Intention-Based Front-Following Control for an Intelligent Robotic Rollator in Indoor Environments

George P. Moustris
School of Electrical and
Computer Engineering
National Technical University of Athens
Athens, Greece
Email: gmoustri@mail.ntua.gr

Costas S. Tzafestas
School of Electrical and
Computer Engineering
National Technical University of Athens
Athens, Greece
Email: ktzaf@cs.ntua.gr

***Abstract*—This work presents an intention-based assistive controller for allowing a robot to follow a human while moving in the front. This task is particularly challenging in indoor environments, as there are situations that are undecidable, namely in junctions. We propose a novel local kinodynamic planner which concurrently detects discrete routes and continuous motion paths, providing path equivalence classes. Furthermore, we detail an intention recognition algorithm to make the robot take the user-intended turn. Our scheme is experimentally tested in a T-Junction set-up using human subjects, and the results are discussed.**

I. INTRODUCTION

As robots become more cognitivelly advanced, their operation in human populated spaces becomes more socialy acceptable and their relation to humans more intertwined. Human-following is an important task in human-robot interaction, emerging in situations where the human must be followed by the robot in the workspace in order to assist him/her with certain tasks. Such scenarios can be seen in hospital assistance robots [1], telepresence robots [2] and companion robots [3], to name some. The major research volume conisders only the case in which the human always walks in the front of the robot. In general however, three cases can be discerned [4]; *behind the leader*, *side-by-side* and, *in front of the leader*.

In the first case ('behind the leader'), the problem can be simplified since the human intention is known *a posteriori* by inspection of his/her trajectory. Cast as a control-theoretic problem, only the human-robot relative position needs to be known, and by retracing the human path the robot can be made to stay behind the user. In contrast, the latter two cases present an increasing difficulty. Specifically, in the 'side-by-side' scenario, the robot has to also monitor the user orientation while in the 'following from the front' case, estimation of the user's intention must be incorporated into the control loop. The difficulty is also exacerbated be the fact that undecidable situations enter the spotlight, which require explicit user input to avoid deadlocks.

In this paper, an intention recognition algorithm is presented which enables the user to walk in an indoor environment while having a robot follow him/her from the front. The algorithm also employs a novel dynamic window local planner which constructs *path equivalence* classes, expressing kinodynamically feasible *discrete* decisions for motion. In turn, each class represents a *continuous* bundle of obstacle-free feasible motion alternatives. This allows the identification of different routes in junctions, and provides safe locomotion by avoiding collisions. The purpose of this control scheme is to allow an active mobility assistance robot (Fig.1) to oversee the patient, walk along with him/her and provide assistance either on demand or autonomously.

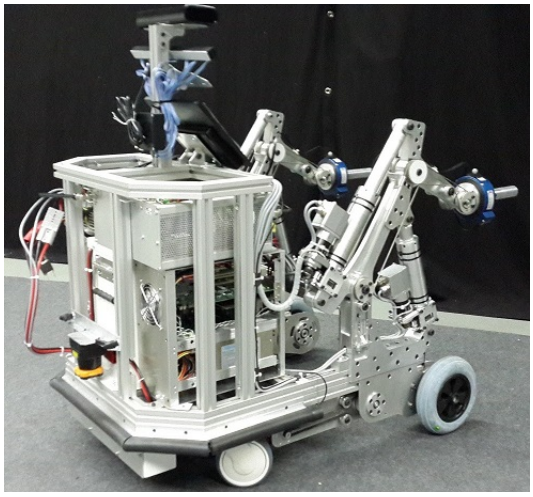

Fig. 1. The MOBOT assistance robot (www.mobot-project.eu).

For the analysis and experimentation, the differential drive model was used i.e.,

$$\begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{bmatrix} = \begin{bmatrix} \cos\theta_R \\ \sin\theta_R \\ 0 \end{bmatrix} v_R + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega_R \qquad (1)$$

where x_R, y_R are the robot coordinates in the world frame, and θ_R its orientation. The control inputs are the linear and angular velocities denoted by v_R, ω_R respectively, obeying the bounding constraints $|v_R| \leq v_{Rmax}, |\omega_R| \leq \omega_{Rmax}$. Furthermore, the *dynamic extension* of the model is considered by augmenting the equations with the linear (α_v) and

angular (α_ω) accelerations. Bounding constraints also apply, viz. $|\alpha_v| \leq a_{vmax}, |\alpha_\omega| \leq \alpha_{\omega max}$.

II. RELATED WORK

The problem of front-following is scarcely met in the literature, with only three research papers discussing the subject. In all three, the free-space following problem is considered that is, following the human in a workspace which has no obstacles. The authors in [5] use a Laser Range Finder (LRF) in order to scan the upper body of the user (torso), which is a more robust target for scanning than the legs. The user pose is estimated by using a particle filter with a constant velocity model. The controller uses the human and robot poses in order to infer a virtual target in the general direction of the user velocity and track it. In [4] an image/depth sensor is used (Microsoft Kinect) in order to track the human position with respect to the robot. Consequently, the nonholonomic human model [6] is deployed to estimate the user's orientation. An Unscented Kalman Filter is also used to provide a smoother estimation of the human linear/angular velocities and orientation. The goal of the controller is to align the pose of the human and the robot while, in the same time, putting it in front of the human.

Lastly, data fusion from a wearable IMU sensor and an LRF is used in [7], in order to estimate the user linear/angular velocities and pose. The authors exponentially stabilize the human-robot pose error using an inverse kinematics controller. The setup is experimentally validated is various paths.

III. DYNAMIC WINDOW ARC-LINE PLANNER

The core problem of the front-following behaviour lies with the handling of *undecidable* situations. A primary example is a crossroads (Fig.2 left). In this scenario, as the robot moves in front of the user, when it arrives at the crossroads it detects completely disjoint and distinct routes . This presents an undecidable situation for the robot, which must be resolved by the human. This *intention* resolution must be performed in a timely manner, as the robot risks invalidating feasible routes. Another example is a T-Junction (Fig.2 right). In this case the risk of collision is exacerbated by the fact that, in the time needed to resolve the user intention when the "left" and "right" routes are detected, the robot might be in a "limbo" state, moving further into the junction and either making the routes infeasible or, worse, hitting the wall.

These two examples reveal a crucial condition; the available routes depend on the environment geometry *as well as* the robot velocity. As such, the planer must take the robot dynamics into account. For example, in the crossroads the robot might be moving *too fast* to make the "left/right" turns, and the only feasible direction would be the "front" route.

In contrast, when moving inside a corridor the available paths vary continuously and the planner must select the best one, using a scoring function. The identification of undecidable situations is the first key step for the "front-following" behaviour. This undecidability must, consequently, be *resolved*. Thus user-intent identification is the second key step.

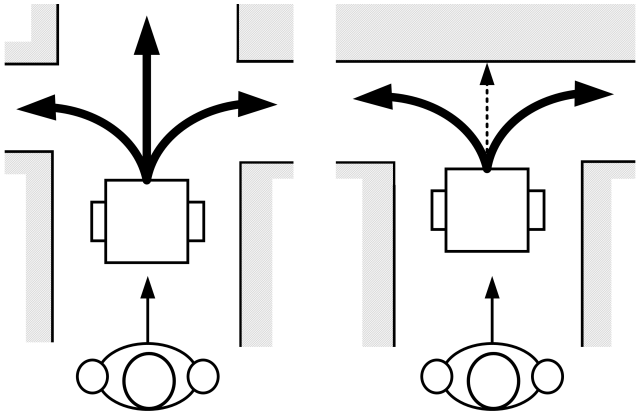

Fig. 2. Example of an undecidable areas; a crossroads (left) and a T-Junction (right).

Intuitively, an undecidable area is characterized by the availability of more than one different "routes". These routes can be described using the notion of *path homotopy*. Two paths, starting and ending from the same points, are called *path homotopic* if one can be deformed to the other in a continuous manner, while not colliding with any obstacles [8]. This standard definition seems to be rather restrictive for *local* path planning because firstly, an ending/goal point is not given and secondly, the path is allowed to deform in any wild fashion hence resulting in paths which might not satisfy the robot's differential motion equations. If we relax the definition and allow the two paths to have different ending points, while simultanesouly we impose constraints on the path characteristics e.g. bounds on the curvature, a more general *path equivalence* can be considered [9]. In this work a new simple way to produce path equivalence classes is presented. This can be performed on-line using a modified dynamic window approach (DWA). The advantage of the proposed planner is that it produces geometrically concise classes (denoted as "path clusters" in the following), and that it allows the construction of simple metrics which convey useful information e.g. span, path mean and median etc.

The Dynamic Window Approach (DWA) [10] is a popular kinodynamic local planner. It searches in the input space (v, ω) for collision-free paths. Given a velocity tuple (v_R, ω_R) of the robot, paths are sampled from the window $[v_R \pm \Delta T \alpha_{vmax}, \omega_R \pm \Delta T \alpha_{\omega max}]$, i.e. the *dynamic window* (Fig.3) , and simulated forward in time. Then the optimal one is selected. The DWA essentially produces *arcs*.

For the front-following task, arcs are a rather unsuitable candidate as we want to check for distinct "openings" in the workspace, not just for obstacle-free paths. We thus propose to test *arc-line* paths instead, that is, arcs that are connected to a straight line at their end. This simple modification is a more intuitive solution for finding available directions of motions than arcs, in indoor environments.

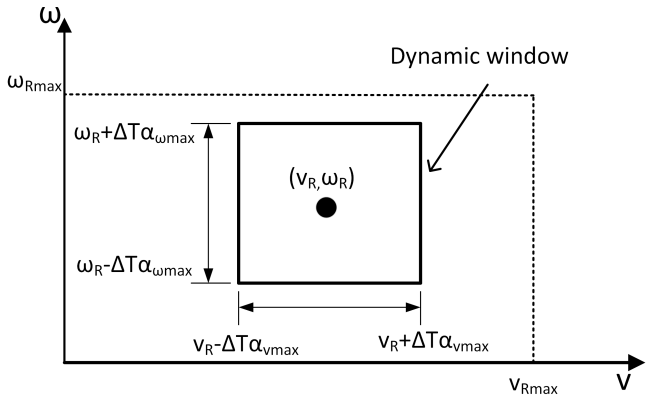

Fig. 3. Depiction of the Dynamic Window

A. Path Calculation

Consider the attainable *curvatures* for the current velocity tuple (v_R, ω_R) in the dynamic window. Denote,

$$\begin{aligned} v_{max} &= v_R + \Delta T \alpha_{vmax} \\ v_{min} &= v_R - \Delta T \alpha_{vmax} \\ \omega_{max} &= \omega_R + \Delta T \alpha_{\omega max} \\ \omega_{min} &= \omega_R - \Delta T \alpha_{\omega max} \end{aligned} \tag{2}$$

the maximum (minimum) velocities in the dynamic window.Then, the maximum and minimum curvatures are,

$$\begin{aligned} \kappa_{max} &= \begin{cases} \omega_{max}/v_{min} & ,\omega_{max} \geq 0 \\ \omega_{max}/v_{max} & ,\omega_{max} < 0 \end{cases} \\ \kappa_{min} &= \begin{cases} \omega_{min}/v_{min} & ,\omega_{min} \leq 0 \\ \omega_{min}/v_{max} & ,\omega_{min} > 0 \end{cases} \end{aligned} \tag{3}$$

Now consider that all generated paths reach a distance R from the robot, i.e. they intersect a circle of radius R. This implies that all their terminal points lie on this circle (Fig.4).

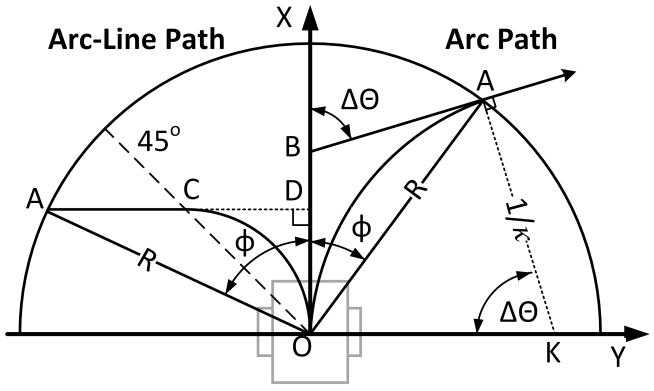

Fig. 4. Calculation of an arc (right) and arc-line (left) path.

Let XOY be a frame aligned with the robot at O. To calculate the arc paths, consider the right side in Fig.4. The robot traces an arc *OA* of curvature κ, and assumes a heading $\Delta\Theta$. Since OB and AB are tangential segments of the same circle, it follows that $OB = AB$ and thus the triangle $\triangle OAB$ is isosceles. Summing up the triangle's angles we get,

$$\phi = \Delta\Theta/2 \tag{4}$$

Taking the arc perpendicular at A, we find the centre K of the circle it is part of. It is clear that its radius is $1/\kappa$ and the subtended angle from K by the arc is $\Delta\Theta$. Since OA is the chord of the arc, its central angle is given by,

$$\Delta\Theta = 2 \arcsin \kappa R/2 \tag{5}$$

and using (4),

$$\phi = \arcsin \kappa R/2 \tag{6}$$

For the calculation of the arc-line (AL) paths (Fig.4 left), we attach a tangent straight segment at the endpoint of each arc (point C), which is *parallel to the y-axis*. By inspection of the figure, we see that AL paths satisfy $\Delta\Theta = \pi/2$. Using (4) we see that all AL paths comply to $\phi = \pi/4$, hence they terminate on the $45\deg$ line, passing from O. The relation between the curvature κ and the angle ϕ in AL paths, can be calculated noting,

$$\left.\begin{aligned} |OD| &= R\cos\phi = |OC|\cos\pi/4 \\ |OC| &= 2\frac{\sin\pi/4}{\kappa} \end{aligned}\right\} \Rightarrow R\cos\phi = \frac{1}{\kappa} \tag{7}$$

and thus,

$$\kappa = \frac{1}{R\cos\phi} \Longleftrightarrow \phi = \arccos\frac{1}{\kappa R} \tag{8}$$

B. Clustering

The *path bundle* is produced by sampling the circle with a fixed angular interval $\Delta\phi$, giving a chord $R\Delta\phi$ of $0.1m$. The min and max of ϕ are,

$$\phi_{min(max)} = \arccos\frac{1}{\kappa_{min(max)} R} \tag{9}$$

Two LRFs are mounted on the robot; the first facing backwards toward the human and the second facing forwards. The latter is used to build a robot-centred rolling costmap which is used to produce a 2D occupancy grid. This way, each cell in the grid is assigned a cost (for more details see *http://wiki.ros.org/costmap_2d*). The edges of each cell are set to a length of $0.1m$.

The planner constructs each path by sampling each arc according to a fixed resolution ($0.1m$ in our experiments). Concurrently each generated sample is tested for collision with obstacles in the costmap. In the case that the sample collides, only the path up to that point is preserved (Fig.5). Each path is attributed with the *minimum cost* among the cells it crosses. If the path collides with an obstacle, the cost is a predefined constant.

Clustering of the path bundle is performed by firstly discarding all colliding paths. Following, we define two parameters

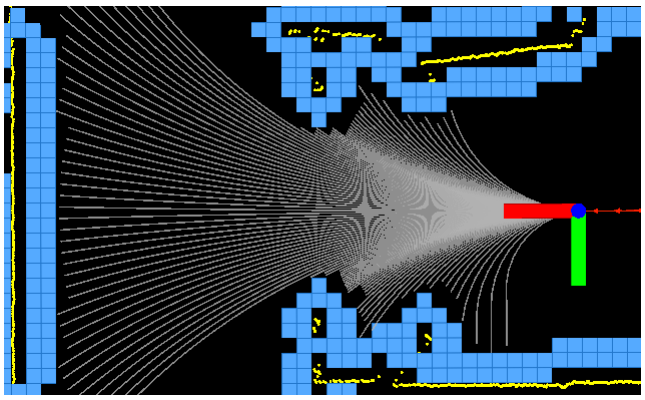

Fig. 5. Path bundle for a T-Junction. Obstacles in the costamp can be seen in cyan. Yellow are the laser points while the AL paths are white. The frame of the Robot is seen as thick red and green lines. Here, the radius is set to $R = 4m$.

which denote the *cluster separation* C_{sep} and the *minimum span* W_{span} of a cluster. Two paths $P_i, P_j (i > j)$, belong to the same cluster if $i - j \leq C_{sep}$. By checking this condition for all successive paths, we produce cluster $C_{k,l}, l > k$, where P_k is the first path of the cluster and P_l the last. By using (8),(6), each path can be associated with its curvature κ and angle ϕ. The *cluster span* $W_{k,l}$ is the chord length of the angle $\phi_l - \phi_k \equiv \Delta\phi_{k,l}$, and given by,

$$W_{k,l} = 2R \sin \frac{\Delta\phi_{k,l}}{2} \tag{10}$$

A minimum cluster span $W_{span} = 0.2m$ is imposed to avoid noise clusters. Smaller clusters are discarded. Each cluster is "represented" by an "optimal" path P_C, which has the lowest cost among the paths of that cluster. If there are sub-clusters within the cluster with equal minimum scores, we take the last one and evaluate the *mean angle* ϕ_C of the (sub)cluster angle. Using (8)(6) we get its corresponding curvature κ_C, called the *cluster curvature*.

C. Multiple Levels

As we can see from Fig.6, the planner produces two clusters as it approaches the T-Junction. In that moment, the robot has to "signal" the user that it has detected an undecidable area and is reading his/her intention. However, the question arises as to what to do until the intention is resolved. It is evident that the robot must, in parallel, scan its immediate area ahead and create *feasible* motion clusters for that short period of time. To accompany this, we define two scanning circles, called *levels*, with different radii R_{far} and R_{near} (set to $4m$ and $2m$ resp. in our study). The *near level* is seen in Fig.6, comprising a single cluster (in red). Until the robot resolves the user intention, it switches from the *far level* to the *near level* for motion commands. Simultaneously, it enters into the "intention estimation" mode and tries to discern in which *far cluster* the user is heading to. Upon resolution, it discards the *near level* and uses again the *far level* for motion.

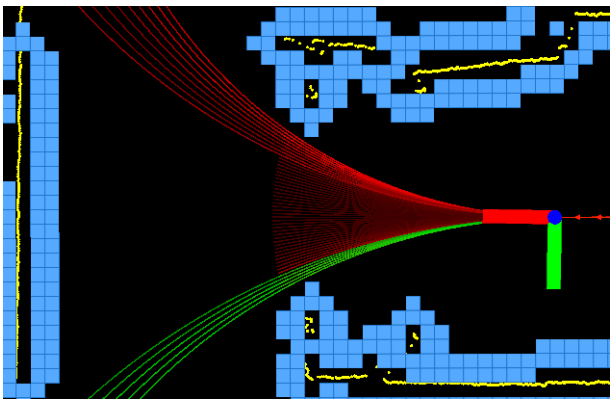

Fig. 6. Far and Near clusters for a T-Junction.

IV. INTENTION ESTIMATION

In the context of the front-following task, intention estimation is referring to the selection of the user-intended *far* cluster, in the presence of undecidability. To this end, consider a reference frame rigidly attached the rear LRF, facing the human (Fig.7), and let (x_H, y_H) be his/her position. Define the *human angle* ϕ_H as,

$$\phi_H = \begin{cases} 0 & ,|y_H| < \epsilon \\ K_\phi \text{sgn}(y_H)(|y_H| - \epsilon) & ,|y_H| \geq \epsilon \end{cases} \tag{11}$$

where ϵ is a *deadband* and K_ϕ an appropriate constant.

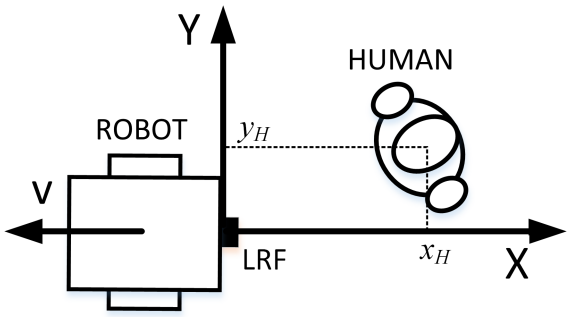

Fig. 7. Human position in the Laser frame.

Using (11), the user essentially selects an angle on the circle of the *far level*. Now consider that there are N *far clusters*, $C_i, (i = 1 \ldots N)$, with respective cluster angles ϕ_{Ci}. During the intention estimation, the robot assigns scores S_i to the clusters, and increases them by picking the one closest to ϕ_H based on their angle. The selected cluster has its score incremented, by adding a vote as follows,

$$V = \begin{cases} 1 & ,|\phi_{Ci}| < \alpha \\ 1 + 2(|\phi_H| - \alpha)/(\beta - \alpha) & ,\beta > |\phi_{Ci}| \geq \alpha \\ 3 & ,|\phi_{Ci}| \geq \beta \end{cases} \tag{12}$$

where α, β are angle constants. The voting mechanism in (12) assigns a lower vote to clusters that are "ahead" than

clusters that are "on the side". This has been selected because, if the user wants to promote side clusters, he/she will have to step *away* from his/her normal walking direction, in order to increase y_H. Such a motion is improbable to have been performed by chance and it is most likely a deliberate user action. Hence, when the robot detects it, the "confidence" is high and the selected cluster is quickly promoted.

After each iteration, the robot selects the top two scoring clusters and compares their scores. If the top score is 50% bigger than the second one, the algorithm terminates and outputs the top cluster. A second condition is that the clusters must have *at least* a predefined number of votes (currently 10). If these conditions aren't satisfied after a *timeout* e.g. $3sec$, it picks the top cluster and exits.

V. CONTROLLER

Using the curvature κ_C of the selected cluster, a geometric path is presented to the robot, which is safe by design, as all paths comprising a cluster are non-colliding. The robot receives a motion command consisting of the velocities (v_R, ω_R) described in (1). The user controls directly the linear velocity of the robot using the following velocity function,

$$v_R = \begin{cases} 0 & , \quad x_H > x_0 \\ k_1(x_H - x_0) & , \; x_2 \leq x_H \leq x_0 \\ v_{walk} & , \; x_1 \leq x_H \leq x_2 \\ v_{Rmax} - k_2 x_H & , \; 0 \leq x_H \leq x_1 \end{cases} \tag{13}$$

Equation (15) describes a piece-wise profile of the robot's linear velocity, consisting of three regions on the *x-axis*; the *approach region*, the *walking region* and the *collision region*, delineated by constants x_0, x_1 and x_2. In the walking region the robot moves with a constant linear velocity, i.e. the *walking velocity* v_{walk}. In this region the human and the robot move in sync. In the case that the user moves very close to the robot, the *collision region* is activated, making the robot accellerate up to the maximum velocity v_{Rmax}. On the other hand, if the user lags behind, he/she enters the *approach region*, making the robot decceleate down to a halt. This velocity controller is based on [11]. The control of the angular velocity ω_R is performed by combining the linear velocity and the curvature of the cluster viz.,

$$\omega_R = \kappa_C v_R \tag{14}$$

Thus the robot selects the available *path* on which to move, while the human controls the linear velocity for traversing it.

VI. EXPERIMENTS

The intention-based front-following controller described in the previous, has been implemented in ROS and ran from a laptop mounted on a Pioneer P3-DX robot. Two Hokuyo UBG-04LX-F01 laser range finders, one facing forwards and one backwards towards the user's legs, were also fixed on the robot frame. The aim of the experiments was to assess the users walking pattern with and without the robot following them from the front. The assistive controller is compared to a teleoperation-like approach utilizing a kinematic controller presented in [11] by the authors. In that work, comparison was made between the kinematic approach and the human baseline gait with no robotic assistive behaviour. A major finding was that the users tend to "drive" the robot to the paths *they* would take when walking normally.

For the baseline experiments, seven healthy subjects were asked to walk normally from an initial predefined position, take a left turn at the junction and stop at a designated position. Each subject performed two runs, thus in total 14 baseline paths were collected. Following, eight subjects performed $2 \times 8 = 16$ runs using the kinematic controller and $2 \times 8 = 16$ runs with the intention-based assistive controller. The traces of the human and the robot are shown in Figs.8 and (9).

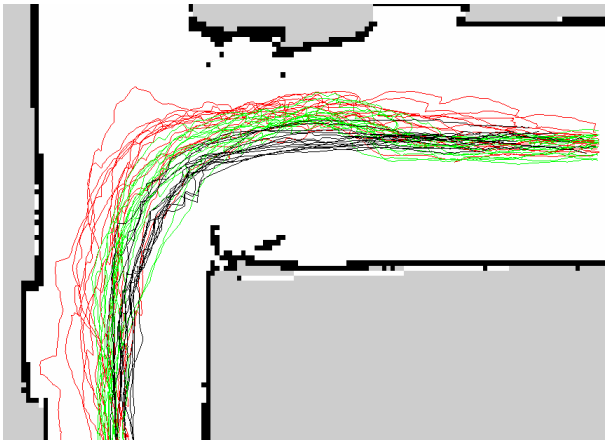

Fig. 8. Human traces. Kinematic(Red), Assistive(Green) and Baseline(Black) trials. The users started from the right and progressed to the left and down.

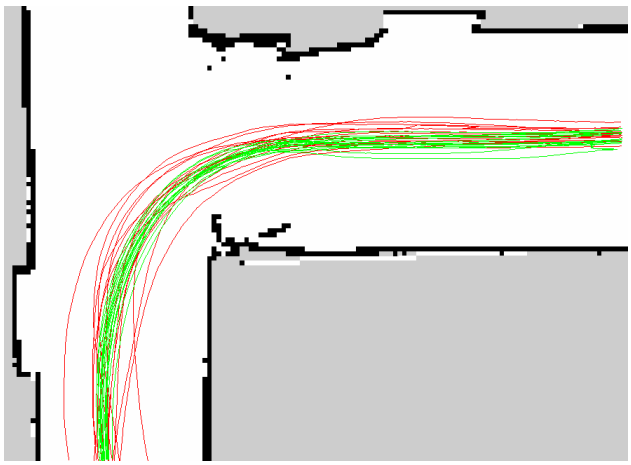

Fig. 9. Robot traces. Kinematic(Red) and Assistive(Green) trials.

In order to perform an analysis of the various paths, the plane was divided into a 48×26 grid of square cells, each with 20cm edge. If we count the number of paths crossing each cell, we can create a 2D histogram for each experiment. In total we have five histograms presented in Figs.(10,11,12,13 and 14) given by,

$$
\begin{aligned}
H_B(i,j) &: \text{Baseline paths} \\
H_{UK(UA)}(i,j) &: \text{User paths kinematic (assistive)} \\
H_{RK(RA)}(i,j) &: \text{Robot paths kinematic (assistive)}
\end{aligned} \tag{15}
$$

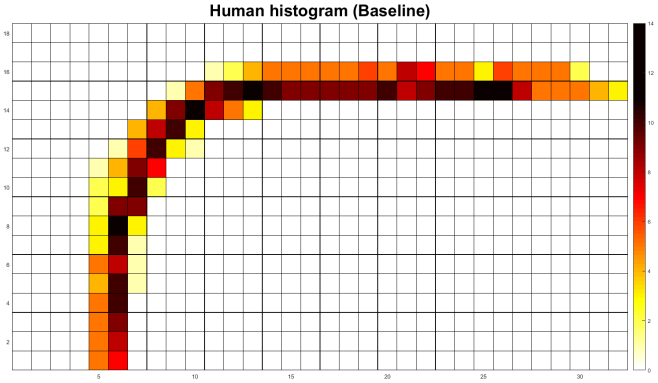

Fig. 10. Histogram of the Human baseline paths.

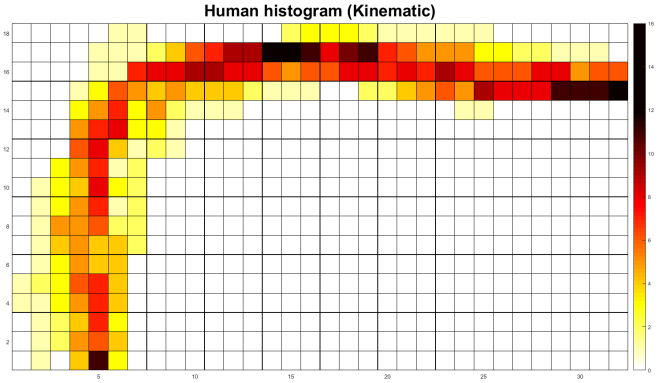

Fig. 11. Histogram of the Human kinematic paths.

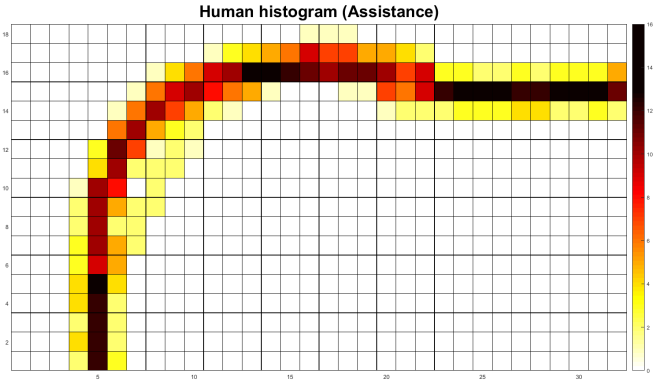

Fig. 12. Histogram of the Human assistive paths.

By dividing each cell count with the total number of paths, we get the probability of a cell being traversed by a path, viz,

$$T_*(i,j) = H_*(i,j)/20 \tag{16}$$

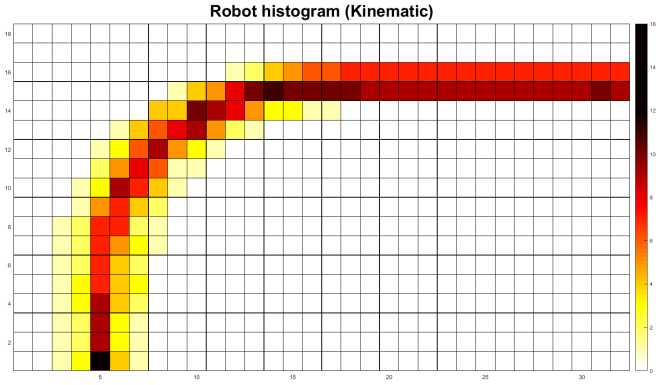

Fig. 13. Histogram of the Robot kinematic paths.

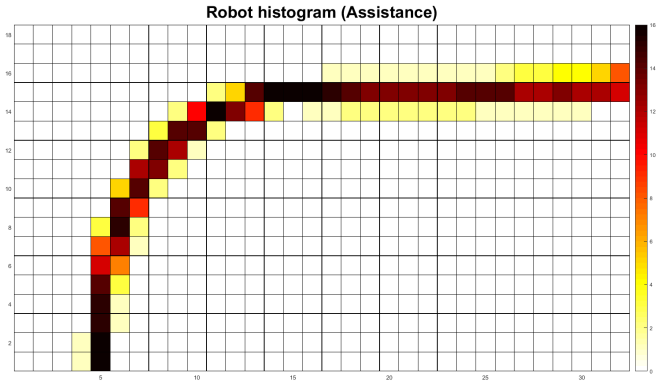

Fig. 14. Histogram of the Robot assistive paths.

TABLE I
HELLINGER DISTANCE OF HUMAN-ROBOT DISTRIBUTIONS

$H(P_{UK}, P_B)$	$H(P_{UA}, P_B)$	$H(P_{RK}, P_B)$	$H(P_{RA}, P_B)$
0.618	0.494	0.294	0.393

A large probability means that the cell is traversed by a large number of paths. A different group of distributions can be produced by dividing each cell with the sum of its respective histogram, i.e.

$$P_*(i,j) = H_*(i,j) / \sum_{i,j} H_*(i,j) \tag{17}$$

These are probability density functions which encode the probability of a user/robot being on a specific cell. To compare the three groups, we resort to the *Hellinger distance* which is a measure of statistical distance between two distributions *P, Q* given by,

$$H(p,q) = \frac{1}{\sqrt{2}} \sum_k (\sqrt{p_k} - \sqrt{q_k})^2 \tag{18}$$

The Hellinger distance ranges from zero to one, with zero being identical distributions and one completely disjoint. The distances between the Human-Baseline and Robot-Baseline distributions are presented in Table I.

TABLE II
CELL COUNT OF THE DISTRIBUTIONS.

	P_{UK}	P_{UA}	P_{RK}	P_{RA}	P_B
count(cells)	178	134	129	93	90
rel.diff. to $P_B(\%)$	97.7	48.8	43.3	3.33	0

We see that the paths in the Human assistive case are almost 25% closer to the paths in the Human Baseline case than the ones in the kinematic case. This implies that when humans use the assistive controller, they deviate less from their natural gait. A second observation regards the robot paths. The robot kinematic distribution has the smallest distance. This reconfirms the finding in [11], where the user actually tend to "drive" the robot the their own optimal path. The distance of the robot assistive paths can be attributed to the automated motion generation of the planner, leading to paths which do no adhere to some native optimality condition found in human locomotion.

Complementary to the Hellinger distance, we measure the dispersion of the distributions using the active cell count by counting the number of cells traced by the paths. Taking the relative differences between the counts of the distributions and the baseline, we can quantify the "ease" of using the kinematic and assistive modes. If the difference is high, the users tend to walk around trying to "drive" the robot.

From Table II we see that the count for the human kinematic paths is double the one in the assistive case. This shows that the intention-based controller measurably "offloads" the cognitive burden of transferring the human intention to the robot by providing collision avoidance, local planning and intent recognition. Even so, the count in the human assistive case is almost 50% higher than the baseline. An explanation for this is that in order to show the human intention to the robot, the human must (unnaturally) deviate from his path and signal a turn. This apparently increases the dispersion of the distribution. Interestingly, the robot paths in the assistive case have a *smaller* dispersion than in the baseline case. This also can be associated with the consistency of the generated paths by the planner.

VII. CONCLUSIONS

This paper describes an intention-based assistive controller for the front-following task for a mobility assistance robot. The controller uses a novel kinodynamic planner which concurrently calculates discrete routes of motion as well as obstacle-free geometric paths. This allows the user to operate indoors in an obstacle laden space. Experimental result reveal that, in comparison with a direct kinematic controller, the assistive controller enables the user to walk with a more naturally gait. It is also shown that the intention-recognition algorithm successfully estimates the human intent and reduces the cognitive load of the task.

ACKNOWLEDGMENT

This research work has been partially supported by two EU-funded Projects: MOBOT (FP7-ICT-2011.2.1, grant agreement no. 600796) and I-SUPPORT (H2020-PHC-19-2014, grant agreement no . 643666).

REFERENCES

[1] R. Gockley, J. Forlizzi, and R. Simmons, "Natural person-following behavior for social robots," in *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, ser. HRI '07. New York, NY, USA: ACM, 2007, pp. 17–24.

[2] A. Cosgun, D. A. Florencio, and H. I. Christensen, "Autonomous person following for telepresence robots," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 4335–4342.

[3] A. Ohya and T. Munekata, "Intelligent Escort Robot Moving together with Human-Interaction in Accompanying Behavior," in *Proceedings of the 2002 FIRA Robot Congress*, 2002, pp. 31–35.

[4] D. Ho, J. S. Hu, and J. J. Wang, "Behavior control of the mobile robot for accompanying in front of a human," in *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Jul. 2012, pp. 377–382.

[5] E. J. Jung, B. J. Yi, and S. Yuta, "Control algorithms for a mobile robot tracking a human in front," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 2411–2416.

[6] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz, "On the Nonholonomic Nature of Human Locomotion," *Auton. Robots*, vol. 25, no. 1-2, pp. 25–35, Aug. 2008.

[7] C. A. Cifuentes, A. Frizera, R. Carelli, and T. Bastos, "Humanrobot interaction based on wearable IMU sensor and laser range finder," *Robotics and Autonomous Systems*, vol. 62, no. 10, pp. 1425–1439, Oct. 2014.

[8] J. Munkres, *Topology*, 2nd ed. Upper Saddle River, NJ: Pearson, Dec. 1999.

[9] R. A. Knepper, S. S. Srinivasa, and M. T. Mason, "An Equivalence Relation for Local Path Sets," in *Algorithmic Foundations of Robotics IX*, ser. Springer Tracts in Advanced Robotics, D. Hsu, V. Isler, J.-C. Latombe, and M. C. Lin, Eds. Springer Berlin Heidelberg, 2010, no. 68, pp. 19–35.

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

[11] G. P. Moustris, A. Dometios, and C. Tzafestas, "User front-following behaviour for a mobility assistance robot: a kinematic control approach," in *International Conference on Integrated Modeling and Analysis in Applied Control and Automation (IMAACA'15)*, vol. 1. CAL-TEK SRL, 2015, pp. 142–149.