EU Horizon 2020 Research & Innovation Program
Advanced Robot Capabilities & Take-Up
ICT-25-2016-2017
Grant Agreement No. 779942

CROWDBOT

Safe Robot Navigation in Dense Crowds

http://www.crowdbot.org

Technical Report

D 3.6: Shared Control Navigation

Work Package 3 (WP 3)

Navigation

Task Lead: University College London (UCL), UK

WP Lead: Swiss Federal Institute of Technology (ETHZ), Switzerland

Table of Contents

Executive Summary

Work Package 3 of the CrowdBot project focuses on navigation. Half of our prototype crowdbots (Pepper and cuyBot) are designed to be fully autonomous and so the navigation algorithm must deal with both global and local aspects of planning. However, the other two crowdbots (smart wheelchair and Qolo) are designed to support human users improving their quality of life. In this case, the human's intention is the high-level global navigation plan, but they may need assistance in planning and executing the local navigation at the operational level. Deliverable 3.6 focuses on the paradigm where the user's continuous control input is safely blended with the robot's local planner in a process we call Shared Control. Within Work Package 3 of the CrowdBot project, the UCL team has concentrated on refining two primary algorithms for shared control (probabilistic dynamic window approach and user intent modelling), both in terms of gathering data in real pedestrian environments and improving performance and user experience in simulation.

Probabilistic shared control using the dynamic window approach and generalised velocity obstacles (PSC-DWAGVO) was selected as an approach due to identified weaknesses in conventional DWA algorithms when it came to avoiding moving pedestrians. Combining the velocity obstacle paradigm with PSC allows the wheelchair to select a trajectory that is most likely to satisfy the user and simultaneously preserve safety, without the risk of freezing in place due to a rigid avoidance strategy. In order to support the development of this algorithm, a large, novel dataset of wheelchair-pedestrian interactions was collected, filling a vital gap in the available data. This dataset was used to create a simulated testing environment with realistic 'pedestrian' agents that allows the algorithm to be rapidly iterated and tested in the absence of human participants (this has become of even more crucial importance during the current pandemic).

Section 2 gives an overview of our custom built smart wheelchair platform and other shared control wheelchairs in the literature. Then, the motion planning and blending strategy is discussed in the Related Work Section 3. First, to give background context, the standard Dynamic Window Approach (DWA) motion planning algorithm is presented followed by the Generalised Velocity Obstacles (GVO) extension for environments with moving obstacles (crowds of people in our case). We then explore the baseline linear blending strategy and its extension to Probabilistic Shared Control (PSC), which formulates blending within a statistical framework and sets the tone for our approach.

Shared control navigation using Probabilistic Shared Control (PSC) is presented in Section 4. Our proposed model is a hierarchical framework for collision avoidance, which treats static and dynamic obstacles separately thus allowing for more flexibility and transparency in control, whilst improving computational performance. The first layer runs DWA to construct sets of Reachable Admissible Velocities (RAV), the second layer formulates Generalised Velocity Obstacles, and the final layer performs the actual blending using PSC, resulting in the final trajectory output.

The above-mentioned model is implemented and validated in a Unity3D ROS simulation environment containing the smart wheelchair platform with static and moving obstacles. All sensors' characteristics apart from the RGBD camera are modelled. In our experimental results, we focus on three core metrics, which stem from Deliverable 1.3 namely: number of collisions, task completion time and "agreement" (between the user and the robot). Our results show that the proposed approach is a good step towards enabling a shared control wheelchair to navigate safely in a highly dynamic and crowded environment.

For 1D and 2D flow of crowds, in our simulated PAMELA facility, results indicate that in a crowd-robot navigation task, our shared control navigation strategy has a small, non-significant, effect on the crowds. This is expected as the current strategy does not aim to minimize such effects.

Section 5 discusses a comparison between our real-world wheelchair-crowd experiments and simulated experiments at PAMELA. Here, the wheelchair was driven manually without shared control and we show that the simulator does reproduce the real-world experiment well.

Section 6 details how we can learn to predict user intent (as represented by their joystick input) from the available visual input. Using data from healthy volunteers, a nonlinear mapping was learned between RGBD camera sequences and user intent as approximated by joystick inputs. A user model of this kind may in future be used to reconstruct how a user with a condition such as hemineglect (which prevents them from perceiving one half of their visual field) would behave if they had access to their unimpaired senses, information which is required for shared control.

1. Introduction

Being able to move freely is key to independence and quality of life. While wheelchairs provide a mobility solution, current designs may not be suitable for people with very high-level motor impairments, especially when combined with sensory and / or cognitive impairments. To address these people's mobility needs, researchers have been working on 'smart wheelchairs'. A smart wheelchair is typically made of a standard powered wheelchair and a collection of sensors for perception purposes. It is able to provide driving assistance to the user by planning and following a collision free path either globally or locally [1]. Based on the level of autonomy, smart wheelchairs can be divided into two categories: fully autonomous [2], [3] and shared control (semi-autonomous) [4]–[6]. Fully autonomous wheelchairs are effective in achieving high-level goals for indoor and outdoor navigation [7], [8]. The user only needs to indicate their preferred destination and the path planner will plan and navigate the wheelchair to the place without collision. However, this type of operating mode ignores users' capabilities and their short-term intentions. In addition, Biddiss et al. (2007) indicated that most end-users do not want to feel useless and expressed their desire to retain as much control as possible [8]. As a result, the control paradigm for shared control wheelchairs could potentially be appreciated by more users due to its collaborative characteristic.

While the research on shared control wheelchairs spans decades there are still challenges in developing navigation strategies for dynamic environments such as crowds. In particular, we have focused on addressing the weaknesses of existing Dynamic Window Approach (DWA)-based shared controllers, namely their poor performance in areas containing many moving obstacles. In support of this, we have begun collecting data on how human pedestrians react in the presence of an instrumented wheelchair, filling a crucial gap in the existing body of data.

In parallel, we have collected data on how unimpaired users control the wheelchair in realistic crowd scenarios. This has allowed us to begin learning predictive models for user intent based on the sensory input available to the smart wheelchair (in particular, visual input captured by an RGBD camera). This aims to address a vulnerability of shared control methods where meaningful assistance cannot be provided in circumstances where the user's intent is unknown (such as in areas that the user is unaware of due to sensory impairment).

2. Shared Control Wheelchairs

A shared control wheelchair is normally built from a standard powered wheelchair with a collection of sensors. It has the ability to sense its environment and make collision-free actions based on the user's input. In general, a shared control wheelchair consists of three essential components: The user's input, a motion/path planner and the control strategy.

In terms of the control strategy, shared control wheelchairs can mainly be divided into two categories. The first category puts the user at the tactical level: Escobedo et al. (2013) and Tomari et al. (2012) adopted a hierarchical framework where the user was in charge of high-level tasks such as choosing a desired manoeuvre whilst the wheelchair took control of the low-level tasks [9], [10]. The user could intervene in the autonomous process by switching the operating mode. Similarly, Simpson et al. (1999) proposed an automatic adaptation strategy, where the motion planner can automatically select the correct operating mode [11]. However, this strategy only assists with driving in specific, predefined scenarios and may not be suited to dealing with uncertainty.

An alternative control paradigm, where the user is considered at the operational level, involves continuously blending input from both the user and the motion planner [5], [6], [12], [13]. For this type of shared control strategy, the wheelchair will not move unless input commands from both parties are received. This characteristic allows the user to retain greater control authority and allows more collaboration between the user and the wheelchair, which seems preferable in the crowd scenario. As a result, we adopt continuously blending shared control as the basis for our shared control strategy.

2.1. Our Smart Wheelchair Platform

The full details of our smart wheelchair platform are given in Deliverable 5.3, however in order to better understand our design structure and experiments, we briefly recap the core setup here. As shown in Figure 1, our wheelchair is built on a standard Salsa M2 electrical wheelchair with a collection of additional sensors. Both hardware and software modifications were made to the platform, including the addition of an RGBD RealSense D435 camera mounted on an adjustable rear sensor frame, a Hokuyo 2D Lidar mounted on the footrest, and ultrasonic sensor clusters mounted on the base corners as shown in Figure 1. We have also equipped the wheelchair with two wheel encoders and one inertial measurement unit (IMU) which are used for dead reckoning. The Robot Operating System (ROS) is used to integrate these sensors, compute their relative transforms, and publish the relevant sensor data as ROS topics. The high-level data processing of the Lidar, RGBD camera and shared autonomy model is performed on an Intel i9 core laptop with 16 GB of RAM and a GeForce GTX 1080 GPU. Low-level data processing of the ultrasonic clusters is performed on dedicated Arduinos and an Odroid Single Board Computer.

The wheelchair is driven via a joystick user interface and can accelerate up to approximately 5m/s. Additional circuitry (and a software module) was also added to be able to directly read this joystick user input and also supply a virtual (modified) joystick input to the robotic wheelchair's motor controller thus allowing it to be driven programmatically. The joystick data is encoded using two axes that capture the translational and rotational velocity inputs. We also developed a 1:1 model of our wheelchair in Unity3D for our experiments in the CrowdBot simulator (see Figure 2).

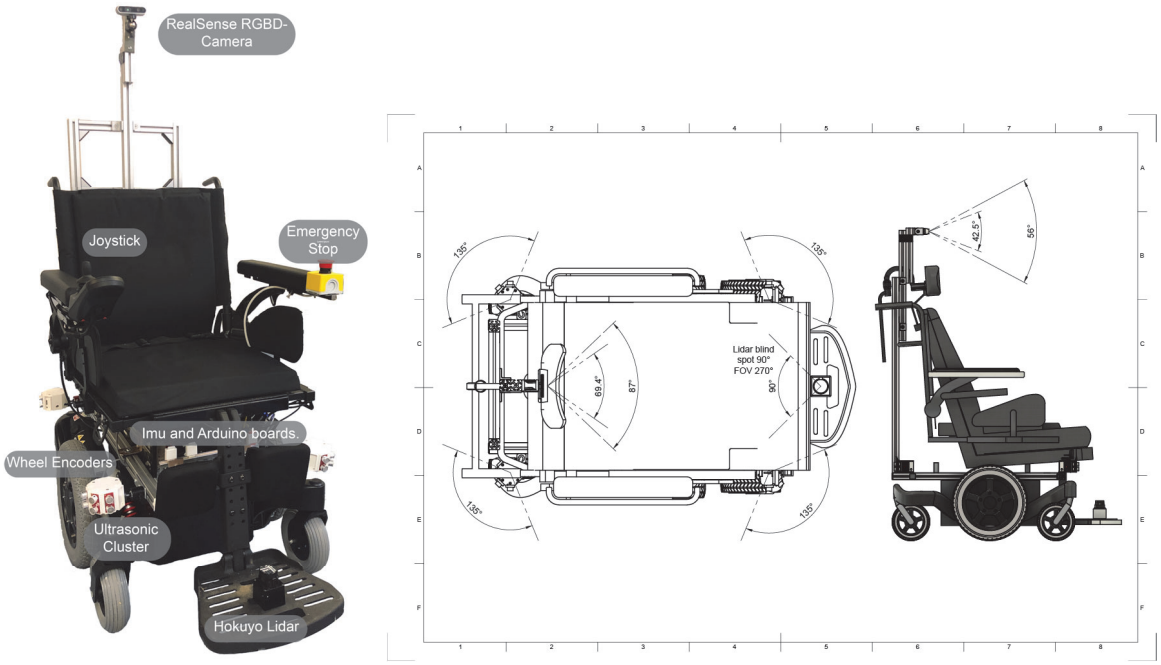

Figure 1. Our custom modified robotic wheelchair platform with mounted RGBD camera, Lidar, IMU, and ultrasonic sensor clusters. On-board electronics such as the single board Odroid computer and processors such as Arduino interface boards are placed under the seat. The sensor coverage is also illustrated**.**

Figure 2. The 1:1 model of our wheelchair in Unity3D for our simulation experiments.

3. Related Work: Planning and Blending

3.1. Motion Planner

One of the essential components of a shared control wheelchair is its motion planner. A detailed planning strategy for other robots such as Pepper was included in Deliverable 3.1. However, the main difference between our shared control wheelchair and fully autonomous mobile robots like Pepper, is that no global planner is used. Due to the nature of shared control at the operational level, it is not necessary to provide a global path as a human driver is involved in the navigation process. In reality the person's short-term global goal often changes during the navigation tasks, according to who they meet along the way and any other personal desires. Thus setting a fixed global path planner may not best reflect the user's intention and would limit the user's interaction with the system.

Therefore, our focus is instead on low latency local planning. In general, there are two types of local motion planner, a reactive planner and a deliberative planner. A reactive planner decides the best action based on the real-time sensor readings, while a deliberative planner uses prior information about the environment to make plans. In the context of this project, the wheelchair is expected to react quickly and safely in an unknown dynamic environment with high uncertainty, thus a reactive planner is more suitable. However, the common downside for reactive navigation is that we may reach a sub-optimal solution and there is no guarantee that the exact goal will be achieved.

Reactive planning has been studied extensively in the context of smart wheelchair navigation. In general, it is either trajectory (path)-based or velocity-based. A trajectory-based approach includes an artificial potential field [14] which was first presented by Khatib et al. (1986) and has been widely adopted for mobile robot navigation [15], [16]. This approach assumes the goal and obstacles act like charged surfaces and the potential creates an imaginary force which attracts the robot to a goal while repelling it from the obstacles. As a result, the robot will move towards the goal while avoiding obstacles. However, the robot may get stuck when the local minimum of the motion equation is reached.

Another method which has been widely used in robot navigation is the vector field histogram (VFH). Introduced by Borenstein et al. (1991), VFH was first used for robot navigation with ultrasonic sensors using dead reckoning [17]. It was further adapted to be used with Lidar in [18]. VFH is a grid-based method, which starts by constructing a 2D histogram of occupancy based on the sensor information. It is then converted to a 1D histogram with polar coordinates. The valley in the histogram demonstrates the direction of free space while the peak represents the direction of obstacles. As a result, a robot will move in the direction of free space with a speed that is proportional to the distance from obstacles. This method allows the sensor error to be considered but does not account for the robot's dimension and kinematics. As a result, Ulrich et al. (1998) introduced VFH+ which extended VFH to deal with robot kinematics constraints and was able to generate smoother trajectories [18]. However, VFH+ may sometimes fail to choose the most appropriate direction due to its purely local nature as indicated in [19]. This is mainly because the robot plans its motion only based on current sensor inputs while ignoring future consequences. This issue is further addressed in another variant of VFH called VFH*, which projects the robot's trajectory several steps ahead and selects the most suitable candidate direction by minimizing a cost function [19].

In addition to these trajectory-based methods, velocity-based reactive planners have also been explored extensively in robot navigation. The most widely used planner is the dynamic window approach (DWA) which was proposed by Fox et al. in 1997 [20]. It has been applied in shared control navigation for non-holonomic robots as well as wheelchairs [5], and demonstrated its effectiveness in avoiding static obstacles. However, naïve DWA does not take obstacle velocity information into account and thus is not well suited to dynamic environments. This issue is addressed by other velocity-based approaches such as velocity obstacle (VO) [21] and optimal reciprocal collision avoidance (ORCA) [22], which have been used to avoid moving obstacles.

The rest of this section will briefly recap DWA and its variants, followed by VO and its extensions. This design choice is consistent with Deliverable 3.1, where DWA and VO have been used in other robots for the purpose of local planning.

3.1.1. Dynamic Window Approach

The Dynamic Window Approach (DWA) which was first proposed by Fox et al. is an obstacle-avoidance method that takes into account the dynamic and kinematic constraints of a mobile robot [20]. DWA can be

implemented in two steps. It starts with defining the suitable search space. DWA uses linear velocity and angular velocity pairs *(v,w)* to describe all the circular trajectories, which reduces the search space to a two-dimensional velocity space. Among all the velocity pairs, a set of admissible velocities which guarantee safety will be computed. A *(v,w)* pair is considered admissible if the robot is able to stop before it reaches the closest obstacle in its selected trajectory [20]. Among all the admissible velocity pairs, a feasible subset of translational and rotational velocities which can be achieved by the robot in a moving time window is extracted to form the final candidate velocity pairs. After the search space pruning process, the second step of DWA is to select the optimal *(v,w)* pair that maximizes an objective function [20]. An illustration of DWA is shown in Figure 3.

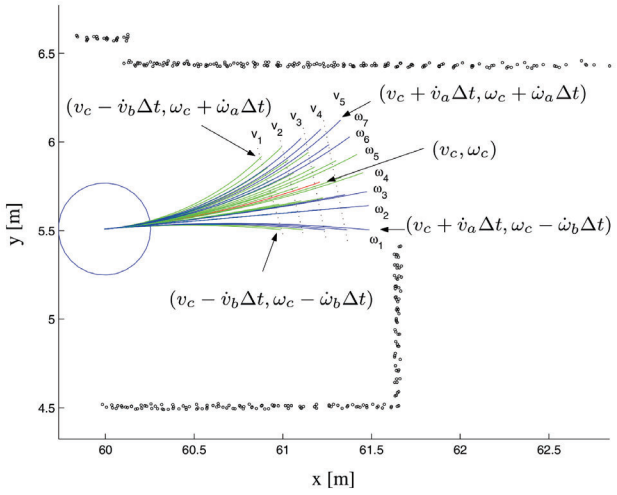

Figure 3. An illustration of candidate trajectories using DWA [23].

Although this approach demonstrates effective obstacle avoidance with static obstacles, the original version of DWA does not incorporate obstacle velocity information and thus is not effective when used in a dynamic environment. An extension of this approach has been proposed in [10], which was designed to overcome the issue of dynamic obstacle avoidance by predicting the future positions of moving obstacles using a dynamic occupancy map. However, this approach is computationally expensive.

3.1.2. Velocity Obstacle & Generalized Velocity Obstacle

Velocity obstacle (VO) was first proposed by Fiorini and Shiller in [21]. It is a velocity-based planning approach, which has been widely used for robot navigation in dynamic environments. In general, it ensures collision-free velocities by constructing so-called collision cones and removing all the velocities that may result in collision in the near future (see Figure 4). While the original VO is designed for holonomic robots only, its extension, Generalized Velocity Obstacle (GVO), deals with non-holonomic constraints, which are more suited to our smart wheelchair platform.

Generalized Velocity Obstacle (GVO) was proposed by Wilkie (2009) [24] to enable the navigation of car-like robots among arbitrarily moving obstacles. GVO differs from VO, which constructs collision cones geometrically, as GVO calculates velocity obstacles algebraically. In detail, for a car-like robot which is

assumed to have a constant speed u_s and steering angle u_φ, GVO calculates the future position A of the robot at time t with respect to its current robot frame as

$$A(t,u) = \left(\frac{L}{\tan(u_\varphi)} sin\left(\frac{u_s \tan(u_\varphi)\, t}{L} \right) - \frac{L}{\tan(u_\varphi)} cos\left(\frac{u_s \tan(u_\varphi)\, t}{L} \right) + \frac{1}{\tan(u_\varphi)} \right) \quad (1)$$

Where L is the wheelbase length.

Obstacles are considered to have linear motion and their future position B_i at time t can be easily calculated as

$$B_i(t) = p_{B_i} + v_{B_i} t \quad (2)$$

And the velocity obstacle can be constructed for which

$$\| A(t_{min}, u) - B_i(t_{min}) \| < r_A + r_B \quad (3)$$

Where r_A and r_B are radii of the robot and the obstacles, respectively. As a result, a feasible speed can be found which falls outside the velocity of the obstacles, and the final command is calculated by solving an optimization problem.

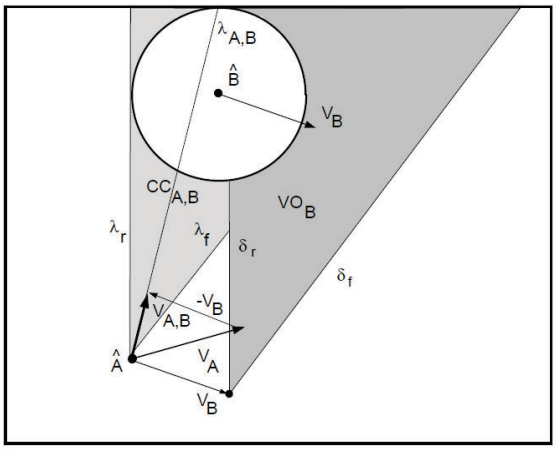

Figure 4. Collision cones representing velocity obstacles, where A stands for a robot and B is an obstacle [21].

3.2. Blending Strategy

Once a motion planner has been designed, its motor commands must be reconciled in some way with those of the user (a 'blending strategy'). Since the choice of blending strategy has a substantial effect on the user's experience of control, significant effort was expended to select the optimal strategy. The notion of optimality from a user perspective (e.g. comfort) is often extremely different from that used in traditional robotics (e.g. shortest path).

3.2.1. Linear Blending

One of the most popular blending approaches is linear blending, where the final control command $u(t)^{LB}$ is a weighted sum of the user's input and the command computed by the motion planner as shown in Figure 5 and the accompanying equations 4 and 5.

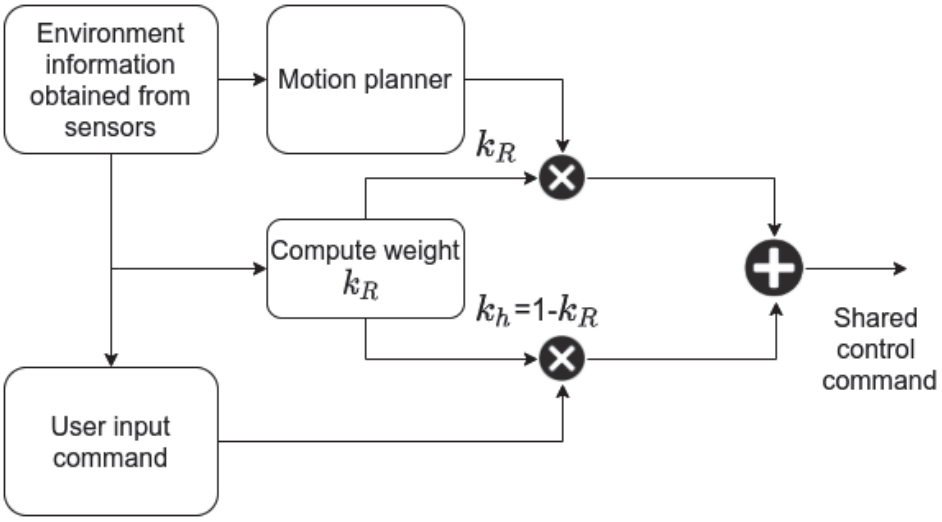

Figure 5. A block diagram of a typical linear blending approach

$$u(t)^{LB} = k_h u(t)^h + k_R u(t)^R \quad (4)$$

$$k_h + k_R = 1 \quad (5)$$

Where k_h and k_R are the weights assigned to the human and the motion planner outputs ($u(t)^h$ and $u(t)^R$), respectively.

Linear blending has been extensively explored by researchers [6], [25], [26]. In most cases, k_h and k_R are dynamic values and are defined based on certain criteria. As a result, the main focus for this approach lies in the weight negotiation between the user and the planner in order to optimize certain objectives.

However, linear blending has some defects. Firstly, it was demonstrated mathematically by Trautman (2015) that simple blending of the two trajectories does not guarantee a collision-free resultant trajectory [27]. Secondly, due to the uncertain nature of the user's intention in cases where there are several similar candidates the final blended trajectory may not align with their desired movement. To address these issues, he proposed probabilistic shared control (PSC) [27].

3.2.2. Probabilistic Shared Control

Probabilistic shared control (PSC) models the human-robot interaction by taking interaction uncertainty into account. This approach adds more flexibility to the user-wheelchair collaboration strategies, while guaranteeing safety mathematically [27]. It assumes both the user's input (interpreted as velocity pairs) and

the candidate velocities (computed by the motion planner), follow a probability distribution. The process is implemented by taking the joint probability of the velocity input from both the user and the planner, and the optimal command can be found by maximizing this joint probability. It can be formulated as:

$$u_{t+1}^{PSC} = u_{t+1}^{R*} \quad (6)$$

$$u_{t+1}^{PSC} = argmax_{u_{t+1}^{R}} \, p(u_{t+1}^{h}, u_{t+1}^{R}, u_{t+1}^{c} | \bar{z}_{1:t}^{h}, \bar{z}_{1:t}^{R}, \bar{z}_{1:t}^{c}) \quad (7)$$

$$p\left(u_{t+1}^{h}, u_{t+1}^{R}, c\right) = \quad \varphi(u_{t+1}^{h}, u_{t+1}^{R}) p(u_{t+1}^{h} | \bar{z}_{1:t}^{h}) p(u_{t+1}^{R}, u_{t+1}^{c} | \bar{z}_{1:t}) \quad (8)$$

where $\bar{z}_{1:t}^{h}$ represents measurements of the user input, $\bar{z}_{1:t} = [\bar{z}_{1:t}^{R}, \bar{z}_{1:t}^{c}]$ in which $\bar{z}_{1:t}^{R}$ defines measurements of the robot trajectory u^{R} and $\bar{z}_{1:t}^{c}$ stands for measurements of obstacles trajectory u^{c}.

$\varphi(u_{t+1}^{h}, u_{t+1}^{R})$ captures how "agreeable" the robot trajectory is with the intention of the user, while $p(u_{t+1}^{R}, u_{t+1}^{c} | \bar{z}_{1:t})$ models the autonomy of the robot by taking human-robot interactions into consideration.

Following this line of research, Ezeh et al. (2017) applied PSC in a wheelchair navigation task and demonstrated that PSC is safer than linear blending through experiments [5]. However, like other state-of-the-art research into the shared control of wheelchairs, the application scenario was limited to a static environment. In this report, we propose a hierarchical framework that extends the implementation of PSC to dynamic environments.

4. Shared Control Navigation using Probabilistic Shared Control (PSC)

4.1. A Hierarchical Framework

Our proposed design uses a hierarchical framework for collision avoidance, which treats static and dynamic obstacles separately. The reasons for this are threefold:

- It allows more flexibility and transparency in control.
- It reduces computational complexity by pruning the subsequent velocity search space for dynamic obstacle avoidance.
- As we need to detect and track pedestrians, we will use Lidar+RGBD camera for this purpose. In order to detect the static obstacles and increase the coverage around the wheelchair, we make use of cheaper sensors such as ultrasonic ones. They are also more computationally efficient.

Figure 6 shows an overview of our collision avoidance and shared control algorithm. In detail, we first search the velocity space for all achievable velocity pairs based on the wheelchair kinematics. Then, the dynamic window approach (DWA) is applied for low-level collision avoidance with static obstacles, which looks into a time horizon Δt. In our setting, where the maximum speed of the wheelchair is limited to 1.2m/s, we found that $\Delta t = 0.5s$ gave a reasonable result. The resultant admissible and achievable speed is further used as the search space for GVO, which filters out any velocity that may result in a collision with moving obstacles. In our implementation, the simulated moving obstacles (pedestrian avatars) each have a radius of

$r_B = 0.33m$. The wheelchair radius is considered as the maximum length on the wheelchair edge from its centre of mass, which is $r_A = 0.6m$ in our case. Additionally, we assign a safety margin $r_{safe} = 0.15m$. As a result, the final candidate velocity u should satisfy:

$$\|A(t_{min}, u) - B_i(t_{min})\| > r_A + r_B + r_{safe} \quad (9)$$

After this pruning process, a cost function is used to evaluate each final candidate, in terms of heading, clearance and velocity. This cost is interpreted as the probability associated with each candidate velocity, which will be used in the shared control stage.

In terms of the shared control blending strategy, PSC serves as our theoretical basis where we extend it to deal with moving obstacles. We make the following assumptions.

- As no global planning is involved in the navigation, a local goal is defined as 2m ahead of the user's intended trajectory.
- The user's intended trajectory is only based on the current input at the most recent time step. This allows us to simplify $p(u^h_{t+1}|\bar{z}^h_{1:t})$ to $p(u^h_{t+1}|\bar{z}^h_t)$.
- As a starting point of this research, obstacles are currently assumed to have linear motion, and their trajectories are only based on the measurements at the most recent time step. This allows us to reduce $\bar{z}^c_{1:t}$ to $\bar{z}^c_t$.
- The motion of the obstacles is assumed not to be affected by the robot, only a simple interaction function between the moving obstacles and the wheelchair has been modelled.

Mathematically, the problem can be simplified to:

$$u^{PSC}_{t+1} = argmax_{u^R_{t+1}} p(\bar{z}^h_t, \bar{z}^R_{1:t}, \bar{z}^c_t) \quad (10)$$

$$p(u^h_{t+1}, u^R_{t+1}, c) = \varphi(u^h_{t+1}, u^R_{t+1}) p(\bar{z}^h_t) p(\bar{z}^R_{1:t}, \bar{z}^c_t) \quad (11)$$

The agreeability between the user and the planner is modelled as:

$$\varphi(u^h_{t+1}, u^R_{t+1}) = exp(-\frac{1}{2\gamma}(\hat{u}^h_{1+t} - \hat{u}^R_{1+t})(\hat{u}^h_{1+t} - \hat{u}^R_{1+t})') \quad (12)$$

where $\hat{u}^h_{1+t}$ and $\hat{u}^R_{1+t}$ are normalized user and robot trajectories (input). The parameter γ which has a range of $\gamma > 0$ models how strongly $\hat{u}^h_{1+t}$ and $\hat{u}^R_{1+t}$ are correlated. As a result, the agreeability φ is between 0 and 1, with 1 being the path planner's decision exactly matches the user's intended trajectory. Through trial and error, we set $\gamma = 100$ in our implementation to achieve an acceptable trade-off.

In addition, the autonomy probability can be written as:

$$p\left(\bar{z}^{R}_{1:t}, \bar{z}^{c}_{t}\right) = f\left(u^{R}, u^{c}\right) p\left(\bar{z}^{R}_{1:t}\right) \prod_{i=1}^{n_t} p\left(\bar{z}^{c}_{t}\right) \quad (13)$$

where n_t is the number of moving obstacles detected by the wheelchair onboard sensors at time t.

Ideally, the most common scenario for robot navigation that involves moving obstacles is in a human-populated environment. As a result, it is important to consider pedestrian-robot interaction by modelling the interaction function $f\left(u^{R}, u^{c}\right)$. In this paper, we assume the pedestrians' short term motion is not affected by the robot, and model the interaction based on robot-pedestrian physical distance:

$$f\left(u^{R}, u^{c}\right) = \prod_{i=0}^{n_t} \left(1 - a * exp\left(-\frac{1}{2q^2}\left|u^{R}_{t+1} - u^{c_i}_{t+1}\right|\right)\right) \quad (14)$$

where $\left|u^{R}_{t+1} - u^{c_i}_{t+1}\right|$ is the Euclidean distance between the robot and each moving obstacle; a is the repulsion force, which can be set between [0,1]; and q is a scaling parameter, which encodes the idea of safety distance implicitly and is set to 0.9. The rationale behind this function is that a low probability is assigned to candidate robot actions which may result in short distance to all moving obstacles that are within the sensor range.

$p(u^{R}_{t+1}|z^{R}_{1:t})$ models the motion planner's trajectory distribution based on prior information. This can be considered as the cost that is associated with each candidate velocity pair that comes from the motion planner. We model the cost function as:

$$G(v, w) = \alpha * heading(v, w) + \beta * clearance(v, w) + \gamma * velocity(v, w) \quad (15)$$

where α, β, γ are weights for each term, '*heading*' measures the alignment of the robot with the goal direction, '*clearance*' represents the distance to the nearest static obstacle on the selected trajectory while '*velocity*' equals the normalized absolute linear candidate velocity. By tuning these parameters, different motion behaviours can be achieved by emphasizing the relative importance of these competing objectives. These parameters together with γ, a and h can be further tuned on a user basis in order to provide the most suitable assistance to the user. Intuitively, we want the wheelchair to run at fast speed if possible and keep a suitable distance from obstacles so that the manoeuvre will not be too aggressive. As a result, in our test, we set α, β, γ to be 0.1, 0.3, 0.6, respectively which gives reasonable behaviour through trial and error. Note this cost function is slightly different from the one proposed in Deliverable 3.1 which instead aims to minimize the distance towards the goal while maximizing the distance to the obstacles.

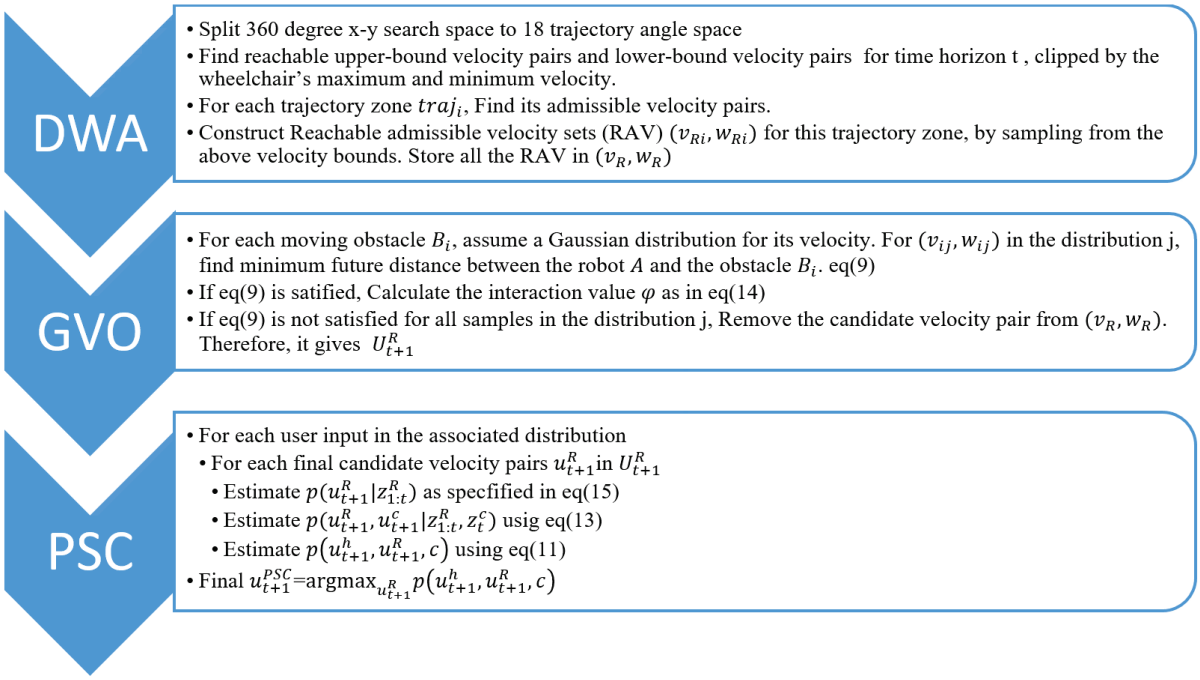

Figure 6. Flowchart of our collision avoidance and shared control algorithm.

4.2. Implementation and validation in a simulator

Before implementing our strategy on the actual wheelchair, we first implemented and validated it in a simulator which is built on Unity 3D. For the details of the simulator, please refer to Deliverable 4.2.

In order to achieve an accurate evaluation, all sensor characteristics (except the RGBD camera) have been modelled in the simulator. Wheelchair dynamics were approximated in the simulator by using Unity physics components. Table 1 shows some key parameters. The center of mass and moment of inertia for the wheelchair body (plus driver), two drive wheels and four caster wheels were calculated automatically from the built 3D models.

Table 1. WHEELCHAIR PARAMETERS IN THE SIMULATOR

Parameters	Value
Body mass	200kg (Wheelchair + a driver)
Drive wheel mass	2.6kg
Caster wheel mass	1kg
Angular drag	0.05N
Motor maximum torque	400Nm

The strategy was implemented as if all inputs were from actual sensors, which requires minimal modification for it to be transferred to a real-world test. The control strategy was implemented in a ROS framework and communication between the simulator and ROS is achieved by RosSharp as illustrated in Figure 7.

To begin with, simple sensor fusion is achieved by implementing an extended Kalman filter in the "robot_localization" package, which takes input from two wheel encoders and the IMU to publish an accurate odometry at 20hz. In terms of perception, 12 ultrasonic sensors and one Lidar sense the surrounding environment, and are used to construct a global occupancy grid map which is published at 10hz. In our implementation, the map has a size of 40m x 40m with a resolution of 0.05m. It is centred at a fixed frame to reduce the computational cost.

Our control strategy takes input from the occupancy grid and implements DWA for static obstacle avoidance. Resultant admissible velocities are used as the search space for the GVO which receives information on the dynamic obstacles from Unity. In the future, this information would be provided by a pedestrian tracker that utilizes the RGBD camera and Lidar data such as that developed as part of deliverable 2.2.

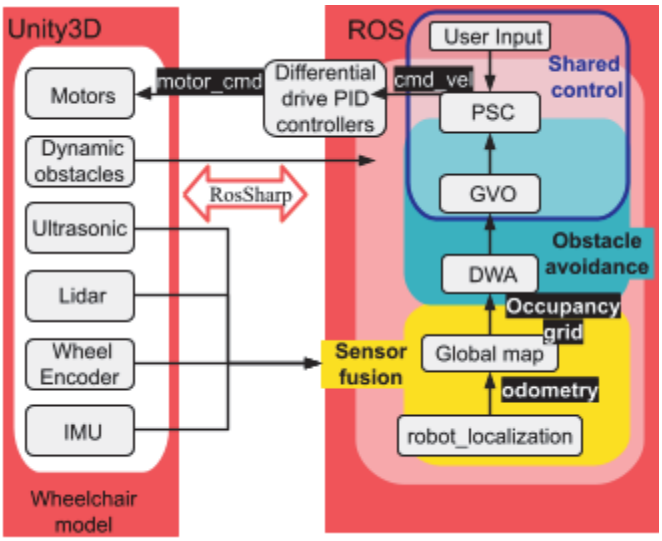

Figure 7. Structure of the proposed design strategy and its communication with Unity3D

The output from GVO is blended with the user input (which is received from a keyboard) using probabilistic shared control. The final velocity command is published at a rate of 10hz and is subscribed by two differential drive PID controllers running at 50hz, which calculates suitable motor torque for two drive wheels so as to achieve the desired linear and angular velocity.

4.2.1. Simple environment with static and moving obstacles

To evaluate the validity and performance of our proposed strategy, we designed an H-shaped course populated with moving pedestrians. The course was designed to simulate daily wheelchair usage, taking into consideration typical manoeuvres from the Wheelchair Skills Test (WST) [28], specifically we chose three of the most used skills, detailed in Table 2.

Table 2. WHEELCHAIR SKILLS THAT HAVE BEEN TESTED

Number	Skills from WST
1	Roll forwards
2	Turn while moving forwards (90°)
3	Avoid moving obstacle

For these initial tests, three pedestrian agents were used. These were programmed to move in a simple manner – moving straight in the direction of their original heading. Pedestrians were placed at three corners of the maze with each instructed to move in a different direction at a mean speed of 1.1m/s, this simulates an average human daily walking speed [29]. The initial positions and walking directions were designed to maximize the potential interaction opportunity with the wheelchair. The wheelchair started from the origin (shown in Figure 8 with a blue star) with a goal of reaching the upper left corner of the H-shaped course (shown in Figure 8 with a red star). The left and right passage of the course has a length of 19m and a width of 3m while the middle passage was 8m long and 3m wide.

During the navigation, all sensor data was published to ROS and a global occupancy grid map was constructed which was fixed at the origin to reduce the computational cost.

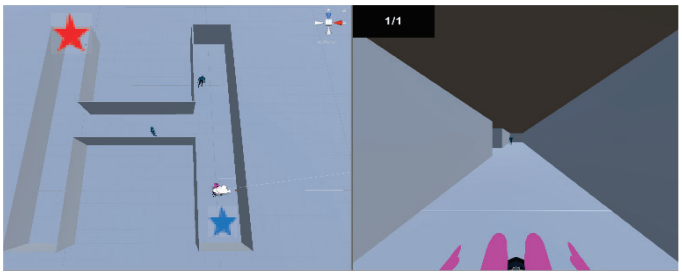

Figure 8. Left: Screenshot of the simulation (top view) where the blue star shows the wheelchair's starting point and the red star represents the goal. Right: First person view.

4.2.1.1. Metrics

Three metrics were used to evaluate the safety and assistance for the proposed shared control navigation design. These metrics align with the performance metrics which have been proposed in Deliverable 1.3.

These are defined as:

- C: Number of collisions (with walls or pedestrians). This metric is the number of collisions that occurred in the scenario and were reported by the simulator.

- T_t: Task completion time (the time that the user required to reach the goal position from the starting position):

$$t_c = t_{end} - t_{start} \quad (16)$$

- A: Agreement. We define agreement in terms of the deviation of the user's heading command from the final shared control's heading command. Note this metric differs from equation 12 which compares the user's command with the path planner's command before the blending process. Mathematically, it is calculated as:

$$\theta(u) = tan^{-1}(\frac{v}{w}) \quad (17)$$

$$a_i = 1 - \frac{|\theta(z_h^i) \ominus \theta(u_{SC}^i)|}{\pi} \quad (18)$$

$$agreement = \frac{\sum_{i=0}^{N} a_i \cdot \Delta t_i}{\sum_{i=0}^{N} \Delta t_i} \quad (19)$$

where v and w are the translational and rotational velocities $u \sim [v\ w]$, a_i is the normalised agreement at time step t_i and u_{SC}^i is the final output of the probabilistic shared control. N is the number of samples available in which data from the user measured input z_h^i coincide in time with u_{SC}^i, and Δt_i is the duration of the user's input command z_h^i.

4.2.1.2. Performance

Due to COVID-19, it was difficult to recruit many wheelchair users for a test with the actual wheelchair. Alternatively, three healthy participants who are not wheelchair users were recruited and participated in driving the wheelchair in the simulator. Two had a background in robotics or computer science while one participant had an unrelated background. Participants were allowed to perform some trial runs to familiarize themselves with the setup before the actual test. In total, three test trials were conducted where each user drove the wheelchair using keyboard arrow keys. It simulates a head-array interface which can only indicate the four cardinal directions and four secondary intercardinal directions [30].

Figures 9 through 13 show the results for one trial from one participant (note for these figures, we use a right-hand coordinate frame which means the x-axis is the vertical one and the y-axis is the horizontal one). As shown in Figure 9 and Figure 10, the final blended velocity command followed the user's input most of the time, except for when there was a collision risk. In detail, the chosen linear velocity at t=11-16s and t=36-38s, angular velocity at 10-15s and 35-37s are largely different from the user's intended velocity. This is also reflected in Figure 12 which describes the agreement between the user and the final chosen command. Figure 13 shows the avoidance trajectory for the wheelchair (plotted in black circles) and the trajectories for three moving pedestrians (plotted in red). The largest disagreements between the user input and the final chosen command are highlighted in Figure 13 by orange stars.

It can be seen that at t=10-16s, the wheelchair detected two moving pedestrians in proximity while the user's input indicates that the user would like to keep moving forward at the maximum speed. Under this scenario, the control strategy filtered out candidate velocities that may result in collision and found the one that guaranteed safety while best obeying the user's intention. As a result, the wheelchair decreased its linear velocity and made a right turn by choosing negative angular velocities. Afterwards, it stopped for a short time and then moved in the user-intended direction.

Similarly, at t=35-39s, the wheelchair encountered a pedestrian who moved across its path. At about t=35-39s, the user kept trying to move straight forward while the controller decided to make a small right turn to avoid future collisions. It then made a big right turn following the user's command.

Table 3 summarizes the test results for each participant. It can be seen that overall the agreement is high for all three participants and no collisions occurred. For Participant 3, a slightly longer completion time was observed which could be explained by their unfamiliarity with such a system.

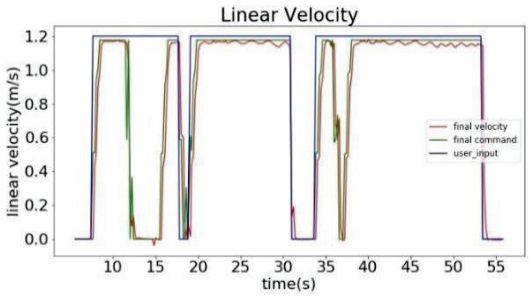

Figure 9. Linear Velocity

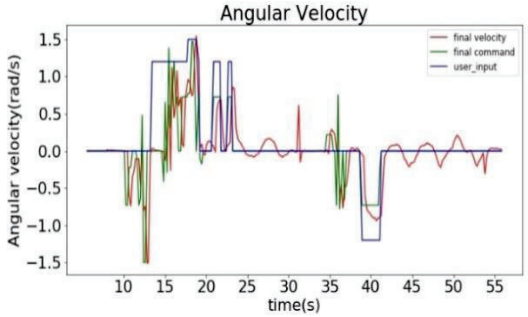

Figure 10. Angular Velocity.

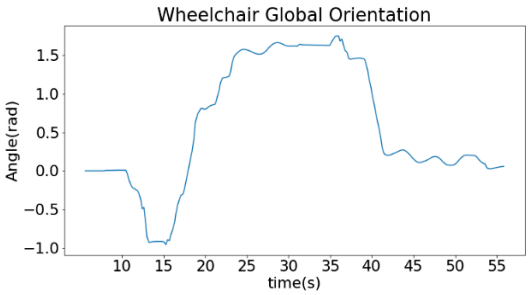

Figure 11. Wheelchair orientation.

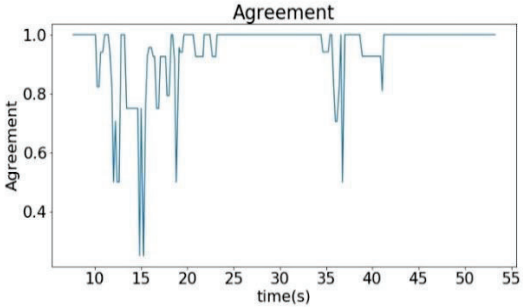

Figure 12. Agreement between the user's input and the final chosen command.

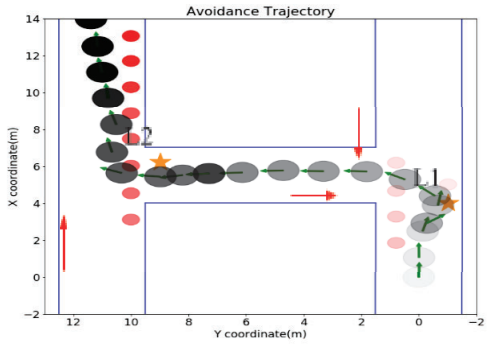

Figure 13. Wheelchair avoidance trajectory, where the wheelchair detects two pedestrians at L1 and one pedestrian at L2, and responds accordingly. The transparency represents the time (alpha from 0-100 corresponds to the time from 0 to 75s). The red arrows represent the moving direction of each pedestrian and the green arrows represent the orientation of the wheelchair).

Table 3. TESTING RESULTS FOR THREE PARTICIPANTS

Participant	**Collisions (C)**	**Completion time, T_t (s)**	**Agreement (A)**
1	0	45.15	0.9777
2	0	46.47	0.9469
3	0	50.25	0.9882

In general, we can see that the final command chosen by the controller generated safe and reasonable motion, which allowed the wheelchair to avoid static and moving obstacles while following the user's intention. The parameters were chosen in a way that favours high speed while guaranteeing a safe distance.

4.2.1.3. Comparison with other methods

We further evaluated our approach (we will call it GVDWA) by comparing it with related works, namely PSC with naive DWA which was originally implemented by Ezeh et al. [5], and PSC with DWA which takes obstacle velocities into account (we will call it VDWA) [31]. For a valid comparison, we used the same parameters that had been used for our method GVDWA for the cost function in all three methods.

In order to examine performance in different human density environments, we designed three scenarios in the simulator (see Figure 14). For each scenario, one user drove the wheelchair three times with each control method, and the performance was evaluated in terms of average number of collisions (C), average computation time (T_c), average agreement (A) and average task completion time (T_t). Table 4 provides a summary of the result.

It can be seen that in general GVDWA+PSC exceeds other methods in terms of reducing the number of collisions, while its computation time, agreement and task completion time are comparable with PSC+naive DWA. Of the three methods, PSC+naive DWA has the highest number of collisions, which is to be expected

as it treats moving obstacles as static and may result in moving towards the direction of the obstacle. For S1, we further tested the capability of naive DWA+PSC and only observed collision free avoidance when the speed of the avatar was decreased to 0.5m/s.

When obstacle velocity information is incorporated in DWA, the number of collisions was reduced at the cost of increased computation time. Of the three methods, our proposed method GVDWA and VDWA both predict moving obstacle positions over a short time horizon based on their velocity, where VDWA+PSC shows the longest computational time due to the use of a dynamic occupancy map.

In terms of agreement, all three methods result in agreement greater than 0.9, while naive DWA+PSC shows a slightly higher value. This can be explained because in some situations, this method moves towards the user's desired direction despite future collision danger with moving obstacles. Although GVDWA+PSC has slightly lower agreement, the deviation normally occurs when there is a risk of collision. Figure 15 shows the avoidance trajectory with its corresponding velocity for GVO+DWA+PSC in the H_maze simulated map with 6 people. Two obvious avoiding manoeuvres can be seen in the two orange stars.

As for the task completion time, GVDWA+PSC takes longer than VDWA+PSC and DWA+PSC. This result is consistent with the computational time and the agreement, and can be explained as the wheelchair deviated from the user's desired path (potentially the shortest) to avoid obstacles, which took a longer time.

Table 4. COMPARISON RESULTS SUMMARY(Mean±SD)

	S1（1 pedestrian）				S2 (H_maze + 3 pedestrians)				S3 (H_maze + 6 pedestrians)			
	C	T_c(s)	A	T_t(s)	C	T_c(s)	A	T_t(s)	C	T_c(s)	A	T_t(s)
GVDWA +PSC	0±0	0.0793±0.0017	0.9589±0.0158	14.78±0.5000	0±0	0.0808±0.0043	0.9540±0.0050	45.01±2.9492	0±0	0.0830±0.0010	0.9356±0.0393	46.87±3.8192
VDWA+ PSC	0±0	0.0832±0.0007	0.9679±0.0030	15.23±0.3915	0±0	0.1061±0.0079	0.9469±0.0048	46.87±0.5834	0.33±0.5774	0.1405±0.0086	0.9502±0.0154	48.10±2.8744
DWA+ PSC	1±0	0.0773±0.0049	0.9889±0.0087	14.96±0.8908	2±1	0.0768±0.0045	0.9856±0.0096	44.6±3.6128	2.66±0.5774	0.0818±0.0053	0.9593±0.0151	46.65±2.205

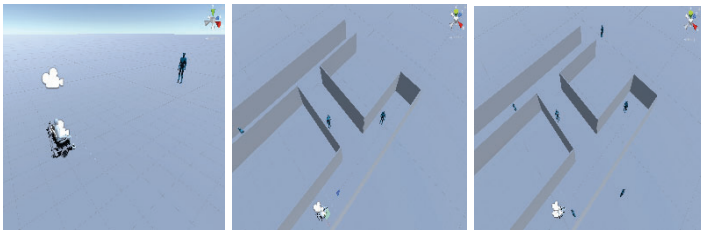

Figure 14. Three test scenarios. From left to right: S1,S2,S3. S1 has one pedestrian moving towards the wheelchair, S2 has three pedestrians moving in different directions in the H maze while the number of pedestrians was increased to six in S3. All pedestrians move at an average speed of 1.1m/s.

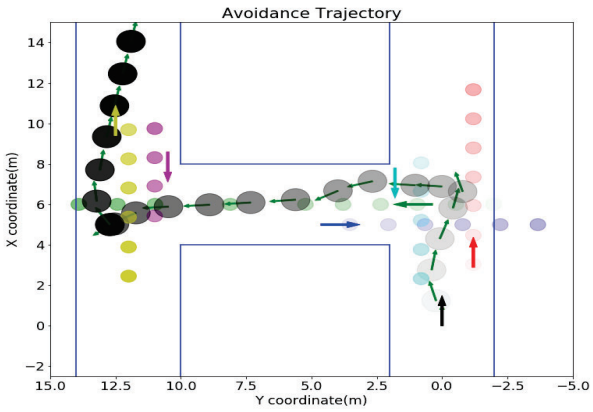

Figure 15. Avoidance trajectory of the wheelchair in S3 using our method. Black circles represent the trajectory of the wheelchair and the green arrows represent its orientation (starting from bottom right and moving towards top left). Circles with other colors show the trajectories of different pedestrians when in close proximity to the wheelchair. The wheelchair successfully avoided all pedestrians.

4.2.1.4. Discussion

The simulation results indicate the validity of our design, which can be seen as a first step towards enabling a shared control wheelchair to navigate in a dynamic environment. Due to the simulator setup and the implementation method, minimal code modifications are needed to transfer this approach onto the actual wheelchair. In the future, the test will be carried out in the physical world, where information about the dynamic obstacles will be provided by a people tracker and the results will be presented as part of the final evaluation in Deliverable 1.5. Nevertheless, we would like to consider some of the current limitations of our design:

1) Computational cost

The DWA is computationally expensive. In order to reduce computational time and still allow the design loop to be executed in real-time, we have limited the velocity search space to those that generate trajectories that fall within [-90,90] degrees of the user's intended trajectory. For example, if the user's intention is going forward, the planner will only search for the candidate with $v \geq 0$. This simplification also takes the user's comfort into consideration, as the user may feel uncomfortable if the wheelchair drives in the opposite direction to the way they intended. In our implementation, we set the horizon for DWA and GVO to be 4s. It allows real-time implementation at a frequency of 10hz on a 2.6GHz Intel Core i7-9750H CPU. However, we expect the computational time to increase when more pedestrians are involved and the crowd density increases.

2) Latency

We observed that there is about a 0.25s latency in the system. This is mainly due to the motor latency and the communication delay between ROS and Unity. In order to reduce its effect on the system, when performing the dynamic window search, we assumed the wheelchair keeps moving with its current velocity for two time intervals, and the search starts with the new wheelchair position.

3) Morphology

One main limitation of GVO is that it assumes both the wheelchair and the pedestrians are circular objects. This poor representation of the morphology of the wheelchair may lead to over-cautious behaviour when

close to other dynamic obstacles. We would like to address this problem in the future by using the method proposed in Deliverable 3.1 which obtains a smooth boundary for each obstacle by fitting a Bezier spline to the occupied cells, although this will clearly have computational implications.

4) Simplified obstacles and interaction function

As mentioned in Section 4.1, we assume the obstacle's next movement is only based on its current trajectory measurement. This may not be true, especially when this "obstacle" is a human. In addition, we use a simplified function to model the interaction between the obstacles (human) and the wheelchair which is similar to a cost function that favours velocity which could result in exaggerated clearance. In reality, pedestrians may adapt their trajectory to avoid the robot. To better understand this problem, we conducted a crowd-robot walking experiment and would like to incorporate our findings into our navigation strategy in the future (for full details of the experiment, please refer to Deliverable 1.4 and see Section 5).

4.2.2. 1D and 2D crowds

We also evaluated our proposed framework in simulated 1D and 2D crowds scenarios. These scenarios take place in a simulated version of UCL's PAMELA facility (see Figure 16 and Figure 17), which hosted the physical crowd-wheelchair experiments already described in Deliverable 1.4. The crowds in these scenarios are goal driven (random starting positions in the green area and random goals in the red area) with RVO used for local avoidance. Although it may not best simulate the user-crowd interaction, it serves as a good starting point for us to evaluate and further improve our shared control strategy.

Figure 16. Simulated 1D crowds.

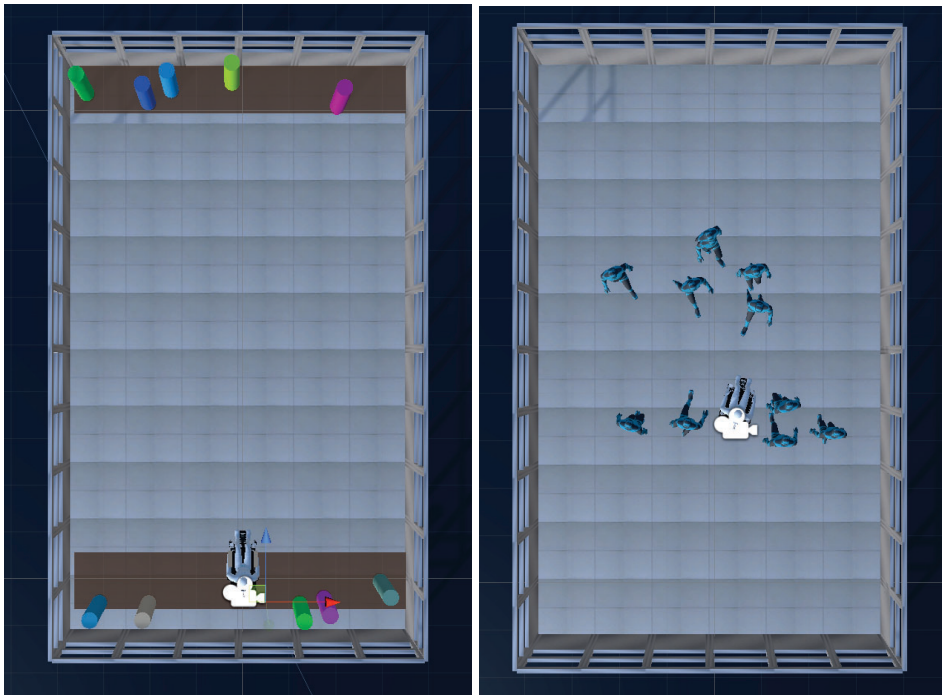

Figure 17. Simulated 2D crowds.

In addition to the shared control metrics, 6 categories of evaluation metrics from Deliverable 1.3 were used, as detailed in Table 5.

Table 5. A summary of metrics

Category	Symbol	Metric
Path efficiency (Compares the robot only scenario with crowd-robot scenario)	T1	Relative time to goal
	L1	Relative path length
Effect on crowd (Compares the crowd only scenario with crowd-robot scenario)	T2	Effect on time
	V2	Effect on velocity
Pedestrian-robot similarities (Compares the robot and averaged pedestrians behaviour in the same crowd-robot scenario)	T3	Time to goal similarity
	L3	Path length similarity
	H3	Heading similarity
	V3	Velocity similarity

Crowd-robot interaction	P4	Proximity
Collision	C5	Number of collisions
Shared control metrics	A6	Agreement
	F6	Fluency of commands

Note: We agreed with Inria to slightly modify the proximity metric (P4), so as to keep it in the range [0, 1] thus making it easier to compare with our other metrics. The new P4 metric is defined as:

$$P4 = 1 - \frac{\frac{1}{T_{robot}} \sum_{0}^{t_{robot}} d_{min}(t)}{l_s} \quad (20)$$

where $d_{min}(t)$ is the minimum distance between the robot and a pedestrian at a given time. l_s stands for the test scale of the environment, which is used to keep the P4 result between 0 and 1. In our implementation, we set it to 5m.

For each scenario, three conditions were tested: crowds only, wheelchair (robot) only and crowd-wheelchair navigation. Five trials have been carried out for each condition. The results are shown in Figure 18 to Figure 22. Statistical significance was tested using a Wilcoxon signed-rank test, where the significance is reported at $p<0.05$.

It can be seen that in terms of path efficiency, the shared control wheelchair spent almost twice the time (T1) and travelled a slightly longer distance (L1) to reach the goal when navigating through crowds.

From the crowd's perspective, the inclusion of a shared control wheelchair slightly increased the time (T2) for crowds to reach the goal, and decreased their velocity (V2).

These results indicate that in a crowd-robot navigation task, our shared control navigation strategy has a small (although not significant) effect on the crowds. This is expected as the current strategy does not aim to minimize such effects. In the future, we would like to add this consideration to our navigation strategy.

In terms of the pedestrian-robot similarities, the wheelchair took a longer time (T3), but traversed a shorter path (L3) to reach the goal compared to the average crowd participant. In addition, a slightly lower velocity (V3) and more variation in heading (H3) were observed for the wheelchair. It is likely that these differences reflect more opportunities for the driver and the controller to disagree in the 2D case: e.g. do I decide to pass the oncoming pedestrian on the left or the right?

Throughout the trials, no collisions (C5) occurred, even though the amount of interaction between the wheelchair and the crowds is quite high (P4). For the 1D scenario, the proximity value between the wheelchair (centre) and the nearest pedestrian (centre) is about 0.7. In the 2D scenario this value is slightly lower (0.65).

For shared control metrics, the user agreement (A6) and fluency of commands (F6) both have relatively high values (>80%) and small variation. This indicates that our shared control strategy generally obeys the user's intention in most situations and only provides assistance when necessary. Note these values may change when a different user or different driving interface is involved and will be investigated further as part of Deliverable 1.5.

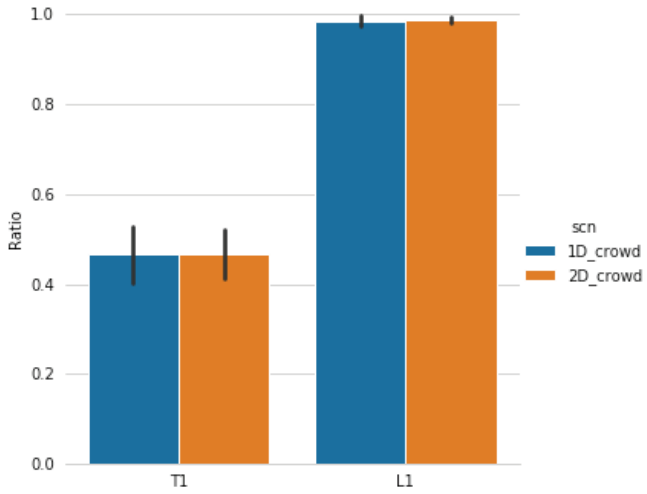

Figure 18. Path efficiency results for the simulated 1D and 2D crowd. In both scenarios, similar results in terms of relative time to goal (T1) and relative path length (L1) can be observed, where the wheelchair spends twice the time with slightly longer distance reaching the goal when it travels with crowds compared to the case when it travels alone.

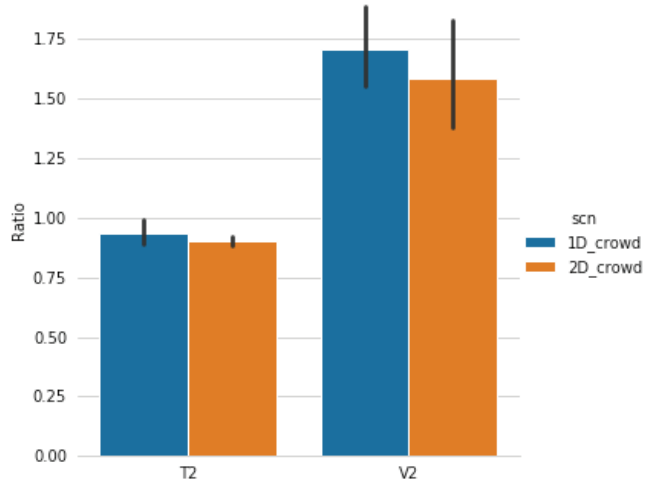

Figure 19. The wheelchair "effect on crowd" for the simulated 1D and 2D crowds. In both scenarios, similar results in terms of relative time to goal (T2) and relative path length (L2) can be observed, where the crowd moves slightly slower and spends slightly more time reaching the goal when it travels with the wheelchair compared to the case when the wheelchair is not present.

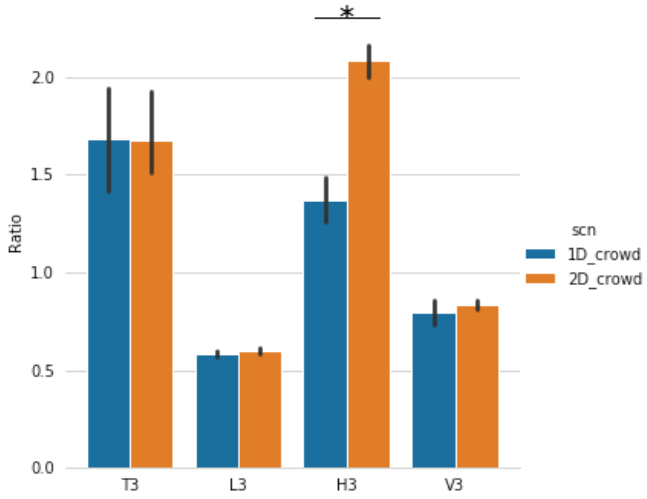

Figure 20. The pedestrian-robot (wheelchair) similarity for the simulated 1D and 2D crowds. Both scenarios yield similar results in terms of time to goal similarity (T3), path length similarity (L3) and velocity similarity (V3). In general, the wheelchair spent around 70% more time, 40% less distance to reach the goal with 20% less velocity. A significant difference in the similarity of heading (H3) ($p<0.05$) is observed between the 1D and 2D crowd scenarios.

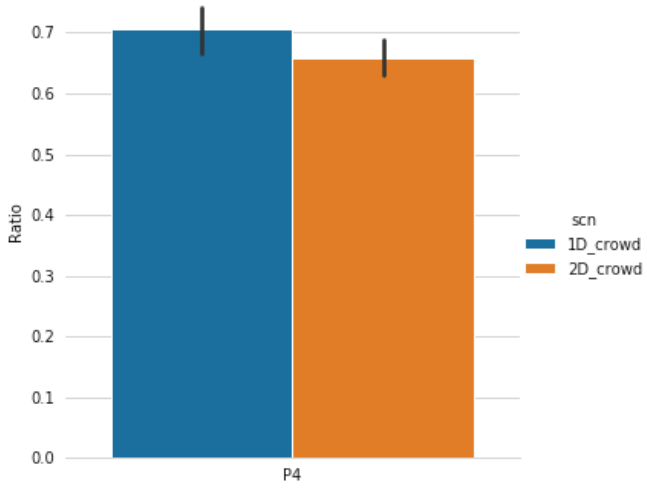

Figure 21. The crowd-robot (wheelchair) interaction metric for the simulated 1D and 2D crowds. Both scenarios yielded similar results in terms of proximity (P4).

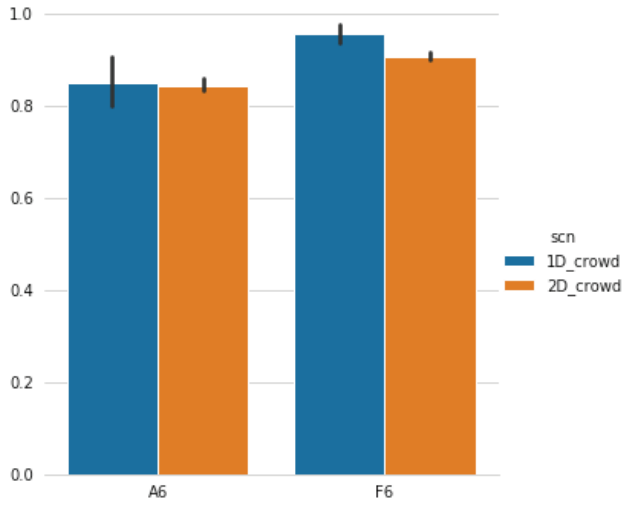

Figure 22. Evaluation of the shared-control performance for the simulated 1D and 2D crowds. In both scenarios, similar results in terms of user-planner agreement (A6) and user command fluency (F6) can be observed. Results indicate high agreement and high command fluency.

5. Wheelchair-Crowd Experiments

As detailed in Deliverable 1.4, we previously conducted an experiment in UCL's Pedestrian Accessibility Movement Environment Laboratory (PAMELA) to better understand the interactions between wheelchair user and crowd. In order to check the validity of our simulator and evaluate our different navigation strategies, we constructed a virtual PAMELA environment in the simulator and ran the same experiment with a similar setup (Figure 23), using the original results as reference. Two conditions were tested: (wheelchair running at) low speed and high speed. For these experiments, the wheelchair was fully controlled by the driver and no shared control was used.

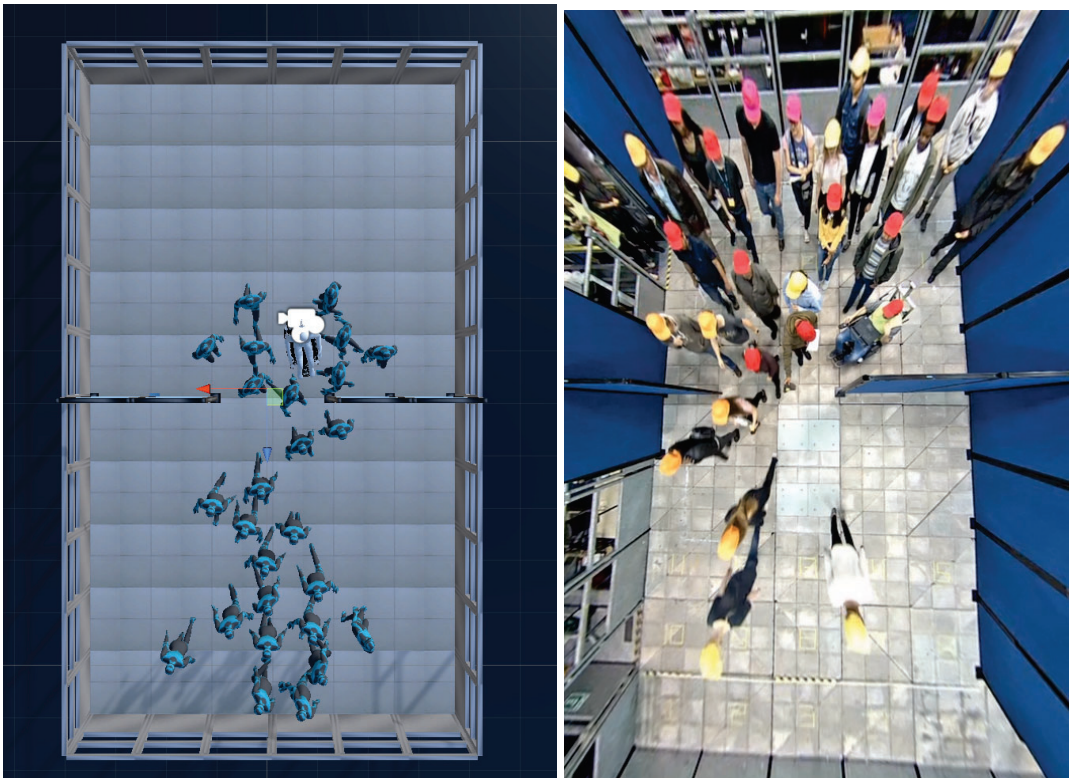

Figure 23. Simulated and actual PAMELA experiment.

To mimic the setup we used in the actual PAMELA experiment, each condition was tested 5 times in the simulation. For each trial, the wheelchair and pedestrian positions were shuffled. For both speed conditions, the wheelchair maximum speed was set to 0.9m/s and 1.3m/s relatively, and resulted in an average speed around 0.75m/s and 0.95m/s in the simulator, which is comparable to the average speed observed in the actual experimental data (0.71m/s and 0.96m/s).

For the low speed scenario, it can be seen from Figure 23 to Figure 26 that the simulated results are similar to the actual results in terms of effect on crowd (T2,V2), and capture the velocity similarity between the pedestrians and the wheelchair (V3). However, the time (T3), path length (L3) and heading (H3) similarities as well as the crowd-robot interaction in terms of proximity (P4) were not well captured. The most probable cause for this variation in similarity metrics is the different input device used in the simulated case (keyboard as opposed to joystick), which led to more discretised movement commands. We would like to point out that this choice of using keyboard inputs was due to limited lab access during COVID, and we would like to repeat the test in the simulator with a joystick in the future.

Similar results can be seen in the high speed scenario (Figure 27 to Figure 30). These comparable results for different metrics and the consistency of the results across different driving speeds indicate that the simulator does mimic the real world experiment fairly well. However, in future, we will evaluate the performance metrics by driving the wheelchair using a joystick in the simulator and further improve the pedestrian's behaviour using the data we collected in PAMELA which would allow a more realistic user (wheelchair)-crowd interaction to be simulated.

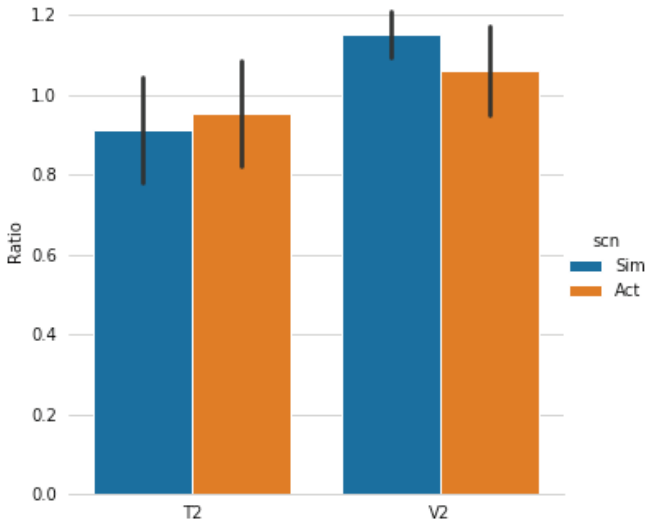

Figure 24. Effect on crowd result for the simulated and actual PAMELA experiment when the wheelchair is in low-speed mode. In both scenarios, similar results in terms of effect on time (T2) and effect on velocity (V2) can be observed, where the crowds move slightly slower and spend slightly more time reaching the goal when it travels with the wheelchair compared to the case when the wheelchair is not present.

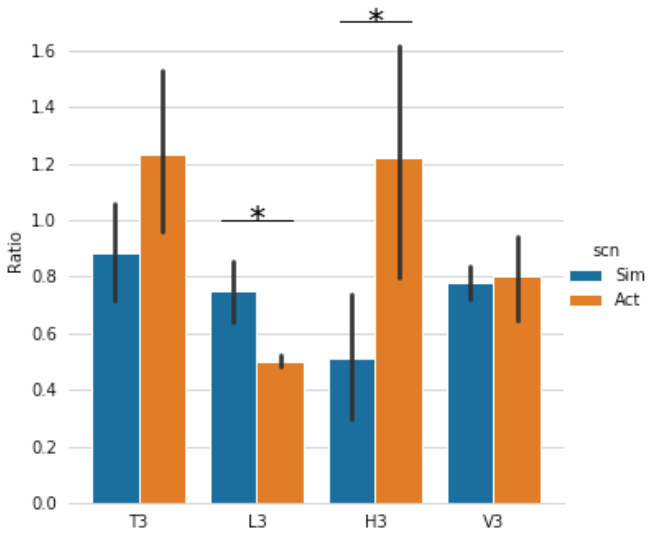

Figure 25. The pedestrian-robot (wheelchair) similarity for the simulated and actual PAMELA experiment when the wheelchair is in low-speed mode. In both scenarios, the velocity similarity is around 80%. A significant difference ($p<0.05$) can be seen in path length similarity (L3) and heading similarity (H3). There is also a noticeable but not significant difference in time to goal similarity (T3).

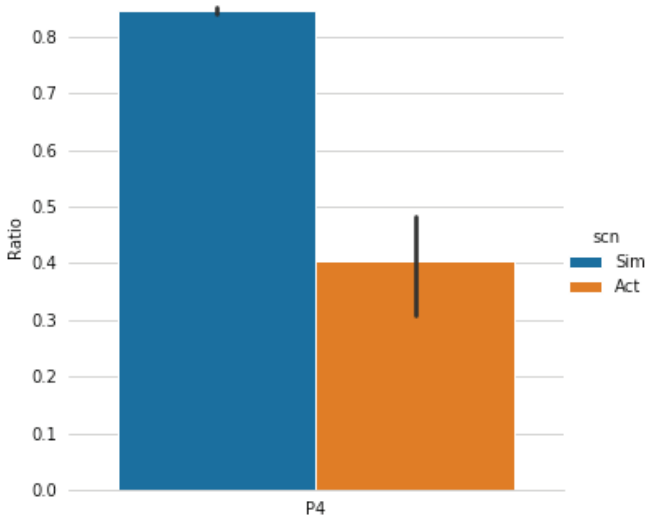

Figure 26. The crowd-robot (wheelchair) interaction for the simulated and actual PAMELA experiment when the wheelchair is in low-speed mode. In the simulated scenario, proximity is higher than in the real-world experiment.

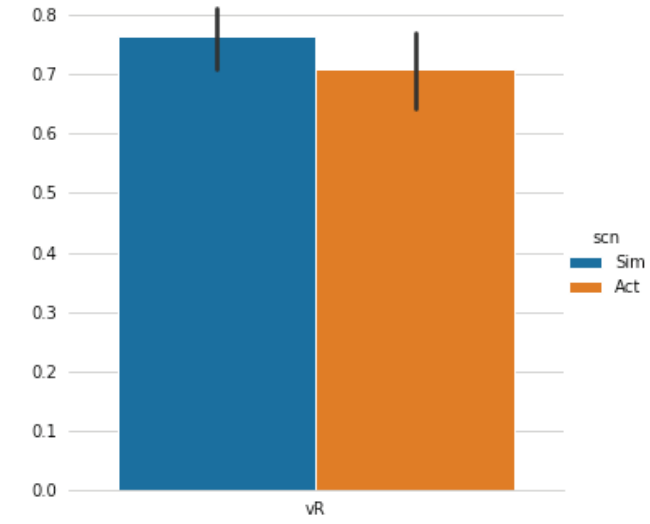

Figure 27. Similar wheelchair speed (low) in the simulated and actual PAMELA experiment.

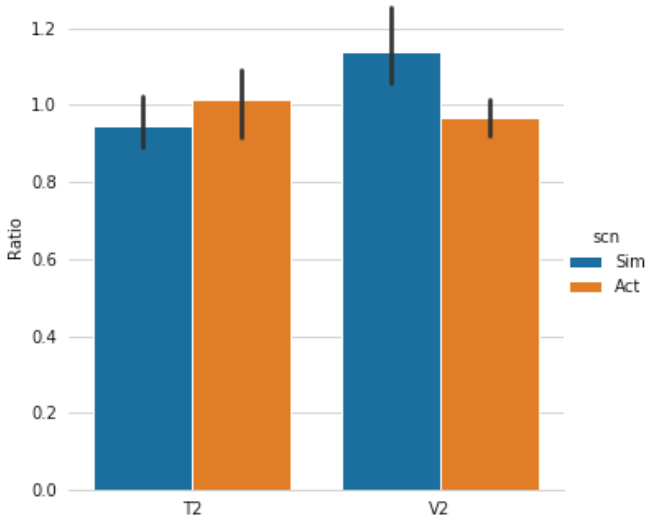

Figure 28. Effect on crowd result for the simulated and actual PAMELA experiments when the wheelchair is in high-speed mode. In both scenarios, similar results in terms of effect on time (T2) and effect on velocity (V2) can be observed, where the crowds move slightly slower and spend slightly more time reaching the goal when it travels with the wheelchair compared to the case when the wheelchair is not present.

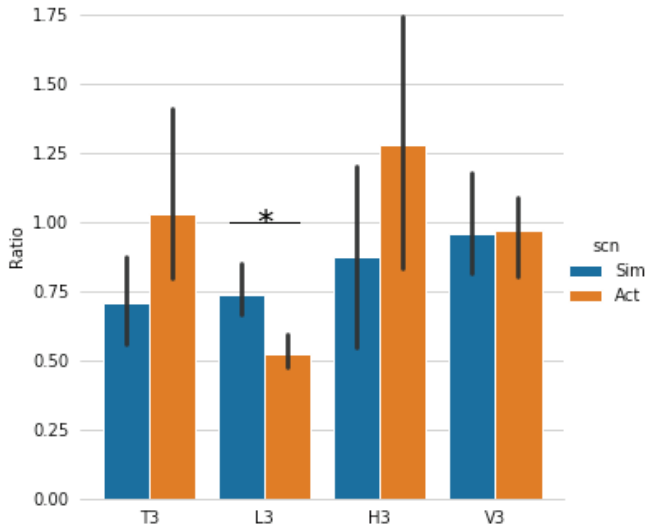

Figure 29. The pedestrian-robot (wheelchair) similarity score for the simulated and actual PAMELA experiments when the wheelchair is in high-speed mode. In both scenarios, the velocity similarity is around 95%. A significant difference ($p<0.05$) can be seen in path length similarity (L3). There is also a noticeable but not significant difference in time to goal similarity (T3) and heading similarity (H3).

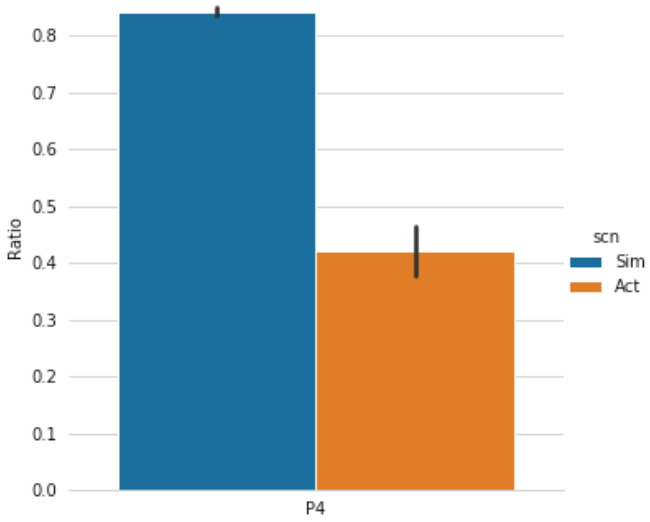

Figure 30. The crowd-robot (wheelchair) interaction for the simulated and actual PAMELA experiment when the wheelchair is in high-speed mode. In the simulated scenario, proximity is higher than in the real-world experiment.

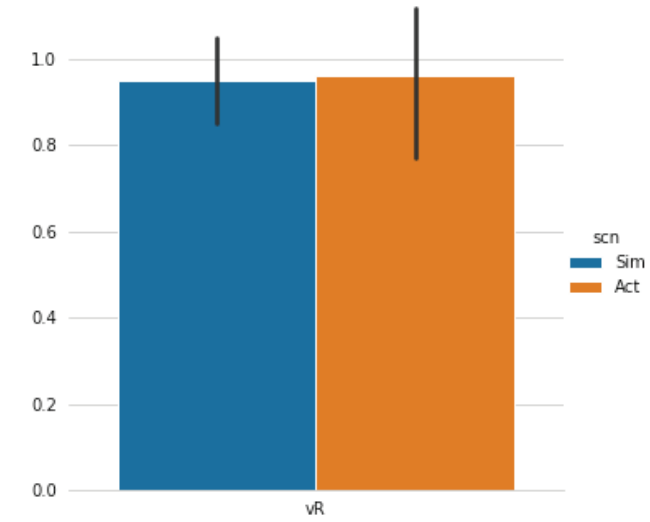

Figure 31. Similar wheelchair speed (high) in the simulated and actual PAMELA experiment.

6. Estimating User Intent for Shared Control

User intent, in the context of robotic wheelchair navigation, refers to the driving intention of a wheelchair user, which is unknown. An approximation can however be derived from the joystick driving input the user provides, which is translated into a drive velocity by the wheelchair's motor controller, as per the previous sections of this report.

Alternatively, rather than explicitly generating wheelchair trajectories, we could instead learn user models for driving intent from vision data. This model is a complex non-linear mapping from image sequences to the wheelchair's joystick drive input. In this section we train and test our models on real-world datasets collected using our robotic wheelchair in dense natural crowds.

6.1. Crowd Dataset

We collected a large-scale shared autonomy robot dataset at the crowded University College London (UCL) Bloomsbury campus. Our robotic wheelchair was driven by four different users through natural crowds over multiple days, both indoors and outdoors, and under varying lighting conditions as summarised by the Table below. Crowd densities varied from very dense (3 people per square meter) to lightly dense.

Table 6: Summary of our robotic wheelchair dataset, collected through natural crowds at UCL

Trials	Day	Weather	Lighting	Route	Driver
1-3	1	Overcast	Day	2	O
4-6	1	Light rain	Day	2	O
7	1	Overcast	Day	2	R
8	2	Overcast	Day	2	O
9-10	2	Light rain	Day	3	A
11	2	Overcast	Dusk	3	R
12	3	Overcast	Day	3	O
13	3	Indoors	Indoor	1	O
14	3	Light rain	Day	3	O
15-16	4	Overcast	Day	2	F

Each trial consists of a driver navigating the wheelchair along various routes on campus. Each route is set to begin at the location pointed to by the red arrow in Figure 31. Route 1 is indoors inside the Engineering cafeteria. Route 2 includes driving through an overpass, through multiple doors and into the main student cafeteria whilst Route 3 also includes using a lift to get to a busy indoor cloister and outdoor quad. Five sensors were recorded for this data collection run including the RGBD camera, lidar, ultrasonic clusters,

IMU and joystick. Our user intent model specifically uses the camera and joystick data. Typical crowd densities encountered during our data collection trials are shown in Figure 32.

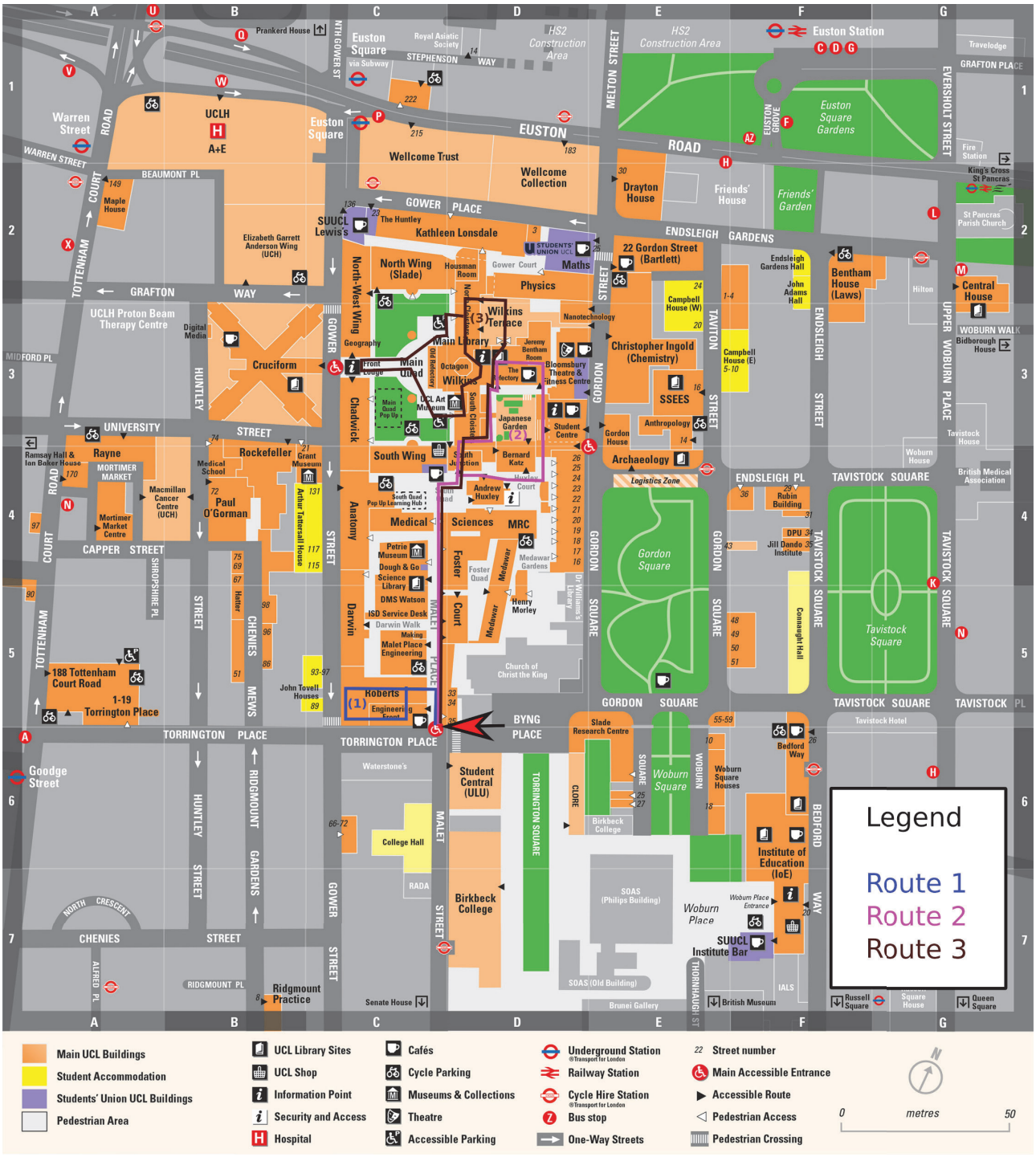

Figure 32. A map of UCL campus, showing the three routes traversed during our data collection trials; the robotic wheelchair moved through natural crowds over multiple days.

Figure 33. Our robotic wheelchair platform shown navigating through natural crowds, both indoors and outdoors, at UCL during our data collection trials.

6.2. Understanding User Intent

Every wheelchair user has their own individual driving style which captures their tendency to accelerate, how sharply they turn, whether they stop (release the joystick) or simply deccelerate (slowly move the joystick backwards) in the face of uncertainty, and how often they perform complex reversing manoeuvres. Crowded spaces add further complexity since there are independent agents (the pedestrians) capable of interacting with the wheelchair in a number of different non-deterministic ways. Different user intent models are therefore required for different scenarios.

We consider a specific scenario, common in rehabilitation facilities [32], where a patient with visual impairment has to safely drive a powered wheelchair. The limitations in the patient's field of view makes moving towards a goal position whilst safely avoiding pedestrians and other moving objects challenging. To achieve safe navigation with dynamic obstacles, we learn a user intent model which can be used to either replace or augment the patient’s noisy joystick input. This model is learnt from data collected from several drivers who do not have any visual impairments.

Specifically, we consider hemineglect, a neuropsychological condition resulting most commonly from right-hemisphere brain injury. A person with hemineglect lacks awareness and attention to one side of their field of view, even though the eyesight is unaffected [33]. This condition manifests in activities such as eating only one side of a plate, applying make-up to one half of the face, only shaving one side of the face and being unaware of large objects (even people) on the neglected side. The effects of left-side hemineglect (caused by right hemisphere stroke) [34] are illustrated in Figure 33 below, using image processing techniques on our robotic wheelchair dataset. Notice how pedestrians are occluded in the hemineglect field of view, making safe wheelchair navigation challenging. Our user intent model seeks to address this challenging problem.

Figure 34. Normal visual field (left) compared to hemineglect partially restricted field of view (right).

6.3. Learning User Intent

The user intent model captures the intention of a user navigating the world using a joystick controlled robotic wheelchair. The model should thus capture a non-linear mapping from image sequences, along the wheelchair's trajectory, to the joystick drive input. To model this complex function, we employ a deep neural network architecture that integrates features and classifiers in an end-to-end multilayer fashion [35] to find compact non-linear low dimensional representations of high dimensional data.

The problem of learning shared autonomy user intent for wheelchair navigation can be framed as sequence learning of the joystick data along the robot's trajectory. To capture both the spatial and temporal aspects of this formulation, we designed a many-to-one CNN-LSTM network architecture for modeling user intent for wheelchair navigation. Our deep neural network architecture comprises of a Convolutional Neural Network (CNN) feature extractor, followed by a Long short-term memory (LSTM) recurrent neural network, and a final Sequential Layer (a fully connected layer followed by ReLu non-linearity and a final Linear layer for the regression output).

Our proposed network architecture is similar to the Long-term Recurrent Convolutional Networks LRCN class of models first proposed by Donahue et. al in [36] for activity monitoring, image captioning and video description. Our network architecture, which solves a regression problem of learning the user's intent (i.e. the joystick input) from an ideal driver is summarised in the Figure below.

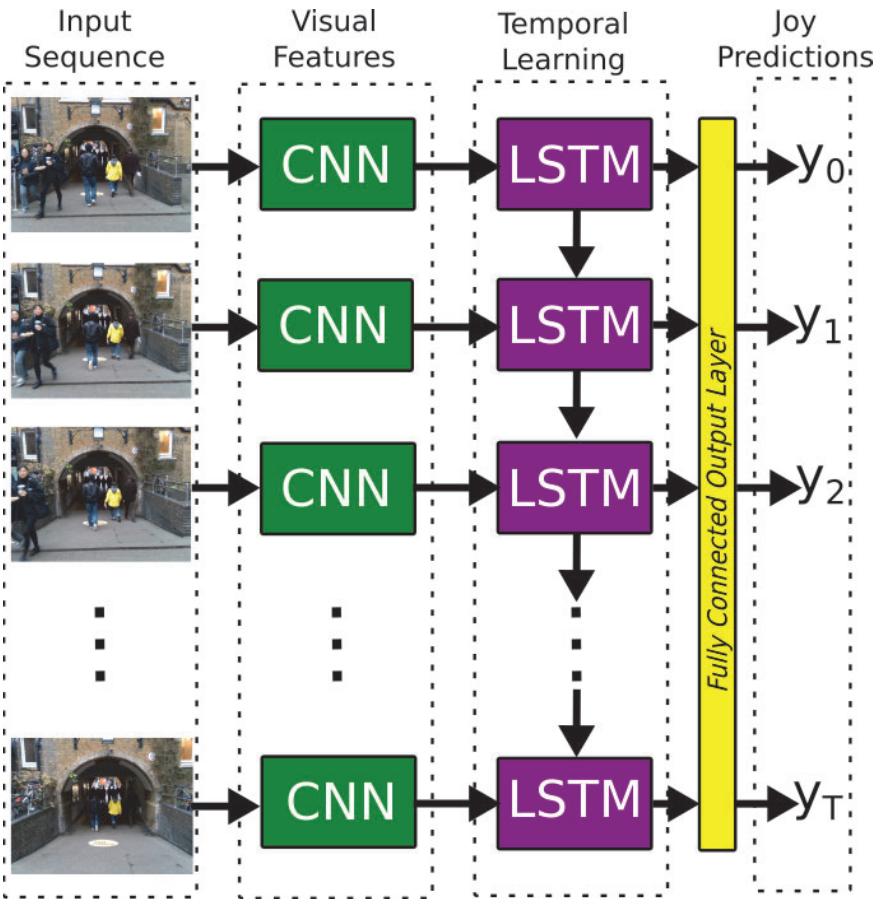

Figure 35. The CNN-LSTM model used for learning user driving intent, given a trajectory.

Specifically, a pre-trained ResNet-18 CNN model is used for feature extraction. We apply transfer learning to extract semantically rich fixed features from a sequence of RGB images along the robotic wheelchair's trajectory. The 512 dimensional feature sequence forms the input to a LSTM. The final regression output is generated by a sequence output layer as described above.

Our experiments showed that decoupling the translation and rotational axes of the joystick input results in better model parameter learning. We therefore train two separate regression models for the separate axes of the wheelchair's joystick input, namely the rotational and translational axes. The model for angular axis is a 5-layer LSTM with 128 hidden dimension size, whilst the radial axis model is a 2-layer LSTM with 256 hidden dimension size.

Formally, Recurrent Neural Networks (RNNs) model temporal dynamics by mapping input sequences to hidden states, and hidden states to outputs using the following recurrence equations:

$$h_t = g(W_{xh} x_t + W_{hh} h_{t-1} + b_h) \quad (21)$$

$$z_t = g(W_{hz} h_t + b_z) \quad (22)$$

where $h_t \in R^N$ and z_t is the hidden state with N hidden units and output at time t respectively. The input is given by x_t , and g represents a non-linear element-wise activation function. The model learns the weights W and biases b from the data. RNNs suffer from the exploding and vanishing gradient problems when propagating the gradients through many layers of the network (i.e. long timesteps) thus making them ill-suited for learning long-term dynamics [37]. LSTMs solve this problem by incorporating memory units such that the network can learn when to forget previous hidden states and how to propagate information through various gates [36]. In our implementation, our input (i_t) , forget (f_t) , cell (g_t) and output (o_t) gates are given by:

$$i_t = \sigma(W_{ii} x_t + b_{ii} + W_{hi} h_{t-1} + b_{hi}) \quad (23)$$

$$f_t = \sigma(W_{if} x_t + b_{if} + W_{hf} h_{t-1} + b_{hf}) \quad (24)$$

$$g_t = \tanh(W_{ig} x_t + b_{ig} + W_{hg} h_{t-1} + b_{hg}) \quad (25)$$

$$o_t = \sigma(W_{io} x_t + b_{io} + W_{ho} h_{t-1} + b_{ho}) \quad (26)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (27)$$

$$h_t = o_t \odot \tanh(c_t) \quad (28)$$

where σ is the Sigmoid function, c_t is the cell state at time t , and ⊙ is the Hadamard product.

Our model is trained using the Huber Loss (Smooth L1) function and the Adam optimiser with a starting learning rate of 0.001. We schedule the initial learning rate to decay on plateau by a factor of 0.1 using a threshold of 1e-4 and patience of 5. Both models are trained on the GPU using a batch size of 16 and timestep (sequence length) of 32 image sequences. The angular and radial axis models are trained for a total of 100 and 50 epochs, respectively.

We choose the Huber Loss over the Mean Squared Error (MSE) loss since it is less sensitive to outliers as defined below:

$$\text{loss}(x,y) = \frac{1}{n} \sum L_i \quad (29)$$

where L_i is given by:

$$L_i = \begin{cases} 0.5(x_i - \hat{y}_i)^2/\alpha & if \ \ | x_i - \hat{y}_i | < \alpha \\ | x_i - y_i | - 0.5\alpha & otherwise \end{cases} \quad (30)$$

We set α to 1 during training and x and y denote the true and predicted values, respectively. In our experiments, we collect ideal driving data from non visually impaired users. This data is used to train our user intent models for both joystick axes. Our empirical experiments showed improved performance when training two separate regression models, compared to a combined model to predict both axes simultaneously. This is, in part, due to the limited amount of training data we have relative to the number of parameters in the model.

6.4. Experimental Results

Our robot wheelchair platform has a motor controller unit that translates noisy user input (joystick) data into smooth velocity commands. The routes we considered also contain sequences of the wheelchair driving in the forward direction and taking multiple turns. The noisy joystick data also arises from the driving behaviours of users such as suddenly letting go of the joystick to come to a complete stop (which translates to a sharp discontinuity to zero in the translational axis).

Our user intent models are trained on a dataset comprising three trial driving sequences and validated on one long sequence. We trained two separate models for each axis of the joystick user interface. The angular axis user intent regression model, which captures the robotic wheelchair's angular rotation, achieved a MSE and RME of 0.048 and 0.219 respectively. The predictions (compared to the groundtruth) user intent model for decoupled rotation is shown in Figure 35.

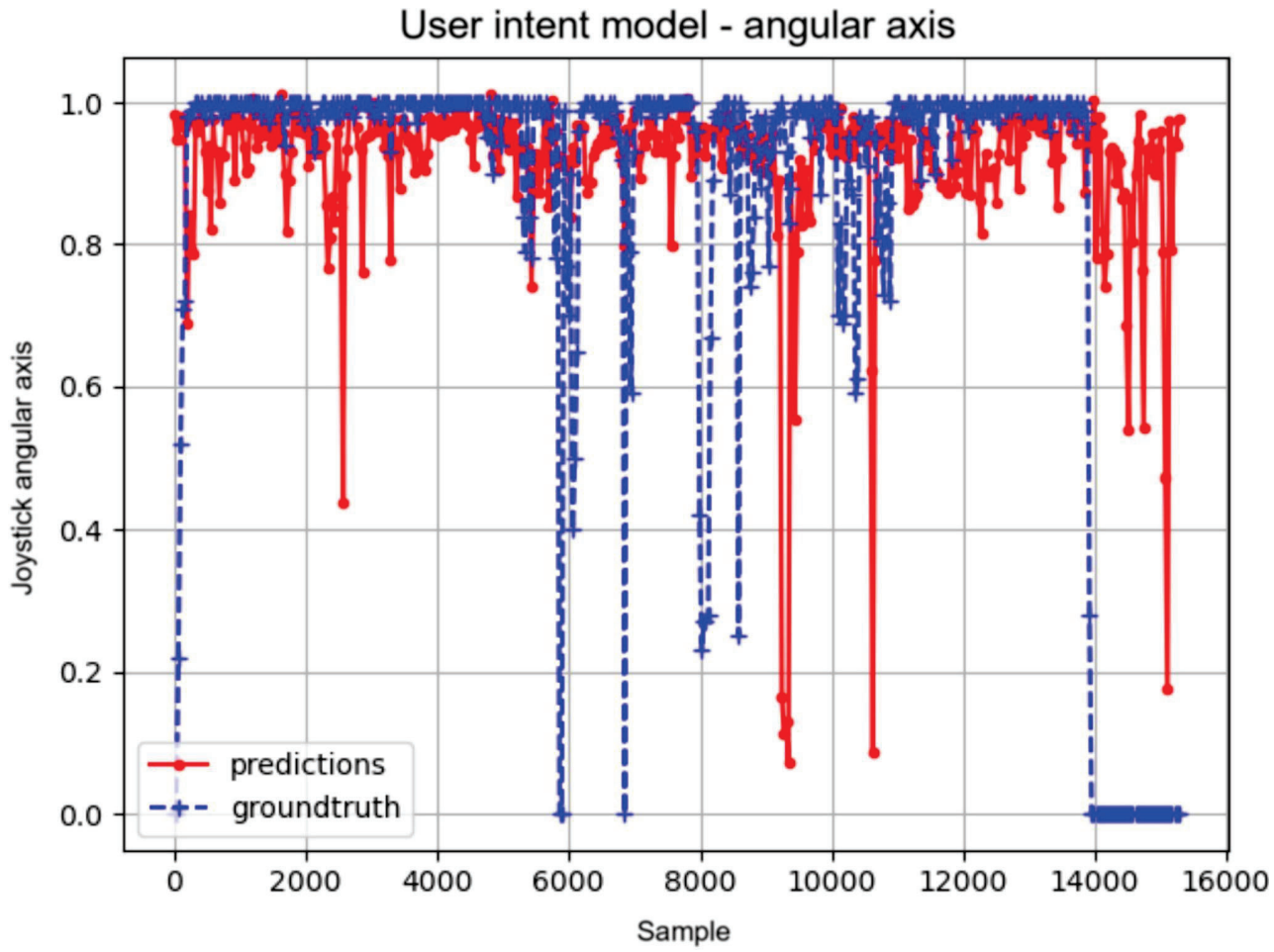

Figure 36. User intent model performance for the rotational joystick axis.

For the robotic wheelchair, a rotation of 1 represents a sharp right turn input signal, while a rotation of -1 represents a sharp left turn input signal. The sensitivity of the input is such that a difference of about 0.2 is not meaningfully noticeable for the wheelchair user.

The radial axis user intent regression model, which captures the robotic wheelchair's linear translational motion, performed with a MSE and RME of 0.100 and 0.316 respectively. The predictions and groundtruth of user intent joystick input for decoupled translation is shown in Figure 36. In this figure, 0 represents a stationary wheelchair and 1.0 is the wheelchair moving forward at maximum speed (for the selected joystick profile). Steering the wheelchair backwards would require a negative joystick input, where -1.0 represents reversing at maximum speed.

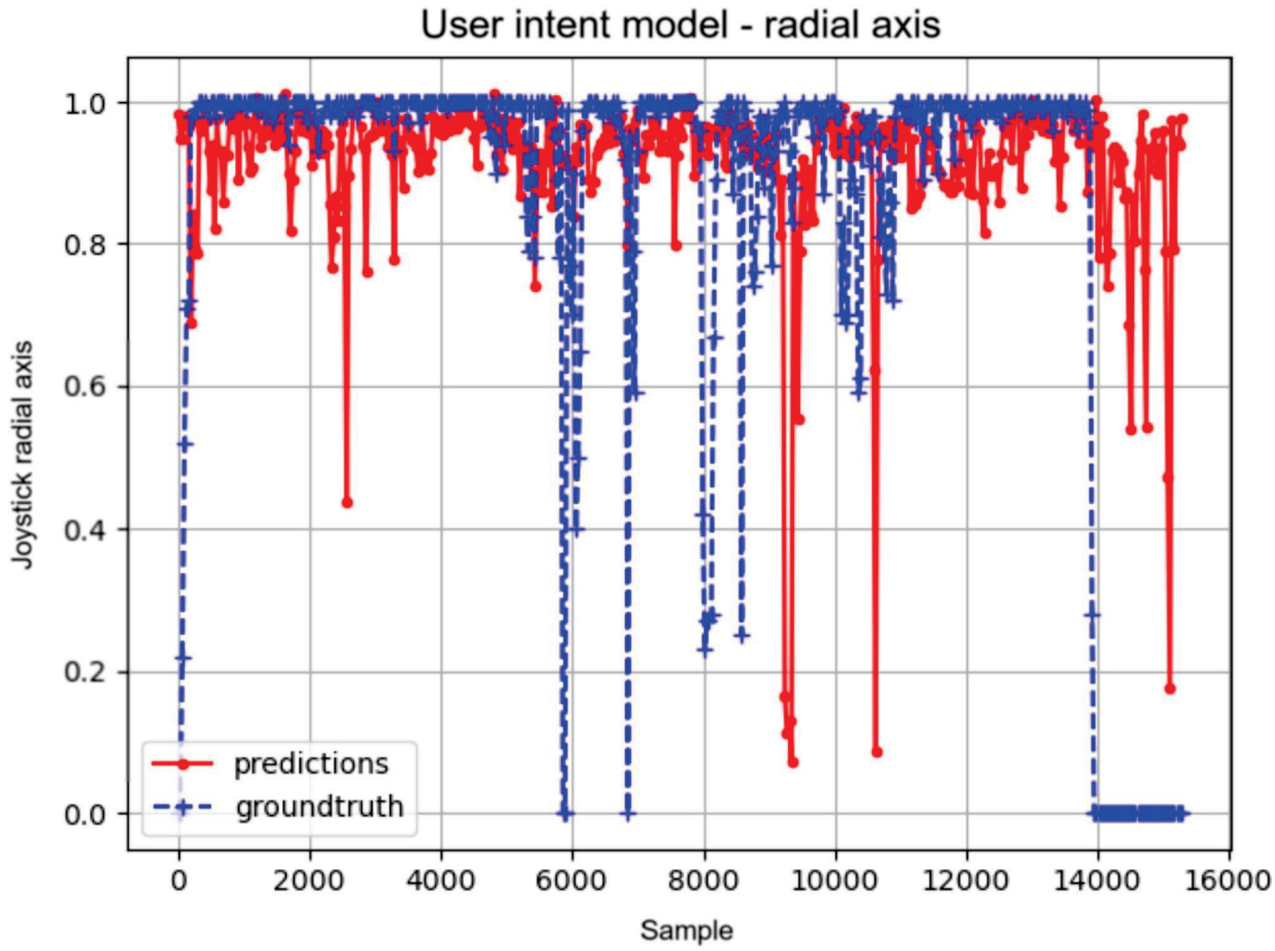

Figure 37. User intent model performance for the translational joystick axis.

Both of the results above (Figure 35, Figure 36) show a period where the wheelchair is stationary (at the end of the trial) and both of the joystick axes are at 0.

Our user intent model predicts varying forward joystick drive inputs which at times deviate from the ground truth (most notably the last section where the wheelchair is stationary but predictions are moving). This can largely be explained by the fact that our model captures, as input, image sequence data which could be drastically changing across the timesteps considered, due to people moving, even though the robotic wheelchair is stationary. Including data which encodes the motion of the wheelchair (for example the wheel encoders) would serve to mitigate this. Reliable wheel encoder data was not available at the time of the data collection.

Both models show that the user joystick input raw data is noisy since this is interpolated by the robot wheelchair's motor controller. Although learning from this noisy data is challenging, our models perform within the observed error tolerance of the raw joystick input data.

6.5. Discussion

User intent refers to the hidden unknown internal state of a wheelchair user. Our model specifically seeks to predict the driving intent of a wheelchair user along a given trajectory which is parameterised by a sequence of images. Our initial results are promising and show that it is possible to learn decoupled user input models from noisy data. In future, we envision that such models could be used to replace or augment incomplete user input, for example that provided by patients with visual impairments (such as hemineglect), thus facilitating increased mobility and independence.

7. Conclusion

In this deliverable, we presented a range of different methods to achieve shared control in a pedestrian-populated environment. We then proposed a novel hierarchical design where we extended the Probabilistic Shared Control (PSC) implementation to account for the probability distribution of dynamic obstacles and their influence on robot navigation. In this approach, the shared control is achieved by maximizing the joint probability between the user, the path planner and the surrounding obstacles. We evaluated this method from six crucial aspects: path efficiency, effect on crowds, crowd-robot interaction, pedestrian-robot similarity, collisions, and shared control performance. Simulator experiment results indicate that even though the modelled pedestrians were strongly affected by the wheelchair's motion, it was able to follow the user's intention (unless there was a collision risk). The wheelchair was able to support the user and navigate efficiently in an environment populated with moving pedestrians, whilst minimising physical contact. When compared with other approaches, our method showed the least number of collisions while obtaining relatively low computational cost and high user agreement.

In addition, we have developed a method for extracting user control input from sensory data through machine learning, closing the shared control loop with an inverse function. In the future, this could enable the replacement of missing or corrupted control input, opening the possibility for methods based on continuous control sharing to be used with a wider range of impairments.

The impact of COVID19 restrictions on this deliverable has been non-trivial. In particular, all non-simulated crowd experiments and user studies were put on hold as a result. Nevertheless we were able to utilise our substantial real-world crowd-robot datasets that we collected prior to the pandemic to underpin our modelling. In the future, we will further improve this method by incorporating a more complex interaction function and perform further experiments in (physical) crowds, which will be reported in Deliverable 1.5.

References

[1] R. Simpson, E. LoPresti, S. Hayashi, I. Nourbakhsh, and D. Miller, 'The Smart Wheelchair Component System', *J. Rehabil. Res. Dev.*, vol. 41, no. 3b, p. 429, 2004, doi: 10.1682/JRRD.2003.03.0032.

[2] E. Prassler, J. Scholz, and P. Fiorini, 'A Robotic Wheelchair Roaming in a Railway Station', Dec. 1998.

[3] B. Kim and J. Pineau, 'Socially Adaptive Path Planning in Human Environments Using Inverse Reinforcement Learning', *Int. J. Soc. Robot.*, vol. 8, no. 1, pp. 51–66, Jan. 2016, doi: 10.1007/s12369-015-0310-2.

[4] T. Carlson and Y. Demiris, 'Human-wheelchair collaboration through prediction of intention and adaptive assistance', in *2008 IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, May 2008, pp. 3926–3931, doi: 10.1109/ROBOT.2008.4543814.

[5] C. Ezeh, P. Trautman, D. Louise, V. Bureau, M. Babel, and T. Carlson, 'Probabilistic vs linear blending approaches to shared control for wheelchair driving', *IEEE Int. Conf. Rehabil. Robot. Proc.*, vol. 2017, pp. 835–840, Jul. 2017, doi: 10.1109/ICORR.2017.8009352.

[6] Qinan Li, Weidong Chen, and Jingchuan Wang, 'Dynamic shared control for human-wheelchair cooperation', in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 4278–4283, doi: 10.1109/ICRA.2011.5980055.

[7] C. Gao, M. Sands, and J. R. Spletzer, 'Towards Autonomous Wheelchair Systems in Urban Environments', in *Field and Service Robotics*, Berlin, Heidelberg, 2010, pp. 13–23, doi: 10.1007/978-3-642-13408-1_2.

[8] E. A. Biddiss and T. T. Chau, 'Upper limb prosthesis use and abandonment: A survey of the last 25 years':, *Prosthet. Orthot. Int.*, Jun. 2016, doi: 10.1080/03093640600994581.

[9] A. Escobedo, A. Spalanzani, and C. Laugier, *Multimodal Control of a Robotic Wheelchair: Using Contextual Information for Usability Improvement*. 2013, p. 4267.

[10] M. R. M. Tomari, Y. Kobayashi, and Y. Kuno, 'Development of smart wheelchair system for a user with severe motor impairment', *Procedia Eng.*, vol. 41, pp. 538–546, 2012, doi: 10.1016/j.proeng.2012.07.209.

[11] R. C. Simpson and S. P. Levine, 'Automatic adaptation in the NavChair Assistive Wheelchair Navigation System', *IEEE Trans. Rehabil. Eng.*, vol. 7, no. 4, pp. 452–463, Dec. 1999, doi: 10.1109/86.808949.

[12] C. Urdiales, J. M. Peula, M. Fernandez-Carmona, R. Annicchiaricco, F. Sandoval, and C. Caltagirone, 'Adaptive collaborative assistance for wheelchair driving via CBR learning', in *2009 IEEE International Conference on Rehabilitation Robotics*, Jun. 2009, pp. 731–736, doi: 10.1109/ICORR.2009.5209575.

[13] M. Fernandez-Carmona, B. Fernandez-Espejo, J. M. Peula, C. Urdiales, and F. Sandoval, 'Efficiency based collaborative control modulated by biometrics for wheelchair assisted navigation', in *2009 IEEE International Conference on Rehabilitation Robotics*, Jun. 2009, pp. 737–742, doi: 10.1109/ICORR.2009.5209573.

[14] O. Khatib, 'Real-time obstacle avoidance for manipulators and mobile robots', in *1985 IEEE International Conference on Robotics and Automation Proceedings*, Mar. 1985, vol. 2, pp. 500–505, doi: 10.1109/ROBOT.1985.1087247.

[15] Y. K. Hwang and N. Ahuja, 'A potential field approach to path planning', *IEEE Trans. Robot. Autom.*, vol. 8, no. 1, pp. 23–32, Feb. 1992, doi: 10.1109/70.127236.

[16] G. Li, A. Yamashita, H. Asama, and Y. Tamura, 'An efficient improved artificial potential field based regression search method for robot path planning', in *2012 IEEE International Conference on Mechatronics and Automation*, Aug. 2012, pp. 1227–1232, doi: 10.1109/ICMA.2012.6283526.

[17] J. Borenstein and Y. Koren, 'The vector field histogram-fast obstacle avoidance for mobile robots', *IEEE Trans. Robot. Autom.*, vol. 7, no. 3, pp. 278–288, Jun. 1991, doi: 10.1109/70.88137.

[18] I. Ulrich and J. Borenstein, 'VFH+: reliable obstacle avoidance for fast mobile robots', in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, Leuven, Belgium, 1998, vol. 2, pp. 1572–1577, doi: 10.1109/ROBOT.1998.677362.

[19] I. Ulrich and J. Borenstein, 'VFH/sup */: local obstacle avoidance with look-ahead verification', in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, Apr. 2000, vol. 3, pp. 2505–2511 vol.3, doi:

10.1109/ROBOT.2000.846405.
[20] D. Fox, W. Burgard, and S. Thrun, ‘The dynamic window approach to collision avoidance’, *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997, doi: 10.1109/100.580977.
[21] P. Fiorini and Z. Shiller, ‘Motion Planning in Dynamic Environments Using Velocity Obstacles’:, *Int. J. Robot. Res.*, Jul. 2016, doi: 10.1177/027836499801700706.
[22] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, ‘Reciprocal n-Body Collision Avoidance’, in *Robotics Research*, vol. 70, C. Pradalier, R. Siegwart, and G. Hirzinger, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 3–19.
[23] M. Seder, K. Macek, and I. Petrovic, ‘An integrated approach to real-time mobile robot control in partially known indoor environments’, *31st Annu. Conf. IEEE Ind. Electron. Soc. 2005 IECON 2005*, 2005, doi: 10.1109/IECON.2005.1569176.
[24] D. Wilkie, J. van den Berg, and D. Manocha, ‘Generalized velocity obstacles’, in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA, Oct. 2009, pp. 5573–5578, doi: 10.1109/IROS.2009.5354175.
[25] H. Emam, Y. Hamam, E. Monacelli, and K. D. Djouani, ‘Power wheelchair driver behaviour modelling’, *2010 7th Int. Multi- Conf. Syst. Signals Devices*, 2010, doi: 10.1109/SSD.2010.5585578.
[26] C. Urdiales *et al.*, ‘An Adaptive Scheme for Wheelchair Navigation Collaborative Control’, Jan. 2008.
[27] P. Trautman, ‘Assistive Planning in Complex, Dynamic Environments: A Probabilistic Approach’, *2015 IEEE Int. Conf. Syst. Man Cybern.*, 2015, doi: 10.1109/SMC.2015.534.
[28] R. L. Kirby, J. Swuste, D. J. Dupuis, D. A. MacLeod, and R. Monroe, ‘The Wheelchair Skills Test: A pilot study of a new outcome measure’, *Arch. Phys. Med. Rehabil.*, vol. 83, no. 1, pp. 10–18, Jan. 2002, doi: 10.1053/apmr.2002.26823.
[29] R. V. Levine and A. Norenzayan, ‘The Pace of Life in 31 Countries’:, *J. Cross-Cult. Psychol.*, Jul. 2016, doi: 10.1177/0022022199030002003.
[30] C. Ezeh, C. Holloway, P. Trautman, and T. Carlson, ‘Comparing Shared Control Approaches for Alternative Interfaces: A Wheelchair Simulator Experiment’, p. 6.
[31] M. Seder and I. Petrovic, ‘Dynamic window based approach to mobile robot motion control in the presence of moving obstacles’, in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, Apr. 2007, pp. 1986–1991, doi: 10.1109/ROBOT.2007.363613.
[32] Stott, Ian, and David Sanders. "New powered wheelchair systems for the rehabilitation of some severely disabled users." *International Journal of Rehabilitation Research* 23.7 (2000): 149-153.
[33] Corbetta, Maurizio, and Gordon L. Shulman. "Spatial neglect and attention networks." *Annual review of neuroscience* 34 (2011): 569-599.
[34] Polanowska, Katarzyna, et al. "Left-hand somatosensory stimulation combined with visual scanning training in rehabilitation for post-stroke hemineglect: a randomised, double-blind study." *Neuropsychological Rehabilitation* 19.3 (2009): 364-382.
[35] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
[36] Donahue, Jeffrey, et al. "Long-term recurrent convolutional networks for visual recognition and description." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
[37] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.
[38] Javdani, Shervin, Siddhartha S. Srinivasa, and J. Andrew Bagnell. "Shared autonomy via hindsight optimization." *Robotics science and systems: online proceedings* 2015 (2015).