4 Actions, Action Update and Utility Update

Five possible actions were defined (see Fig. 9) consisting in two big adjustments of the users heading, two small adjustments and one action representing that there is no need to adjust the intent from the user. The actions were defined in order to avoid sudden motion transitions caused by the learned actions in the behavior of the walker, thus creating smooth transitions in the local planning and actuation of the walker’s motors. In order to associate a utility to the RL model output, the five actions were modeled using the form of sigmoidal functions on the edges and Gaussian functions for each action in-between. The equation that represents the computation of the utility (**ActionUtility** in Algorithm 1) is given by:

$$U_{RL}(a, x) = \begin{cases} K_B + \frac{K_U}{(1+e^{(-K_S(-x+A(a)))})}, & \text{if } a = 1 \\ K_B + \frac{K_U}{(1+e^{(-K_S(x-A(a)))})}, & \text{if } a = N \\ K_B + K_U e^{-\left(\frac{(x-A(a))}{\sigma}\right)^2}, & \text{otherwise} \end{cases} \tag{12}$$

where N is the number of actions, K_B is a minimum cost parameter, K_U a scaling parameter, K_S a sigmoid’s shape parameter, σ the Gaussian function shape parameter and $A(\cdot)$ the centroid for each action, considering the variation between the user’s intent and a defined candidate (x).

Fig. 9 Representation of the utility (Eq. 12) for each RL output action



Equation 8 is updated for the RL procedure and is given by:

$$U_c = W_O U_O + W_{ui} U_{ui} + W_t U_t + W_{RL} U_{RL} \tag{13}$$

where W_{RL} is the RL associated weight. As stated earlier, the output of the RL model is computed considering, an weighted histogram instead of other techniques such as ϵ -greedy or replay/preview techniques.

This was mainly due to the nature of the problem of aiding a user while driving an assistive platform. The initial guess, when no prior knowledge of the output of an action or when each action represents a negative outcome, should be handled by the local planner. In this particular scenario, when the user is driving the walker platform, we cannot randomly select an output action by risking the safety of the user. In this context, the best possible solution is to let the local planner select the appropriate decision considering the user intent and the local environment, and then let the model learn from the decisions output. This means that if the Q-value of the bin with maximum weight is negative, no action is issued, and an utility of zero is considered. The output of the RL model is updated after the decision of the motion planning, considering the variation between the user’s intent and the selected candidate. The action is updated (**RLActionUpdate** in Algorithm 1) by

$$S_{k-1} = \operatorname{argmax}_{a=1:N} U_{RL}(a, \|u_{ui} - u'\|) \tag{14}$$

4.4 RRT Explorability Analysis and Candidate Refinement

To increase the method’s safety, we introduce a RRT [16] inspired algorithm to assess the explorability of each candidate on the S_C set (see Algorithm 1). In the scope of this work, we define explorability as an indicator of a candidate’s ability to further generate valid solutions on a determined region.

The RRT algorithm is a sample-based approach that is probabilistically complete and has the advantage of being able to find a feasible path relatively quickly (when a feasi-

Algorithm 3: RRT-E - RRT Explorability index

```

Input: Motion candidate ( $c$ )
        User intent ( $u_{ui}$ )
        Simulation step ( $t_r$ )
        Maximum number of iterations ( $K$ )
        Local environment model ( $M_{local}$ )

1 Initialization:
2 RRT Tree ( $G$ ) with initial configuration;
3 ( $G$ .InitializeTree( $c$ )).
4 InvalidCount  $\leftarrow$  0;

5 for  $k=1$  to  $K$  do
6    $v_{rand} \leftarrow$  SampleRandomVelocity( $u_{ui}$ );
7    $x_{rand} \leftarrow$  SelectRandomExpansion( $G$ );
8    $x_{new} \leftarrow$  Steer( $x_{rand}, v_{rand}, t_r$ );
9   if CollisionFree( $M_{local}, x_{rand}, x_{new}$ ) then
10     $G \leftarrow G \cup (x_{rand}, x_{new})$ ;
11  else
12    InvalidCount  $\leftarrow$  InvalidCount + 1;
13  $i_e \leftarrow \frac{K - \text{InvalidCount}}{K}$ ;
Output:  $i_e$ 
    
```

ble path exists), even in high-dimensional spaces, but is not reliable in finding optimal solutions or in ensuring two equal solutions to the same motion planning problem. In this particular application of the RRT inspired algorithm, the goal is not to obtain a valid path, but to compute an explorability index related to the growth of the RRT. By sampling velocities biased towards the user intent, and evaluating the reach of each node, the RRT explores the surrounding space since the growth of the RRT is only constrained by surrounding obstacles. The proposed variation of the RRT algorithm is presented in Algorithm 3.

For each motion candidate c in the set S_C , the proposed RRT inspired approach is instantiated and iterated K steps. For each iteration, a random velocity biased on the user intent is sampled and used to guide the search component of the algorithm. The proposed algorithm has four main procedures enumerated by the same order of execution, being **SampleRandomVelocity**, **SelectRandomExpansion**, **Steer** and **CollisionFree**. The **SampleRandomVelocity** procedure provides a new velocity sample v_{rand} biased on the user intent u_{ui} or sampled in the R_u

window. In this context, biased on the user intent means that the user intent will be selected for exploration more than any other intent pair, sampled from the R_u window (e.g., considering a bias factor of 30%, for 10 samples, 3 will be the user intent and 7 sampled from the R_u window). The **SelectRandomExpansion** procedure selects a random element x_{rand} on the initial tree to be explored using v_{rand} . To avoid unnecessary explorations, the direction of exploration is included in this procedure guaranteeing that the exploration goes only forward or backward (except for the initial node that allows both directions). The **Steer** procedure computes the new trajectory (and a new node x_{new}) using Eq. 1. The **CollisionFree** procedure uses the local representation M_{local} to assess possible collisions between the proposed trajectory and the environment, using the walker's footprint dimensions. When a collision is detected a counter containing the number of invalid explorations is incremented and the node is blocked for further exploration in **SelectRandomExpansion**. If the trajectory is collision free, the node is added to the tree and can be selected for expansion in its motion direction.

The output of the algorithm is the explorability index given by:

$$i_e = \frac{\#ValidExplorations}{\#ExploredNodes} = \frac{K - InvalidCount}{K} \quad (15)$$

Applying the proposed algorithm to each candidate in the set S_C creates a new set S_e containing the explorability index for each candidate. The candidate that maximizes the utility and explorability is then given by:

$$u' = S_v \cap S_C \cap \arg \max_{i_e \in S_e} (i_e) \quad (16)$$

Scenarios that represent near collisions often provide solutions with a small explorability index for strong user intents. The final solution u' will be selected from a set of candidates which, while maximizing the utility, may represent unwanted solutions due to the small explorability. If a node is selected with an explorability index smaller than a given threshold it is discarded and a warning is triggered. The same behavior is applied in solutions that offer small values of total utility. Solutions with small values of total utility in the set S_C represent situations where the requested user intent cannot be satisfied, either because the region could not be explored or there are many obstacles in the robots pathway.

Triggering a warning forces the walker to stop and the user to be warned that the desired intent is invalid (see **Visual Aid/Debug** in Fig. 11). This action guarantees that in cases where a collision is almost eminent the user still has the ability to rotate the walker and provide an updated intent. The final form of the proposed algorithm for robot-assisted navigation is presented in Algorithm 1 and a representation

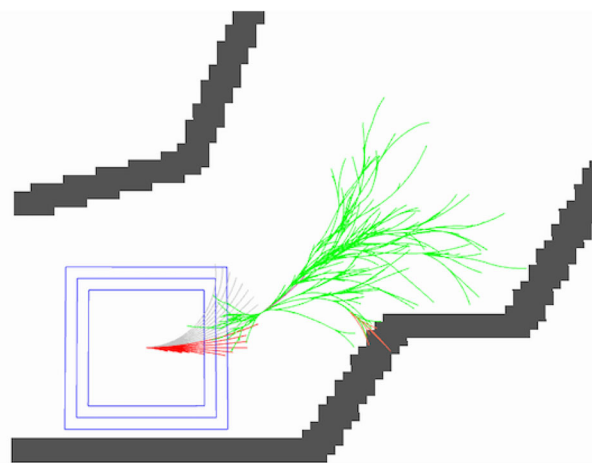


Fig. 10 Representation of the proposed method: in red, excluded nodes; in light grey, valid candidates; in dark grey occupied cells in the local model; and in green, the RRT exploration for one candidate. (Color figure online)

is shown in Fig. 10. It is important to note that the RRT inspired algorithm allows the proposed approach to avoid local minimum obstacle configurations.

5 Experimental Results

5.1 ISR-AIWALKER Software Architecture

In order to deploy the proposed approach in the walker platform, the ROS based software architecture shown in Fig. 11 was developed. The architecture is composed by three main modules; walker interface; local environment perception; and robot-assisted navigation. The walker interface is composed by 3 sub-modules (walker's HMI, sensor acquisition and walker driver) responsible for all sensor acquisition and wheel actuation. The walker's HMI sub-module was developed in-house and connects ATI Omega160 NetBox Force/Torque sensor using TCP/IP and provides a ROS twist message. Sensor acquisition provides the point cloud from a Microsoft Kinect One sensor, based on the available ROS driver.¹ The walker driver implements bidirectional communication between the robotic walker (low level) and the ROS environment providing two topics; a ROS odometry message and a custom message for low level variables (e.g., battery status, encoders). The module also receives a ROS twist message to drive the robotic walker motors. The local environment perception module provides a local representation of the environment taking into account non-trivial obstacles that are in the robot's pathway (e.g., floor outlets, stairs) and

¹ Kinect One driver Package—https://github.com/code-iai/iai_kinect2.

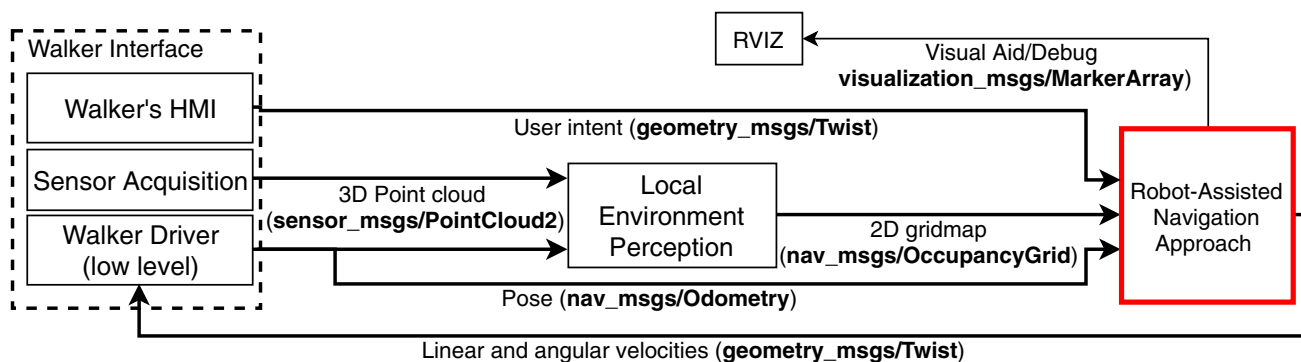


Fig. 11 Deployed ROS based software architecture in the ISR-AIWALKER platform. In bold are highlighted the ROS message types for each module

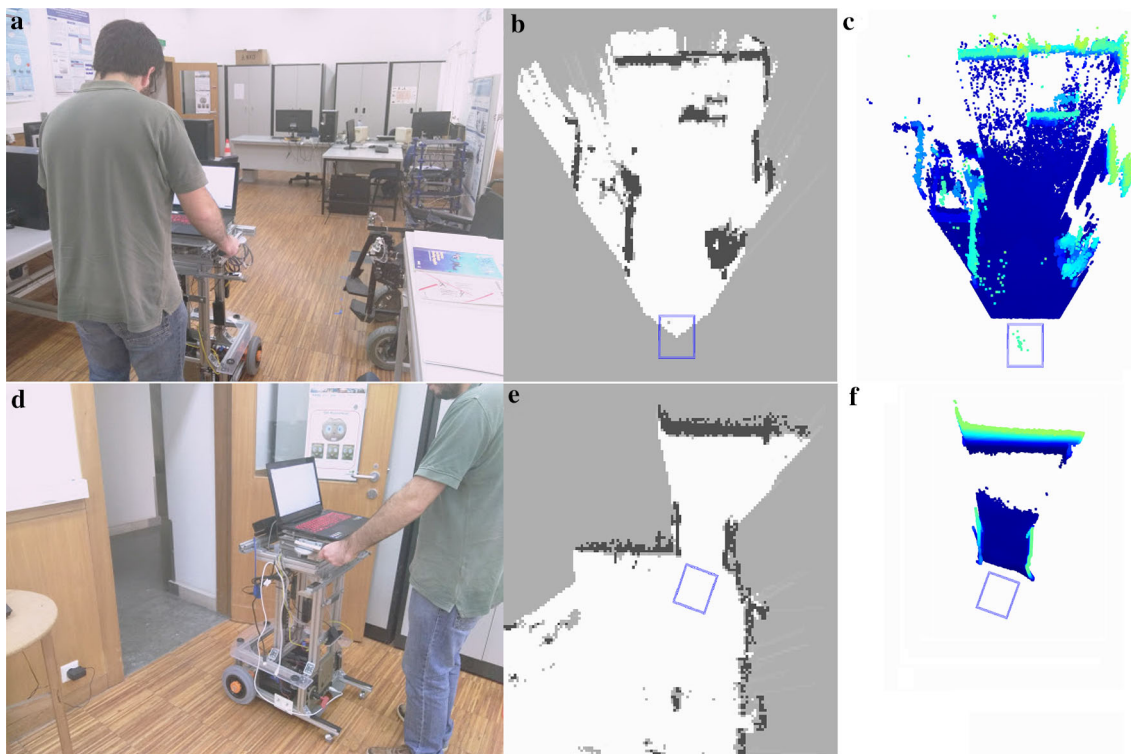


Fig. 12 Snapshots with a user driving the walker, from the office traversal and door entrance (a, d). Images b and e correspond to the local occupancy map for a office traversal and door entrance, c and f to the Microsoft's Kinect One point cloud in both scenes

other obstacles (e.g., chairs, tables, walls). The module provides a local ROS occupancy grid-map message (see [10] for more details). Figure 12 shows a volunteer driving the ISR-AIWALKER platform in two scenarios, as well as the local environment model and 3D point cloud for each scenario. Finally, the robot-assisted navigation module implements the proposed approach (see Fig. 6) and provides a ROS twist message. Table 2 presents the parameters used during the validation process with the proposed robot-assisted navigation module.

5.2 Experimental Protocol

To validate the proposed robot-assisted navigation approach in the ISR-AIWALKER, two sets of tests were defined. The first set of tests were conducted in a virtual environment (see Fig. 13) where a group of twelve volunteers was asked to drive the assistive platform in three modes: no navigational aid (u_{ii} drives the walker platform), proposed approach (UTRL-RRT) and proposed approach without the learning stage (UT-RRT). The virtual scenario was inspired in the real world scenario presented in Fig. 14.

Table 2 Validation parameters

Parameter	Value	Parameter	Value	Parameter	Value
μ_o	0.8	d_{max}	1.0	K	400
σ_o	0.25	W_O	1.0	α	0.5
d_o^{max}	3.0	W_{ui}	1.0	K_b	1.0
N	75	W_t	1.0	K_r	2.0
σ_{ui}	0.25	t_r	0.3	K_x	5.0
b	0.45	K_s	10.0	K_y	1.0
K_B	0.1	K_U	0.9	σ	0.1
W_{RL}	1.0	n_{rows}	300	n_{cols}	300
Δv_{ui}	0.1	Δw_{ui}	0.4	v_{min}	-0.22
v_{max}	0.22	w_{min}	-0.75	w_{max}	0.75
T_{LA}	1.0	Δt	0.1	$A(N = 5)$	-0.5, -0.2, 0, 0.2, 0.5



Fig. 13 Screenshots of the virtual environment

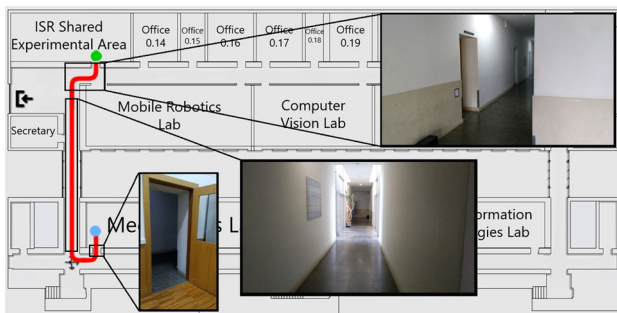


Fig. 14 Scenario for the experimental validation of the proposed approach. In blue the start of the test, in green the end of the test and in red the path each volunteer had to perform. (Color figure online)

In the second set of tests, and considering the scenario presented in Fig. 14, a group of four volunteers was asked to drive the assistive platform with two different navigation

pipelines. Following the work proposed in [9], each one of the volunteers was asked to drive the walker with RNLA and with the proposed UTRL-RRT approach. A new RL model is initialized for each user, guaranteeing that each model only adapts to the user it is learning from. The test consisted on exiting an office environment (through a door) traversing a long corridor with multiple 90° turns and a final step consisting in performing a door entrance (see Fig. 14). It is important to note that in both sets of tests, the same goal and start points are provided (equal in both sets of tests) and the volunteers are expected to perform similar trajectories.

In order to evaluate the performance of the robot-assisted navigation approaches, based on the data gathered from both sets of experimental tests, we consider the following performance indices [9]: the average robot’s speed estimate (U_R), and the average control command and the user’s intent (U_C , U_{UI}); average variation of the robot’s speed estimate (C_R), and average variation of the control command and the user’s intent (C_C , C_{UI}); minimum distance to obstacles (d_o) and test duration time (T_s).

Prior to the start of each test, a familiarization with the system and the objectives of the experiment was performed by each volunteer (for the virtual environment and ISR-AIWALKER). Each volunteer was asked to drive safely while avoiding obstacles and to report scenarios of distress (e.g., stress while performing certain motions or uneasiness with any type of motion). In cases of distress the volunteer would stop the platform and the scenario would be restarted if possible.

5.3 Results in the Virtual Environment

The main results obtained with the experimental tests in the virtual environment are summarized in Table 3 and the performed trajectories are shown in Fig. 15. All the volunteers were able to achieve the goal position with the virtual walker without critical collisions.

Table 3 Performance results of the different navigation methods related to experimental tests in the virtual environment

	Volunteer 1 (V1)			Volunteer 2 (V2)			Volunteer 3 (V3)		
	u_{ui}	UT-RRT	UTRL-RRT	u_{ui}	UT-RRT	UTRL-RRT	u_{ui}	UT-RRT	UTRL-RRT
C_C	0.01 ± 0.01	0.01 ± 0.02	0.02 ± 0.03	0.02 ± 0.04	0.03 ± 0.04	0.02 ± 0.04	0.03 ± 0.04	0.02 ± 0.04	0.02 ± 0.04
C_R	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.02 ± 0.02	0.02 ± 0.02	0.01 ± 0.02	0.02 ± 0.03	0.02 ± 0.02	0.01 ± 0.01
C_{UI}	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.02 ± 0.04	0.03 ± 0.04	0.02 ± 0.02	0.03 ± 0.04	0.02 ± 0.03	0.01 ± 0.02
U_C	0.21 ± 0.03	0.2 ± 0.02	0.19 ± 0.03	0.21 ± 0.06	0.22 ± 0.07	0.22 ± 0.05	0.22 ± 0.06	0.22 ± 0.05	0.22 ± 0.05
U_R	0.21 ± 0.03	0.2 ± 0.02	0.19 ± 0.03	0.21 ± 0.05	0.21 ± 0.06	0.21 ± 0.04	0.22 ± 0.05	0.22 ± 0.05	0.22 ± 0.05
U_{UI}	0.21 ± 0.03	0.2 ± 0.02	0.19 ± 0.03	0.21 ± 0.06	0.22 ± 0.07	0.21 ± 0.04	0.22 ± 0.06	0.22 ± 0.06	0.22 ± 0.05
d_o (m)	0.9 ± 0.27	0.88 ± 0.24	0.9 ± 0.24	0.91 ± 0.29	0.85 ± 0.22	0.95 ± 0.28	0.88 ± 0.24	0.87 ± 0.23	0.92 ± 0.25
T_s (s)	205	217	222	217	224	199	212	206	201
	Volunteer 4 (V4)			Volunteer 5 (V5)			Volunteer 6 (V6)		
	u_{ui}	UT-RRT	UTRL-RRT	u_{ui}	UT-RRT	UTRL-RRT	u_{ui}	UT-RRT	UTRL-RRT
C_C	0.13 ± 0.1	0.13 ± 0.1	0.01 ± 0.03	0.12 ± 0.1	0.12 ± 0.1	0.02 ± 0.05	0.01 ± 0.01	0.01 ± 0.02	0.01 ± 0.01
C_R	0.02 ± 0.01	0.02 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.02	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01
C_{UI}	0.13 ± 0.1	0.13 ± 0.11	0.01 ± 0.01	0.12 ± 0.1	0.12 ± 0.1	0.01 ± 0.02	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01
U_C	0.15 ± 0.1	0.16 ± 0.1	0.21 ± 0.03	0.15 ± 0.1	0.15 ± 0.1	0.21 ± 0.04	0.16 ± 0.05	0.17 ± 0.06	0.19 ± 0.04
U_R	0.17 ± 0.04	0.16 ± 0.03	0.21 ± 0.03	0.16 ± 0.03	0.15 ± 0.03	0.21 ± 0.03	0.15 ± 0.05	0.17 ± 0.06	0.19 ± 0.04
U_{UI}	0.15 ± 0.1	0.15 ± 0.1	0.21 ± 0.03	0.15 ± 0.1	0.15 ± 0.1	0.21 ± 0.04	0.16 ± 0.05	0.17 ± 0.06	0.19 ± 0.04
d_o (m)	0.86 ± 0.19	0.87 ± 0.19	0.93 ± 0.25	0.96 ± 0.24	0.91 ± 0.22	0.94 ± 0.25	1.0 ± 0.33	1.0 ± 0.36	0.9 ± 0.28
T_s (s)	358	388	199	361	395	202	279	258	229
	Volunteer 7 (V7)			Volunteer 8 (V8)			Volunteer 9 (V9)		
	u_{ui}	UT-RRT	UTRL-RRT	u_{ui}	UT-RRT	UTRL-RRT	u_{ui}	UT-RRT	UTRL-RRT
C_C	0.03 ± 0.04	0.02 ± 0.02	0.02 ± 0.05	0.01 ± 0.02	0.02 ± 0.03	0.02 ± 0.04	0.12 ± 0.1	0.11 ± 0.09	0.02 ± 0.04
C_R	0.02 ± 0.02	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01
C_{UI}	0.03 ± 0.04	0.05 ± 0.02	0.01 ± 0.02	0.01 ± 0.02	0.01 ± 0.02	0.01 ± 0.02	0.12 ± 0.1	0.11 ± 0.09	0.01 ± 0.01
U_C	0.13 ± 0.09	0.18 ± 0.05	0.17 ± 0.06	0.2 ± 0.03	0.19 ± 0.03	0.19 ± 0.03	0.15 ± 0.1	0.13 ± 0.09	0.2 ± 0.04
U_R	0.13 ± 0.08	0.18 ± 0.05	0.17 ± 0.05	0.2 ± 0.03	0.19 ± 0.03	0.19 ± 0.03	0.16 ± 0.04	0.14 ± 0.03	0.2 ± 0.04
U_{UI}	0.13 ± 0.09	0.18 ± 0.05	0.17 ± 0.06	0.2 ± 0.03	0.19 ± 0.03	0.19 ± 0.03	0.15 ± 0.1	0.13 ± 0.09	0.2 ± 0.03
d_o (m)	0.7 ± 0.35	0.95 ± 0.3	0.87 ± 0.3	0.96 ± 0.29	0.95 ± 0.26	0.93 ± 0.26	0.87 ± 0.22	0.88 ± 0.2	0.93 ± 0.28
T_s (s)	471	235	245	211	228	227	353	438	215

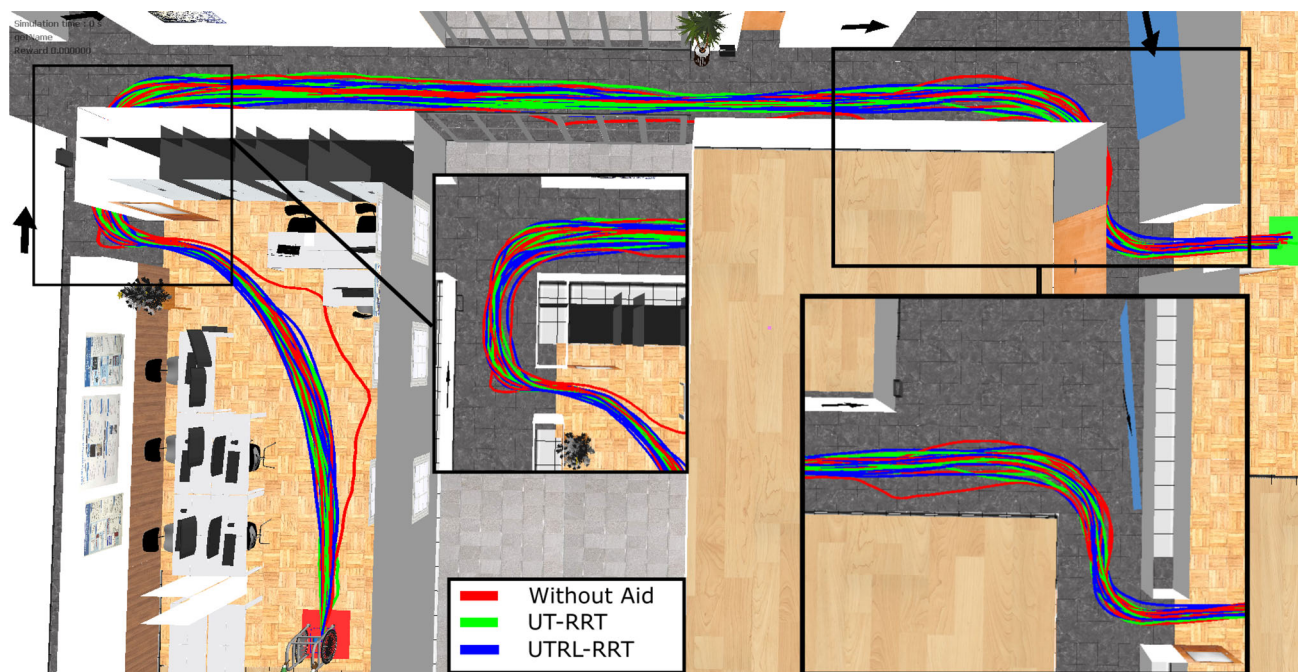


Fig. 15 Overlap of all the geometric paths performed by the volunteers in the virtual environment. In red are the geometric paths obtained without a robot-assisted navigation approach, in green the geometric paths

followed under the proposed approach without the RL model and in blue the geometric paths generated with the proposed approach. (Color figure online)

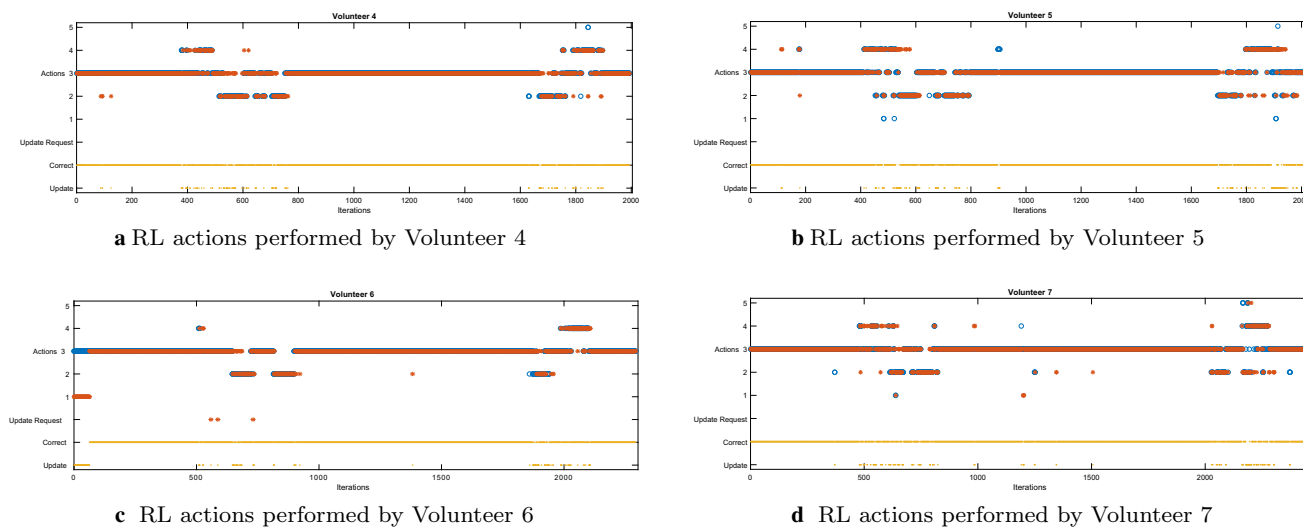


Fig. 16 Snapshot of the performed RL actions, by four volunteers (V4, V5, V6 and V7) in the virtual environment, using the proposed UTRL-RRT approach. In red the RL model actions and in blue the actions taken by the platform (considering Eq. 12). (Color figure online)

more closely follow the user's intent in the UTRL-RRT approach while in the RLNA the variations are higher. The standard deviation for the average control and average variation of control in the proposed approach is slightly smaller for all four volunteers which indicates that the motion transitions are smoother. Due to the utility rule that governs the distance to lateral obstacles and considering the walker's footprint (see Fig. 2) the proposed approach provides, for most volunteers,

the desired wall approaching behavior to leave one way of the pathway free. Due to some difficulties while performing door traversals the average distance to obstacles decreases with the RLNA. With the proposed approach each door traversal was performed without problem. No resets or critical problems were detected in the aforementioned tests.

For the proposed RL approach, in Fig. 18, the results are shown for one of the tests. It is noticeable that for most iter-

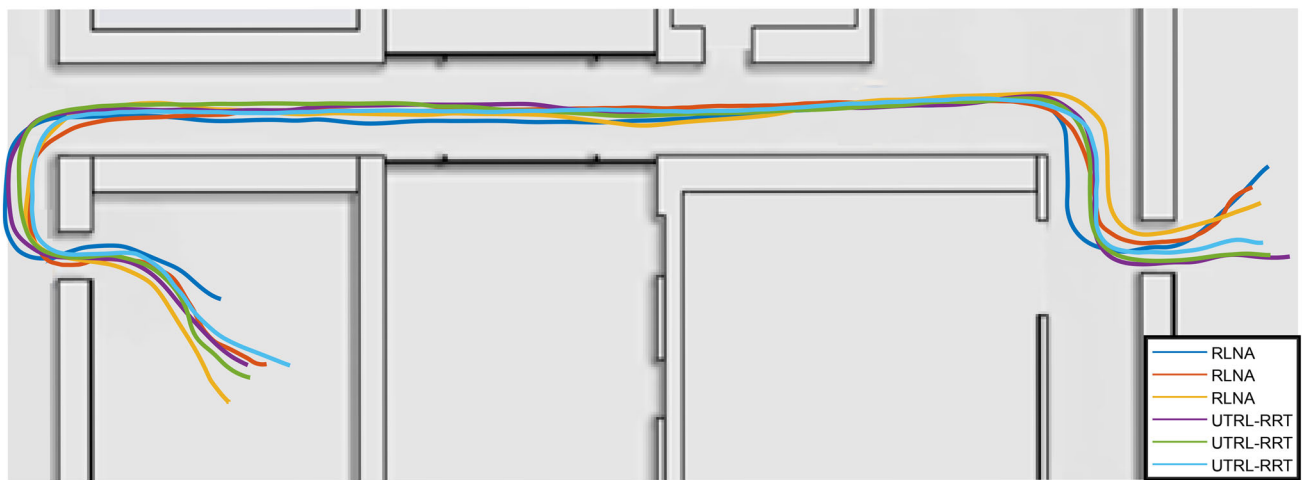


Fig. 17 Sample of the performed geometric paths overlapped with the scenario map

Table 4 Performance results for the validation tests

	Volunteer 1		Volunteer 2		Volunteer 3		Volunteer 4	
	RLNA	UTRL-RRT	RLNA	UTRL-RRT	RLNA	UTRL-RRT	RLNA	UTRL-RRT
C_C	0.05 ± 0.10	0.02 ± 0.04	0.04 ± 0.08	0.03 ± 0.04	0.05 ± 0.09	0.03 ± 0.04	0.06 ± 0.07	0.04 ± 0.05
C_R	0.03 ± 0.02	0.03 ± 0.02	0.03 ± 0.02	0.03 ± 0.02	0.03 ± 0.02	0.03 ± 0.02	0.03 ± 0.02	0.03 ± 0.02
C_{UI}	0.03 ± 0.03	0.02 ± 0.02	0.03 ± 0.03	0.02 ± 0.02	0.03 ± 0.03	0.02 ± 0.02	0.04 ± 0.05	0.03 ± 0.04
U_C	0.23 ± 0.09	0.22 ± 0.04	0.21 ± 0.07	0.23 ± 0.05	0.23 ± 0.11	0.24 ± 0.05	0.25 ± 0.09	0.25 ± 0.08
U_R	0.20 ± 0.05	0.22 ± 0.04	0.19 ± 0.05	0.22 ± 0.04	0.19 ± 0.07	0.23 ± 0.05	0.22 ± 0.07	0.23 ± 0.06
U_{UI}	0.24 ± 0.06	0.21 ± 0.04	0.23 ± 0.07	0.21 ± 0.05	0.25 ± 0.09	0.22 ± 0.05	0.26 ± 0.09	0.25 ± 0.08
d_o (m)	0.76 ± 0.34	0.87 ± 0.39	0.94 ± 0.63	1.02 ± 0.70	0.79 ± 0.57	0.87 ± 0.25	0.87 ± 0.30	0.84 ± 0.24
T_s (s)	176	180	205	174	219	166	187	175

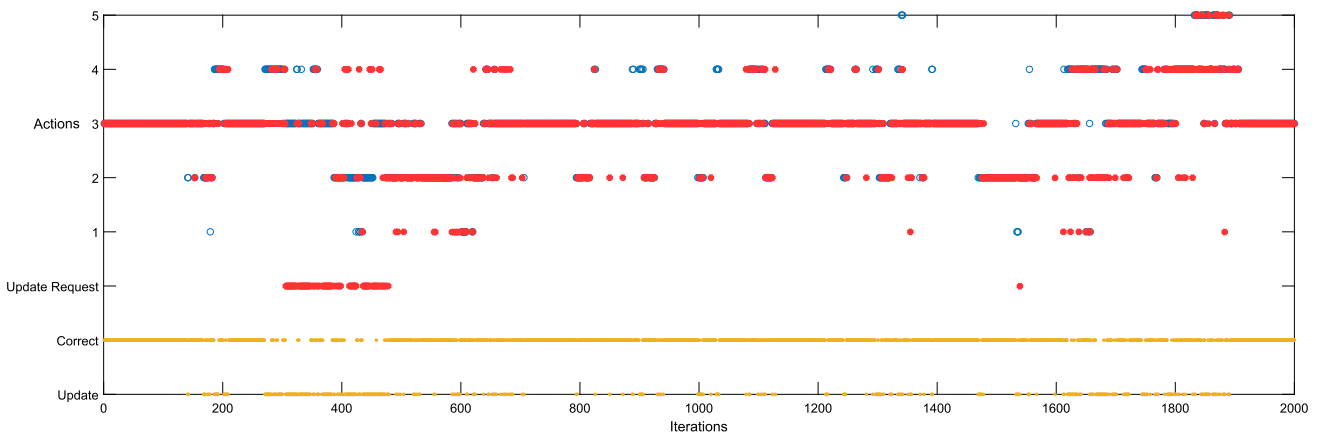


Fig. 18 Snapshot of the performed RL actions, by one volunteer, using the proposed UTRL-RRT approach. In red the RL model actions and in blue the actions taken by the platform (considering Eq. 12). (Color figure online)

ations, the RL action and the updated action in the model are the same, meaning that the model ends up learning important corrections. It is also important to note that most mismatches between actions are from neighbouring actions (2–3 and 3–4), which is normal for a initial model biased towards a non correcting action. On the other end, when the Q-values were negative, it correctly forced the update by learning from the computed output action. As can be seen in Fig. 18 (the actions shown in red), the RL model provided important corrections that aided the assistive platform.

6 Conclusion

In this paper we proposed a new robot-assisted navigation approach, validated and deployed in the ISR-AIWALKER platform. It is a solution to aid users with mobility impairments, that usually are faced with complex environments, inherently difficult to navigate due to the presence of obstacles and narrow passages, which demand dexterous walker operation. In our studies, the proposed approach presented promising results both in the virtual environment and in the real-world scenario. The proposed approach improved the performance of the user on several scenarios, which are commonly encountered in real-life situations. The performance of the volunteers, when using the proposed UTRL-RRT approach, improved significantly for the presented metrics, taking less time for three volunteers to complete the scenario and improving the presented control-related metrics for all volunteers. In the future, we plan to extend our work with the incorporation of dynamic obstacles and introduction of additional user's behaviors, as well as performing further tests with individuals belonging to the target population of these assistive devices to better ascertain the efficacy of our method.

Acknowledgements This work was supported by the Portuguese Foundation for Science and Technology (FCT) under the Ph.D. Grant with reference SFRH/BD/88459/2012 with funds from QREN-POPH and the European Social Fund from the European Union. It was also partially supported by the Project funded by FCT with reference FCT/COMPETE 2020/P2020 through Grants UID/EEA/00048/2013 and MATIS - CENTRO-01-0145-FEDER-000014, Portugal.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

- Balash Y, Hadar-Frumer M, Herman T, Peretz C, Giladi N, Hausdorff J (2007) The effects of reducing fear of falling on locomotion in older adults with a higher level gait disorder. *J Neural Trans* 114(10):1309–1314
- Batani H, Maki BE (2005) Assistive devices for balance and mobility: benefits, demands, and adverse consequences. *Arch Phys Med Rehabil* 86(1):134–145
- Burns B, Brock O (2007) Single-query motion planning with utility-guided random trees. In: 2007 IEEE international conference on robotics and automation, pp 3307–3312
- Chuy Jr O, Hirata Y, Wang Z, Kosuge K (2005) Motion control algorithms for a new intelligent robotic walker in emulating ambulatory device function. In: 2005 IEEE international conference mechatronics and automation
- Constantinescu R, Leonard C, Deeley C, Kurlan R (2007) Assistive devices for gait in Parkinson's disease. *Parkinsonism Relat Disord* 13(3):133–138
- Duan Y, Cui B, Yang H (2008) Robot navigation based on fuzzy RL algorithm. In: International symposium on neural networks. Springer, Berlin, pp 391–399
- Dubowsky S, Genot F, Godding S, Kozono H, Skwersky A, Yu H, Yu LS (2000) PAMM: a robotic aid to the elderly for mobility assistance and monitoring: a helping-hand for the elderly. In: IEEE International conference on robotics and automation
- Ferrari F, Divan S, Guerrero C, Zenatti F, Guidolin R, Palopoli L, Fontanelli D (2019) Human-robot interaction analysis for a smart walker for elderly: the acanto interactive guidance system. *Int J Soc Robot*. <https://doi.org/10.1007/s12369-019-00572-5>
- Garrote L, Paulo J, Perdiz J, Peixoto P, Nunes UJ (2018) Robot-assisted navigation for a robotic walker with aided user intent. In: 2018 27th IEEE international symposium on robot and human interactive communication (RO-MAN), pp 348–355
- Garrote L, Rosa J, Paulo J, Premevida C, Peixoto P, Nunes U (2017) 3D point cloud downsampling for 2D indoor scene modelling in mobile robotics. In: 2017 IEEE international conference on autonomous robot systems and competitions (ICARSC)
- Geravand M, Werner C, Hauer K, Peer A (2016) An integrated decision making approach for adaptive shared control of mobility assistance robots. *Int J Soc Robot* 8(5):631–648
- Khrijji L, Touati F, Benhmed K, Al-Yahmedi A (2011) Mobile robot navigation based on Q-learning technique. *Int J Adv Robot Syst* 8(1):4
- Kirby R, Simmons R, Forlizzi J (2009) COMPANION: a constraint-optimizing method for person-acceptable navigation. In: IEEE international symposium on robot and human interactive communication (RO-MAN)
- Kober J, Bagnell JA, Peters J (2013) Reinforcement learning in robotics: a survey. *Int J Robot Res* 32(11):1238–1274
- Lacey G, Namara SM, Dawson-Howe KM (1998) Personal adaptive mobility aid for the infirm and elderly blind. In: Mittal VO, Yanco HA, Aronis J, Simpson R (eds) Assistive technology and artificial intelligence: applications in robotics, user interfaces and natural language processing. Springer, Berlin, Heidelberg, pp 211–220. <https://doi.org/10.1007/BFb0055980>
- Lavalle SM (1998) Rapidly-exploring random trees: a new tool for path planning. Report No. TR 98-11, Computer Science Department, Iowa State University
- Lee G, Jung EJ, Ohnuma T, Chong NY, Yi BJ (2011) JAIST robotic walker control based on a two-layered Kalman filter. In: 2011 IEEE international conference on robotics and automation (ICRA)
- Lopes A, Rodrigues J, Perdigo J, Pires G, Nunes U (2016) A new hybrid motion planner: applied in a brain-actuated robotic wheelchair. *IEEE Robot Autom Mag* 23(4):82–93
- Lutz W, Sanderson W, Scherbov S (2008) The coming acceleration of global population ageing. *Nature* 451(7179):716–719
- Makarenko, AA, Williams SB, Bourgault F, Durrant-Whyte HF (2002) An experiment in integrated exploration. In: IEEE/RSJ

- international conference on intelligent robots and systems, 2002, vol 1, pp 534–539
21. McLachlan S, Arblaster J, Liu D, Miro JV, Chenoweth L (2005) A multi-stage shared control method for an intelligent mobility assistant. In: 9th international conference on rehabilitation robotics, 2005. ICORR 2005, pp 426–429
 22. Morris A, Donamukkala R, Kapuria A, Steinfeld A, Matthews JT, Dunbar-Jacob J, Thrun, S (2003) A robotic walker that provides guidance. In: 2003 IEEE international conference on robotics and automation, vol 1, pp 25–30
 23. Mou WH, Chang MF, Liao CK, Hsu YH, Tseng SH, Fu LC (2012) Context-aware assisted interactive robotic walker for Parkinson's disease patients. In: 2012 IEEE/RSJ international conference on intelligent robots and systems (IROS)
 24. Moustiris GP, Geravand M, Tzafestas C, Peer A (2016) User-adaptive shared control in a mobility assistance robot based on human-centered intention reading and decision making scheme. In: IEEE ICRA workshop: human–robot interfaces for enhanced physical interactions
 25. Paulo J, Peixoto P, Nunes U (2017) ISR-AIWALKER: robotic walker for intuitive and safe mobility assistance and gait analysis. *IEEE Trans Hum Mach Syst* 47:1110–1122
 26. Pérez-Higueras N, Caballero F, Merino L (2018) Teaching robot navigation behaviors to optimal RRT planners. *Int J Soc Robot* 10(2):235–249
 27. Ramrattan RS, Wolfs RC, Panda-Jonas S, Jonas JB, Bakker D, Pols HA, Hofman A, de Jong PT (2001) Prevalence and causes of visual field loss in the elderly and associations with impairment in daily functioning: the Rotterdam study. *Arch Ophthalmol* 119(12):1788–1794
 28. Russell S, Norvig P (2009) *Artificial intelligence: a modern approach*, 3rd edn. Prentice Hall Press, Upper Saddle River
 29. Simpson R, Boninger ML (2008) Clinical evaluation of guide robotic walker. *J Rehabil Res Dev* 45(9):1281
 30. Simpson R, LoPresti E, Cooper R (2008) How many people would benefit from a smart wheelchair? *J Rehabil Res Dev* 45(1):53–72
 31. Truong XT, Ngo TD (2016) Dynamic social zone based mobile robot navigation for human comfortable safety in social environments. *Int J Soc Robot* 8(5):663–684
 32. Wachaja A, Agarwal P, Zink M, Adame MR, Möller K, Burgard W (2017) Navigating blind people with walking impairments using a smart walker. *Auton Robots* 41(3):555–573
 33. Xu W, Huang J, Yan Q (2015) Multi-sensor based human motion intention recognition algorithm for walking-aid robot. In: 2015 IEEE international conference on robotics and biomimetics (ROBIO), pp 2041–2046
 34. Yu KT, Lam CP, Chang MF, Mou WH, Tseng SH, Fu LC (2010) An interactive robotic walker for assisting elderly mobility in senior care unit. In: 2010 IEEE workshop on advanced robotics and its social impacts (ARSO)

Luís Garrote is a PhD student (PhD fellowship granted by Portuguese Foundation for Science and Technology) and has a Master's degree in Electrical and Computer Engineering by the Faculty of Science and Technology of Coimbra University, obtained in 2011. Since 2010 he has collaborated on 4 research projects at the Institute of Systems and Robotics (ISR-University of Coimbra). His current research interests include robot motion planning, reinforcement learning applied in human-centered mobile robots and environment perception.

João Paulo has received his PhD degree in Electrical and Computer Engineering from the University of Coimbra (UC) in June, 2018, with the subject “A Multimodal HMI Approach for Interaction, User Modeling and Automatic Gait Analysis on a Robotic Walker”. He is currently collaborating in several projects and holds a position as post-doctoral researcher at the ISR-UC. His main research interests are assistive and rehabilitation robotics particularly mobility assistance. He focuses on perception and machine learning for gait analysis and Human-Robot Interaction.

Urbano J. Nunes is a Full Professor with the Coimbra University and Senior researcher of the Institute for Systems and Robotics (ISR-UC). He is the coordinator of the ISR-UC Human-Centered Mobile Robotics Lab. He has research interests in several areas in connection with human-centered mobile robotics with axes on mobile service robotics, mobile assistive robotics, technologies for autonomous vehicles (navigation and perception), pattern recognition and machine learning. He is author/coauthor of more than 160 publications on international scientific journals, book chapters, and refereed conference proceedings. Prof. U.J. Nunes serves as Associate Editor the *IEEE Transactions on Intelligent Vehicles* (2015-). He was co-organizer of eight journal special issues in topics related to Technologies for Intelligent Vehicles. He was with several international conferences: General co-chair of the 11th Int. Conf on Advanced Robotics (ICAR2003); Program Chair of IEEE Intelligent Transportation Systems Conf. (ITSC2006); Program co-chair of the IEEE ITSC2008; General Chair of the 13th IEEE Intelligent Transportation Systems Conf. (ITSC2010); a General Chair for the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012), and the General Chair of the IEEE ROMAN2017.