EU Horizon 2020 Research & Innovation Program

Advanced Robot Capabilities & Take-Up

ICT-25-2016-2017

Safe Robot Navigation in Dense Crowds

http://www.crowdbot.org

Technical Report

D 1.5: 2nd Round Test Evaluation Report

Work Package 1 (WP 1)

Scenarios Co-Design & Evaluation

Task Lead: University College London, UK

WP Lead: University College London, UK

Table of Contents

Executive Summary

In this final deliverable of WP1, we discuss the second series of evaluations performed with each of our integrated Crowdbots: the smart wheelchair, Qolo, Pepper and cuyBot. The evaluations are both quantitative and qualitative and range from testing the efficacy of core components that are developed within the project and underpin the Crowdbots — namely: sensing, localisation, simulation, planning — to fully integrated tests of each platform for some of the scenarios identified in D1.3 "Specification of Scenarios Requirements Update".

Two distinct evaluations are performed on the smart wheelchair. First, the performance of our proposed reinforcement learning method for creating a more personalized shared control experience, which is validated in the CrowdBot simulator. Second, we evaluated our previously proposed probabilistic shared control method PSC-DWAGVO (D3.6) on the physical wheelchair in a controlled sparse crowd environment, before demonstrating its feasibility in a natural sparse crowd on the main UCL campus in London, UK.

Qolo has been tested in natural crowds in Lausanne, Switzerland. Reactive navigation without a high-level planner was successful and the shared control felt very natural, with people going by without noticing the robot. By contrast, the direct point-to-point controller had problems of not interacting properly with the environment and resulted in less intuitive behavior. People detection was reported to work very well, however, tracking was a bottleneck when there were many people surrounding Qolo. All the controllers, and dataset of experiments have been made publicly available along with analysis tools for pedestrian and robot performance analysis.

For our fully autonomous Crowdbots, training robust sensor-to-control policies for robot navigation remains a challenging task. We trained two end-to-end, and 18 unsupervised-learning-based architectures, and compared them, along with existing approaches, in unseen test cases. We also demonstrated our approach working on the Pepper robot in a real environment. Our results show that unsupervised learning methods are competitive with end-to-end methods. We also highlight the importance of various components such as input representation, predictive unsupervised learning, and latent features. Our models have been made publicly available, along with the training and testing environments, and tools.

Finally, the compliant motion control on cuyBot was evaluated during physical experiments with human participants. The experiment demonstrated that the current implementation of the compliant motion control on the cuyBot robot was safe, even with inexperienced users. Our results suggest that the concept of robots getting into close contact with humans may be accepted by the public, as long as the functionality of compliant motion is implemented in a safe and intuitive way. Once persons gathered experience with the robot, on average they have a better than neutral attitude towards close proximity and physical contacts.

In summary, we have demonstrated that our two semi-autonomous systems have both shown promising results for the use of shared control in crowded environments. Similarly, our two fully-autonomous Crowdbots have demonstrated the feasibility of both navigation capability in dynamic human environments, as well as the safety and reasonable acceptance of our compliant motion control when making physical contact with pedestrians.

1. Introduction

This second-round evaluation report builds upon our learnings from the first-round evaluations (D1.4 “Initial evaluation report”) and in particular addresses the scenarios and specifications that were consequently updated in D1.3 “Specification of Scenarios Requirements Update”.

In this report, we discuss the final evaluations performed with each of our integrated Crowdbots: the smart wheelchair, Qolo, Pepper and cuyBot. The evaluations are both quantitative and qualitative and range from testing the efficacy of core components that are developed within the project and underpin the Crowdbots — namely: sensing, localisation, simulation, planning — to fully integrated tests of each platform for some of the scenarios identified in D1.3.

Each robot has been evaluated using the scenarios and metrics most appropriate for its intended purpose and each test was planned in advance using the methodologies developed in D1.3. Due to Covid, some of the planned physical tests were postponed and additional evaluations were made utilising the CrowdBot simulator, developed in WP4. Nonetheless, in this deliverable we detail a range of exciting evaluations and demonstrations of our physical Crowdbots in the real world with human participants and natural crowds.

The rest of this deliverable is organised by the prototype Crowdbot, beginning with the smart wheelchair, Qolo, Pepper and finally cuyBot. Each of these sections is structured, beginning with a clear research question before moving on to describe the experiments undertaken and present the CrowdBot specific results. The deliverable concludes that our two semi-autonomous systems have both shown promising results for the use of shared control in crowded environments. Similarly, evaluations of our two fully-autonomous Crowdbots have demonstrated the feasibility of both navigation capability in dynamic human environments, as well as the safety and reasonable acceptance of our compliant motion control when making physical contact with pedestrians.

2. Smart wheelchair

In this section we report on the two main evaluations that were performed on the smart wheelchair, as a result of the updated specification of scenarios requirements and our continued stakeholder engagement. First, the performance of our proposed reinforcement learning method for creating a more personalized shared control experience is validated in the CrowdBot simulator. Second, we evaluated our previously proposed probabilistic shared control method PSC-DWAGVO (D3.6) on the physical wheelchair in a controlled sparse crowd environment, before finally demonstrating its feasibility in a natural sparse crowd on the main UCL campus in London, UK.

2.1. Research question

We have seen that traditional navigation methods could lead to a "freezing robot" issue (Trautman et al., 2015) when the crowd density increases. To address this problem, it is important to understand the interactions and/or cooperation between pedestrians and pedestrian-robots.

In D1.4, we presented and evaluated a hierarchical framework DWAGVO-PSC for shared control navigation in human environments with simple hand-crafted interaction functions. In reality, interaction and cooperation between pedestrians, as well as between pedestrians and robots are difficult to model. To this end, deep reinforcement learning (RL) approaches have been actively studied due to their reported high performance and robustness to changes in the environment. While state-of-the-art work focuses on navigation for fully autonomous robots, the shared control wheelchair poses additional requirements on the problem as a user is involved in the interaction and control loop. As an assistive technology, shared-control should provide assistance that is consistent with the user's driving style.

In this report, we explore the user-based shared-control crowd navigation problem from a reinforcement learning way. Therefore, our research question here is:

- How to learn different user driving styles in complex environments?
- Would a reinforcement-learning based shared-control framework work for wheelchair navigation in crowds?

2.2. Reinforcement learning based shared-control navigation

2.2.1. Background

2.2.1.1. Proximal Policy Optimization

Reinforcement learning (RL) has received increasing attention in recent years for tasks that may be too complex to model, however easier to be learnt. In general, state-of-the-art RL approach is either value-based or policy-based. One of the most widely used policy-based algorithms is called proximal policy optimisation (PPO). Proposed by Schulman et al. (2017b), PPO has demonstrated promising results and has become the default reinforcement learning algorithm at OpenAI. In this work, we decided to use PPO as our main RL algorithm as it handles continuous action spaces, and directly modify the policy during training, which is more suitable for navigation applications. While many RL methods suffer from stability issues, PPO guarantees stability during training by setting a trust region (Schulman et al., 2017a). The main working principle for policy gradient methods lies in computing an estimator of the policy gradient and plugging it into a stochastic gradient ascent algorithm. In general, the estimator ˆg can be obtained by differentiating an objective function $L^{PG}(\theta)$.

$$L^{PG}(\theta) = E_t[log\pi_\theta(a_t|s_t)A]$$
$$g = E_t[\nabla_\theta log\pi_\theta(a_t|s_t)A]$$

where s_t is the state and a_t is the action, π_θ is a stochastic policy and A_t is an estimator of the advantage function at timestep t.

The standard state-action value function Q_π and the value function V_π are,

$$Q_\pi(s_t, a_t) = E_{s_{t+1}, a_{t+1}\ldots}[\sum_{t=0}^{\infty} \gamma^l r(s_{t+l})]$$

$$V_\pi(s_t) = E_{a_t, s_{t+1}\ldots}[\sum_{t=0}^{\infty} \gamma^l r(s_{t+l})]$$

The advantage term is defined as their difference.

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$$

PPO constructs its objective function as follow,

$$L^{CPI}(\theta) = E_t[\pi_\theta(a_t|s_t)/\pi_{\theta old}(a_t|s_t)A_t] = E_t[r_t(\theta)A_t]$$

The maximization of $L^{CPI}(\theta)$ would lead to an excessively large policy update and yield instability. While trust region policy optimisation (TRPO) guarantees to treat this issue by limiting the KL divergence between the new policy and the old policy, PPO simplifies it by clipping the objective function. Consequently, the main objective function for PPO is defined as,

$$L^{CLIP}(\theta) = E_t[min(r_t(\theta)A_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t]$$

$$clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon) = \begin{cases} r_t(\theta) & \text{if } r_t(\theta) \geq 1-\varepsilon \\ 1-\varepsilon & \text{if } r_t(\theta) \leq 1-\varepsilon \\ 1+\varepsilon & \text{if } r_t(\theta) \geq 1+\varepsilon \end{cases}$$

where ε is a hyperparameter that modifies the clipping length. By taking the minimum of the clipped and unclipped objective, the final objective becomes a lower bound on the unclipped objective. Schulman et al. (2017b) show that PPO outperforms other online policy gradient methods, while maintaining a favourable balance between sample complexity, simplicity, and computation time.

2.2.1.2. Residual Policy Learning

Shared control can be formed as a POMDP problem where the user's goal (or short-term intention) is partially observed by the agent (wheelchair). State-of-the-art data driven approaches infer the user's goal using inverse reinforcement learning (Abbeel and Ng, 2004; Ziebart et al., 2008) or hindsight optimization (Javdani et al., 2015). However, these methods normally require a known user goal space or a transition function which can be difficult to obtain. Recently, Silver et al. (2019) proposed Residual Policy Learning (RPL), which aims to improve nondifferentiable policies using model-free deep reinforcement learning. The main idea of RPL is to learn a residual policy $\pi_\theta(s)$ from a residual function $f_\theta(s)$ and augment on top of some arbitrary initial policy π(s).

$$\pi_\theta(s) = \pi(s) + f_\theta(s)$$

By evaluating the performance on a robot picking task, with a hand designed policy and a model predictive controller (MPC) as initial policies, Silver et al. (2019) showed that RPL does not only improves on initial policies but is also more data-efficient than learning from scratch. While this method could help with imperfect controllers, it can also be combined with imitation learning where the user's demonstration is imperfect.

2.2.1.3. Learning based shared autonomy

While deep reinforcement learning has been widely used in fully autonomous robots, there is very limited research on using deep RL in a control sharing setting. Recently, Reddy et al. (2018) addressed this problem by decomposing the reward function into two parts. One part captures general requirements such as collision avoidance, while the other captures user generated feedback. The former part is known to the agent and is updated every timestep, while the second part is not known and is used as a terminal reward which only emits a large value based on the user's feedback when each episode is completed. In this way, the control sharing is achieved by involving user feedback in the reward function. In addition, they claimed that by using deep neural networks, the agent can discover arbitrary relationships between user controls and observations of the physical environment directly, eliminating the need to explicitly assume the existence of a goal. A main limitation for

this method is it requires discrete human input during training, which could be impractical and problematic when continuous input is required. In this work, we aim to address this issue by combining residual policy learning with deep reinforcement learning. The inspiration is drawn from the work presented by Schaff and Walter (2020), where they created a surrogate user by behaviour cloning, and augmented this user policy with a learnt residual policy. By evaluating their methods on continuous gaming tasks (eg. Lunar Lander, Lunar Reacher, and Drone Reacher), they showed it greatly improved the performance of human operators.

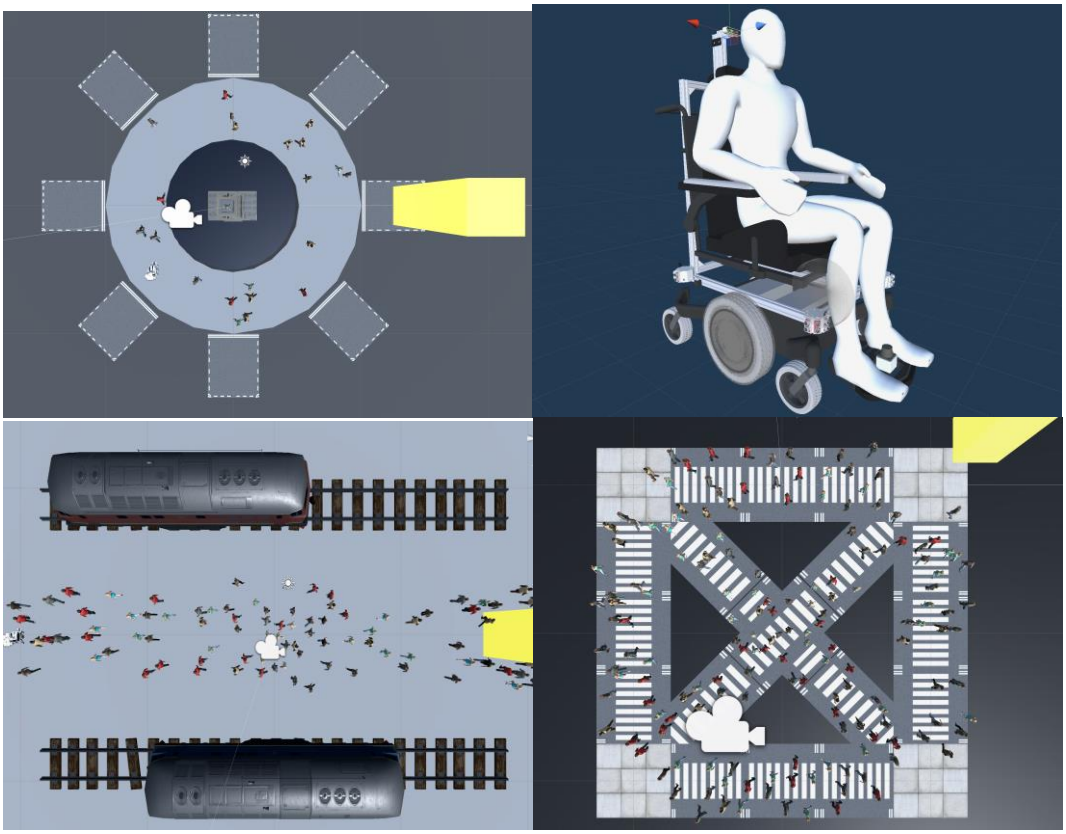

Figure 2.1. The virtual wheelchair and three trained crowds' scenarios. In Each Scenario, the participant has to drive the wheelchair from the starting location to reach the yellow goal location.

2.2.2. Method

2.2.2.1. Problem Formulation

As an assistive device, a shard-control wheelchair should be able to provide user-centred assistance in appropriate time. Imagine a naive wheelchair user, their driving style could be cautious and polite. If they were assisted by an aggressive motion planner, they would potentially be confused and uncomfortable. Conversely, imagine an experienced wheelchair user being assisted in a cautious way, they may find it annoying. This requires the system to learn personalized user policies. Therefore, we break down the navigation problem into two sub-problems. The first one concerns the learning user driving style, while the second one takes the user model and achieves shared-control navigation in real-time.

2.2.2.2. Simulator Setup

To facilitate various driving styles and test the generalization of our approach, three scenarios have been designed in Unity (see Figure 2.1). Each scenario is populated with crowds and is designed to capture various movement patterns (circle, 1d flow, 2d flow). Pedestrian agents are designed with realistic dynamics and their collision avoidance is governed by ORCA (van den Berg et al., 2011). For more detail of the simulator, see Grzeskowiak et al. (2021). In each scenario, a random 10% of the pedestrians are designed to ignore the wheelchair (i.e. no collision avoidance to the wheelchair), this simulates the people who are distracted. Pedestrians are set at a target velocity of 1.3m/s with a Gaussian noise (Levine and Norenzayan, 1999). The

simulated wheelchair receives the final linear and angular command, which is sent to a differential PID controller that governs the movement of two wheels. The non-holonomic constraints are considered in its kinematic model. The maximum linear velocity for the wheelchair is 1.3 m/s while the maximum angular velocity is set to 0.785 rad/s. These values are within the velocity limit of our wheelchair and are set to be comparable with the human walking speed for observing potential various interaction behaviours.

2.2.2.3. Learning User Driving Style

To obtain user data, 20 healthy participants from the UCL psychology participant pool were recruited, all of them are over 18, with unimpaired vision and wrist mobility. Each of them was given 10 minutes to get familiar with the setup before the experiment. They were then instructed to drive a simulated wheelchair with a wheelchair joystick for 45 minutes in three simulated scenarios (15 minutes for each scenario). Participants were asked to drive the wheelchair in their own style and reach a goal (highlighted in yellow) in each scenario. Once the goal was reached, they were brought to a new scenario generated in a random sequence.

During the data collection, participants have the first-person view of the wheelchair. All driving data and environment data were collected at the frequency of 0.1s. Collected data were then pre-processed, which included cleaning unrealistic samples caused by transition and then segmenting the whole data sequence into short sequences with a length of 2s each. After evaluation of various features and their combination, we found "The average distance to the nearest pedestrian", "The average linear velocity", "The average collision number" and "The average local density" gave us the best description of the driving data.

Principal component analysis (PCA) was used to further reduce the dimensionality, which produced two principal components with explained variance > 99.5%. To explore the existence of various driving styles, clustering was performed by using K-means. The number of clusters was determined by the elbow method. In the end, two clusters were produced and the result was verified with a silhouette coefficient > 0.6. While an individual user could potentially exhibit different driving styles, for simplicity we assumed their driving was consistent during the experiment. This allowed us to categorize the collected user data into two driving styles (See Figure 2.2).

Behaviour cloning and imitation learning has been used to learn user policies from human demonstrations in previous works (Cèsar-Tondreau et al., 2021; Kuefler et al., 2017). When it comes to crowds navigation scenarios, the various states make it impracticable to directly learn a motion output from limited and potentially imperfect human demonstrations. As a result, we use Generative Adversarial Imitation Learning (GAIL) (Ho and Ermon, 2016) to learn a user policy, which not only leverages the information in the demonstration but also allows exploration of other unvisited states.

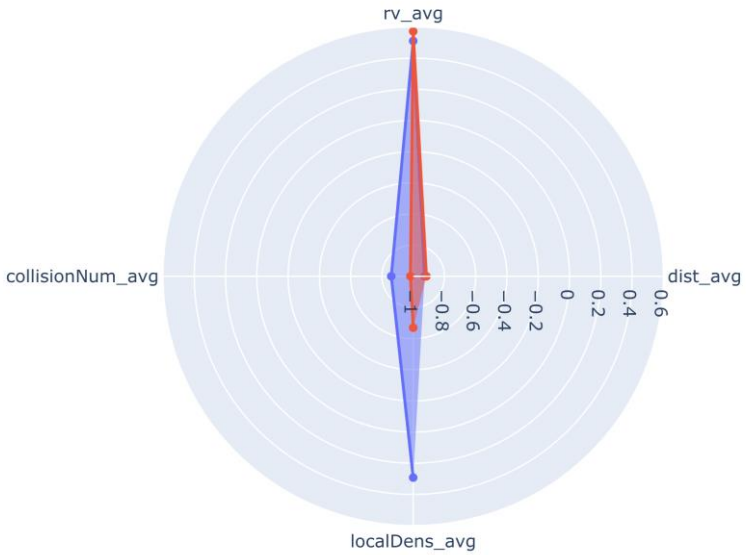

Figure 2.2. Radar plot of the two classified driving style.

2.2.2.4. Shared-control via reinforcement learning

For shared-control navigation, continuous user input is required throughout the whole process. This characteristic differs itself from previous work where discrete user input is provided as feedback (Reddy et al., 2018). In addition, continuous user input will require extensive user interaction during training, the amount of training time may take tens of hours based on the difficulty of the task and the complexity of the scenario. It is impractical for every wheelchair user to be supervised for that long before they can actually use the wheelchair.

As a result, the problem was approached by residual policy learning where the initial policy comes from a surrogate user that is trained offline. During training, the user input is sampled from the surrogate user policy, which is augmented with the robot states. The control sharing is achieved by shaping the rewards as a joint probability. To be more specific, the state of the robotic wheelchair is set as $[g_x, g_y, v_r, w_r, p_{ix}, p_{iy}, p_{ivx}, p_{ivy}, v_u, w_u]$. g_x, g_y is the position of the goal, v_r, w_r represent the velocity of the wheelchair. In this paper, i takes range from 0 to 9 which includes the 10 nearest pedestrians within 5 meters of the wheelchair. This value is set by considering the range of the people tracker which gives us a local density about $1p/m^2$. If fewer than 10 people are detected around the wheelchair, the position values are padded with the maximum range and the velocity values are filled with 0. All values are wheelchair-centric. During training time, the wheelchair and environment related states are received from the simulator, while v_u and w_u are inferenced from the surrogate user model. In this work, we assume the wheelchair is only aware of the long-term goal (i.e. reaching the final goal) while not having direct access to the user's short-term intention. We assume the user's intention is encoded in its user policy and is only partially observable. As a result, we use a window of 3 time-steps (0.3s) for the states to make it suitable for the recurrent neural network. A key in achieving control sharing via reinforcement learning is shaping its reward. Inspired by the work in (Reddy et al., 2018; Schaff and Walter, 2020), we divided the reward function R^t into two parts, one part ${R^t}_r$ that solves basic robot navigation requirements such as avoid collisions ${R^t}_c$, reach the final goal ${R^t}_g$ and encourage smoother trajectory by penalizing sudden big change in angular velocity ${R^t}_{com}$, while the other part (${R^t}_u$) rewards following the user's intention. We form our reward in a joint probability way. Figure 2.3 shows a high-level summary of our proposed approach.

$$R^t = (R^t_r) * (R^t_u)$$

$$R^t_r = R^t_g + R^t_c + R^t_{com}$$

$$R^t_g = \begin{cases} r_d * (\left\|\mathbf{p_r^{t-1}} - \mathbf{g}\right\| - \left\|\mathbf{p_r^t} - \mathbf{g}\right\|) & \text{Otherwise} \\ r_g & \text{if } \left\|\mathbf{p_r^t} - \mathbf{g}\right\| \leq 2 \end{cases}$$

$$R^t_c = \begin{cases} 0 & \text{Otherwise} \\ r_c & \text{if } \left\|\mathbf{p_r^t} - \mathbf{p_i^t}\right\| \leq 0.66 \\ r_{cc} * \left\|\mathbf{p_r^t} - \mathbf{p_i^t}\right\| & \text{if } \left\|\mathbf{p_r^t} - \mathbf{p_i^t}\right\| \leq 1.5 \end{cases}$$

$$R^t_{com} = -0.01 * |w^t|$$

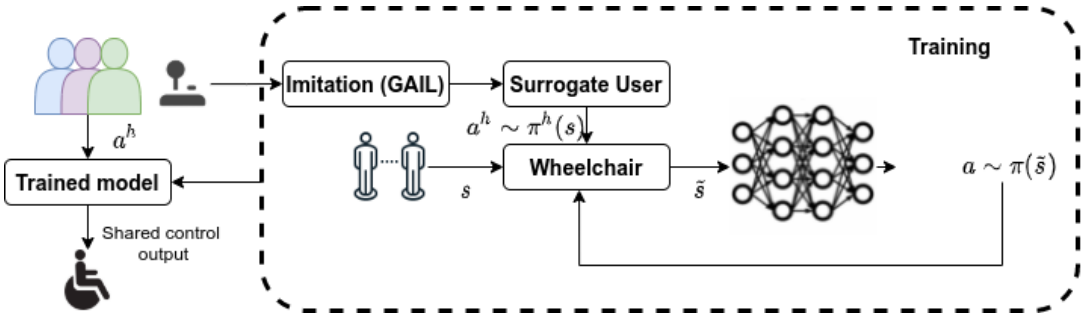

Figure 2.3. A summary of our proposed approach. The state input to the neural network is a combination of the robot(wheelchair) information, environment(crowds) information and the predicted input comes from the surrogate user.

Where ${p_r}^{t-1}$ is the position vector of the robot at time t-1 and g represents the position of the goal. In our implementation, $r_d = 5, r_g = 50, r_c = -20$. In terms of the user, we use a function to evaluate the consistency of the wheelchair motion candidate with the surrogate user policy output.

$$R_u = r_a * exp^{\lambda(\|{v_r}^t - {v_u}^t\|^2 + \|{w_r}^t - {w_u}^t\|^2)}$$

Parameter λ controls how the user input and the final motion are correlated. Through trial and error, we found λ =-0.5 gives us a reasonable trade-off. r_a is set to 0.8.

2.3. Simulator Evaluation

2.3.1. Implementation details

All training and testing in the simulator are implemented in Unity 3D 2019.3.14f, with mlagents version 15. The laptop has Intel® Core™ i7-9750H CPU @ 2.60GHz × 12, with graphics card GeForce RTX 2070 with Max-Q Design/PCIe/SSE2. The imitation learning was performed using GAIL without extrinsic rewards. The user policies converge after approximately 50k steps.

In terms of the final shared-control policy, we adopted a curriculum learning strategy due to the complexity of the task. During some initial trials, we noticed that populating all the human agents at the very beginning of the training may result in undesirable behaviours, which includes the wheelchair wandering around in the free area or hit with obstacles immediately as it expected more costs to reach the goal. Therefore, we break down the learning objectives by first training the wheelchair to reach the goal in a human-free environment. After that, we keep populating crowds until the local density reaches $1p/m^2$. The final policy converged after about 200k steps with high agreeability between the surrogate user input and the wheelchair motion (See Figure 2.4 and Figure 2.5).

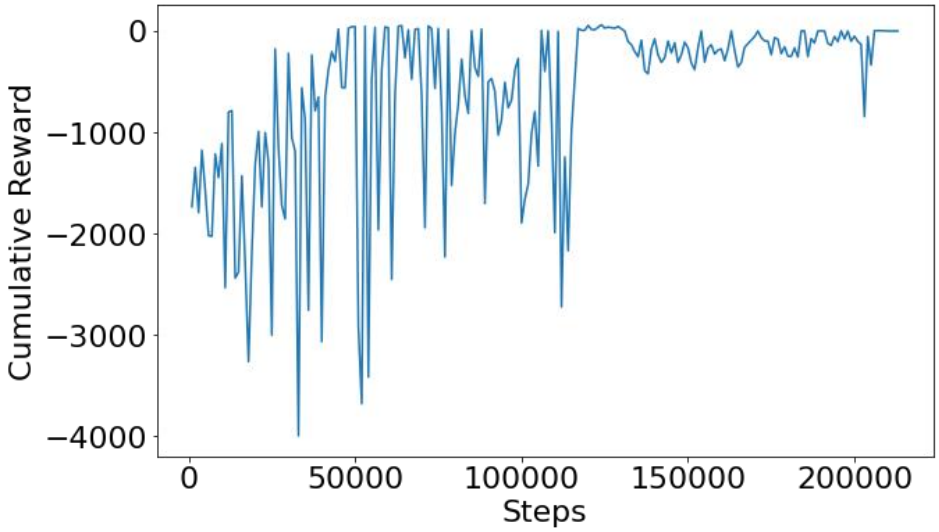

Figure 2.4. Cumulative reward for the shared-control navigation policy.

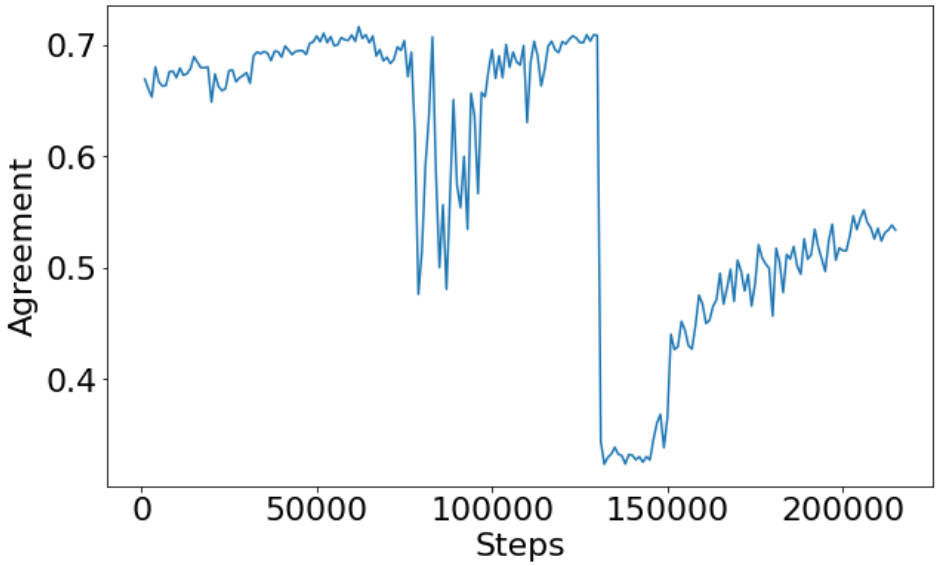

Figure 2.5. Average user-wheelchair agreeability.

2.3.2 Metrics

We first evaluated the performance of our proposed model and made comparisons with some baseline models using three basic metrics.

- 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):

$$t_c = t_{end} - t_{start}$$

- A: Agreement. We define agreement in terms of the deviation of the user's heading command from the final shared control's heading command. Mathematically, it is calculated as:

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

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

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

where v and w are the translational and rotational velocities u~$[v\ w\ ]$, 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.

We further used the evaluation metrics proposed in D1.3, specifically Path efficiency, Effect on crowd, Crowd-robot interaction, Collision, and Shared control quality.

2.3.3 Results

We tested the performance of our approach (RLPSC) with one user giving input through a joystick, in a simulated circular crowd scenario (See Figure 2.1) with local crowd density $\geq 1p/m^2$. It was compared with our previous work which uses a velocity-based probabilistic shared control (see D3.6). In addition, no assistance mode was used as a baseline.

The wheelchair started at a random position which is consistent in trials with different methods. The user drove the wheelchair to the goal and completed one trial. Each method is tested for 5 trials and Figure 2.6 to Figure 2.8 give a summary of the evaluation results.

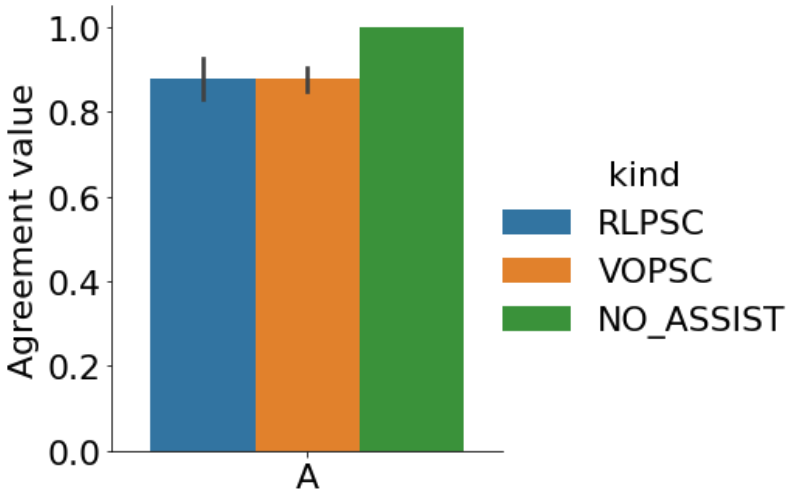

Figure 2.6. User-robot agreement for the tested three methods.

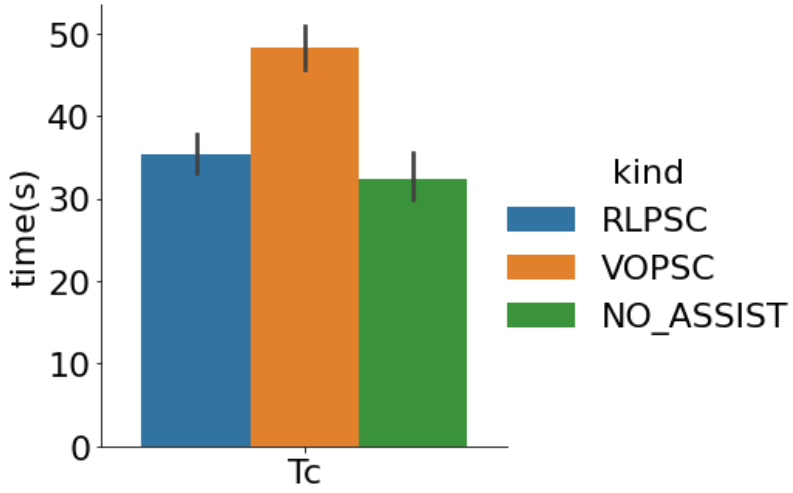

Figure 2.7: Time to complete for the tested three methods

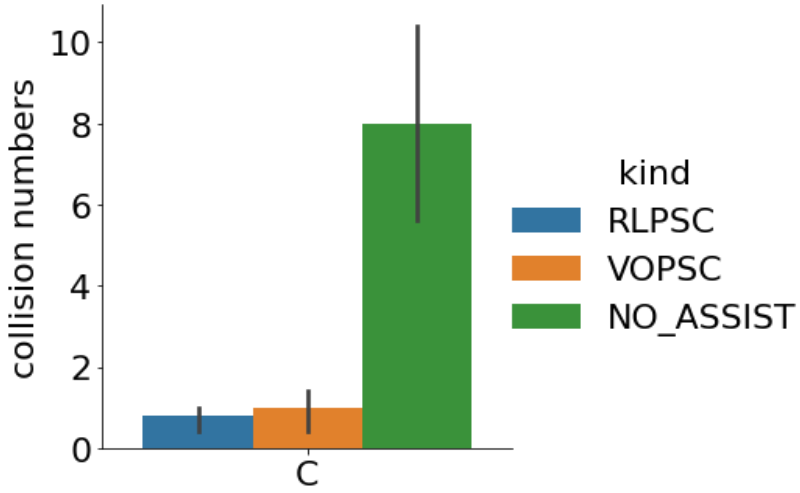

Figure 2.8: Number of collisions for the tested three methods.

It can be seen that both RLPSC and VOPSC reduced the number of collisions with pedestrians substantially compared to the one without assistance. While the learning-based methods achieve collision avoidance by setting negative rewards, they do not guarantee collision-free behaviour, especially in challenging scenarios. As safety is essential to the wheelchair user, a potential solution could be adding Model Predictive Control (MPC) for redundancy.

On the other hand, while velocity-based PSC took longer to reach the goal, potentially due to "freezing" when the crowd density increases, the RL-based method had a similar time to completion as when no assistance was given. This implies that the proposed method is promising in learning crowd interactions, and moving the wheelchair in a safe and social way. In terms of the agreement, both RLPSC and VOPSC had a similar value at about 0.88, which shows good control sharing performance in general.

We further evaluated our method in terms of T1(the robot's relative time to goal), L1(the robot's relative path length), J1(the robot's relative jerk), T2(the robot's effect on time),V2(the robot's effect on velocity), P4(proximity), C5(number of collisions), A6(Agreement), H6(User input entropy) and F6(Fluency of command). Note H6 is evaluated as,

$$\theta(u_h) = tan^{-1}(\frac{v_h}{w_h})$$

$$H = -\sum_{i=0}^{14} p(\theta_i) log(p(\theta_i))$$

where we discretized the angles into 15 sections and calculated $p(\theta_i)$. Ideally, a better shared control algorithm would have smaller user input entropy.

Figure 2.9 shows the summarized result.

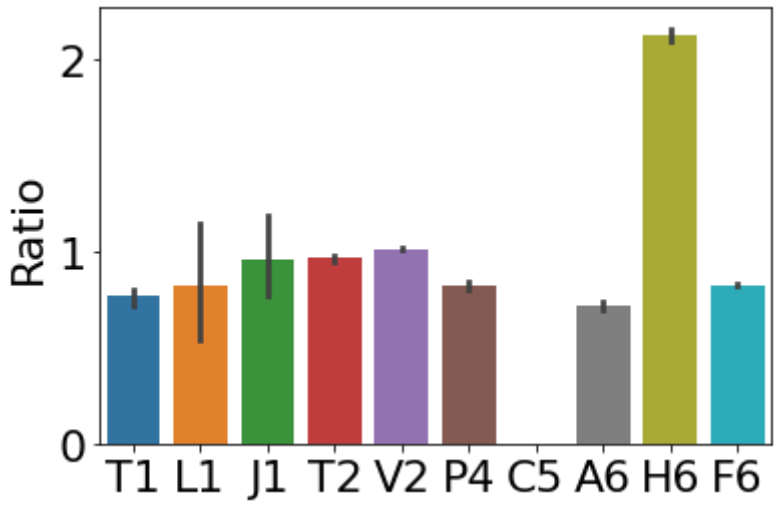

Figure 2.9. Results for the 10-evaluation metrics. Our proposed RLPSC allows the robot to reach the goal in slightly longer time by taking a slightly further path, while obeying the user's command more than half of the time. The effect on crowds is subtle and there were no collisions.

2.4. Field Test

It remains challenging at this stage to directly port RLPSC onto the physical wheelchair, mostly due to differences between the actual sensors and the characteristics of the sensor models in the simulator. This means the learned behaviours in the CrowdBot simulator will need further adjustment for deployment in the physical world. Nevertheless, we further evaluated our previously proposed method (PSC-DWAGVO) on the physical wheelchair in a number of real-world scenarios. First, we began in a controlled indoor environment, before demonstrating the wheelchair in uncontrolled indoor and outdoor environments with naturally occurring crowds.

2.4.1. Scenario 1: Controlled 1D Sparse Crowd

2.4.1.1. Implementation

Scenario 1 was conducted in IOMS Student Centre on UCL's Stanmore Campus. We used a valid interaction area of 5m x 5m, where the sparse crowds were constructed using 4 students at a time, from a total pool of 10 participants that were constantly rotated. The students were instructed to walk from one side to the other side of the room (1D) altogether in a natural way. An optitrack motion capture system was used to track pedestrians for ground-truth data, and the trajectories were streamed to ROS bag. A photograph of one of the experimental runs is shown in Figure 2.10, where you can see the smart wheelchair in the centre and each participant is wearing a baseball cap with the optitrack markers for tracking. During the experiments, we collected data with "wheelchair and crowd" and "crowd" alone, which also enabled us to measure the effect of the shared controlled wheelchair on the crowd dynamics. Each experiment was repeated 5 times.

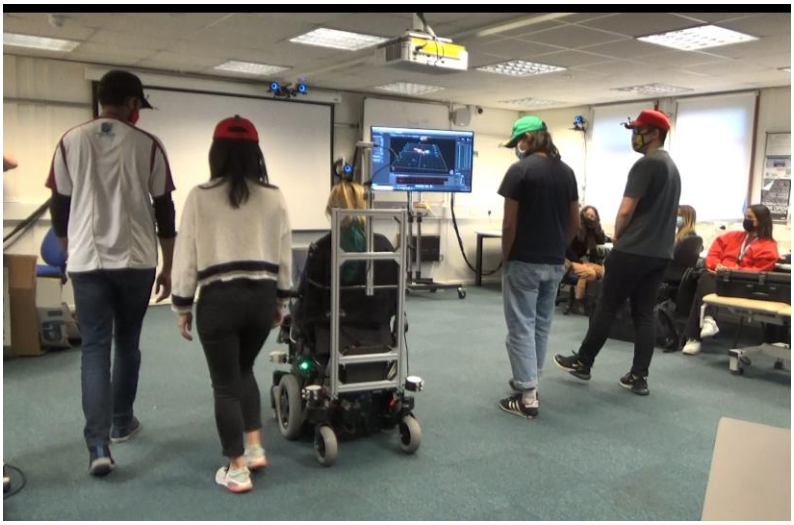

Figure 2.10. The wheelchair driving amongst a controlled 1D Sparse Crowd.

2.4.1.2. Results

We evaluated our proposed method in terms of "effect on crowds", "crowd-robot interaction" and "shared control performance". The results are summarized in Figure 2.11. In general, driven by an experienced user, our proposed PSC-DWAGVO method allows the robot to navigate in sparse 1D crowds without collisions, while obeying the user's command most of the time. Such a wheelchair was observed to increase the pedestrians' time to goal, whilst slightly decreasing their walking speed.

In addition, when compared with the 1D crowd simulation result, we observed similarities in terms of crowd-robot interaction and shared control, while some differences exist in the effect on crowds. The field test shows more effect on pedestrian's time to goal.

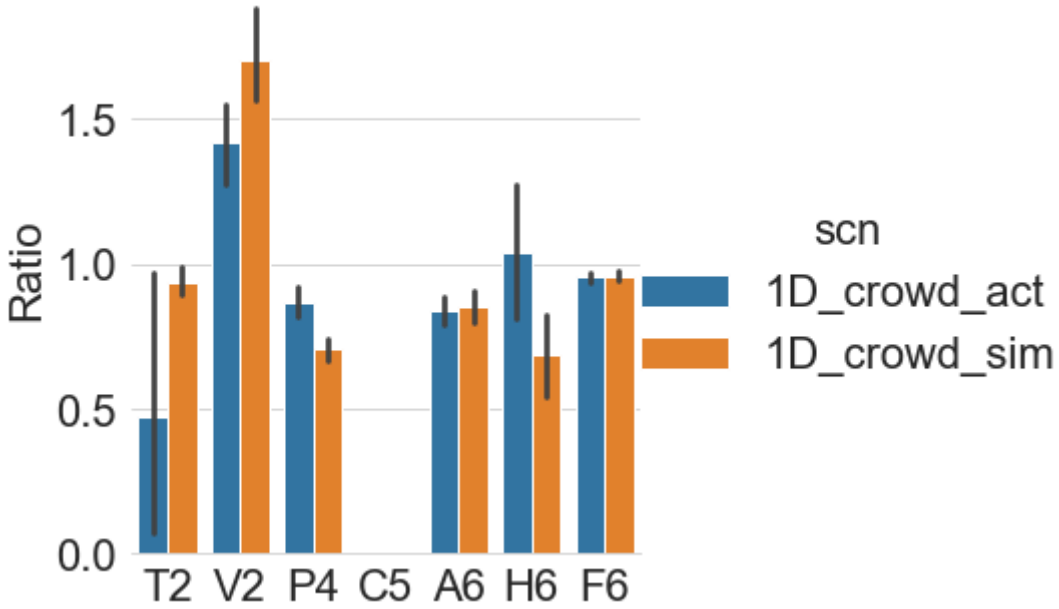

Figure 2.11. Results for the controlled 1D sparse crowd scenario (in blue) and its comparison with the corresponding simulation result reported in D3.6.

2.4.2. Scenario 2: Demonstration in indoor/outdoor environment

2.4.2.1. Implementation

Finally, we demonstrated the smart wheelchair in a natural 1D/2D sparse crowd scenario on the UCL Bloomsbury campus. Figure 2.12 shows the route we took, which consists of an outdoor environment (the entrance) and an indoor environment (the cloisters). The same route was used for data collection as reported in D3.6. The data was collected from 13:00-16:00, but due to Covid restrictions the passageways were less busy than we would expect in a normal year.

Some example stills from the data we collected are shown in Figure 2.13, where you can see an external view of the wheelchair and environment in the left image and the corresponding sensor data and user input in the right image.

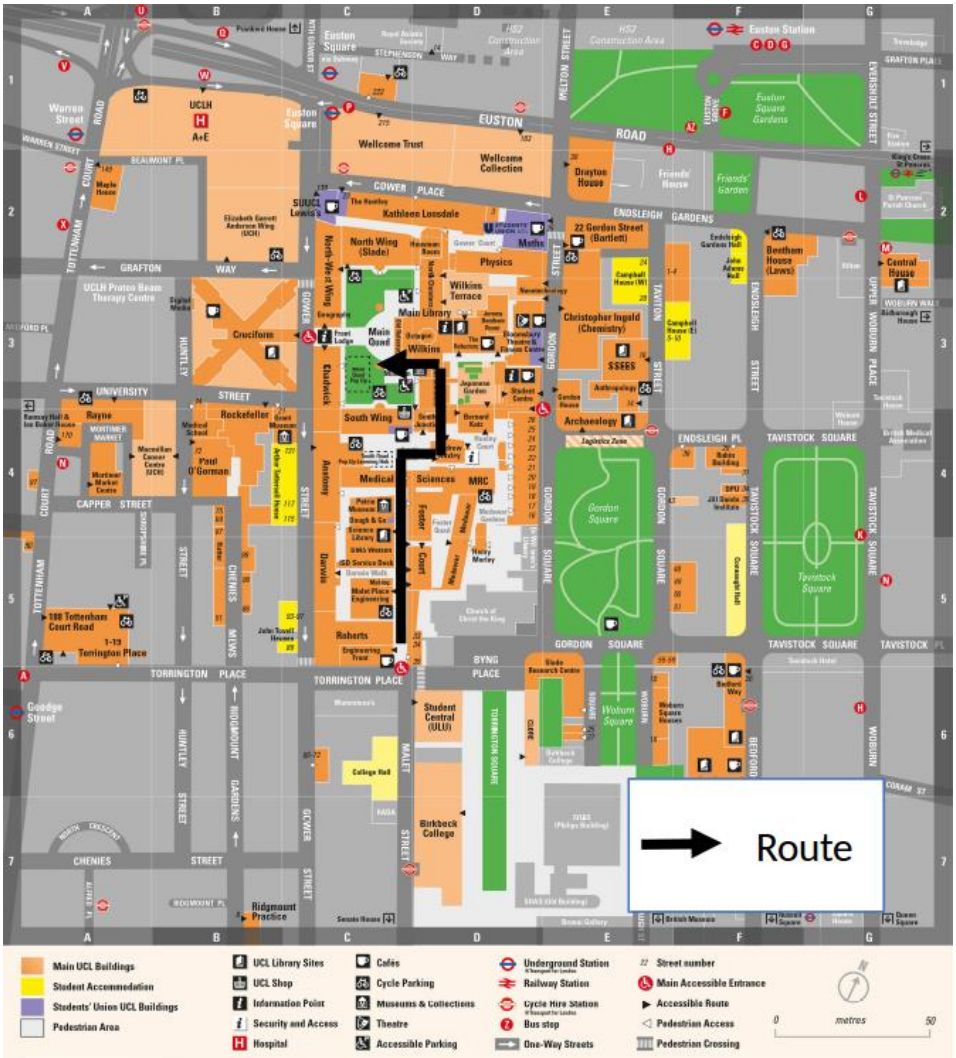

Figure 2.12. A map of UCL Bloomsbury campus, showing the route traversed during our data collection trials. This route is the same as route 3 reported in D3.6.

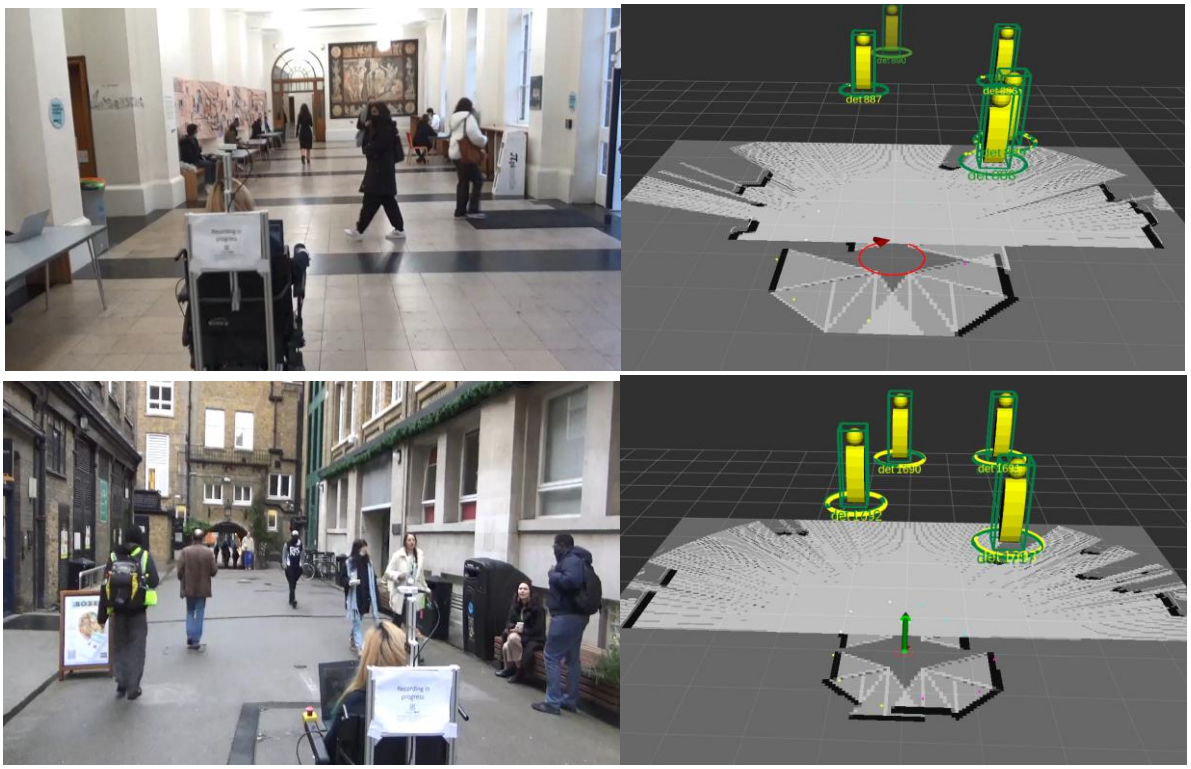

Figure 2.13. Natural 1D/2D sparse Crowd (Indoor and Outdoor) and the corresponding data visualization in Rviz. Green arrow shows the linear velocity while the red circle stands for the angular velocity. The driver intends to move forward while the final command makes a small turn to avoid the front pedestrians.

2.4.2.2. Results

Figure 2.14 and 2.15 show the user intended and final chosen linear and angular velocity. It can be observed that high agreement is maintained most of the time while slight deviation (more in angular velocity) happens when there is a risk of collision. Figure 2.16 illustrated the performance evaluation in terms of crowd-robot interaction and shared control.

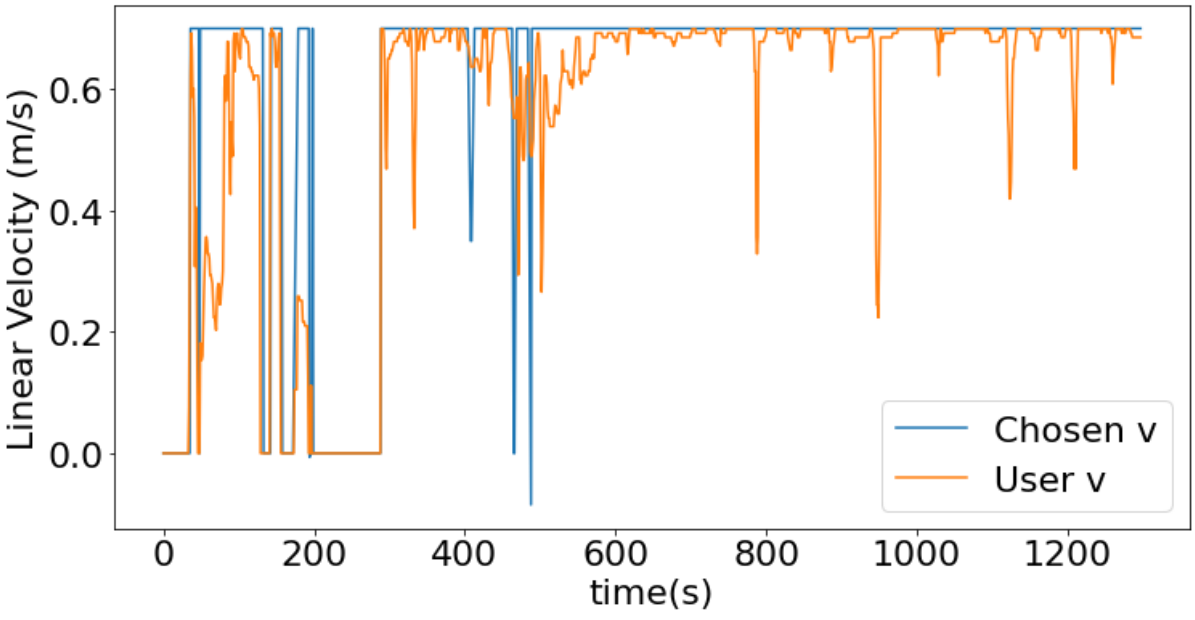

Figure 2.14. User's input and the final linear chosen velocity, selected by the PSC-DWAGVO algorithm.

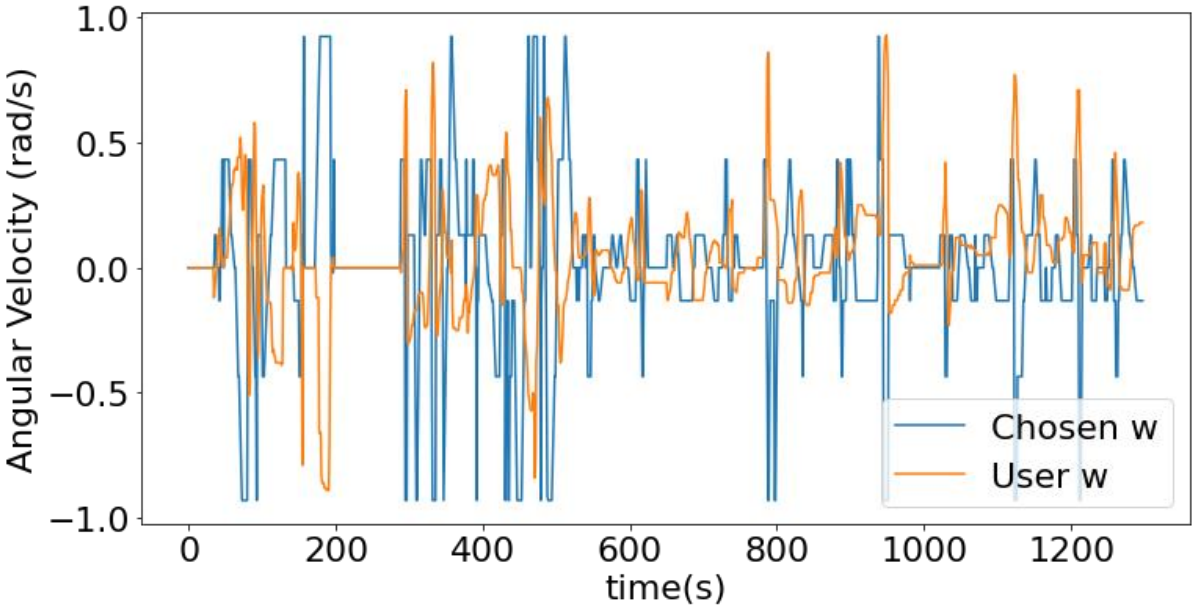

Figure 2.15. User's input and the final chosen angular velocity, selected by the PSC-DWAGVO algorithm.

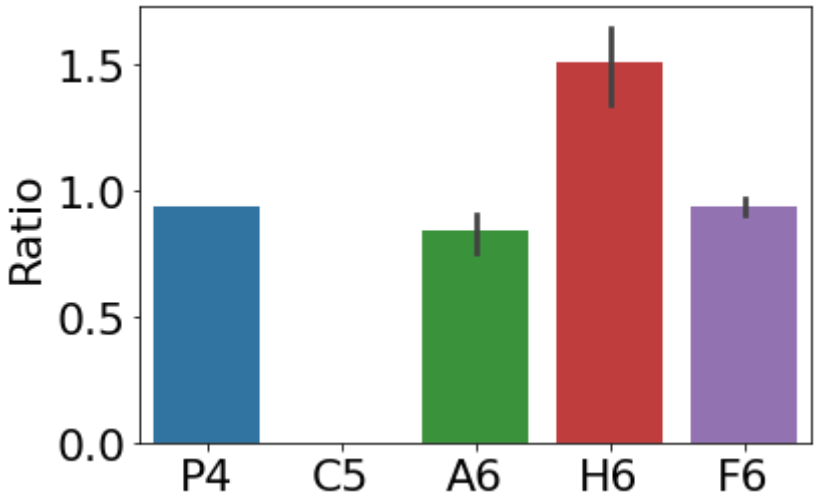

Figure 2.16. Results for the natural 1D/2D sparse crowd scenario. Driven by an experienced user, our proposed PSC-DWAGVO method allows the robot to navigate in natural 1D/2D crowds without collision, while closely obeying the user's command most of the time.

2.5. Conclusion

In this report, we first proposed a reinforcement learning based shared control approach for wheelchair navigation in crowds. To address the challenge of involving humans in the training loop, a surrogate user was trained by GAIL that learns the driving style from the collected user data. The final shared-control policy combined the predicted user input, the wheelchair state as well as agent-level crowd information, and was trained to provide suitable driving assistance. The performance of our proposed RL method has been evaluated in a simulated circular crowds scenario and showed promising navigation performance while obtaining relatively high user agreement when compared with other approaches.

In the future, we would like to evaluate the model performance and its usability in various challenging crowds scenarios such as 2D crossing. In addition, we are also interested in exploring generalized sim-to-real transfer in robot learning.

In addition, we evaluated our previously proposed PSC-DWAGVO (see D3.6) in a controlled 1D sparse crowd environment and demonstrated its feasibility in a natural 1D/2D sparse crowd in our campus. In the future, we would like to further validate it in a denser crowd environment.

3. Qolo

In this section, we report on the tests performed with Qolo in natural crowds in Lausanne, Switzerland. We begin by describing two different formulations of reactive navigation. We then describe 3 different scenarios, in which we perform over 5 hours of navigation, traversing more than 12 km in moderately crowded market environments.

3.1. Research Question

We have seen that traditional navigation methods could lead to a "freezing robot" issue (Trautman et al., 2015) when the crowd density increases. To address this problem, we have formulated 2 different methods of reactive collision avoidance applied to crowd navigation for different robot types (as detailed in D3.4).

The first, modulated dynamical systems (MDS) method focuses on holonomic robots with round shape representation. The second, reactive driving support (RDS) based on velocity obstacles optimization for non-holonomic and non-circular robots. Both methods focus on immediate reaction to the unexpected obstacle's motion and slide smoothly along the obstacle when possible, in place of freezing. Nonetheless, contact might be unavoidable when the robot's kinematic and dynamic constraints are below pedestrian's. Therefore, a post-collision reactive control has been developed to further extend the safe navigation capabilities of the robot even under contact. The active compliance combined with MDS provides a way to slide around obstacles while in contact and continue moving towards the goal. Sliding in contact is done in such a way to not exceed force thresholds determined as safe.

In D1.4, we presented evaluations of the methods: testing the MDS (Huber, Billard & Slotine. 2019) on a holonomic platform and in simulation. RDS (Gonon, Paez-Granados & Billard. 2021) was tested on a non-holonomic robot Qolo (Paez-Granados, et al. 2022) in collaboration with Prof. Kenji Suzuki (University of Tsukuba, Japan) in laboratory settings and on sparse crowds around the corridors of the EPFL's campus.

In the current experimental evaluation, we explored each method of reactive navigation in raw crowds in the city of Lausanne, focused on:

- How do the proposed reactive navigation methods perform when used in sparse and flow crowds?
- How does the whole sensing, detection, tracking, and control perform as a reactive collision avoidance system?

3.2. Reactive Navigation Control

3.2.1. Problem Formulation

The main objective of the reactive control module is to provide a layer for local navigation assistance and immediate responsiveness to unmodeled or unpredicted object occurrences or motions. Therefore, this algorithm assumes a continuous dynamical system guiding the robot's motion to be existing and given by a high-level algorithm that plans for optimality towards its intended goal.

The Modulated Dynamical System (MDS), represents obstacles analytically as star-shaped level sets of a distance function that absorb the robot's footprint, thus, allowing the robot to be represented as a point moving in cartesian space. In the case of Qolo, we control a single point in the bumper area as a holonomic point as documented in D3.4. It guarantees to lead the robot to its goal (hereafter called attractor) by assuming a virtual boundary as a circle (Huber, Billard & Slotine. 2019).

The method termed Reactive Driving Support (RDS) replicates the behaviour of the MDS locally. While it sacrifices the guarantee to reach the global goal (by itself, i.e. without an additional path planner), it allows representing more accurately the robot shape detail (see, Fig. 3.1) and non-holonomic constraints of robot kinematics (Gonon, Paez-Granados & Billard. 2021).

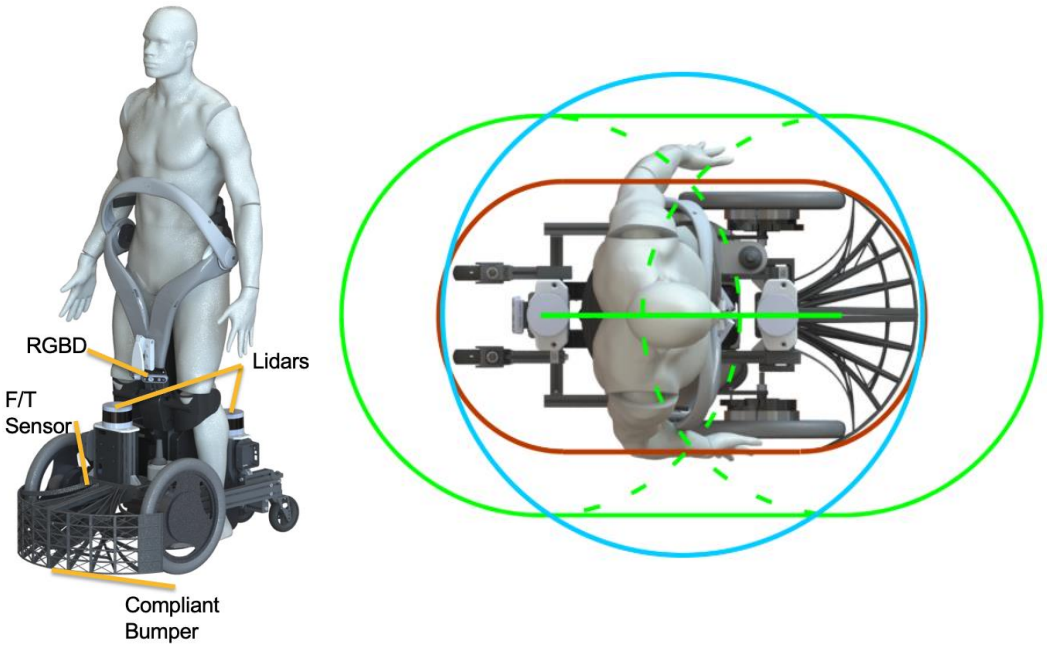

Figure 3.1. Left side: Qolo robot with implemented sensors. **Right side:** In red the actual robot area, in blue the circle representation used for MDS obstacle avoidance, and in green the virtual representation used for RDS obstacle avoidance.

Guaranteeing obstacle avoidance during navigation in highly occupied areas would be unattainable for current mobile service robots bounded by energy storage capacity, computational resources, and actuation hardware expected to behave as pedestrians, i.e. holonomic, reactive, communicative, and knowledgeable of proxemics and other social rules.

To mitigate risks and allow the robot to navigate in post-contact scenarios, we designed a controller that combines online reactivity to obstacle avoidance as provided by the MDS system with compliant control using a passive dynamical system approach offered in (Kronander and Billard. 2016). Unlike the admittance controller presented in D3.4, the current approach uses a dynamical system framework for providing a sliding behaviour for overcoming initial collision and allowing the robot to advance in crowd scenarios (Paez-Granados, Gupta, Billard. 2021).

We further generate a compliant response by extending the explicit force control framework in (Amanhoud, Khoramshahi, Billard, 2019) to post-collision, where, the location of the contact surface is unknown a priori, therefore, we make use of a closed-loop force control for achieving a sliding control over the surface of the robot while the underlying dynamical system continues to be modulated by the obstacle avoidance.

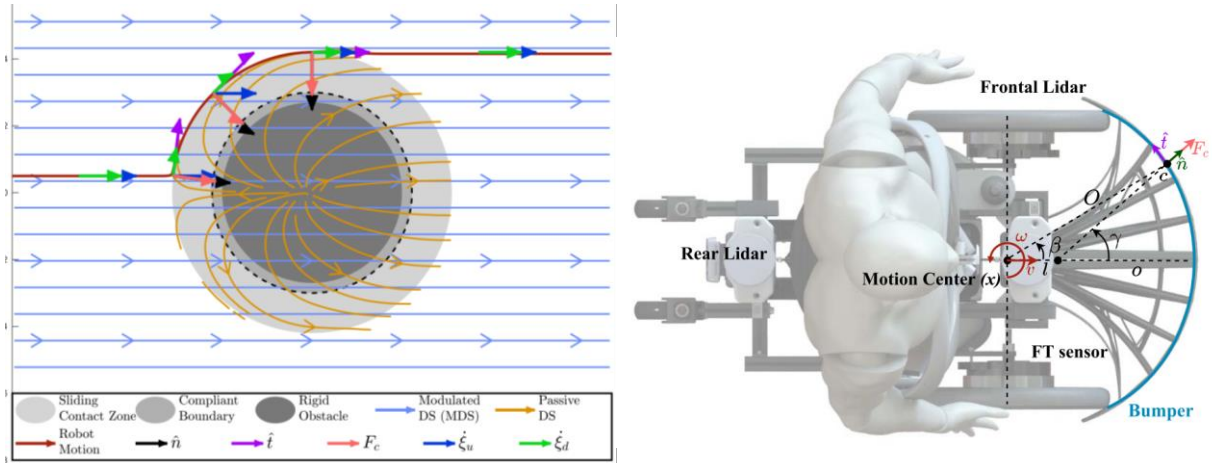

Figure 3.2. Sliding DS formulation for limiting contact forces while moving along an underlying desired motion. **Left side: the robot is represented as a point-mass,** when the robot enters in contact with the obstacle (light-grey zone) the desired motion is controlled by the reaction force at the boundary guaranteeing a limited contact force Fn to the obstacle and allowing a sliding motion ξd around it. **Right side: implementation on the Qolo robot's bumper.**

We design a nominal trajectory controlling for the robot's speed and generated through a function $f(\xi) \in R^2$. We then control for the torques using a damped tracking controller over f(ξ), effectively controlling the robot during contact as depicted in Figure 3.2,

$$f(\xi) = f_u(\xi) + f_n(\xi)$$

where $f_u(\xi)$ represents the driving, force generated by the nominal DS (ζ_u') input tangential ($\hat{t}$) to the collision surface which can be transformed into the contact dynamics as:

$$f_u(\xi) = \hat{t}^T \frac{M\dot{\xi}_u}{T_s}\hat{t}$$

where T_s accounts for a discretizing time constant, and M represents the desired reflected inertia of the robot. Finally, $f_n(\xi)$ describes the force control function as:

$$f_n(\xi) = \frac{F_n + F_c}{\lambda_t}\hat{n}$$

where F_C represents the measured contact force, and F_n was chosen as the contact force limit bounded by safety and acceptability during unexpected collisions. e.g., during leg contacts we chose an $F_n = 45N$, well below the pain threshold at the lower legs of 130 N [ISO/TS 15066].

Finally, the controller can be expressed as:

$$\tau_c = \lambda_t f_u(\dot{\xi}) + (F_n + F_c)\hat{n} - D\dot{\xi}$$

where the damping effect on the matrix D was controlled by normal and tangential parameters over the surface of the obstacle, λ_n and λ_t, respectively.

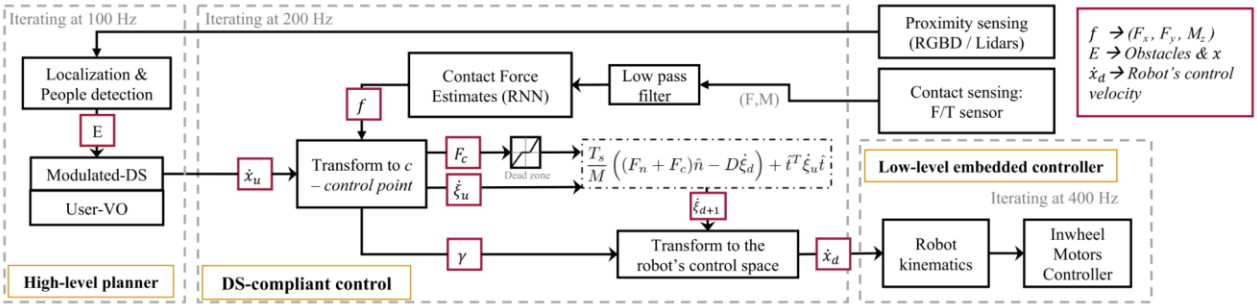

Figure 3.3. Controller architecture considering that the robot is driven by a high-level controller that takes its desired velocities. The closed-loop controller with contact sensing relies on an RNN for estimation of the contact forces and location of the contact over the bumper.

The control architecture results in a continuous controller that handles three states of the mobile robot, namely obstacle avoidance, contact, and post-collision control. as shown in Figure 3.3.

As well, the controller is composed by three parts: first, a high-level closed-loop motion planner drives the dynamics of the robot in its control space ($\zeta_u{}'$), which includes localization, obstacle detection and tracking. We used a modulated DS (Huber et al. 209) for obstacle avoidance, and the velocity-based planners in RDS (Gonon et al. 2021) for controlling this input.

Second, the compliance and contact control through sliding method using a known sensing surface over the robot's hull with a limited contact force Fn.

Third, a low-level controller that handles the execution to actuators in real-time closed-loop control.

Further details on the controller and evaluation can be found in [Paez-Granados, Gupta, & Billard, 2022].

3.3. Experimental Setup

The experimental setup reported here follows the protocol established on D6.3 "Proceedings of ESAB Workshops & Report on Ethical Protocols", for EPFL.

The experiments were carried out in the city of Lausanne with the approval and cooperation of the office for mobility, the police and the office for parks, and public domain of the city of Lausanne. Approval numbers: 395128 and 416008

At the designed areas in the city of Lausanne we ran experiments in 2 sets of scenarios from the requirements on D1.3 "Specification of Scenarios Requirements Update", following the protocol set in point 1.1.1 of travelling 50m goal at different times of the day for achieving different crowd densities:

3.3.1. Scenario 1: Sparse Crowds

Place: Place de l'Europe & Rue de Saint Laurent (see, Fig. 3.4)
Preliminary recordings during non-rush hours of the weekdays (mid-afternoon 14:00-17:00).

- Point-to-point recordings of 50m trips x 10 times.

Recordings during non-rush hours of the weekdays (mid-day 11:00-13:00).

- Point-to-point recordings of 50m trips x 10 times.

The goal of this scenario was to assess the feasibility of the controller to navigate in increasingly dense crowds (see, Fig. 3.5).

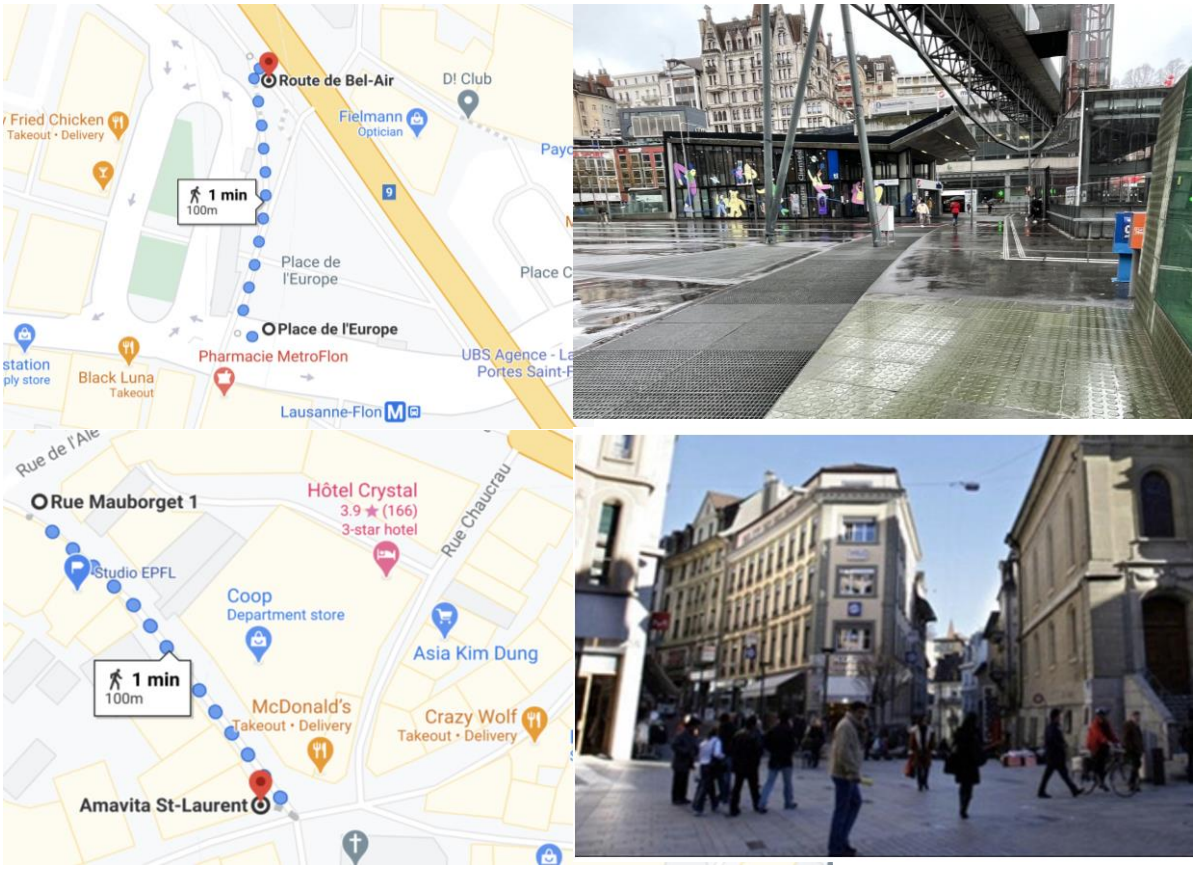

Figure 3.4. Top: Place de l'Europe. Bottom: Rue de Saint Laurent.

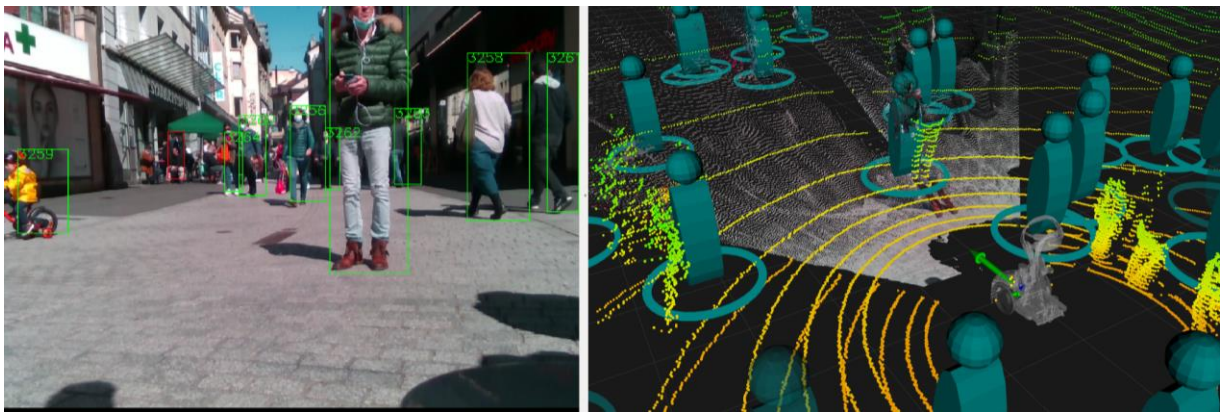

Figure 3.5. Left side: Onboard view from the Qolo robot's camera with pedestrian bounding boxes. Right side: 3D capture of the onboard data from the robot, including 3D lidar point cloud, RGBD point cloud, pedestrian detections, and the robot's velocity control state (green arrow).

3.3.2. Scenario 2: Point-to-point navigation in flow crowd

In this scenario, we target assessing how compatible is the reactive navigation with the flow of pedestrians. We compared the different definitions of the robot's reactivity in MDS, and RDS through executing the task of avoiding obstacles while reaching a target 20 m head.

In this scenario, we considered a **1D flow navigation** at the street of "**Rue de Saint-Laurent**" (see, Fig.3.6). The experiment was repeated on several dates in order to get similar crowds at the market of the city of Lausanne. We expected to find several pedestrians moving in between the 6 streets at the intersection (see, Fig. 3.7), thus, we set the direction of the robot to follow one of the street's flow.

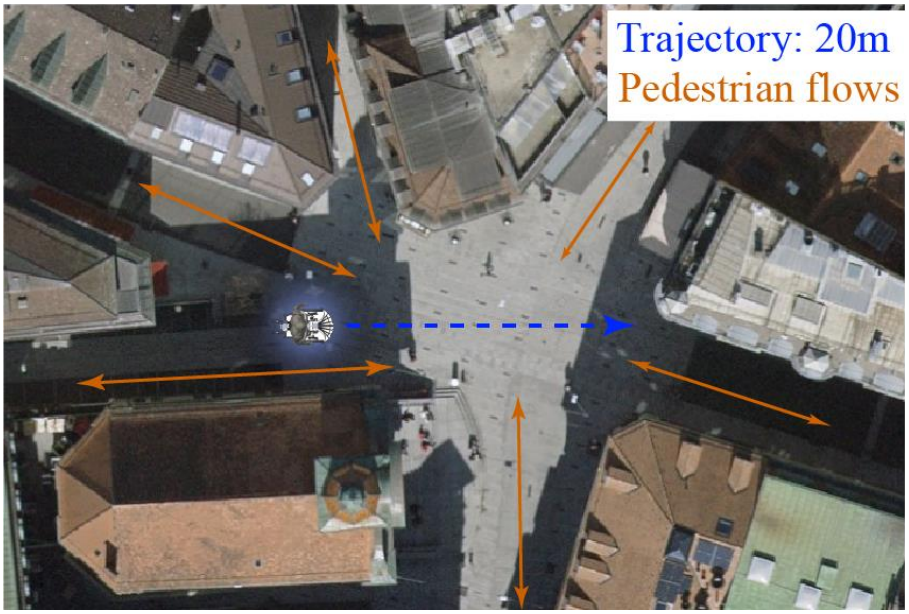

Figure 3.6. Top view of the selected scenario for multiple evaluations in crowd flows.

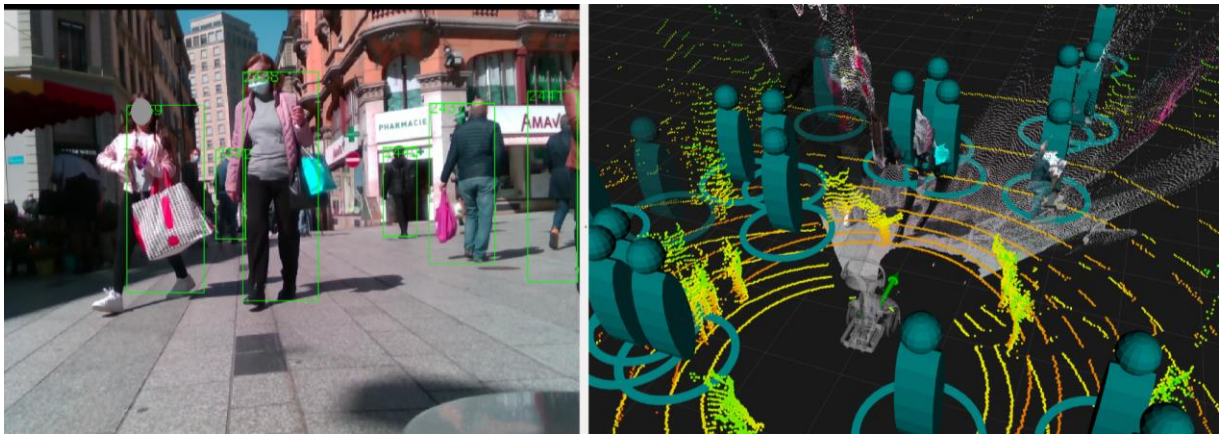

Figure 3.7. Qolo's sensor view of the scenario during experiments.

3.3.3. Scenario 3: Navigation in dense mixed crowd

In this scenario, we considered a **1D flow navigation** at the street "**Place de l'Europe**". The recordings were made during the Christmas market of the city of Lausanne (see, Fig. 3.8), therefore, we expected to find denser mixed crowds, formed by people lining up, and flows of pedestrians (see, Fig. 3.9). We followed the same protocol of traversing a set path with a start and end goal location of 30 m apart in this case.

Figure 3.8. Recordings of the navigation scenario in a highly crowded area.

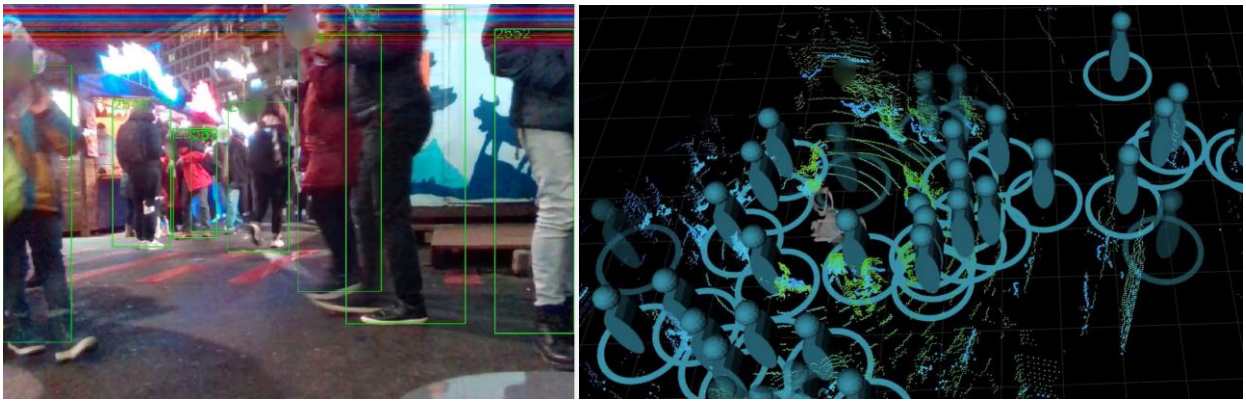

Figure 3.9. View from Qolo's sensors in the dense crowd navigation.

3.3.4. **Qolo's Implementation details**

All testing was implemented on the robot Qolo as detailed in D5.4 “System integration”. Two embedded computers Upboard Squared Intel® Core™ CPU @ 2.20GHz × 8, and one Nvidia Jetson Xavier AGX (8-core ARM v8.2 64-bit CPU), 512-core Volta GPU.

3.3.5. **Data Recordings:**

We collected over 5 hours of navigation data (recorded at 20 Hz), and over 12 km of travelled distance in moderately crowded market environments, encountering sparse crowds and mixed flows. Based on the multiple testing at the selected location, we finally chose a path of 20 m over the intersection of the streets where sufficient influx of pedestrians allowed to perform a test in moderately crowded environments (values ranging from 0.1 to 0.5 ppsm).

Initial round of recordings over scenario 1 reached over 3 km of recordings, in sets of 50 m round trips, with 13 recordings of autonomous driving using RDS for reactive control and 20 recordings of shared control. Subsequent testing in scenario 2, successfully recorded 16 MDS trials, 30 RDS trials, and 45 shared-control trials, each test including 2 minutes of data with 20 m round trips, for a total of 3.6 km.

Measurements to be taken during each trial encompass:

1. Pedestrian's motion information in the form of a set of 2 point-clouds around the robot, including all surrounding people and obstacles in a range of up to 50 m.
2. Pedestrian's motion data from a forward looking RGBD camera, with people labelled and blurred.
3. Force/Torque information gathered by the contact sensors at the robot’s fences.
4. Recordings of the navigation interface input given by the user/driver of the robot.
5. Motion data gathered from the robot inertia sensors and velocity sensors.
6. Video recording of the scene from the robot’s perspective without personal identification recording, all faces will be blurred out of the images.
7. Video recording of the participant-robot driving in a scenario, with no personal identification, all faces will be blurred out.

Data format:

One recording dataset includes approximately 120s of recordings, in a rosbag format for Qolo's sensors that is about 11GB:

- Localization - Pose
- 2 x 3D point-cloud
- 1x RGBD camera
- 3 x people detectors
- 1x Tracker
- Qolo state and controller
- Contact Forces

The dataset of recording will be publicly available as: Paez-Granados D., Hen Y., Gonon D., Huber L., & Billard A., (2021), “Close-proximity pedestrians’ detection and tracking from 3D point cloud and RGBD data

in crowd navigation of a mobile service robot.", Dec. 15, 2021. IEEE Dataport, doi: https://dx.doi.org/10.21227/ak77-d722.

Scripts for extraction and post-processing of the crowd data have been released in the repository: https://github.com/epfl-lasa/crowdbot-evaluation-tools

3.4. Results

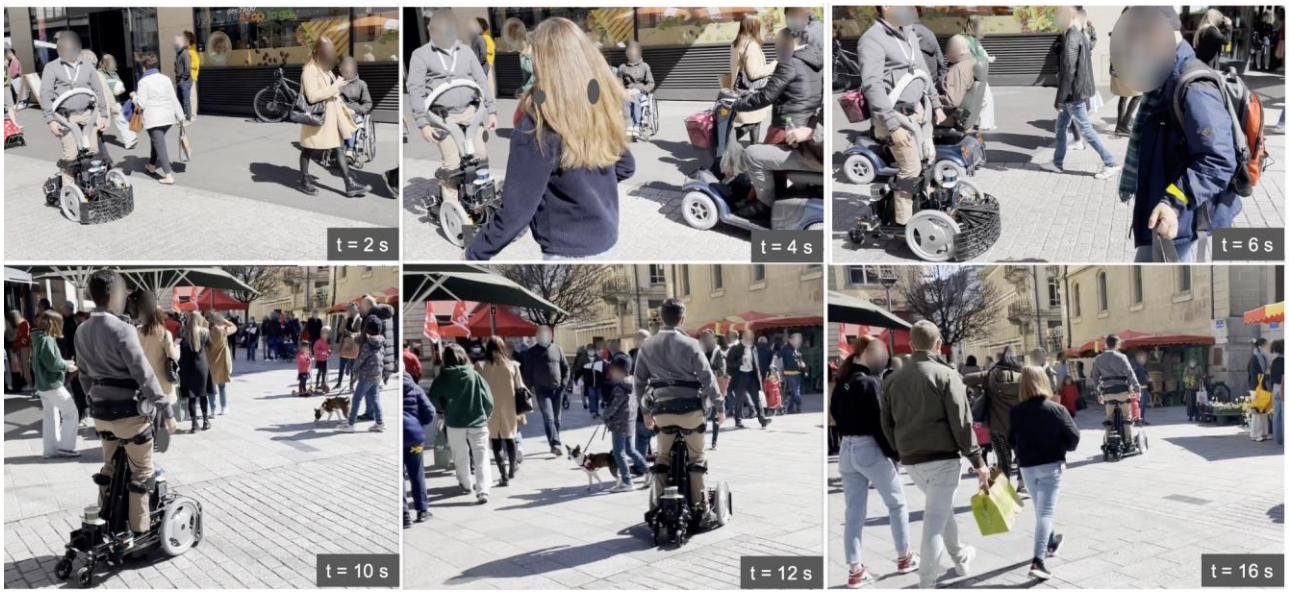

Figure 3.10. Time-lapse of crowd navigation experiments of Qolo in scenario 1.

The reactive navigation controller in full autonomy with a simple underlying dynamical system driving towards an attractor 50m ahead of the starting position showed to be compatible with the obstacle avoidance of pedestrians and other bystanders. i.e. the robot was successful in arriving at the desired destination (see, Fig. 3.10)

Notwithstanding, the simple method is not sufficient to overcome complex scenarios of mixed crowds where static pedestrians could block the path while lining up, or sitting. In comparison, using shared control where the input of general direction was given by the driver made the navigation more compatible with the complexity of the mixed traffic scenes observed.

We validated through scenario 1 the feasibility of driving the robot with reactive navigation around pedestrians, with sparse crowds ranging from 0.05 ppsm to 0.1 ppsm. The maximum crowd densities observed in these recordings were 0.4 ppsm.

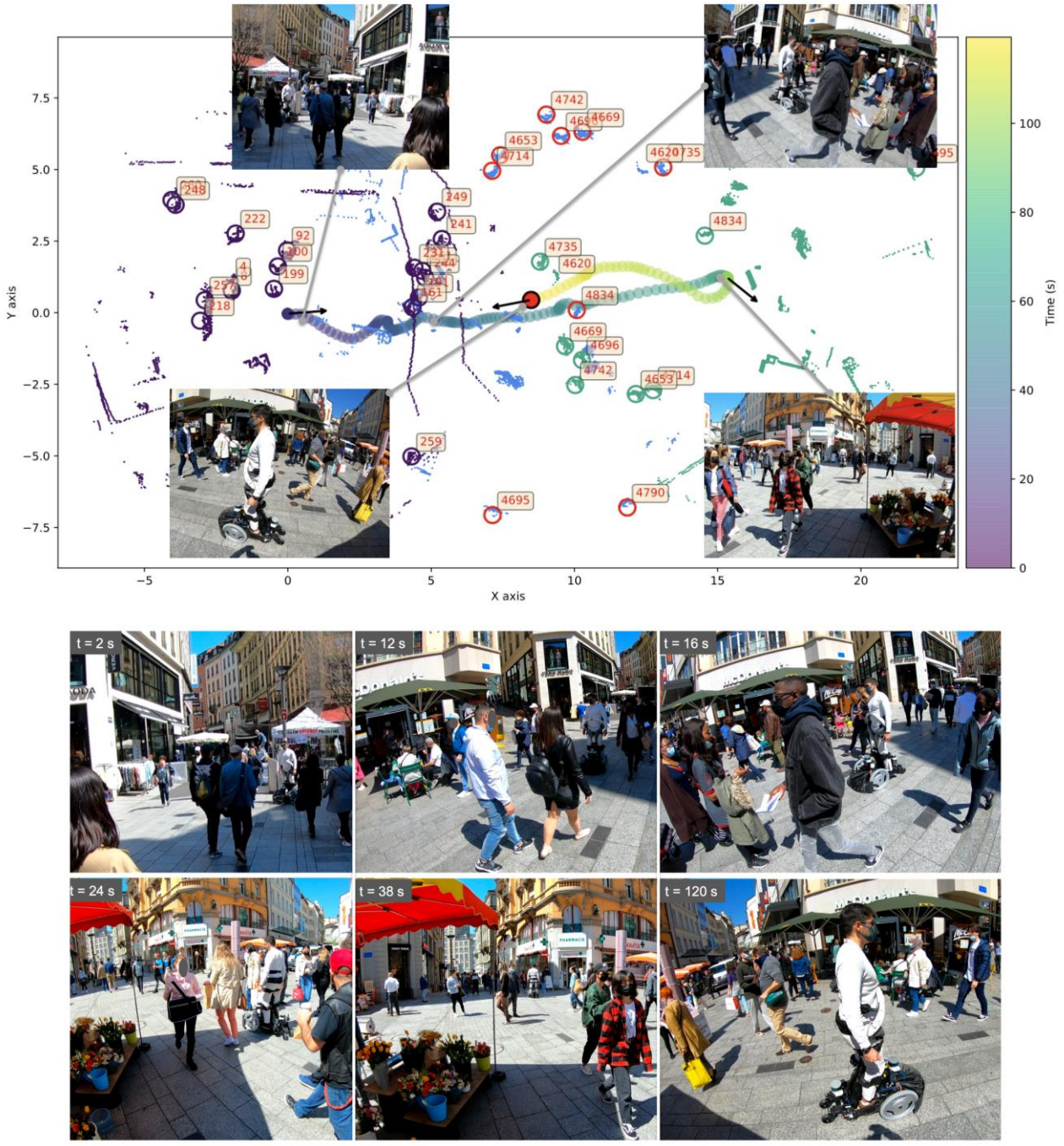

Figure 3.11. Trajectory example of a single execution of scenario 2.

Scenario 2 on crowd flows (see, Fig. 3.11) showed density of crowds with mean 0.15 ppsm and up to 0.7 ppsm max density. In the following section we detail the controller comparison on this scenario.

3.4.1. Crowd Metrics

Crowd Density:

Using the synchronized detections at a given timestamp within a radius r around the robot. We measured the mean and max pedestrian density for comparing the multiple methods in similar scenarios.

A drawback in the current metric estimation is the limited field of view of the robot as it is embedded in the crowd, therefore we report on the density within 2.5, 5 and 10m around the robot (see, Fig. 3.12), which gives a better understanding of the scenario.

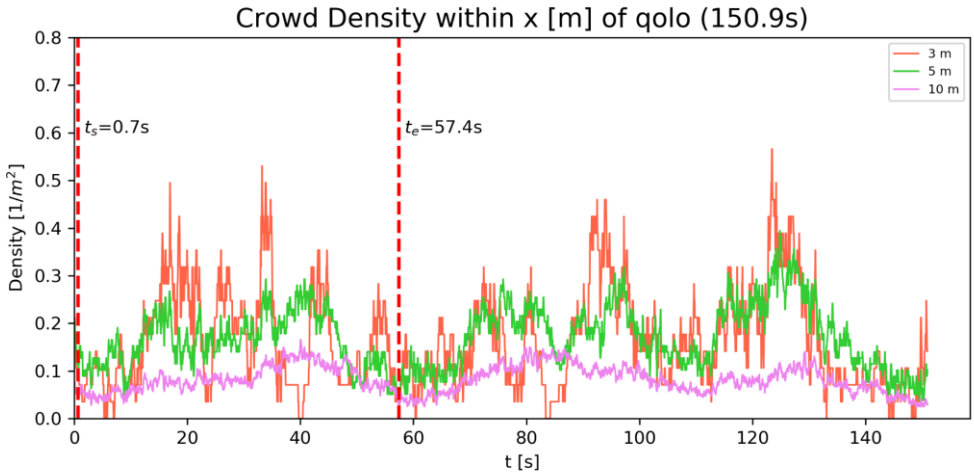

Figure 3.12. Crowd density example for 2.5, 5 and 10 m around the robot. The start and end time of the task for scenario 2 is marked as vertical dotted lines.

Proximity: Using the pedestrian detection output, we can find the mean proximity as:

$$Prox = 1 - 1/t_f \sum_{t=0}^{t_f} d_{min}(t)/R$$

where d_{min} is the distance to the closest agent, and R is the range in meters. Figure 3.13 displays an example of the closest proximity (Prox) for one data recording.

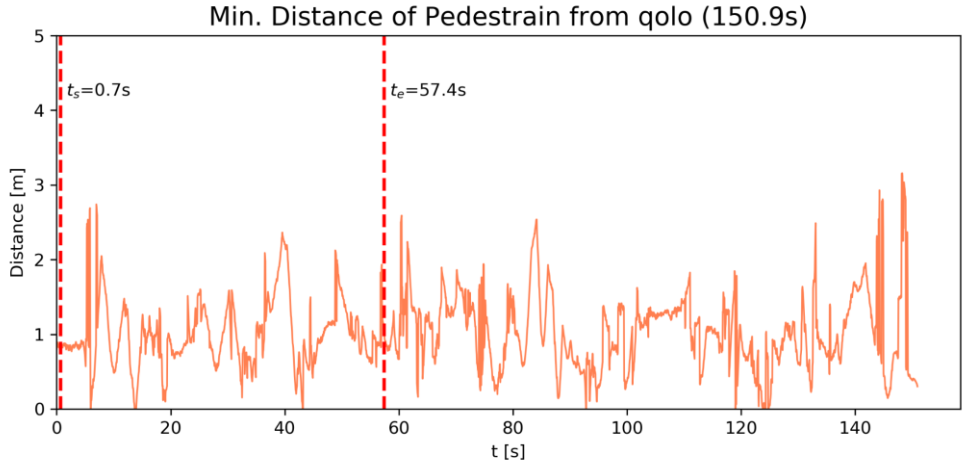

Figure 3.13. Distance of the closest pedestrian to the robot Qolo.

Comparison among controllers:

A total of 16 MDS recordings, 30 RDS recordings, and 45 shared control recordings were successful in achieving the navigation task.

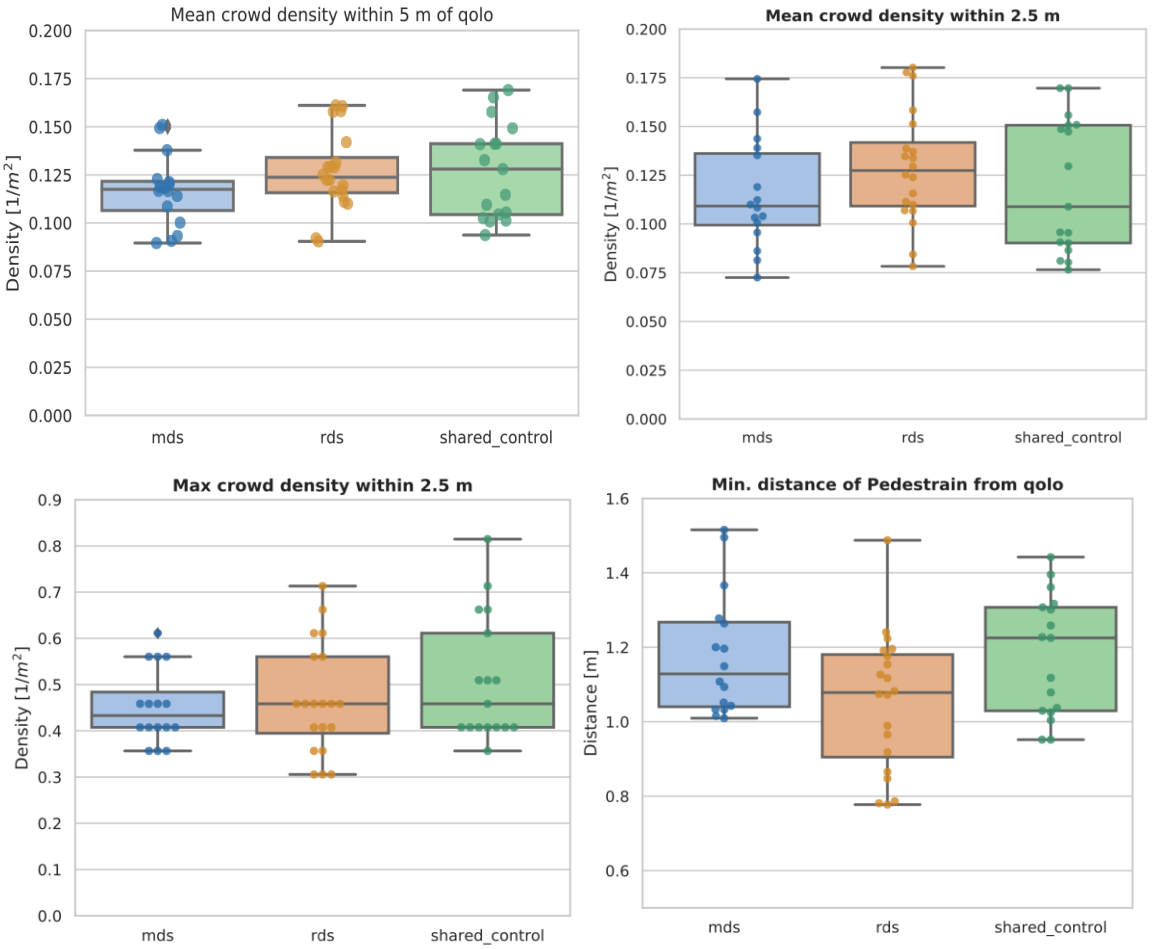

Figure 3.14. Crowd density measured from the lidar-based people detector in a radius of 2.5 m and 5 m around the robot.

The overall crowd density (as shown in Figure 3.14) measured from the onboard sensing fluctuates around 0.12 ppsm, with peaks of up to 0.4 ppsm. Although it is worth noting that the current measurements are limited to the proximity sensors, therefore, it could be possible that higher densities were not visible from the robot's view point.
The results of crowd density showed no significant difference among the set of trials. Therefore, we can conclude that the evaluations occurred in similar crowds in terms of density and flow around the robot.

The minimal distance to pedestrians during the trials showed to be lower in the case of the RDS controller, compared with MDS ($p < 0.01$), which is expected for the RDS controller having a tighter definition of the robot's shape.
Nonetheless, in shared control where the obstacle avoidance is provided by RDS, there was a significantly higher minimal distance to surrounding pedestrians. Which suggests that the user preference would be of a proxemic social distance higher than the values set on the controller.

3.4.2. Metrics of path efficiency

We used the evaluation metrics proposed in D1.3, specifically path efficiency, collisions, and shared control quality.

Relative time to goal: The relative time to goal compares the time taken by the robot to reach its goal when it is alone to the time it takes when it is in a crowd, thus a T_{rtg} of 1 would be a full free path.

$$T_{rtg} = t_{free} \,/\, t_c$$

Where the completion time is defined as: $t_c = t_{end} - t_{start}$
$T_{free} = L_{goal}/vel$, where *L_goal* represents the distance to the goal and *vel* is the set desired velocity.

For our experiments, we took the desired velocity of the robot and the linear distance to (L_goal = ||pose_final - pose_init||). for estimating the relative time to the goal.

Relative Path length: The relative path length compares the length of the path taken by the robot to reach its goal when a crowd is present or not. Mathematically it is given by:

$$L_r = L_{goal}/L_{crowd}$$

Figure 3.15 shows a path example with the deviations corresponding to obstacle avoidance, where the first path in red is the autonomous path by the controller and the return path (blue) is a shared control driven path.

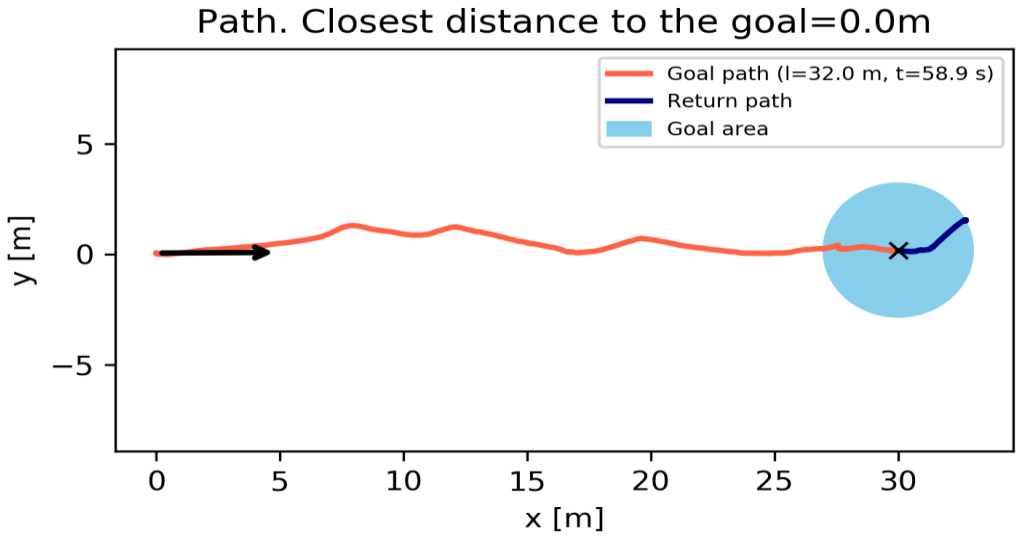

Figure 3.15. Example round trip trajectory during experiments.

Relative Jerk: The relative jerk evaluates the smoothness of the path taken by the robot to reach its goal when it is alone compared to when it is in a crowd. It is given below:

$$J = (1/t_f)\sum_{t=0}^{t_f} \Delta t\sqrt{(J_v{}^2 + J_\theta{}^2}$$

$$J_{rp} = J_{crowd}\,/J_{manual}$$

where J_{crowd} is the jerk of the autonomous controller test, and J_{manual} is a reference jerk of a manual driven test of the Qolo robot by the operator. Each computed as J, where J_v and J_θ represent the linear and angular jerk, respectively. Δt corresponds to the sample time window over the period $t = 0 \sim t_f$

Results comparison among controllers:

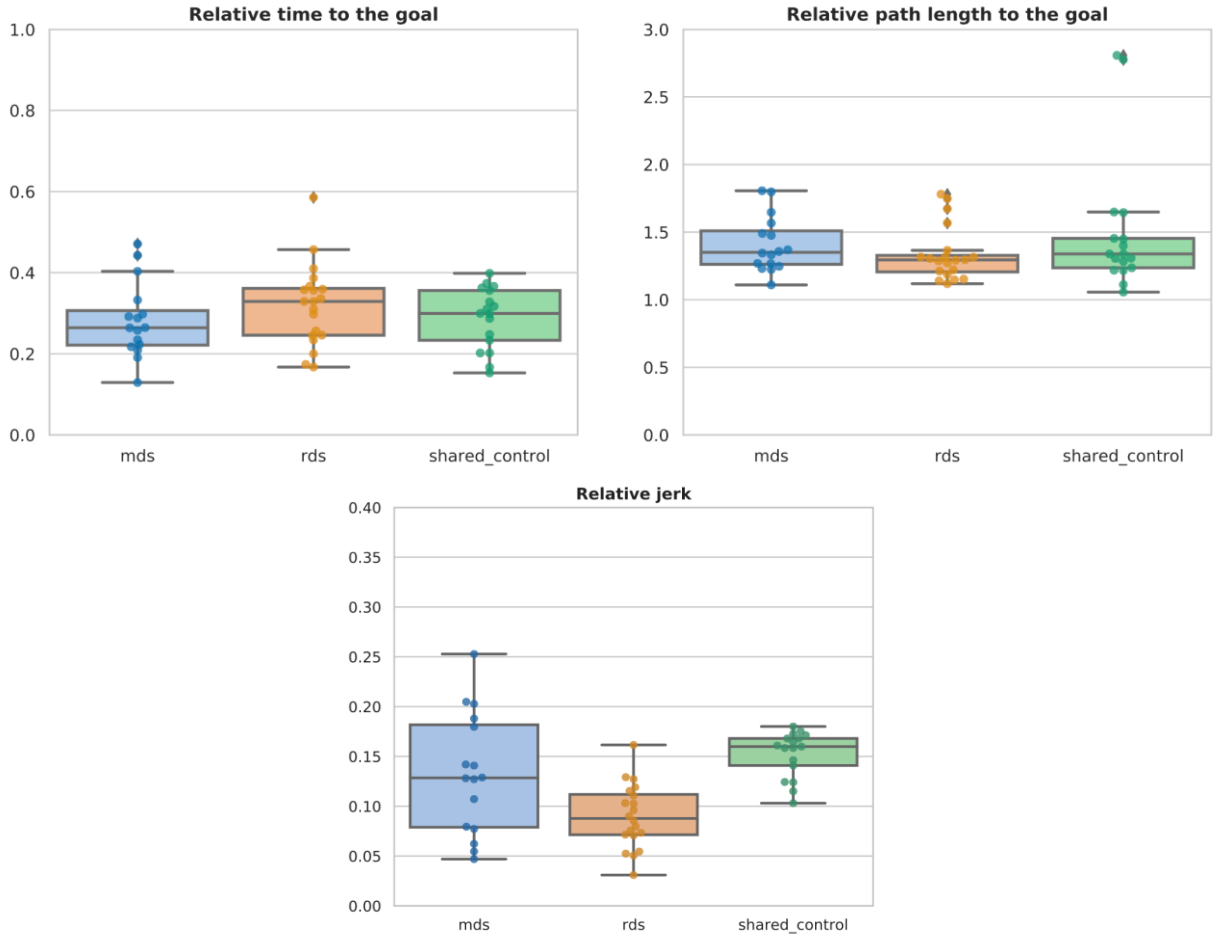

Figure 3.16. Relative time to goal and path length of the different controllers.

The path efficiency results show a relative time to the goal similar among controllers, with relative times from 30% up to 60% the efficiency of a free path (see, Fig. 3.16). No significant difference was observed among the methods. This suggests that the reactive navigation for obstacle avoidance does not hinder performance when compared with a shared control where the user can take lead over the direction of motion.

The path length was similar among all recordings of MDS (21.5 ± 5.9 m), RDS (20.2 ± 6.2), and shared control (27.1 ± 12.8), showing no significant difference (evaluated through a one-way ANOVA). Where all tests arriving within 3 meters of the target location were considered successful. This was set accounting for the drift in localization which could cause this level of error in the motion data.
The main difference in shared control shows somewhat longer paths still achieve similar time efficiency, which proves what was learned from scenario 1. A high-level plan from the user improves the navigation in the mixed traffic and mixed crowd scenarios.

The relative jerk showed to be significantly different between MDS and RDS controllers (p <0.1, F= 7.35), with RDS being 5 % lower. On the contrary, comparing SC and MDS did not show a significant difference. A more in-depth analysis of the controllers performance is presented in the next section through measurements of agreement and contribution to the obstacle avoidance.

3.4.3. Comparison through shared control metrics

The current tests aimed to estimate the level of assistance that is provided by the navigation algorithm to the driver, and the level of agreement (or disagreement) that the driver would have from the executed motion by the robot.
We observe the fluency of commands of the user (temporal continuity) given that the reactive navigation is intervening with its desired motion, as a reference measurement of the disagreement with the decisions of the assistive navigation. These measurements were selected from shared control studies metrics [Erdogan, A., and Argall B. 2017; Ezeh, C.et all. 2017].

Agreement: We defined agreement in terms of the deviation of the direction of the user's commands from the direction of the final shared control's velocity. Described as follows:

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

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

$$agreement = \sum_{i=0}^{N} a_i \cdot \Delta t_i \; / \; \sum_{i=0}^{N} \Delta t_i$$

where v and w are the translational and rotational velocities u~$[v\ w\ ]$, a_i is the normalised agreement at time step t_i and u_{SC}^i is the final output of the robot. 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.

Controller Contribution: C is calculated as C = ||Ur - Uh|| / ||Uh||, over a finite number of discrete samples N. Where the Ur is the user command in the robot command space (linear and angular speed),
Ur is the output command of the robot.
Both normalized to the maximum linear and angular velocities.
In case of full autonomy with a high-level input controller, we evaluated the contribution as $C = (\frac{||U_r||}{||U_{DS}||})$.
In here, || x || represents the L2-norm of x.

Fluency of Commands: We observe the fluency of commands of the user (temporal continuity) given that the reactive navigation is intervening with its desired motion, as a second reference measurement of the disagreement with the decisions of the assistive navigation.

$$F = \frac{1}{N} \sum_{t=t_0}^{t_N} 1 - |u_h^t - u_h^{t-1}|$$

where ut represents the user given command at time t, and N the number of samples taken in the interval h $t0\ totN$.

Number of collisions: Collisions with walls or pedestrians. This metric is the number of collisions that occurred in the scenario and were reported by the experimenters and post-analysis of the recordings.

Comparative Results:

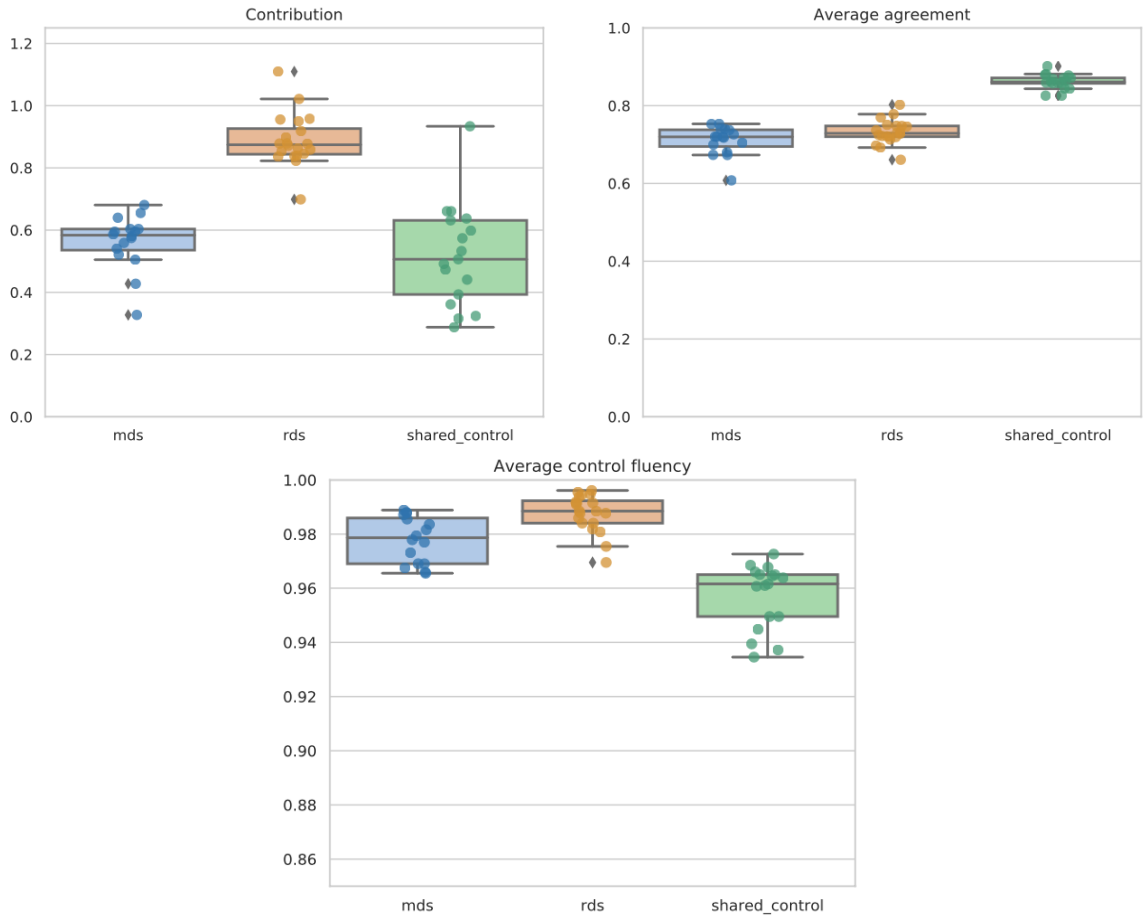

Figure 3.17. The relative comparison of the controller response in crowd navigation, showing navigation contribution, agreement and fluency for each method.

From the previous section we validated that the scenario of each controller was performed in similar crowds and the resulting navigation had non-significant difference in time or path length compared to the user driven recordings.
In contrast, the contribution to obstacle avoidance given by the controller showed to be significantly different between the methods ($p < 0.01$, $F = 37.0$).
In shared control the results (Figure 3.17) show that the obstacle avoidance contributed to 50%, with agreement over 85%, which means that the user was in control of the motion direction 50% of the time, and the obstacle avoidance assisted mostly in the velocity magnitude over the tests.

In the case of autonomous driving towards the set goal without high-level user planning, RDS guided most of the velocity components with an average 87% contribution, whereas the MDS controller provided 60% of contribution. The difference in the approach to avoid obstacles is clear, with RDS controlling the magnitude of the velocity, whereas MDS executes higher control over the angular component.

In turn, the fluency of the controllers showed to be significantly different among all tested controllers ($p < 0.01$, $F = 74.6$), however, only in shared control tests we observed an appreciable difference with a 2% drop in the fluency of the user-robot commands. Overall, this high fluency shows the compatibility of the reactive controllers to the high-level planner.

Collisions:

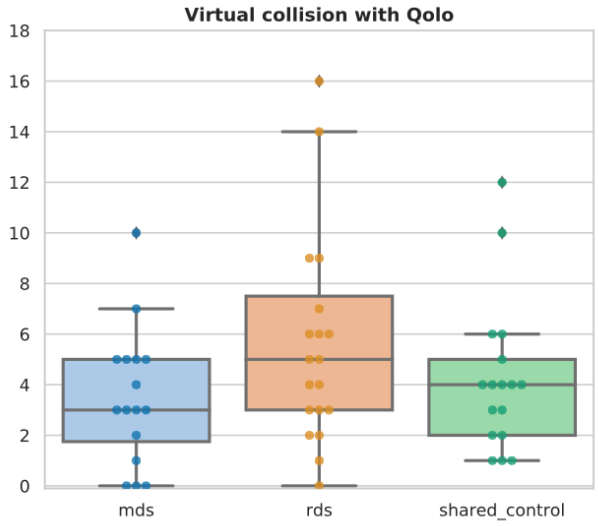

Figure 3.18. Virtual collision count with a mean number of collisions per trial of MDS 3.5±2.7, RDS 7.0±7.0 and shared control 7.9±4.2.

The virtual collisions showed to be slightly lower in the case of MDS control compared with RDS ($p < 0.1$, $F = 2.9$), with no significant difference to shared control (see, Fig. 3.18).

This result is expected given the larger size of the robot representation used in MDS, which forces the robot to be further away from pedestrians.

On the other hand, actual reported collisions are shown in the table 3.1.

Table 3.1. Data record of valid tests used for comparative analysis of the controllers and reported number of collisions on the overall recordings.

	Number of Recordings	Crowd flow tests	Number of Collisions
Shared Control	45	17	3
RDS	30	20	2
MDS	18	15	2

As detailed in table 3.1, we observed, across all conditions, 7 collisions leading to contacts with pedestrians. All of which occurred at the frontal bumper of the robot.
In 5 of the collisions, either the operator onboard the robot, or the external supervisor of the tests pressed the emergency stop, therefore, no post-collision control was recorded on those occasions.
In the remaining 2 contacts (when using the RDS model) the contact was with a pedestrian's wheeled shopping bag, and the contact was very short (<1 s), so that, there was not any perceived reaction from the compliance controller, nor was the collision perceived by the pedestrian.

We documented each collision case and found the following probable causes for the controller behaviour. Reasons for collisions:

1. Limited acceleration on the robot that could not match the pedestrian's acceleration. Probable cause of one collision with RDS test, and one with SC.
2. Non-holonomic constraints in dense crowds limit reactivity, probable cause for one collision with MDS.
3. Pedestrians being very close to a surrounding obstacle generated false negatives in the pedestrian detection. Probable cause for two collisions with shared control and MDS tests.

Comparative results per crowd and density type:

We selected a subsample of the overall tests where the robot was driven in shared control mode through reactive driving support (RDS) for analysing the effect of crowd type and density of the controller response.

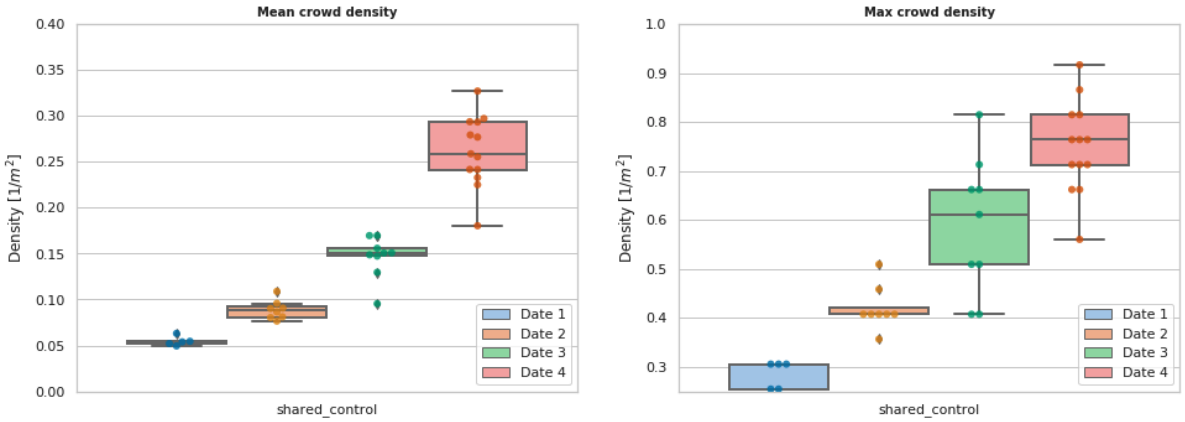

Figure 3.19. Subsample of recordings on shared control with RDS showing variations in crowd density and proximity to the robot in 3 scenarios: on date 1 with sparse crowd up to 0.3 ppsm, scenario 2 on dates 2 and 3 with crowd flows density up to 0.7 ppsm, and scenario 3 on date 4, with mixed crowds up to 1.0 ppsm. A one-way ANOVA showed all crowd densities to be significantly different ($p<0.01$, $F=105$).

Clustering by dates (see, Figure 3.19) shows the type of crowd and density roles in the controller strategy and provides us feedback into how the controller responds to different situations. Here, we analyse the behaviour of the crowd with a user providing high-level direction commands as described in Figure 3.3, with the RDS controller assisting to provide obstacle avoidance and the active compliance providing post-collision control.
We observed sparse crowds in date 1 (mean 0.05 ppsm, max:0.28 ppsm), compared with flow crowds in dates 2 (mean 0.09ppsm, max 0.42 ppsm) and date 3 (mean: 0.15ppsm, max: <0.8 ppsm). Finally, date 4 shows a highly dense environment (mean: 0.27 ppsm, max: 0.92 ppsm) of mixed crowds (both flow, static, and sparse). With all crowd types significantly different in mean and max density ($p<0.01$, $F=105$).

In Figure 3.20 we depict all the path efficiency metrics. On the one hand, we found a significant difference between date 1 - date 3 ($p<0.01$, $F=10.8$), and date 1-date 4 ($p<0.05$, $F=7.8$) on the relative time to goal, which suggests that the RDS controller is more efficient in sparse crowds compared with flow and dense crowded environments. On the other hand, no significant difference was found between highly dense crowds (~1 ppsm) and flow crowds (~0.5 ppsm).

In contrast, relative path length showed no significant difference among the tested crowd environments. Hence, the controller performed comparatively in all the crowds, in terms of path efficiency.

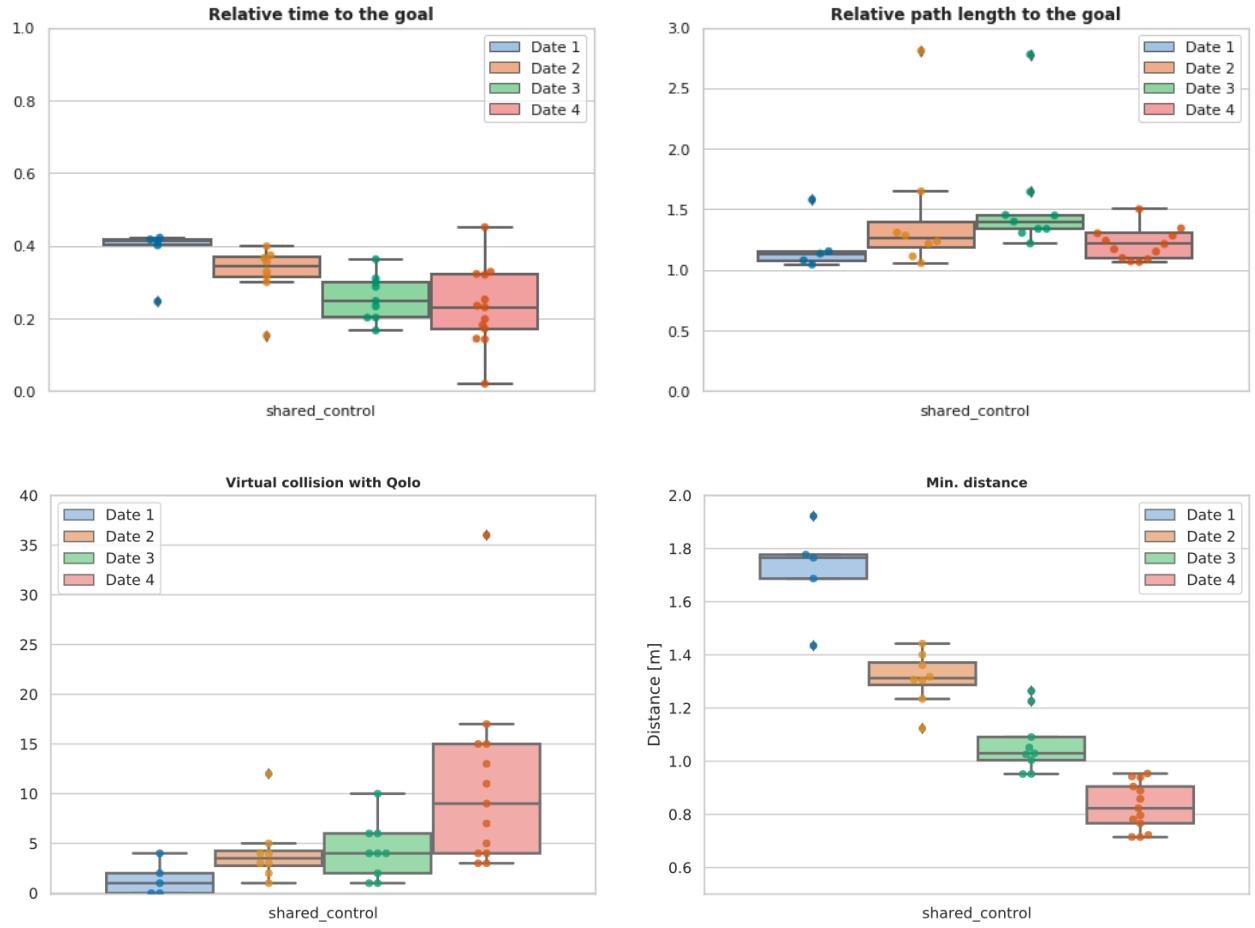

Figure 3.20. Comparison of the resulting path metrics in shared control navigation among the different crowd scenarios.

Figure 3.20 shows the minimal distance between the robot and the closest pedestrian, as well as, the number of virtual contacts in all tests.

The proximity clearly drops with the crowd density, whereas the average number of virtual collisions increases. We observe, in sparse crowds a mean of 1.4 collisions per test, 4.2 collisions per test in flows, and 10.9 collisions per test, in dense crowds.

Nonetheless, it is worth noting that in dense crowds no collisions were recorded. We recorded physical contacts on the flow crowds on dates 2 and 3, with 2 and 1 contacts respectively.

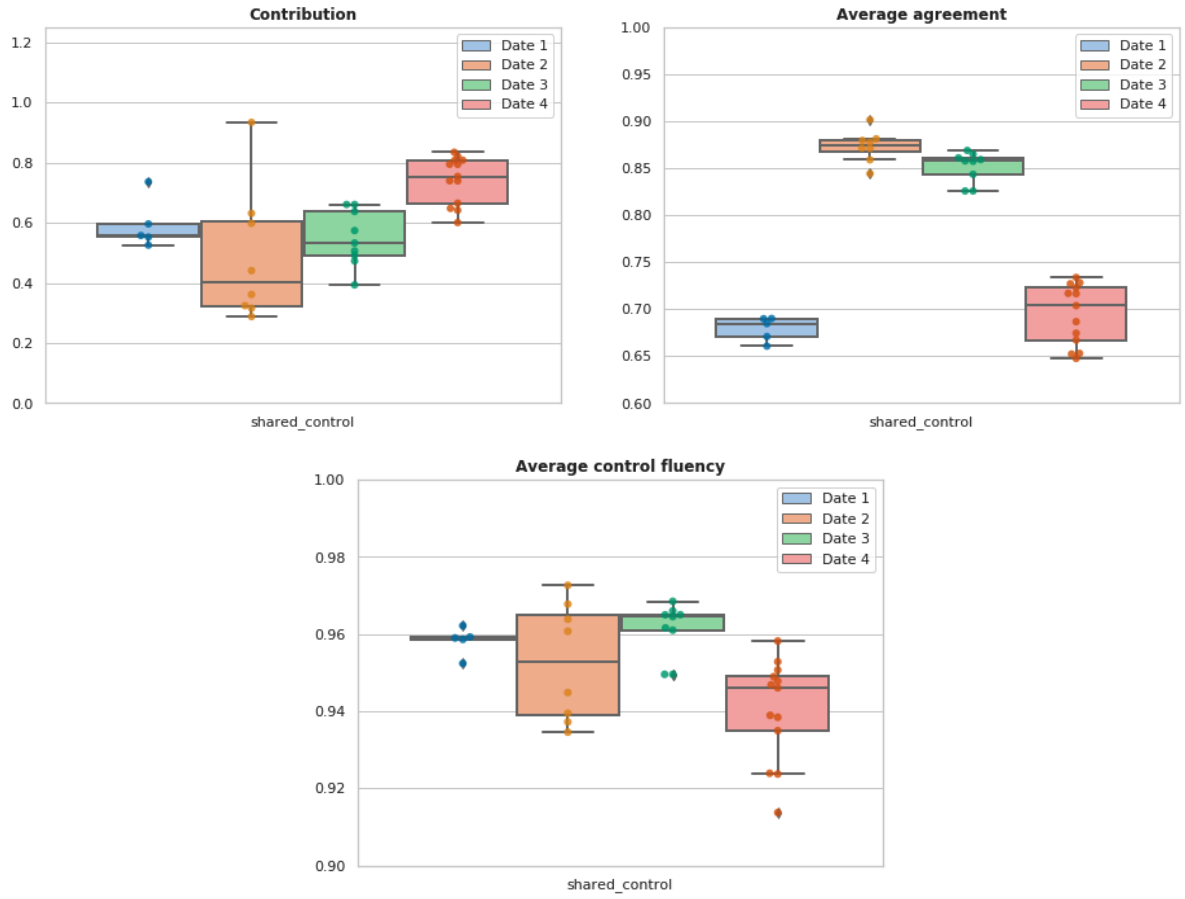

Figure 3.21. Comparison of shared control navigation (RDS) among all crowd scenarios, showing navigation contribution, agreement, and fluency for each scenario.

Figure 3.21 shows the comparison of the controller contribution for the navigation. We observe no significant difference between dates 1, 2 and 3, with only date 4 (dense crowds) showing a significantly higher contribution ($p < 0.01$, $F=17.1$) with mean 74%.

Equally, the user fluency was observed to be different only on high dense crowds with a drop of 2% (from average 95% to 93%). This might be caused by the user prompting to switch motion direction and speed given the highly dynamic crowd.

In agreement metric with the user we observe high values (89% and 87%) in the flow crowds, whereas in sparse crowds it drops to 68%, and in dense crowds to 69%. Behavior in sparse crowds is likely due to the result of the controller having enough free space to perform larger avoidance manoeuvres, whereas in flows the user and the controller agree in the direction as there is not much free space. In the case of dense crowds (date 4), we observe the controller performing similar manoeuvres depending on the crowd type (mixed static and sparse).

3.5. Conclusions

In the current evaluation of the reactive evaluation methods, we have shown it is feasible to navigate around raw pedestrian crowds demonstrating the controller's compatibility with the behaviour of bystander pedestrians. During all 100 trials in sparse crowds, crowd flows, and mixed crowds, we did not report on any incidents, or negative comments from the bystanders. On the contrary, several people approached the operators asking about the device Qolo that was being tested and the utility of it. In most cases the comments were positive about the perceived utility of having assistive navigation technology for mobility impaired users.

The control pipeline showed to perform well, from sensing and detection to control, however, there was a clear limitation in the people tracking capabilities given the small computing power of the embedded GPU used in the robot. This was reflected in some of the tests with collisions and unsuccessful recordings. Nonetheless, improvements were made during the period by the team in RWTH for creating focused attention areas for people tracking, herewith, allowing better performance.

Furthermore, the whole set of data created during these experiments was made open access and should enable future researchers on understanding people navigation around robots, and improving the detection and tracking methods specially from 3D lidar sensing.

The reactive navigation controllers showed to perform similarly in terms of trajectory efficiency, and fluency during the motions. The controllers differ in the way they avoid obstacles, one mainly deviating through angular velocity (MDS) and the other mostly by reducing speed (RDS). It is hence expected that they lead to measurable differences such as faster displacement to the goal for MDS than RDS (confirmed by results), larger jerkiness for MDS than RDS (confirmed by data). One may have expected that MDS leads to more collisions than RDS but this is not confirmed by the results. However, the number of collisions is so small that it may be insignificant to draw conclusions.

A significant difference found in the reactive controller response was given by the type of crowd, where sparse and mixed traffic crowds showed to be more difficult to navigate with lower agreement with the user. Whereas in contrast the flow crowd made it easier for the controller to respond to a somewhat uniform response of the surrounding pedestrians.

When a simple DS autonomously guided the robot in the tests, there were some problems of not interacting properly with the environment. i.e., getting inside stores, getting in the middle of people lining up, and getting to shops in the wrong direction, this behaviour is expected as there was no situational awareness of the complex mixed crowd environments. In comparison, shared control was much more natural and most of the tests ran smoothly with people going by without even noticing that there was a robot.

4. Pepper

Compared with the smart wheelchair and Qolo, Pepper is a fully autonomous Crowdbot and therefore needs to be able to navigate in highly dynamic environments without input from the user. In this section, therefore, we explore how unsupervised learning methods can assist reinforcement learning in robot navigation tasks. We compare two end-to-end, and 18 unsupervised-learning-based architectures, which we compare with existing approaches, in unseen test cases. We end this section by demonstrating our approach working on the Pepper robot in a real environment.

4.1. Research Question

Robot navigation is a task where reinforcement learning approaches are still unable to compete with traditional path planning. State-of-the-art methods differ in small ways, and do not all provide reproducible, openly available implementations. This makes comparing methods a challenge. Recent research has shown that unsupervised learning methods can scale impressively, and be leveraged to solve difficult problems. In this work, our research question is:

- How can unsupervised learning be used to assist reinforcement learning for robot navigation?

To answer this research question, we trained two end-to-end, and 18 unsupervised-learning-based architectures, and compared them, along with existing approaches, in unseen test cases. We also demonstrated our approach working on a real-life robot. Our results show that unsupervised learning methods are competitive with end-to-end methods. We also highlight the importance of various components such as input representation, predictive unsupervised learning, and latent features.

Our models are publicly available, as well as training and testing environments, and tools (https://github.com/ethz-asl/navrep). This release also includes OpenAI-gym-compatible environments designed to emulate the training conditions described by other papers with as much fidelity as possible. Our hope is that this helps in bringing together the field of RL for robot navigation, and allows meaningful comparisons across state-of-the-art methods.

4.2. NavRep: Unsupervised Representations for Reinforcement Learning of Robot Navigation in Dynamic Human Environments

4.2.1. Navigation simulation environment

The NavRepSim environment is designed with RL applications in mind. It aims to apply to any range-based sensor navigation problem. The goal of open-sourcing this simulator is to make it easier for anyone to reproduce state-of-the-art solutions for learning-based navigation.

The simulator offers 23 fixed maps as well as the possibility to procedurally generate maps containing random polygons (see Figure 4.1 for examples of the available maps). The offered maps were designed to imitate those used in prior work. Note that pre-made gym environments are made available to reproduce the specific training domains of other papers. We also created several new maps from real LiDAR data. Overall, we provide environments that range from simplified and highly synthetic to more realistic maps containing real-world sensor noise.

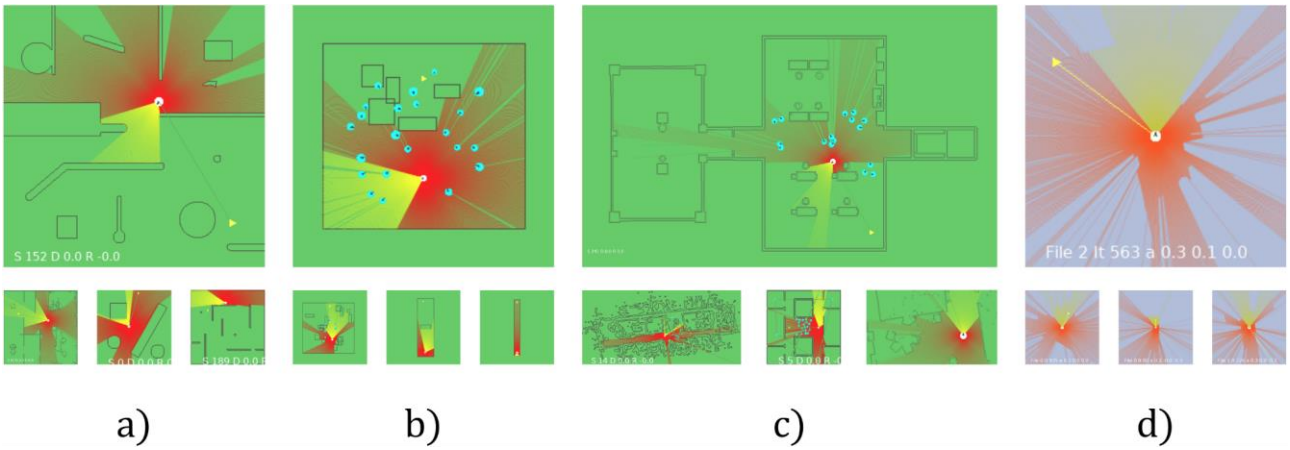

Figure 4.1. We make NavRepSim openly available, a simulator which contains efficient, gym-compatible environments for end-to-end navigation, allowing all to reproduce scenes from (a) Pfeiffer et al. (2017), Pfeiffer et al. (2018), (b) CADRL and SOADRL (Chen et al., 2017; Chen et al., 2019; Liu et al., 2020), CrowdMove (Fan et al., 2020) (not shown), (c) IAN (Dugas et al., 2020), and (d) real data.

In addition to having different maps, the learning environments can also vary in terms of the simulated robot(s), the simulated human agents as well as the learning curriculum. The simulated robot can be either holonomic or differential-drive. The inertial physics of the robot base can be simulated with high fidelity or simplified as instantaneous velocities. Further, collisions can either lead to damage (reflected as a negative reward) or instant episode termination. Users can also specify the number of robots in the environment, since joint training of several robots in the same simulation is also possible. The number and behaviour of the simulated human agents can also be adjusted. Agents in the environment may be static or dynamic, they may be rendered as moving legs or ellipses, and they can be controlled using different policies (constant velocity, ORCA (Van den Berg et al., 2008), global planning). Finally, we offer two learning curricula. The first uses a constant episode set-up, picking different maps and agent locations for every episode, while the second varies the environment difficulty (more human agents and more obstacles in the case of procedurally generated maps) based on the policy's success.

Care is taken to maintain simulation efficiency, with most environments running at more than 100 iterations per second in a single thread (on a laptop, Intel Core i7-6600U CPU @ 2.60GHz x 4). For comparison, the CarRacing gym environment used in Ha & Schmidhuber (2018) runs at the same speed on the same machine.

4.2.2. Learning-based navigation

A core purpose of this work is to study the possible variants of learning-based robot navigation. The basic setup common across many state-of-the-art methods is a ground robot equipped with a range sensor (often a 2D planar LiDAR). The state space used in the learning formulation is derived from the incoming range data. Several works also assume that live pedestrian tracks are available to the robot, however in our work we do not make this assumption. The robot action space is continuous and is represented as a vector in R^3, $a = [v_x, v_y, \omega]$. Each of v_x, v_y, $\omega \in [-1,1]$ correspond to the robot's translational and rotational velocity in its base-link frame. Here we consider a holonomic robot, however the work can be extended to diff-drive robots by simply requiring $v_y = 0$.

The scalar reward function for time-step *t* is

$r^t = {r_s}^t + {r_c}^t + {r_d}^t + {r_p}^t.$

The first term, ${r_s}^t$, is the success reward, which is 100 if the goal is reached and 0 otherwise. The collision reward, ${r_c}^t$, is -25 if the robot collides and 0 otherwise. The danger reward, ${r_d}^t$, is equal to -1 if the robot is closer than 0.2m to an obstacle or agent, and 0 otherwise. Finally, ${r_p}^t$ is the progress reward, which is equal to the difference between the previous distance to goal and new distance to goal.

4.2.2.1. State space variants

The state representation s^t is a combination of the LiDAR observations ${s_l}^t$ and the robot velocity and goal position in the robot frame ${s_r}^t$ (see Figure 4.2). In this work, we consider two methods for representing LiDAR measurements and compare their effects when used as inputs to a learned navigation policy. The first representation is simply a 1-dimensional vector of the 1080 raw distance values from the LiDAR normalized to [0, 1]. We refer to this as the **1D** representation. We also consider a probabilistic representation for the input LiDAR data. This representation maps the space around the robot as a polar grid ($R^{64\times64}$), with evenly distributed discrete angular sections and exponentially distributed discrete radius intervals, leading to a greater resolution closer to the robot compared to regions that are farther away. Given the LiDAR scan, each grid cell is assigned a value $p \in \{0,0.5,1\}$ for being unoccupied, unknown or occupied, respectively. We refer to this two-dimensional probabilistic LiDAR representation as the **rings** representation.

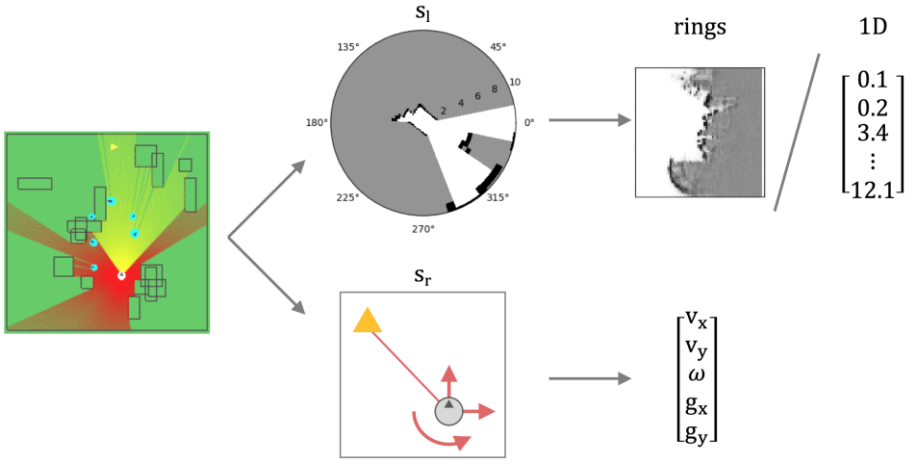

Figure 4.2. The state representation obtained from the world (left) combines both the LiDAR observation s_l, and goal/velocity observation s_r. The LiDAR observation is either represented as a 64×64 matrix of occupancy probability (rings) or a 1080 vector of distances (1D).

4.2.2.2. NavRep: Unsupervised representations for navigation

In contrast to end-to-end learning approaches, NavRep exploits unsupervised representations. Following the approach of Ha & Schmidhuber (2018), we make use of three main modules,

- V (LiDAR encoding): a variational auto-encoder (VAE) trained to reconstruct the LiDAR state (Figure 4.3),
- M (prediction): a sequence-to-sequence model trained to predict future state (Figure 4.4),
- C (controller): a 2-layer fully connected network (FCN) trained to navigate using s_r and the latent representations of V and M, as input. Its output is the probability of taking action a.

For brevity, we refer to the latent encoding of V as z, and the latent encoding of M as h.

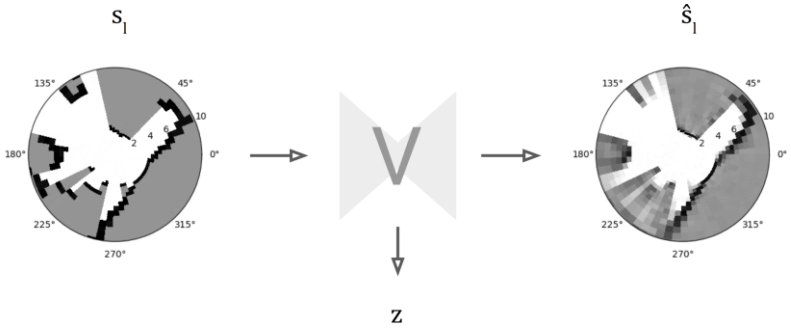

Figure 4.3. Visualization of the V auto-encoding module's inputs and outputs. The LiDAR observation $s_l$and reconstruction $\hat{s_l}$can be either in the rings or 1D representation.

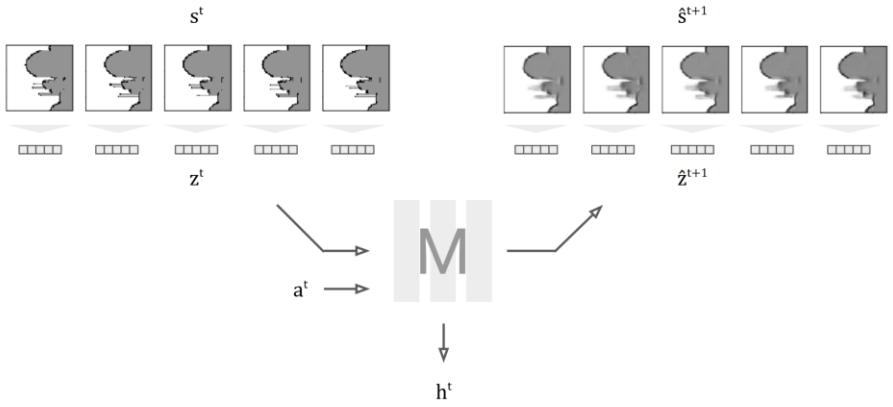

Figure 4.4. Visualization of the M prediction module's inputs and outputs. The *z* latent features for each observation in the sequence are first encoded by the V module. These are passed to M along with the sequence of actions *a*. Each predicted $\hat{z}^{t+1}$ in the sequence is then decoded by V.

1) *Modular training:* From this set of modules, we offer our first NavRep learning variant. The V, M, and C modules are all trained separately, an LSTM is used for M, as in Ha & Schmidhuber (2018).
2) *Joint training:* Our second NavRep learning variant is designed to provide insight into the effect of training V and M jointly or separately. In Ha & Schmidhuber (2018), the authors first train V to minimize reconstruction loss before training M to minimize the *z* prediction loss while keeping the V network weights fixed. Here we compare their **modular** training regime with one that trains V and M **jointly**. An LSTM is used for M.
3) *Transformer architecture:* Rather than an LSTM, we use the attention-based **Transformer** architecture (Vaswani et al, 2017) in M to generate predictions of the future LiDAR encodings.
4) *Input features to C:* We consider the effect of the inputs used by the controller to compute the robot action, splitting each NavRep variant into three. We compare between using **z**, **h** or **[z, h]**. An overview of all the learning architecture and training variants that we study is provided in Figure 4.5.

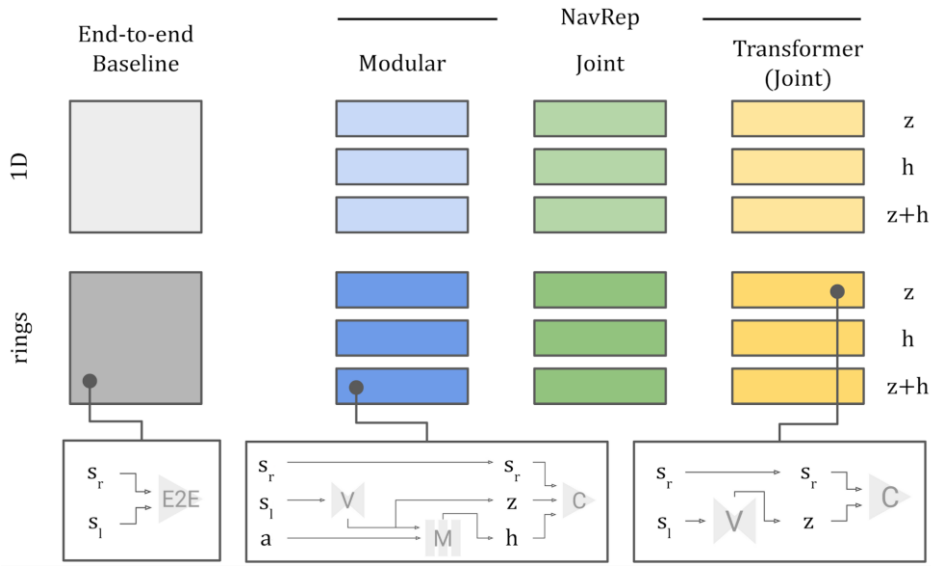

Figure 4.5. Overview of the architectures trained in this work. For all approaches there are 2 LiDAR input variants. In addition, the 3 unsupervised-learning-based approaches each have 3 latent feature variants. Simplified schema of the components is detailed for 3 arbitrarily selected variants.

4.2.2.3. Network implementation and training

1) *Hyperparameters:* All V modules are composed of a 4-layer CNN encoder, followed by 2-layer FCN, and 4-layer CNN decoding blocks. *z* latent features are of size 32, *h* latent features are of size 64. The LSTM-based M module has 512-cells. The Transformer-based M module is composed of 8 causal-

self-attention blocks with 8 attention heads each. The modular V and M modules have respectively 4.3 million and 1.3 million trainable parameters. The V+M transformer model has 4.8 million trainable parameters. More details available in the open-source implementation.

2) *V:* was trained using minibatch gradient descent, with a typical VAE loss,

   $L_V = L_{rec} + KL(z^t)$,

   where $KL(z^t)$ is the Kullblack-Leibler divergence between the latent feature distribution parameterized by the encoder and the prior latent feature distribution. The reconstruction loss L_{rec} is the mean squared error between input and output,

   $$L_{rec} = H(\hat{s}^t, s^t), \tag{4.1}$$

   where *H* denotes the binary cross entropy, s^t denotes the observed state (LiDAR representation) at time *t* in the sequence.

3) *M:* was trained on batches of sequences. We define the loss function as the latent prediction loss,

   $$L_M = H(\hat{z}^{t+1}, z^{t+1}) + H(\hat{s}_r{}^{t+1}, s_r{}^{t+1}). \tag{4.2}$$

   This adds the binary cross entropy loss between the predicted ($\hat{z}^{t+1}$) and the true (z^{t+1}) latent features to that of the predicted ($\hat{s}_r{}^{t+1}$) and the true ($s_r{}^{t+1}$) goal-velocity states.

4) *Joint V+M:* was trained with a loss function composed of the sum of the reconstruction and prediction losses from Eq. 4.1 and Eq. 4.2. However, in this case, for the rings variant the prediction loss is,

   $L_{V+M} = H(\hat{s}^{t+1}, s^{t+1})$,

   the binary cross entropy loss between the reconstructed state prediction $\hat{s}^{t+1} = V(\hat{z}^{t+1})$ and true next state s^{t+1}. For the 1D LiDAR variant we used the mean squared error instead of the cross-entropy loss as the predictions are not distributions, but direct values.

5) *C:* was trained using the PPO algorithm (Schulman et al., 2017; Hill et al, 2018). Each C module was trained up to 3 times with varying random seeds in order to mitigate initialization sensitivity in the results. Training was stopped after 60 million training steps.

6) *Training data:* The V and M modules were trained with data extracted from the same environment in which the C module was later trained. To generate training sequences, the ORCA (Van den Berg et al., 2008) policy was used to control the robot, and the number of agents in the scene was increased to ensure sufficient observations of dynamic objects and prevent imbalance in the dataset. A simple model of leg movement was used to render dynamic agents as moving legs in the generated LiDAR data and a time-step of 0.2s was used for updating the agents' positions.

   The C module itself was trained in the SOADRL-like environment (shown in Figure 4.1), with randomly generated polygons and agents (every episode is unique). A bounding obstacle had to be added to prevent the policy from learning a simple risk-averse heuristic which consists in going around the entire area. During training, actions for the C module were limited to x-y movement, with 0 rotation. However, the robot's initial angular position was randomized for each episode.

   Curriculum learning was used to improve the training of the policy: every failed episode leads to a lower number of polygon obstacles and agents, conversely, every successful episode leads to an increase. The maximum number of agents and polygons was capped to 5 agents and 10 obstacles.

4.2.2.4. End-to-end learning baselines

As a baseline, we implement a typical end-to-end RL approach. This approach attempts to learn a navigation policy $\pi(a, s)$ predicting action probabilities directly from LiDAR inputs.

Two variants of end-to-end learning are implemented: the first taking the 1-dimensional LiDAR representation as the state input, and the second taking 2-dimensional rings representation as the state input. For the value and policy model, we use a CNN for feature extraction, followed by fully connected layers. To make comparisons fair, the same CNN architecture and dimensions are used as in the V and C models described in the following subsection.

4.3. Evaluations

4.3.1. Simulation Experiments

4.3.1.1. Worldmodel error for unsupervised learning architectures

The six joint, modular and transformer architectures (Figure 4.5) have differing loss functions, which makes it difficult to compare their losses during training. To this end, we define the "worldmodel error", a next-step prediction error term that allows meaningful comparison across architectures,

$$E_{worldmodel} = MSE(\hat{s}^{t+1}, s^{t+1}).$$

However, this error function does not produce comparable results for the rings vs. 1d LiDAR inputs, as they do not have the same dimensions and scale. Instead, these comparisons are shown separately in Figure 4.6. The worldmodel errors shown here are calculated for the training sets, which compares each architecture's ability to learn in an unsupervised manner. It can also be understood as the accuracy of 'Dreams' generated by each architecture. In our results (Figure 4.6) joint outperforms modular training, and Transformer outperforms LSTM.

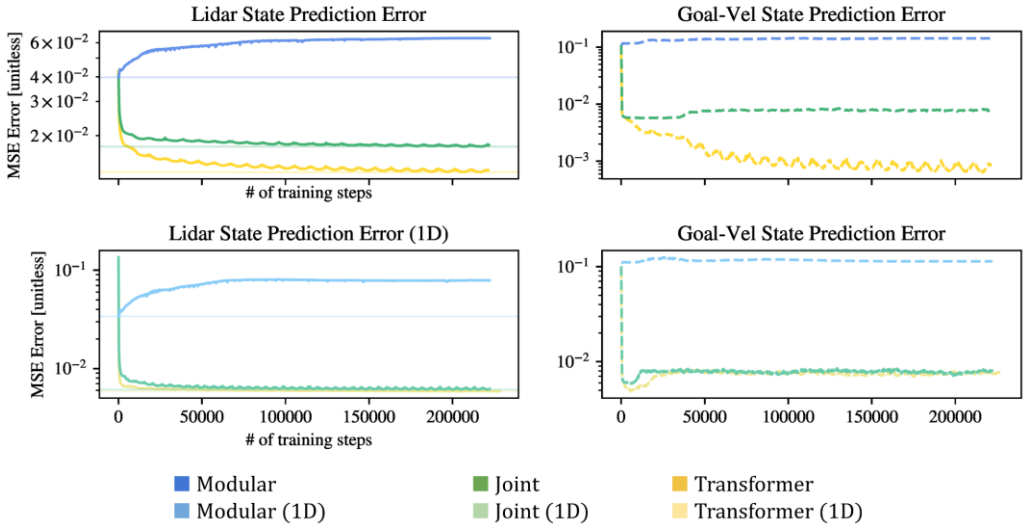

Figure 4.6. Comparison of the *worldmodel error* for the joint, modular and transformer architectures, for their (top) rings LiDAR state representation variant, and (bottom) 1D variant. This error is calculated and shown separately for the LiDAR state s_l (left) and goal-velocity state s_r (right).

4.3.1.2. Controller training performance

Though the training environment features curriculum learning through adaptive difficulty, a separate validation environment with fixed difficulty is used to quantify the absolute progress of each model every 100'000 training steps. This validation is similar to the training environment, except that it remains on the maximum difficulty, i.e. maximum number of agents and obstacles, and that it consists of 100 episodes with arbitrary but deterministic initializations. Success rates (number of validations runs in which the robot reaches the goal without collision) are shown in Figure 4.7.

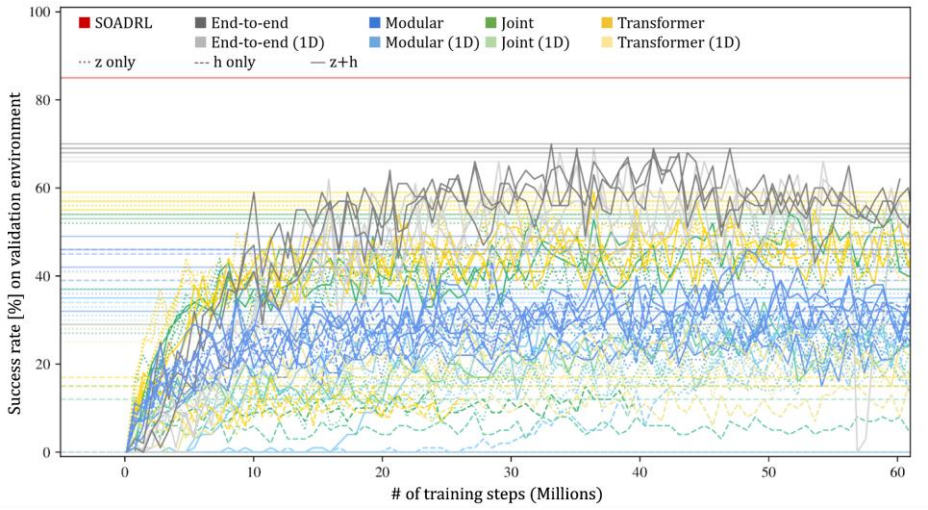

Figure 4.7. 3 training sessions were performed for each of the two end-to-end and 18 NavRep architectures to account for the influence of network initialization on the results (legend as in Figure 4.6). Horizontal lines mark the maximum success rate of each curve. The SOADRL success rate allows for comparison with methods that use exact human locations as input.

Looking at the modular, joint and transformer models, it seems that the worldmodel error and learned controller performance are correlated. This suggests that better reconstruction and prediction performance in the V and M modules leads to latent features which are more useful for the control task. Nevertheless, it can also be seen that the end-to-end approaches perform better in the validation tasks than the NavRep controllers. Finally, none of the models are able to surpass the performance of SOADRL, which has access to exact positions of pedestrians.

4.3.1.3. Test performance

Each model was tested in 27 unseen scenarios (3 maps, 9 scenarios each). At least 300 episodes were run in each scenario, for a total of 10'000 test episodes per model. These scenarios are designed to represent a wide range of possible real-life situations that can help to identify specific issues with the learned controller, such as global planning, agent avoidance and crowd aversion. For each map, the first 6 scenarios are taken from Dugas et al. (2020), and the remaining 3 are (7) a trivial scenario where the robot is close to the goal with no obstacle in between, (8) an easy scenario with a close goal, and a single obstacle between robot and goal, and (9) a typical dynamic avoidance task where all agents are placed on a small circle with goals on the opposite side and no obstacles in between. The scenario definitions and selection were fixed before any testing took place to avoid bias.

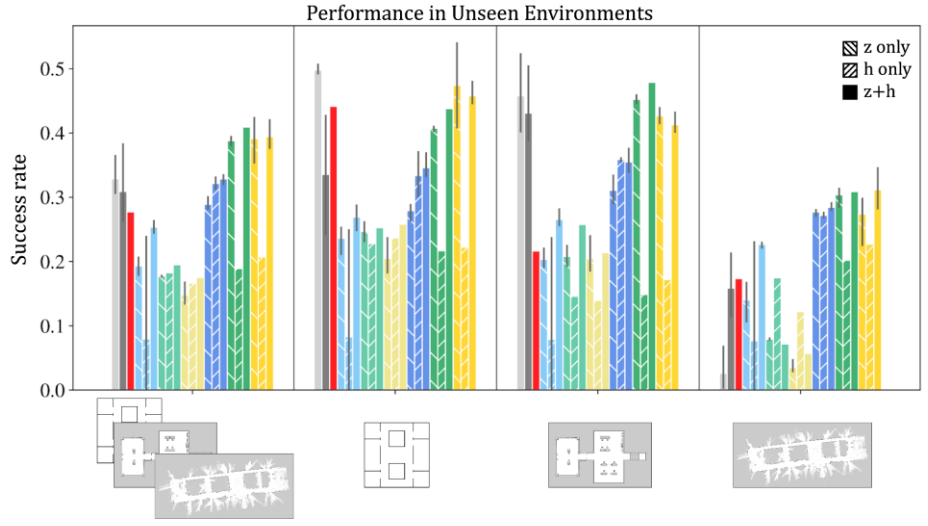

Figure 4.8. Mean performance of each model in the testing environments. Legend as in Figs 4.6, 4.7, with error bars denoting the minimum and maximum score across random seeds. Overall performance is shown (left), as well as per-map performance from the simple, complex, and realistic maps.

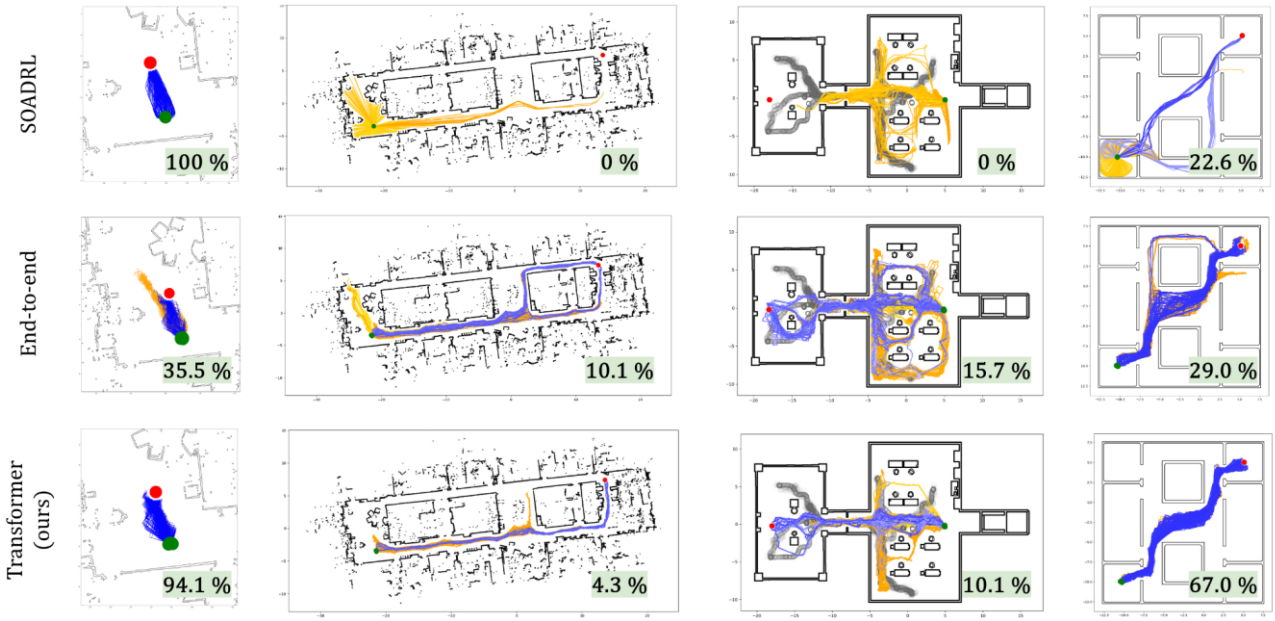

Figure 4.9. Comparison of trajectories executed by the best performing models during testing in unseen environments. From left to right, the scenarios are 1) simplest scenario in realistic map, 2) global planning task in realistic map, 3) full planning task (global, static, dynamic avoidance) in complex map, 4) global planning in simple map. Orange trajectories are trajectories which resulted in failure to reach the goal, through collision or time-out. Blue trajectories are successful trajectories. Grey circles indicate the positions of dynamic agents. Overlaid is the percentage of successful trajectories.

In general, it can be seen that the end-to-end model performance suffers a greater drop in performance from training to testing, whereas the NavRep models have more consistent performance (Figure 4.8). In particular, the 1D LiDAR end-to-end model fails completely in the realistic map even though it is a top performer during training. In addition, when investigating the low performance of SOADRL, we saw that most failures were due to crashes into obstacles and not agents.

In the same test environment, a traditional planner based on the TEB approach from the widely-used ROS navigation stack achieves 76% success rate.

The low success rates of all models on long-range navigation scenarios (Figure 4.9) indicates that global navigation is a challenge for RL-based methods. We also see that several works (Chen et al., 2017; Liu et al., 2020; Everett et al., 2018) avoid this problem by dealing with local navigation only. Looking at other failures, we see that in several cases, the end-to-end and unsupervised models don't learn to proceed slowly in the presence of "danger" (other dynamic agents). Instead they move as fast as possible, or are stuck when available space becomes narrow. This can perhaps be improved through reward design. Finally, in the challenging dense static crowd scenario, though most attempts fail, a few are able to succeed by avoiding the dense area, which implies taking a significantly longer route. As shown in Dugas et al. (2020), this is a challenging situation even for traditional planners.

4.3.1.4. Analysis of learning variants

1) *Impact of rings:* Rings encodes the space around the robot as values between 0 and 1 in a pseudo-probabilistic representation of occupancy. It seems that allowing the NavRep model to infer from and predict occupancy uncertainties explicitly makes its task easier than inferring directly from the raw LiDAR scan. In addition, rings provide a higher-level abstraction for the CNN to operate in, meaning that it requires less depth, while scaling the spatial resolution information to focus on regions close to

the robot. This is in line with how sensors typically work and also matches the way that, in obstacle avoidance, more attention should usually be paid to close objects. The difference in resolution sensitivity between the 1D and rings representations could also explain the dramatic performance drop observed in the realistic map for the end-to-end model. Local features in the LiDAR representations for the simple vs. realistic maps vary more in the 1D case than in the rings. Moreover, rings more easily allow using the same model with different sensors of varying resolution.

2) *Joint vs modular training:* Previous work (Ha & Schmidhuber, 2018) recommends modular training, that is, training V, M and C separately, and to use both *z* and *h* as inputs to the C module. However, in this work we find that jointly trained V+M models provide superior performance in almost all tasks. The computational cost of joint training is furthermore not significantly greater than for modular training, though memory usage does increase due to having to store sequences of rich observations rather than compressed latent features.
3) *Latent feature usefulness:* When looking at model performances both in seen and unseen environments, our results do not support the idea that using both *z* and *h* latent features is advantageous. Though in some cases models trained with *z*+*h* features performed slightly better, this improvement is not significant when compared to the variance between random seeds. In our experience, the performance cost of including *h* latent features in the C training regime is great: training a *z*+*h* C model takes approximately 9 times longer than for an equivalent *z*-only C model. *h*-only models have it worst, with low test performance and slow training times.

4.3.2. Hardware Experiments

We took our Transformer architecture (rings, *z*-only) and implemented it on a Pepper robot (Figure 4.10). Our robot has a quasi-holonomic base and two 270° LiDARs merged together to form a 1080-vector combined range scan. The robot was given several waypoints in a space mixed with people and obstacles, and tasked to patrol between waypoints.

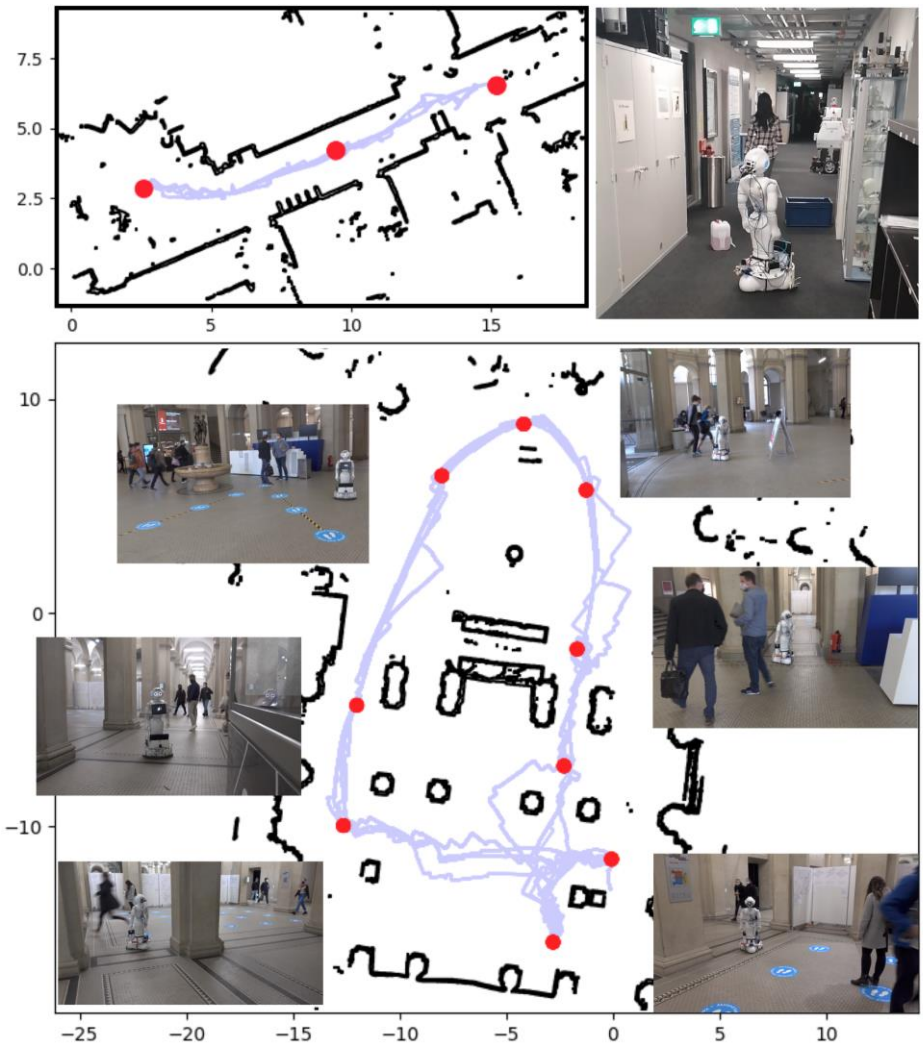

Figure 4.10. We test the proposed method on a real robot, where the learned policy is repeatedly able to avoid obstacles and people and reach its goals. Goals are shown in red, successful trajectories in blue, images taken during the experiment are displayed next to the locations they were taken in. Each sequence was run three times in a row.

The learned controller achieved 100% success in the real environment, reaching its goal for all trajectories. However, we observed issues which show a need for improvement.

(i) Smoothness: the commands selected by the controller are aggressive, leading to jerky motion.

(ii) Reluctance to go into tight spaces: when faced with narrow but traversable areas, the controller oscillates around the entrance but eventually makes it through.

(iii) Getting stuck: we observed the planner spend several seconds oscillating in front of an obstacle before finally passing on one side.

These issues appear to be in part due to the following causes:

(i) Inertia in the dynamics of the robot is not modelled in the training environment, which causes sub-optimal control in the policy.

(ii) Sensor noise is not present in the simulated environment. As a result, when it occurs in the real scenario, the controller can react unpredictably.

We surmise that there are further unknown reasons. Such “unknown unknown” effects could possibly be addressed via continual learning by refining the model in the real world.

4.4. Conclusion

Training robust sensor-to-control policies for robot navigation remains a challenging task. NavRep methods do not dominate over end-to-end methods, but we do see interesting properties and trade-offs between the two. Though less suited to learning specific tasks, unsupervised representations display several benefits, such as more consistent performance across general tasks, modularity, and the potential to generate "Dream" environments (Ha & Schmidhuber, 2018), which could assist or replace simulation. The popularity of these methods is additionally propelled by the continued increase in compute and memory availability. Moving forward we expect to see larger unsupervised models with richer modalities, that create more convincing approximations of reality and provide more useful and discerning features.

Nevertheless, navigation-RL still has a long way to go, should one aim to replace traditional planners. Among the current issues are global planning, sim-2-real constraints, and safety. Potential avenues include more rigorous exploration of reward functions, more realistic training environments, and continual learning. Given that standardized gym-like environments could help harmonize future research, we hope that by making NavRep and NavRepSim publicly available and easy to use, progress in this domain can be accelerated and to improve collaboration between researchers.

5. CuyBot

Like Pepper, cuyBot is a fully autonomous robot that cannot rely upon an operator to help out when traversing complex dynamic and crowded environments. It is inevitable at some point that there will be contact between an autonomous robot and pedestrians or bystanders. Therefore, in this section we investigate the safety and acceptability of cuyBot's compliant motion control system. We report on a physical experiment with 20 participants at University Bonn-Rhein-Sieg in Germany.

5.1. Research question

In previous navigation experiments (D1.4), the cuyBot robot was driving within a university building amidst different types of crowds. Although one of the features of the robot is its compliant motion control, it was hardly observed that a person would want to make use of it and try to push the robot away. Even in situations with higher crowd densities people would rather avoid the robot and move around. Since then the motion control of the robot platform has been significantly improved. The robot's special actuators called KELO Drives, compact modules of wheel-pairs that are powered by gearless hub-motors, enable compliant motion behaviour even without additional force or contact sensors on the robot's periphery. In this new experiment we conduct an empirical evaluation of the new compliant control with the following research questions:

- What are reasons for the hesitation as well as conditions for the acceptance of getting into direct physical contact with a moving robot?
- What influence do practical experiences with a robot's compliant behaviour have on the level of comfort when being in close proximity to the robot?

As mobile service robots enter more and more areas of daily life, interactions with humans become an increasingly frequent phenomenon, not only for technology-affine people, but for any person that may happen to be in the vicinity of the robot. The presence of a robot thereby may evoke a wide range of emotions, from positive approval and curiosity to fear and discomfort. This topic gains even more importance when the distance between robot and human gets smaller, up to direct contact as it may happen in crowded situations. The meaning of distances between humans is studied in the field of proxemics and gained extensive attention in the context of social-aware robot navigation (Mumm and Mutlu, 2011).

(Takayama and Pantofaru, 2009) investigated influences on the proxemic behaviours during human-robot interaction and conducted a user study to evaluate four hypotheses. They found evidence that previous experiences with robots (as well as owning pets) leads to a decrease of people's personal space around robots, while the direction of approach (human towards robot or vice versa) has no noticeable effect. Also, negative attitudes towards robots were found to have an effect of increased distancing. The experiment with cuyBot is closely related to this study, including the additional element of physical interaction in the form of pushing.

The general acceptance of robots has been studied in numerous studies. (Leite et al., 2013) provides a survey of long-term studies on human-robot interaction with social robots. Variables that influence the acceptance have been for example identified in (DeGraaf and Allouch, 2013). (Horstman and Krämer, 2020) derive in a study that the robot's actual behaviour had more impact on people's evaluation than their initial expectations or backgrounds. (Carreno-Medrano et al, 2021) evaluate the perception and expectation of users before and after conducting a workshop with a humanoid robot, listing five categories (likeability, perceived intelligence, trust, reliability, expected success) that all decrease on average after the workshop was conducted. In this experiment with the cuyBot robot we want to investigate if the experience with the robot's compliant control behaviour may lead to similar effects.

5.2. Experimental setup

The experiment was conducted as a combination of a physical demonstration with a robot and a questionnaire. It took place in the University Bonn-Rhein-Sieg in Germany, partly in the university's main entry hall and partly in a lab room. Two interviewers were controlling a cuyBot robot via joypad at these locations. Passengers passing by were asked if they would like to join a robot experiment and short survey. In total 20 persons were found to take part.

Figure 5.1. Steps of test procedure (shown with one of the interviewers). In the first step the robot approaches a person who pushes the robot back with his hand as well as with the foot. In the second step the robot nudges the person from the side.

For each participant the following procedure was executed:

1. The scope of the experiment was explained and participants were asked about their familiarity with robots in general.
2. The interviewers demonstrated the compliant behaviour of the robot in two steps (Figure 5.1):
    a. First, let the robot approach them from the side with a slow velocity of about 0.10 m/s and when getting close they push it away with the hand against the robot's upper trunk, as well as with the foot against the base frame.
    b. Second, they let the robot get into contact and slightly nudge the person with a quasi-static force of about 25 N, indicating the request to make way.
3. The participants were asked how comfortable they would feel in the two situations with robot contacts as just demonstrated. The level of comfort was given on a scale from 1 to 5, with 1 denoting strong reluctance, 3 being neutral and 5 having no concerns.
4. The participants redid the steps in 2. and experienced the behaviour of the robot themselves.
5. Afterwards they were asked about changes of their initial evaluations, reasons for discomfort and conditions on the acceptance for such situations.

The answers and reactions were noted by one of the interviewers on paper. The survey was anonymous, only the gender was noted as personal data.

5.3. Evaluation

5.3.1. Participants

20 participants took part in total, with 16 male and 4 females. Due to the small number of female participants gender differences have not been analysed.

40% of the participants said that they have no or only minor experiences with robots, such as small domestic robots. 60% of participants felt familiar with robots in general. The high level is due to the situation in the university where mostly students or staff of technical courses were present.

In the experiment, all participants tried to push the robot away as shown before by the interviewers. All of them pushed with the hand against its upper trunk, 85% also against the base. The concrete reaction is not emphasized further, since participants were influenced by the previous demonstration by the interviewers. No participant showed a noticeable reaction of fear or uneasiness.

5.3.2. Level of comfort in situations with robot contact

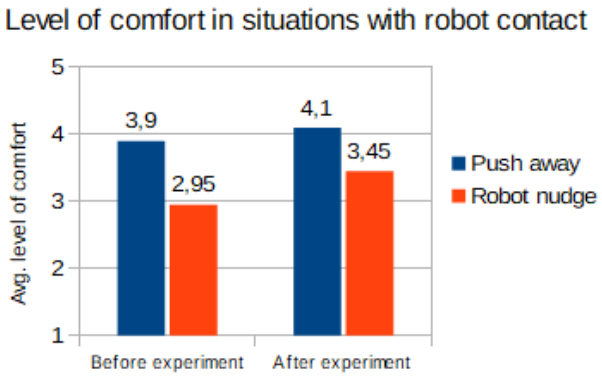

Figure 5.2. Average level of comfort in situations with direct robot contact.

On average the participants felt more comfortable in a situation where they could push the robot away in comparison to when the robot would touch and nudge them (Figure 5.2). This holds for both, before and after making the practical experience with the robot. The initial comfort level when self-pushing was high with 3.9 (on the scale 1 to 5), the comfort level when being nudged was just below neutral (2.95).

After having the practical experience with the robot, the estimation of the situation to push the robot only slightly changed by 0.2 points, with 5 participants increasing their evaluation and 2 decreasing. The evaluation of being nudged showed a bigger influence with a change of 0.5 points. 12 participants increased their evaluation, while 4 decreased it.

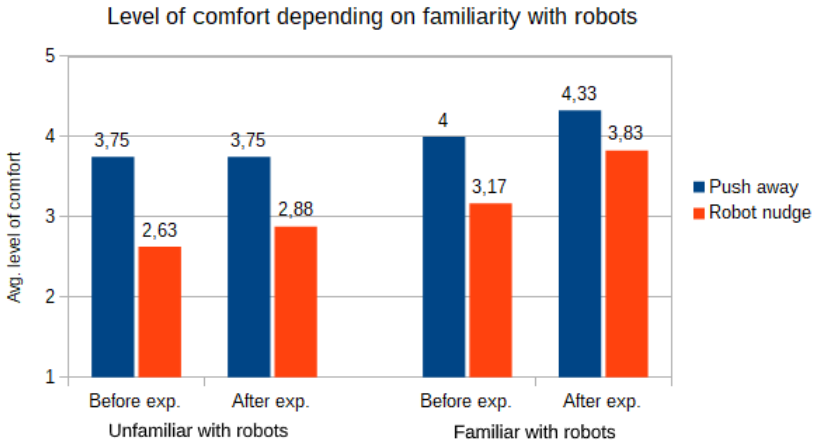

Figure 5.3. Average level of comfort depending on familiarity with robots.

The relation between the participants' familiarity with robots and the change of evaluations was rather contrary to initial expectations. For participants with few experiences with robots, the evaluations only slightly changed (Figure 5.3). For pushing the robot there was actually no change on average, and for being nudged only by 0.25 points.

On the other hand, participants that claimed to be familiar with robots gave a significantly higher score after the experiment, increasing by 0.33 points for pushing and 0.66 points for robot nudging. Originally, it was

expected that people new to robots should be much more influenced by the novel experience than those that have already a good understanding of related technologies. A possible explanation for this effect is that the robot's behaviour confirmed positive expectations, but avoided known possible hazards. If the expectations are rather vague, the single experience with one robot might not be sufficient yet for a significant change of attitude.

5.3.3. Reasons for reduced comfort level

Participants were asked about the reasons for their previous evaluation, without providing preselected options in order to avoid bias and with the option of multiple answers. The following set of replies were given concerning the two situations of close contact. The percentage in number indicates how many participants mentioned that concern.

Concerns if person pushes the robot:

- Fear to damage the robot (15%): It was not clear if applying forces could break something. In particular external elements such as sensors (RGB-D camera in robot's trunk) and the touch display were mentioned to possibly have issues when being pressed.
- Too high resistance (15%): The participants expected to require only very small force to make the robot back off.
- Unintuitive continuous pushing (10%): For the purpose of the experiment the robot would continuously try to keep moving forward. In a real scenario it would be rather expected that the robot should possess some kind of situational awareness and understand a manual push as a clear indication to not approach again.
- Intrusive proximity (10%). The close proximity of the robot felt intrusive. This was the reason for both of the worst evaluations with scores of 1. Here the resultant behaviour of the robot after contact hardly mattered, as getting so close was already considered highly undesired.

Concerns if robot nudges the person:

- Dependency on robot behaviour (35%): The behaviour was regarded as acceptable, but only under the assumption that the robot behaves as in the experiment. In particular it should be similarly slow and not exert much more force.
- Missing awareness of robot (20%): If the participant is unaware of the robot and gets surprised by the robot suddenly being so close, the reaction might be different and more negative. This issue is discussed in the next two paragraphs.
- Restraint of space (15%): When the robot got into direct contact with the participants, they felt their own movability disturbed, in particular due to lack of space for the legs.
- Intrusive proximity (10%): Similar as for manually pushing the robot, the general close proximity was considered as highly undesired, leading to two evaluation scores of 1.
- Unclear robot intention (10%): The robot's intention why it got so close would be unclear. In addition, the participant would be unsure which action might trigger what kind of reaction from the robot, maybe touching it could cause an alarm signal to set off.

5.3.4. Robot approach direction

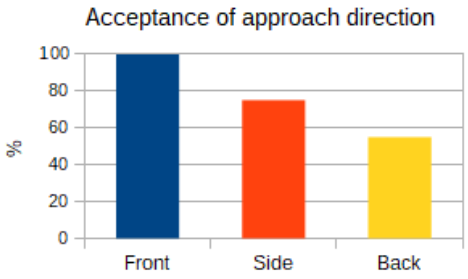

Figure 5.4. Acceptance of robot approaching from different sides.

All participants (100%) accepted to be approached by the robot from the front, where they would see it coming (Figure 5.4). Most of them (75%) also accepted to be approached from the side, while only about half (55%) would still accept it to approach from behind.

Approaches from the back were several times only seen feasible under the condition that the robot would issue a warning before. If the contact happens as a surprise it could lead to eventual stumbling over the robot. Some participants would also expect to be informed about such situations upfront e.g. with warning signs placed in the robot's operation area.

The dependency on the direction poses a considerable challenge for the robot's perception system. For approaching from the front, the robot in the presented form also had the practical issue that it could squeeze a foot under the robot's hull. In this version of the robot the hull had some clearance in order to be able to drive on ramps which on the other hand does allow small objects to get below the edge of the hull.

5.3.5. Expected types of warning before contact

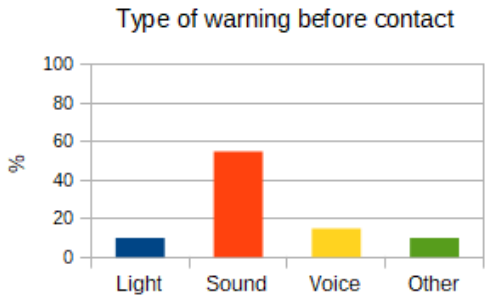

Figure 5.5. Expected types of warning before a contact.

The majority of participants (75%) stated that they would expect a warning signal by the robot before it gets into direct contact (Figure 5.5). The by far most often (55%) requested modality is making some sound, while voice (15%) and light (10%) seemed less helpful.

Sound was estimated to be crucial in particular when the robot does not approach from the front. In that case lights would probably not be seen at all. Three participants noted that the sound should be similar to that of a car. A system similar to an Acoustic Vehicle Alerting Systems (AVAS) could be thought of here, like it is required for electric vehicles in the EU. The cuyBot robot emits some noises due to ventilators and Lidar sensors inside, but when driving slowly it can hardly be heard in an environment with ambient noise like in the university hall. Beep sounds were explicitly noted to be undesired and annoying.

5.3.6. Concerns for other types of robots

The final question aimed to investigate the dependency on the robot type. Two other robots were shown, one in reality and one as a picture (Figure 5.6). The one shown as real had very distinct characteristics, a human-size UV-C disinfection robot developed at Locomotec that was placed at the side of the interviewers. A Pepper humanoid robot was shown in a picture.

Figure 5.6. Robots shown to participants: cuyBot used in experiment (left), UV-C robot shown in reality (centre), Pepper robot shown as picture (right).

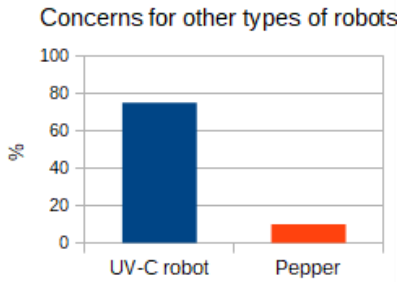

Figure 5.7. Number of concerns for other types of robots.

The majority of participants (75%) had strong concerns about having the same experience with the big UV-C robot (Figure 5.7). The main reason (60%) mentioned was the size, causing participants to feel scared and intimidated by it. They were also not sure if the robot would be safe to push, if something might get damaged or the robot even falls over. Furthermore, it would not be clear whether it might be dangerous to touch it at all, with open compartments for the UV-C lamps. The strong negative reaction towards the tall robot is somewhat in contrast to Walter (2008), who found the robot height not having a systematic influence on comfortable approach distances.

Concerning Pepper, the small size was highlighted as being important, which would leave the feeling of safety and having control over the situation; apparently assuming that the small size also implies smaller forces involved. Two issues mentioned were the danger of the robot being overlooked due to its size and the possible inconvenience of having to touch rather low.

5.4. Conclusion

The hypothesis of the experiment has been partly confirmed. The comfort level for situations with physical contact with the robot increased for most participants after they could experience the behaviour of the robot by themselves. For situations where the humans themselves would push the robot away the change was rather marginal.

Participants that felt familiar with robots showed a significant higher level of comfort. This result matches the observation in (Takayama and Pantofaru, 2009). Though the causes why the effect of the experiment led to a much bigger increase of comfort for these groups remain unclear.

Several participants stated that their evaluation is strongly linked to the experienced robot behaviour. That way the experiment conforms with (Horstman and Krämer, 2020) on the importance of the actual behaviour. Though in contrast also a noticeable relation to the background of the participants was observed.

The evaluations must be seen in context of the situation during the experiment. Only few people were present at university at those times due to the pandemic situation. The estimation of getting close to the robot could be quite different in a crowded environment where it is more apparent that closer distances might become unavoidable.

This experiment demonstrated that the current implementation of the compliant motion control on the cuyBot robot was safe when getting into contact also with inexperienced users. This observation was done under the prerequisite that the robot's velocity had been reduced before contact. Impacts with higher velocities would result in very different impact forces and reduce the acceptability considerably. The timely reduction of velocity must be ensured by appropriate sensing and environment perception, which was not focussed in this experiment.

Still there were some concerns about the demonstrated implementation. The continuous pushing of the robot was criticized, expecting a better situation awareness of the robot. Furthermore, the resistance of the robot was felt as too high several times, expecting that pushing the robot should be more or less effortless. This requires adapting the active compliance controller fusing with external sensors and will be part of ongoing work.

Although the number of only 20 participants hardly allows for generalized statements, this experiment provided hints on various aspects related to the motion of robots in densely crowded situations. Most notably the experiment suggests that the concept of robots getting into close contacts with humans may be accepted by the public as long as the functionality of compliant motion is implemented in a safe and intuitive way. Once persons gathered experience with the robot, on average they have a better than neutral attitude towards close proximity and physical contacts.

6. Conclusion

We have successfully evaluated each of our integrated Crowdbots: the smart wheelchair, Qolo, Pepper and cuyBot. These evaluations were both quantitative and qualitative and ranged from testing the efficacy of core components that are developed within the project and underpin the Crowdbots — namely: sensing, localisation, simulation, planning — to fully integrated tests of each platform for some of the scenarios identified in D1.3 (Specification of Scenarios Requirements Update).

Our two semi-autonomous systems, the smart wheelchair and Qolo, have both shown promising results for the use of shared control in crowded environments. Both prototypes have been tested in controlled and uncontrolled indoor and outdoor environments. In these evaluations, we have shown that it is feasible to navigate around pedestrian crowds, demonstrating the controller's compatibility with the behaviour of pedestrians. With over 100 trials in sparse crowds, crowd flows, and mixed crowds, we did not experience any incidents, or negative comments from the bystanders. We have also shown how a surrogate user can be trained by GAIL that is able to learn the driving style from the collected user data and this has been validated in simulation, although translation to the physical robots still remains a challenge.

Evaluations of our two autonomous Crowdbots have demonstrated the feasibility of both navigation capability in dynamic human environments, as well as the safety and reasonable acceptance of our compliant motion control when making physical contact with pedestrians. However, to maintain acceptance in these situations, the timely reduction of velocity must be ensured through appropriate sensing and environment perception.

References

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. InProceedings of the Twenty-First International Conference on Machine Learning, ICML'04, page 1, New York, NY, USA, 2004. Association for Computing Machinery. ISBN1581138385. doi: 10.1145/1015330.1015430. URL https://doi.org/10.1145/1015330.1015430.

W. Amanhoud, M. Khoramshahi, M. Bonnesoeur, and A. Billard, "Force Adaptation in Contact Tasks with Dynamical Systems," in *IEEE International Conference on Robotics and Automation, ICRA-2020*, 2020, no. March, pp. 6841–6847, doi: 10.1109/icra40945.2020.9197509.

B. Cèsar-Tondreau, G. Warnell, E. Stump, K. Kochersberger, and N. R. Waytowich. Improving autonomous robotic navigation using imitation learning.Frontiers in Robotics and AI,8, 2021.

F. Grzeskowiak, D. Gonon, D. Dugas, D. Paez-Granados, J. J. Chung, J. Nieto, R. Siegwart,A. Billard, M. Babel, and J. Pettré. Crowd against the machine: A simulation-basedbenchmark tool to evaluate and compare robot capabilities to navigate a human crowd. InICRA 2021 - IEEE International Conference on Robotics and Automation, pages 3879–3885, Xian, China, May 2021. IEEE. URL https://hal.archives-ouvertes.fr/hal-03206221.

D. J. Gonon, D. Paez-Granados, and A. Billard, "Reactive Navigation in Crowds for Nonholonomic Robots with Convex Bounding Shape," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4728–4735, 2021, doi: 10.1109/LRA.2021.3068660.

L. Huber, A. Billard, and J.-J. Slotine, "Avoidance of Convex and Concave Obstacles With Convergence Ensured Through Contraction," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1462–1469, 2019, doi: 10.1109/lra.2019.2893676.

K. Kronander and A. Billard, "Passive Interaction Control With Dynamical Systems," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 106–113, 2016, doi: 10.1109/LRA.2015.2509025.

J. Ho and S. Ermon. Generative adversarial imitation learning, 2016.

S. Javdani, S. S. Srinivasa, and J. A. Bagnell. Shared autonomy via hindsight optimization,2015.

A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer. Imitating driver behavior withgenerative adversarial networks, 2017.

R. V. Levine and A. Norenzayan. The pace of life in 31 countries.Journal of Cross-Cultural Psychology, 30(2):178–205, 1999. doi: 10.1177/0022022199030002003. URLhttps://doi.org/10.1177/0022022199030002003.

S. Reddy, A. D. Dragan, and S. Levine. Shared autonomy via deep reinforcement learning,2018.

C. Schaff and M. R. Walter. Residual policy learning for shared autonomy, 2020.

J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization, 2017a.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimiza-tion algorithms, 2017b.

T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling. Residual policy learning, 2019.

P. Trautman, J. Ma, R. M. Murray, and A. Krause. Robot navigation in dense human crowds:Statistical models and experimental studies of human–robot cooperation.The InternationalJournal of Robotics Research, 34(3):335–356, 2015.

J. van den Berg, S. Guy, M. Lin, and D. Manocha.Reciprocal n-Body Collision Avoid-ance, volume 70, pages 3–19. 04 2011. ISBN 978-3-642-19456-6. doi: 10.1007/978-3-642-19457-3_1.

B. Zhang, C. Holloway, and T. Carlson. A hierarchical design for shared-control wheelchairnavigation in dynamic environments. In2020 IEEE International Conference on Systems,Man, and Cybernetics (SMC), pages 4439–4446, 2020. doi: 10.1109/SMC42975.2020.9282838.

B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, et al. Maximum entropy inversereinforcement learning. InAaai, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, “From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots,” in 2017 IEEE international conference on robotics and automation (icra). IEEE, 2017, pp. 1527–1533.

M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, “Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations,” IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 4423–4430, 2018.

Y. F. Chen, M. Liu, M. Everett, and J. P. How, “Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning,” in 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 285–292.

C. Chen, Y. Liu, S. Kreiss, and A. Alahi, “Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning,” in 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 6015–6022.

L. Q. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, “Robot navigation in crowded environments using deep reinforcement learning,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, pp. 5671-5677.

T. Fan, P. Long, W. Liu, and J. Pan, “Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios,” The International Journal of Robotics Research, pp. 856-892, 2020.

D. Dugas, J. Nieto, R. Siegwart, and J. J. Chung, “IAN: Multi-behavior navigation planning for robots in real, crowded environments,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, pp. 11368-11375.

J. Van den Berg, M. Lin, and D. Manocha, “Reciprocal velocity obstacles for real-time multi-agent navigation,” in 2008 IEEE International Conference on Robotics and Automation. IEEE, 2008, pp. 1928–1935.

D. Ha and J. Schmidhuber, “Recurrent world models facilitate policy evolution,” in Advances in Neural Information Processing Systems, 2018, pp. 2450–2462.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” arXiv preprint arXiv:1707.06347, 2017.

A. Hill, A. Raffin, M. Ernestus, A. Gleave, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert et al., “Stable baselines,” GitHub repository, 2018.

M. Everett, Y. F. Chen, and J. P. How, “Motion planning among dynamic, decision-making agents with deep reinforcement learning,” in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018, pp. 3052–3059.

P. Carreno-Medrano, et al., “Aligning Robot's Behaviours and Users' Perceptions Through Participatory Prototyping,” arXiv preprint arXiv:2101.03660, 2021.

M. A. De Graaf and S. B. Allouch, “Exploring influencing variables for the acceptance of social robots,” Robotics and Autonomous Systems 61, 1476–1486, 2013.

A. C. Horstmann and N.C. Krämer, “Expectations vs. actual behavior of a social robot: An experimental investigation of the effects of a social robot’s interaction skill level and its expected future role on people’s evaluations,” Frontiers in Psychology 10, 939, 2020.

J. Mumm and B. Mutlu, “Human-robot proxemics: Physical and psychological distancing in human-robot interaction.” 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp. 331-338, 2011.

D. Paez-Granados, V. Gupta, and A. Billard, “Unfreezing Social Navigation : Dynamical Systems based Compliance for Contact Control in Robot Navigation,” in *[Under review] IEEE International Conference on Robotics and Automation (ICRA)*, 2022, vol. 1, no. 1, pp. 1–7, [Online]. Available: https://youtu.be/y7D-YeJ0mmg%0Ahttp://infoscience.epfl.ch/record/287442?&ln=en .

D. Paez-Granados, M. Hassan, Y. Chen, H. Kadone, and K. Suzuki, “Personal Mobility with Synchronous Trunk-Knee Passive Exoskeleton: Optimizing Human-Robot Energy Transfer,” *IEEE/ASME Trans. Mechatronics*, vol. 1, no. 1, pp. 1–12, 2022, doi: 10.1109/TMECH.2021.3135453.

L. Takayama and C. Pantofar, “Influences on proxemic behaviors in human-robot interaction,” IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009.

M. L. Walters, “The Design Space for Robot Appearance and Behaviour for Social Robot Companions”, University of Hertfordshire Dissertation, 2008

Y. Chen, D. Paez-Granados, H. Kadone, and K. Suzuki, “Control interface for hands-free navigation of standing mobility vehicles based on upper-body natural movements,” in *IEEE International Conference on Intelligent Robots and Systems*, 2020, pp. 11322–11329, doi: 10.1109/IROS45743.2020.9340875.