Indoor Turn-By-Turn Navigation Assistance for Robotic Rollators

Polihronis-Panagiotis Drakopoulos
National Technical University of Athens
Athens, Greece
a_drakop@windowslive.com

George Moustris
Athena Research Center Institute of Robotics
Athens, Greece
gmoustri@mail.ntua.gr

Costas Tzafestas
National Technical University of Athens
Athens, Greece
ktzaf@cs.ntua.gr

Petros Maragos
National Technical University of Athens
Athens, Greece
maragos@cs.ntua.gr

Abstract

Navigating indoor environments poses significant challenges for individuals with mobility and cognitive impairments, impacting their independence and quality of life. While robotic rollators have shown potential in providing mobility assistance, existing systems often rely on static navigation approaches that lack a capacity to dynamically adapt to path changes and unforeseen deviations. This paper addresses this gap by introducing a dynamic turn-by-turn navigation system designed to provide real-time, user-centered guidance. The system aims to enhance spatial orientation and wayfinding through online dynamic planning, addressing critical limitations of current solutions. We present results from user trials in a simulated environment, which show its potential to improve mobility. Possible areas of improvement are also discussed, paving the way for more effective assistive technologies in real-world settings.

CCS Concepts

• **Human-centered computing** → **Auditory feedback**; *Natural language interfaces*; • **Computer systems organization** → **Robotics**.

Keywords

assisted navigation, robotics, interaction design, turn-by-turn navigation, audio commands, robotic rollator, smart walker

ACM Reference Format:
Polihronis-Panagiotis Drakopoulos, George Moustris, Costas Tzafestas, and Petros Maragos. 2025. Indoor Turn-By-Turn Navigation Assistance for Robotic Rollators. In *The PErvasive Technologies Related to Assistive Environments (PETRA '25), June 25–27, 2025, Corfu Island, Greece.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3733155.3733211

1 Introduction

Navigating indoor environments can present a significant challenge for individuals with mobility and cognitive impairments, particularly in unfamiliar or complex spaces such as hospitals, malls, or care facilities. Effective navigation relies on maintaining a coherent sense of spatial orientation along with localizing one's self with respect to the surroundings and constructing a mental map of the premises. However, navigational skills tend to diminish with age [10], a decline exacerbated by cognitive impairments prevalent in geriatric populations, which affect a significant portion of patients in nursing homes, rehabilitation centers, and acute care hospitals [5, 11].

Difficulties in spatial orientation and navigation, often observed in both familiar and unfamiliar environments, are among the earliest indicators of dementia [2, 3]. The loss of these essential skills can have profound consequences, including reduced mobility, autonomy, and independence [1]. Robotic rollators are being increasingly investigated as a tool to assist patients with everyday task, offering various services such as proactive support, navigational assistance, internet connectivity and more. Incorporating smart functionalities to support spatial orientation and wayfinding could offer significant benefits, particularly for frail older adults facing navigation difficulties.

In this paper we present a turn-by-turn (TbT) navigation functionality for a robotic rollator to assist the users navigate indoor environments and guide them to the desired location using audio commands. The algorithm uses a pre-built known map of the premises and estimates the robot's location using localization techniques, fusing odometric and distance information from a Light Detection and Range (LIDAR) sensor. It continuously monitors the robot's surrounding constructing an obstacle map, computes safe directions of motion using the Dynamic Window Arc-Line (DWAL) local planner [7] and uses Dijkstra's algorithm for the overall path to the goal position, issuing appropriate motion commands to the user.

The functionality has been tested in a realistic virtual environment depicting a hospital floor. We have constructed a virtual representation of the rollator, being guided by the user who sees the world from a first person point-of-view. We conducted a series of randomized experiments with real users to assess the effect of the TbT navigational assistance on the way-finding performance of the users. The results are presented and discussed in the following.

This work comes as a follow-up to a string of papers presented by the authors [6, 9] describing a similar TbT functionality in robotic rollators. The assisted navigation in that work used a predefined set of guard point on the map, comprising fixed routes. When the user entered a guard point, essentially a circle of a given radius, the rollator voiced a predefined turning command. This setup limited

PETRA '25, Corfu Island, Greece

ACM ISBN 979-8-4007-1402-3/25/06
https://doi.org/10.1145/3733155.3733211

the applicability of the TbT algorithm since the routes had to be defined a priori and did not allow for any flexibility during navigation. Our current work removes such limitations as it is purely dynamic, without preset routes and accommodates the current position of the user by on-line planning a safe route to the desired goal.

2 Methods

2.1 System Overview

As already mentioned, our algorithm considers navigation in an indoor environment with a known map. The robot is equipped with a LiDaR sensor, which scans the surrounding environment, as well as wheel encoders which measure the distance traveled by the robot. The evolution of the robot's pose from its initial starting position, i.e. the *odometric* information, is estimated using *dead reckoning*. This involves the incremental integration of the differential wheel motion over time. The estimation of the robot's pose on the map, the so-called *localization* problem, is computed online using the Adaptive Monte Carlo Localization (AMCL) method [8]. AMCL is a widely used probabilistic algorithm that employs a particle filter to estimate the robot's pose by fusing information from the odometry and the laser scanner.

During navigation, the user's position and the goal point they want to reach are always known. A *global planner* running at 10Hz, computes the path from the user to the goal point using Dijkstra's algorithm. To account for the local changes in the environment and also provide general "directions of motion", the DWAL local planner is deployed. DWAL creates a local representation of the obstacles around the robot crafting a 2D costamp from the LIDAR's readings. This results in an occupancy grid where each cell's value represents the cost of traversing through it. It is based on the well know Dynamic Window Approach [4] but instead of projecting arcs ahead, it uses a combination of arcs and lines. These are checked for collision up to a circle of set radius ahead, the so-called "*level*". The DWAL produces clusters of motion which are kinodynamically feasible and safe.

Based on the intersection of the global path and the motion clusters produced by DWAL, an appropriate turning command is issued. The general architecture of our system can be seen in Fig.1.

We use *three* Levels for the DWAL, set at different distances ahead; the "near" level (R=1.25 m), the "'mid" level (R=2.5 m) and the "far" level (R=4.0 m), as seen in Fig. 2. We also use a fourth *back up level* at R=1.0 m in the case where the first three are "OFF". These levels produce a respective turning command based on their *level architecture*. The commands are *lexical tokens* from a command vocabulary {"continue straight", "turn right", "turn left", "turn around", "You have arrived at your destination"}. Using a simple First-past-the-post voting process, we produce a *candidate command*. If the voting is a draw, then the candidate command occurs from the level that is further away from the robot. If all the levels are "OFF", then the "Back up" level command will be considered.

To compute the final *user command*, we introduce the *Command Scheduler*, which is responsible for determining which command should be delivered to the user and when it should be played. The primary principle guiding this process is that a new command should only be issued if a significant change has occurred, such as a turn, or if a substantial amount of time has passed. This approach ensures the user receives timely confirmations about their direction and avoids confusion through appropriate reminders.

To achieve this, the Command Scheduler maintains a buffer of the last command issued to the user along with the timestamp of its delivery. When a new candidate command is generated, the scheduler evaluates its necessity based on the following criteria:

- **Repetition of the Same Command**: If the candidate command matches the previous user command, the scheduler activates a timer starting from the last issued command. If this timer reaches 10 seconds, the command is repeated to reinforce the user's navigation instructions and maintain clarity.
- **Change in Command**: If the candidate command differs from the previous user command, the scheduler checks whether at least 2 seconds have elapsed since the last command. If this condition is met, the new candidate command is issued to the user. This delay prevents overwhelming the user with consecutive commands and acts as a filtering mechanism to ignore transient or random events that may not warrant a command.

The scheduler logic can be represented by the pseudocode in Algorithm 1.

Algorithm 1 Command Scheduler

Given:
User Command $Cu(T-1)$ at time *T-1*
User Command $Cu(T)$ at time T
Candidate Command $Cc(T)$ at time T
Time $Tcmd$ elapsed since last issued user command
Command-Repeat time threshold $Tr = 10sec$
Command-Update time threshold $Tu = 2sec$
Do:
if $Cc(T) = Cu(T-1)$ **then** ▷ same command arrives
    $Cu(T) \leftarrow Cc(T)$
    if $Tcmd > Tr$ **then**
        $Tcmd \leftarrow 0$
        play audio of $Cu(T)$
    else
        wait for next command
    end if
else ▷ new command arrives
    if $Tcmd > Tu$ **then**
        $Cu(T) \leftarrow Cc(T)$
        $Tcmd \leftarrow 0$
        play audio of $Cu(T)$
    else
        discard $Cc(T)$
    end if
end if

The use of timers in the scheduling process serves two purposes: avoiding excessive or rapid command delivery and providing a de-noising effect. Considering the slower walking speed of elderly users, the 2-second delay ensures commands are deliberate and contextually relevant. Whenever the Command Scheduler decides to issue a command, it interfaces with an appropriate module which which is responsible for playing the corresponding .wav audio file.

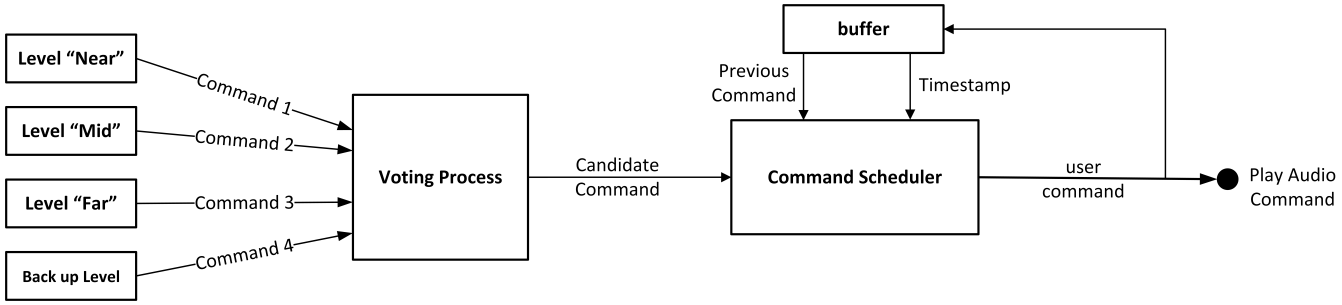

Figure 1: General overview of the Turn-by-Turn navigation algorithm.

2.2 Level Commands

Each DWAL level produces a turning command which is introduced to the voting process as a possible candidate command. To calculate this turning command we compute the intersection of the global path with *the front* of the available level clusters (essentially with the cluster arcs on the level's circumference). Since the DWAL clusters signify feasible, and safe, directions of motion, this ensures that the turning command points the user to a, similarly, safe direction. If the global path does not intersect any clusters or no available clusters exist, then this level is considered "OFF".

Technically, since both the global path and the cluster arcs consist of a sequence of 2D points, there intersection is computed iteratively by considering their Euclidean distance. If this distance is below a preset, small, threshold, then we mark the corresponding cluster point as the intersection point.

To create a prompt that is understandable by the user, the intersection point is translated to a *lexical command* selected from the vocabulary. This is performed by observing the central angle of the ray to the intersection point and the*X-axis* (see Fig. 2-Right). Depending on the specific intervals the angle falls in, it is assigned to a corresponding command. Note that these intervals are different for each level since the sense of what is considered "left", "straight" or "right" depends on the distance. The intervals used in this work are given below:

Table 1: Intersection angle ϕ to lexical command

Level	Straight	Left (Right)	Turn Around
Far	$\{-22°, 22°\}$	$-(+)\{22°, 135°\}$	$\{-135°, 135°\}$
Mid	$\{-28°, 28°\}$	$-(+)\{28°, 135°\}$	$\{-135°, 135°\}$
Near	$\{-33°, 33°\}$	$-(+)\{33°, 135°\}$	$\{-135°, 135°\}$
Backup	$\{-18°, 18°\}$	$-(+)\{18°, 120°\}$	$\{-120°, 120°\}$

The lexical level commands are produced at a rate of 10Hz. Since the geometry of the clusters is dependent on the configuration of the surrounding space, as the user walks through the premises, the intersection point can present frequent switching. To prevent his behavior and produce a stable outcomes, we feed the commands to a *persistence filter*. This is essentially a queue of 15 places. Each command is pushed to the top, discarding the oldest one at the bottom. A *stable level command* is produced only if the last (oldest) 10 commands in the queue are identical. Otherwise the level is considered OFF.

The introduction of three different levels, each some distance ahead from the other, ensures that the required direction of motion presents spatial persistency and is not sensitive to fast or frequent switching. For example, in the turn depicted in Fig. 2, the intersection of the global path with "Far" level implies a turn to the right; however the "Mid" and "Near" levels might suggest a further movement ahead, before turning. This anticipatory behavior is served by the use of multiple levels.

In the case where all levels are OFF, the system falls back to the "Backup Level". This is a simplified level without clusters, which is very close to the user (R=1 m). Its utility is to produce a motion command to allow the user to re-orient and face towards open space, where the other levels can produce directions of motion. This behavior has been experimentally observed in cases where the user is very close to obstacles (walls, furniture etc) and the "backup level" instructs them to turn around. Since no clusters exist, the intersection point with the path is calculated with the level's front.

2.3 Simulation Environment

To perform the experimental evaluation of our algorithm, we used an open source Gazebo simulated Hospital World developed by Amazon Web Service Robomaker [1]. The creation of the world's ROS map was achieved via a Gazebo plugin (ros 2Dmap), which can automatically generate a 2D occupancy grid map from a Gazebo 3D simulated world. To emulate the rollator, the geometry of the MOBOT rollator [7] was imported and augmented with differential drive kinematics. The users had control over the linear and angular velocities of the robot through the keyboard, via a custom teleoperation node.

Our goal was to have a complete simulation of the human capabilities and thus a camera node was added to emulate the human point of view. The camera was attached to a fixed distance above the robot, matching the position of the user's head in the virtual environment. It allowed two rotational degrees of freedom; pitch and yaw. The user could control the camera with the mouse while the image was streamed to a window via a ROS topic, encoded with the 'theora" codec. Damping and friction were also added to all robot joints to make the simulation more stable and realistic.

[1] https://github.com/aws-robotics/aws-robomaker-hospital-world

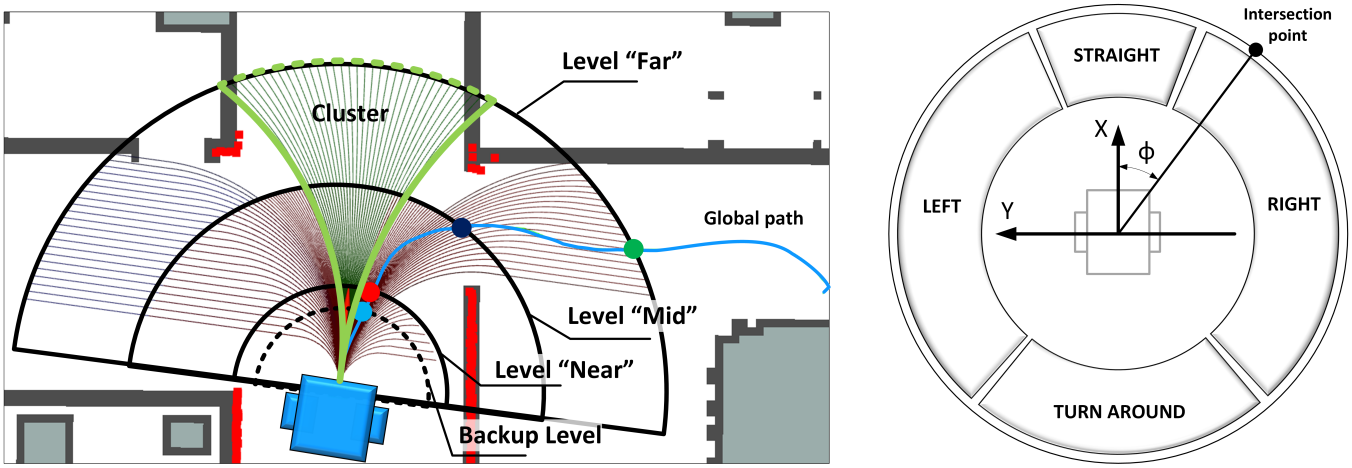

Figure 2: LEFT: Snapshot of the DWAL planner in action. The four levels ("Far", "Mid", "Near", "Backup") can be seen. The motion clusters are arc-line paths which start from the robot and reach the level front, grouped in similar color. The intersection point with the global path for each level is also depicted. RIGHT: Translation of the intersection point to lexical command.

3 Simulation Experiments

3.1 Experimental Setup

During the experimental session, each user was seated in front of a monitor which streamed the virtual simulation of the hospital environment from the Gazebo simulator. The user controlled the motion of the rollator via keyboard, and the virtual head POV with the mouse. In total, 6 users were used for the algorithm evaluation. Each one ran 6 trials, amounting to a total of $6 \times 6 = 36$ trials for all users.

For the experiments, we defined *three* pairs of starting and ending positions (called points "A" and "B" resp.) on the hospital map. The tasks of the user was to navigate to the goal point (point B") while starting from the initial point "A", following the audio commands of our TbT navigation algorithm. When the user approached the ending position, an appropriate audio message was heard ("you have reached your destination"), marking the completion of the trial. Each pair appeared *twice* during the session, while the trial sequence was randomly generated and not known to the user.

Before starting the trials, each user was given 5 minutes to drive the robot around the world to get familiar with the controls. Each trial was recorded with ROS in bag files and the results were analyzed in post processing. The users were 3 males and 3 females with their age ranging from 22 to 67 years old. Each pair corresponded to a plausible real-life scenario in the hospital e.g. go from the bed in the hospital ward to the floor kitchen. This semantic interpretation was unknown to the user. Thus, during the trials the users did not know for example that they were searching for a kitchen. The three pairs are described in the following list:

- Pair 1: User should go from the hospital's reception to their room and find their bed.
- Pair 2: User should go from their bed to the kitchen room
- Pair 3: User should go from their bed to floor's reception

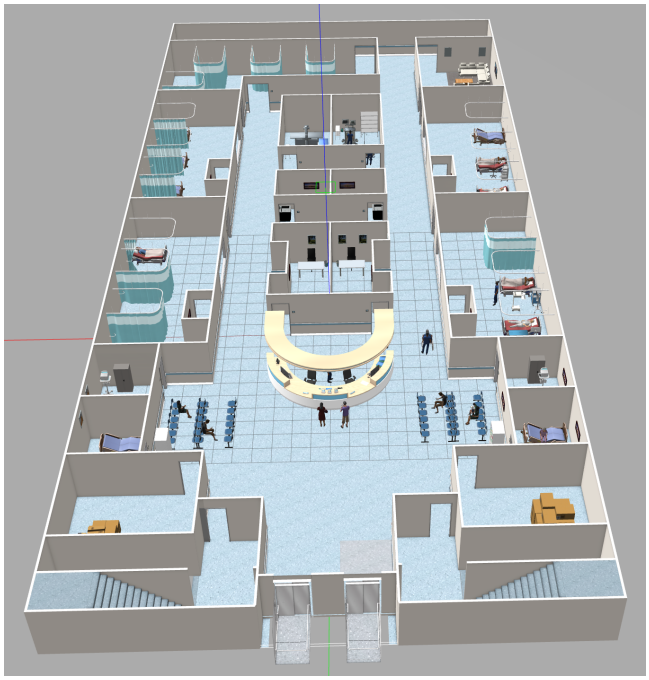

Figure 3: Top-down view of the virtual hospital floor environment.

To assess the performance of the TbT algorithm, we calculated the outcome measures described in Table 2, also used in similar studies [6, 9]. The results are presented in the following section.

3.2 Results

To calculate the outcome measures, we extracted the path each user traversed during each trial. This is essentially the positions of the user/robot pair on the map, as given by the localization algorithm.

Table 2: Description of the Outcome Measures of the experimental evaluation

Outcome measure	Details
Path Completion Time - **T (sec)**	The time each user takes to go from the starting position to the ending position
Number of Stops - **K**	The number of the time intervals where the user has a velocity less than 0.1 m/s, for at least 1 s
Total Stop Time - $\mathbf{T}_{stop}$ **(sec)**	The sum of all time intervals where the user has a velocity less than 0.1 m/s, for at least 1 s
Walking Distance - **S (m)**	The geometrical distance a user travels to go from the starting position to the ending position
Walking Velocity - **Vm (m/sec)**	The average user velocity calculated as the Walking Distance over the Total Walking Time. The Total Walking Time is computed as the sum of all time intervals in which the user has a velocity $\geq$ 0.1 m/s

To have similar starting and stopping conditions across the trials, the actual paths that were considered started a distance of 1 m from the starting point and ended when the remaining length of the global path segment from the user to the ending position, was less than 2 m.

When discussing the distance, velocity, and time associated with a user during a trial, it is important to clarify that these measurements actually correspond to the respective quantities of the robot. This is because, in the simulated environment the user is rigidly attached to the robot. In real life however, the user walks while holding the robot by its handles, causing the two to move together as a single, unified entity. The user's instantaneous walking velocity was estimated during post-processing from the odometric data using the Euler approximation. Specifically, this involved calculating the difference between two consecutive localization measurements and dividing it by their corresponding time interval. To reduce the noise, the velocity was filtered using a robust locally weighted scatterplot smoothing (RLOWESS) filter. The user paths for all three pairs are seen in Fig. 4. The results are presented per pair, to have a similar reference between trials.

The results for the outcome measures are presented in Table 3, expressing the average value and standard deviation across all 12 trials for each pair. Note that the Number of Stops (K) presents the number of trials for each pair where the users stopped "K" number of times during their trial. These metrics provide a comprehensive assessment of user performance and system behavior during the navigation experiments.

As a general comment, we first note that all users were successful in reaching their intended goal. Thus the primary objective of the TbT algorithm was fulfilled. Analyzing the results, we can see the following; the "Path Completion Time (T)" reflects the overall duration users took to traverse each path, accounting for walking distance, stops, and system guidance. The variability in completion times across trials, as indicated by standard deviations, highlights the impact of user-specific behavior and environmental interactions on navigation performance.

The "Number of Stops (K)" and "Total Stop Time (Tstop)" offer insights into user-system interaction dynamics and the complexity of the process. While some trials exhibited smooth navigation with minimal stops, others showed more frequent or prolonged pauses, suggesting challenges in interpreting commands or adjusting to unexpected situations.

The "Walking Distance (S)" aligns with the geometry of each path, showing consistent results across trials. The stable "Walking Velocity (Vm)" across all paths (0.3 ± 0.01 m/s) suggests that users maintained a steady pace, regardless of the path length or complexity. At first glance, this indicates that the navigation system effectively guided users without causing excessive hesitation or disruptions in their walking behavior. However, this measurement is highly affected from the experimental setup, where users controlled the robot via a keyboard, and the way the custom teleoperation node works (maximum linear velocity was limited to 0.31 m/s). In a real-world scenario where users interact with a physical robot, we expect greater variation in average velocity between users and a higher standard deviation.

3.2.1 Outliers. Out of all trials, 9 were labeled as outliers due to significant deviations in the user's trajectory, compared to other trials of the same pair. The main causes of these outliers are summarized below:

(1) Global Path Initialization Variability In three trials, (Pair 1-green,cyan, Pair 2-cyan) the global planner produced a different initialization path despite seemingly identical conditions. This variability, likely caused by factors such as sensor noise, floating-point precision, and costmap updates, but did not significantly impact the algorithm's performance.
(2) Computational Overloading Hardware overloading due to running the simulation and the TbT algorithm concurrently on the same cpu, caused a significant delay in processing "intersection points" in one trial (Pair 1-blue), leading to desynchronization of the simulator with the turning motions. However, the system recovered after 27 seconds and navigated the user correctly afterwards.
(3) DWAL's sensitivity and Backup System Limitations In one trial (Pair 3-purple), all primary navigation levels temporarily went OFF, due to DWAL's sensitivity in closed and narrow space, activating the backup system that issued one incorrect command. This rare event, likely related to environmental constraints, revealed limitations in the backup system's tuning for such scenarios. Despite this, the algorithm corrected the commands dynamically and navigated the user to the goal without any further issue.
(4) Users Misunderstanding Commands In two trials (Pair 1-red, Pair 1-orange), users misunderstood the audio command "turn left", assuming it referred to a door in their line of sight rather than their immediate position. This caused initial deviations, but the algorithm immediately adapted dynamically, issued corrective commands, and successfully guided users to a U turn and finally to the goal.

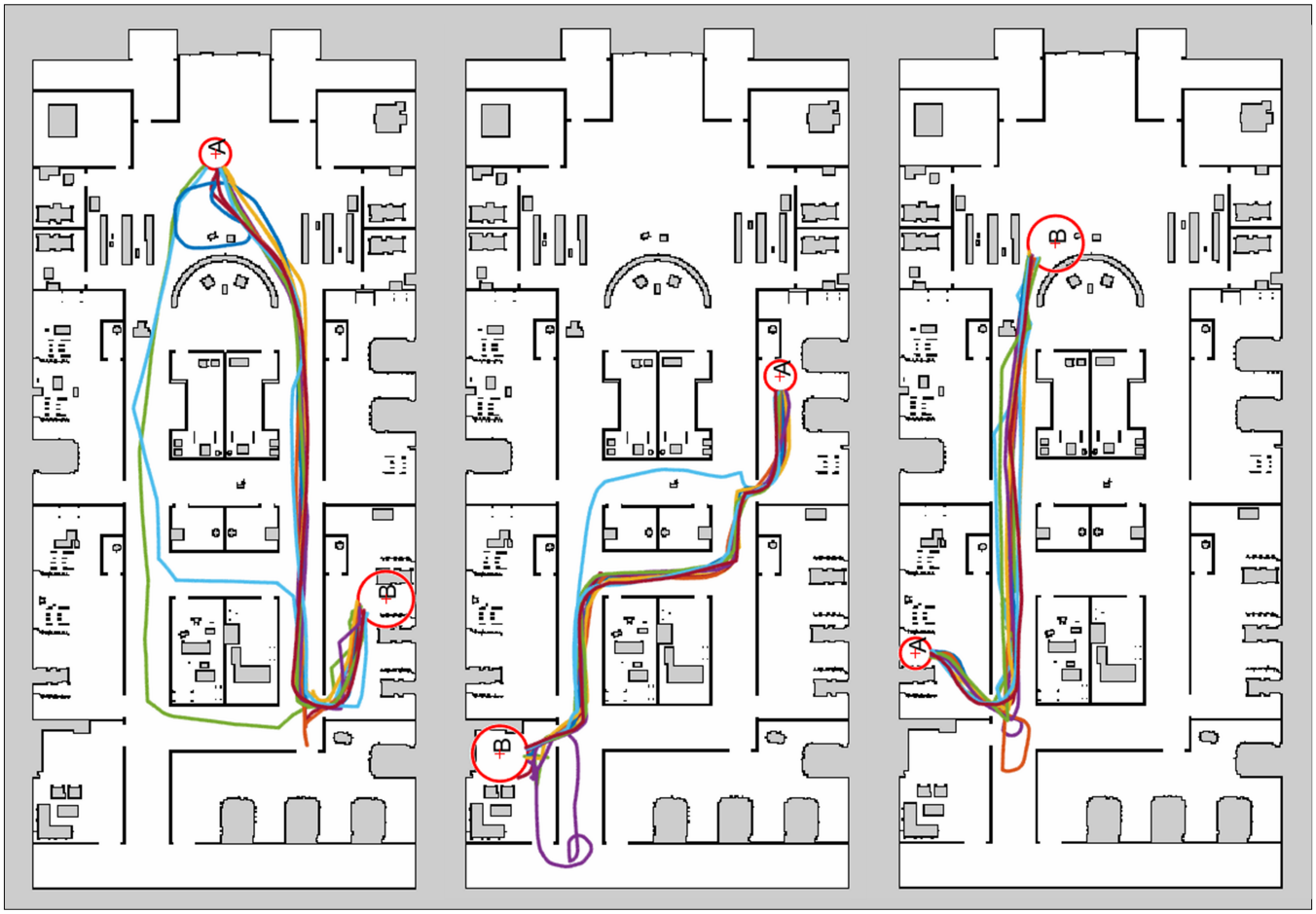

Figure 4: Paths of the users during the trials for all the start/goal pairs. From left to right: Pair 1 to Pair 3.

Table 3: Results of the experimental evaluation

Outcome measure		**Pair 1**	**Pair 2**	**Pair 3**
Path Completion Time - **T (sec)**		173.89 (24.19)	142.11 (35.83)	131.62 (12.10)
Number of Stops - **K**	K=0	2	4	3
	K=1	3	1	2
	K=2	2	1	2
	K=5	3	3	2
	K>5	2	3	3
Total Stop Time - $\mathbf{T}_{stop}$ **(sec)**		13.06 (11.93)	14.17 (15.69)	8.10 (7.86)
Walking Distance - **S (m)**		48.42 (5.03)	37.94 (7.94)	37.32 (2.89)
Walking Velocity - **Vm (m/sec)**		0.3 (0.01)	0.3 (0.01)	0.3 (0.01)

*results are presented as *mean* (sd)

(5) User Errors and Assumptions In two trials (Pair 2-purple, Pair 3-orange), users deviated from the given commands due to personal errors. In the first case, the user ignored a "turn left" command and continued forward for unknown reasons. In the second, the user disregarded navigation instructions near the goal and turned to the wrong direction. However in a real life such a scenario could have been avoided given that users know what they are searching for. Both errors caused temporary deviations, but the algorithm adapted and successfully guided users back to the correct location.

4 Discussion

The results of this study demonstrate the effectiveness of TbT navigation in a known static map with audio commands for robotic rollators, in guiding users through an indoor virtual hospital environment. It is important to note that all participants successfully navigated from the start to the goal point, demonstrating high level of reliability and robustness of the system in achieving its primary objective. This achievement reinforces the potential for dynamic on-the-fly planning for assistive navigation, which addresses the limitations of previous work that relied on predefined routes and guard points.

Despite the presence of outliers and occasional user or system misunderstandings, the system managed to adapt to deviations in user behavior, provide updated corrective navigation commands and successfully redirect users. This adaptability is critical for assistive navigation and indicates the potential of the algorithm architecture to be applied in real world scenarios, where environmental changes are unpredictable and user behaviors are inevitable.

However, the study also revealed areas for improvement. In some of the trials, users experienced significant confusion and difficulties interpreting audio commands, leading to particular wrong turns and navigation errors. This issue was particularly evident in scenarios where users misunderstood or ignored commands, highlighting the importance of precise, context-aware instructions and spatial references. For example, commands such as "turn left" could be enhanced with spatial clarifiers like "turn left in 3 meters". This improvement could mitigate confusion and reduce the time required for corrective navigation.

The metrics also showed that there were users who navigated faster and with fewer stops than others, indicating that not all users interacted with the system in the same way or found it equally intuitive. This variation suggests that individual differences, such as cognitive abilities, familiarity with technology, or hearing sensitivity, can influence the effectiveness with which users can follow the instructions provided. It is also important to note that the study was conducted in a simulation environment in Gazebo, which likely introduced factors that could affect user performance. For example, users ability to interact with the computer and it's external components, navigate the simulation setup, and adapt to virtual environments, may have influenced the metrics recorded during the trials. These factors highlight the need to test the system in real-life environments, where users would physically push the rollator and interact with tangible surroundings. Such testing would provide a more accurate assessment of the system's usability and effectiveness under realistic conditions, accounting for human-environment and human-robot interactions. In such case, more certain conclusions will be extracted.

5 Conclusion & Future Work

The results of this study validate the feasibility of audio-based TbT navigation for robotic smart rollators in indoor environments, offering a promising solution to enhance mobility for older adults and individuals with disabilities. By leveraging dynamic planning and robust adaptability, the system demonstrated the potential to overcome challenges associated with static predefined routes.

Future improvements should focus on making the navigation system even more intuitive and user-friendly in all terms. Refining audio commands to make them more specific, clearer and easier to understand could reduce confusion in some cases, but we should consider that in real world noisy environments will be met, where elderly users have a high probability of struggling to hear the audio commands. A solution to this problem could be incorporating additional feedback methods, as well as multimodal navigation. Such could be vibrations on the handgrips of the robot which indicate additional turns, adding haptic shared-control functionality by modulating the resistance or "heaviness" of the robot when the user deviates from the intended path, or simple visual indicators like arrows via augmented reality. Finally, an appropriate display in a good placement on the robot to be seen from the user, might also help users who struggle with audio instructions. At this point it is important to mention that adding more interactions and giving extra information to the user has a trade off, as it could potentially lead to cognitive overload and distract users attention from the environment, ultimately reducing the effectiveness of the navigation system. Balancing the quantity and complexity of information delivered is crucial to ensure that assistive cues enhance, rather than hinder, the user's situational awareness and ability to safely navigate their surroundings. Thus, any added feature or integrations with other technology systems, should be thoroughly tested in order to draw clear conclusions about the overall effectiveness of the system.

These enhancements would build on the strengths of the current system and address its limitations, paving the way for its use in real-world settings to improve mobility and independence for older adults and individuals with disabilities.

Acknowledgments

This research work was supported by the project "Applied Research for Autonomous Robotic Systems" (MIS 5200632) which is implemented within the framework of the National Recovery and Resilience Plan "Greece 2.0" (Measure: 16618—Basic and Applied Research) and is funded by the European Union—NextGenerationEU.

References

[1] Peter C Burns. 1999. Navigation and the mobility of older drivers. *The Journals of Gerontology Series B: Psychological Sciences and Social Sciences* 54, 1 (1999), S49–S55.

[2] Yi-Chen Chiu, Donna Algase, Ann Whall, Jersey Liang, Hsiu-Chih Liu, Ker-Neng Lin, and Pei-Ning Wang. 2004. Getting lost: directed attention and executive functions in early Alzheimer's disease patients. *Dementia and geriatric cognitive disorders* 17, 3 (2004), 174–180.

[3] Gillian Coughlan, Emma Flanagan, Stephen Jeffs, Maxime Bertoux, Hugo Spiers, Eneida Mioshi, and Michael Hornberger. 2018. Diagnostic relevance of spatial orientation for vascular dementia: a case study. *Dementia & neuropsychologia* 12 (2018), 85–91.

[4] D. Fox, W. Burgard, and S. Thrun. 1997. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine* 4, 1 (1997), 23–33. https://doi.org/10.1109/100.580977

[5] Fiona E Matthews and Tom Dening. 2002. Prevalence of dementia in institutional care. *The Lancet* 360, 9328 (2002), 225–226.

[6] George Moustris, Nikolaos Kardaris, Antigoni Tsiami, Georgia Chalvatzaki, Petros Koutras, Athanasios Dometios, Paris Oikonomou, Costas Tzafestas, Petros Maragos, Eleni Efthimiou, et al. 2021. The i-walk lightweight assistive rollator: First evaluation study. *Frontiers in Robotics and AI* 8 (2021), 677542.

[7] George P Moustris and Costas S Tzafestas. 2016. Intention-based front-following control for an intelligent robotic rollator in indoor environments. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 1–7.

[8] Sebastian Thrun. 2002. Probabilistic robotics. *Commun. ACM* 45, 3 (2002), 52–57.

[9] Christian Werner, George P Moustris, Costas S Tzafestas, and Klaus Hauer. 2018. User-oriented evaluation of a robotic rollator that provides navigation assistance in frail older adults with and without cognitive impairment. *Gerontology* 64, 3 (2018), 278–290.

[10] Shuying Yu, Alexander P. Boone, Chuanxiuyue He, Rie C. Davis, Mary Hegarty, Elizabeth R. Chrastil, and Emily G. Jacobs. 2021. Age-Related Changes in Spatial Navigation Are Evident by Midlife and Differ by Sex. *Psychological Science* 32, 5 (2021), 692–704. https://doi.org/10.1177/0956797620979185 PMID: 33819436.

[11] Sheryl Zimmerman, Philip D Sloane, and David Reed. 2014. Dementia prevalence and care in assisted living. *Health Affairs* 33, 4 (2014), 658–666.