Benchmarking Multi-Robot Coordination in Realistic, Unstructured Human-Shared Environments

Lukas Heuer[1], Luigi Palmieri[2], Anna Mannucci[2], Sven Koenig[3] and Martin Magnusson[1]

***Abstract*— Coordinating a fleet of robots in unstructured, human-shared environments is challenging. Human behavior is hard to predict, and its uncertainty impacts the performance of the robotic fleet. Various multi-robot planning and coordination algorithms have been proposed, including Multi-Agent Path Finding (MAPF) methods to precedence-based algorithms. However, it is still unclear how human presence impacts different coordination strategies in both simulated environments and the real world. With the goal of studying and further improving multi-robot planning capabilities in those settings, we propose a method to develop and benchmark different multi-robot coordination algorithms in realistic, unstructured and human-shared environments. To this end, we introduce a multi-robot benchmark framework that is based on state-of-the-art open-source navigation and simulation frameworks and can use different types of robots, environments and human motion models. We show a possible application of the benchmark framework with two different environments and three centralized coordination methods (two MAPF algorithms and a loosely-coupled coordination method based on precedence constraints). We evaluate each environment for different human densities to investigate its impact on each coordination method. We also present preliminary results that show how informing each coordination method about human presence can help the coordination method to find faster paths for the robots.**

I. Introduction

Multi-robot systems have many potential applications. Currently, the biggest application is warehouse automation, where the storing or sorting of many different objects is enabled by large multi-robot systems [1]. However, there are applications and environments where multi-robot systems are not yet common. Examples include airports, hospitals, restaurants or loading docks in logistics [2]–[4]. Such environments pose two major challenges when employing multiple robots. First, they do not always have a fixed and simple structure. Semi-static obstacles and non-regular corridors or intersections can make it harder employ efficient coordination algorithms. Second, they are often not devoid of humans. This demands the fleet decision-making system to account for the delays of individual robots that can occur when interacting with humans.

[1]L. Heuer, and M. Magnusson are with the Center for Applied Autonomous Sensor Systems (AASS), Örebro University, Sweden {lukas.heuer, martin.magnusson}@oru.se.

[2]L. Palmieri and A. Mannucci are with Robert Bosch GmbH, Corporate Research, Stuttgart, Germany {luigi.palmieri, anna.mannucci}@de.bosch.com.

[3]S. Koenig is with the Department of Computer Science, University of Southern California, Los Angeles, United States of America skoenig@usc.edu.

This work was partly supported by the EU Horizon 2020 research and innovation program under grant agreement No. 101017274 (DARKO) and NSF grant 1837779.

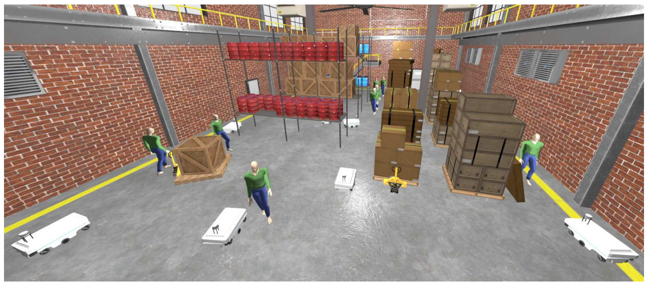
Fig. 1: The *depot* environment, which we use in our framework to test different multi-robot planning algorithms. We focus our attention on realistic, human-shared environments.

The literature on autonomous navigation consists of two main branches: single-robot navigation and multi-robot coordination/path-planning. Single-robot navigation [5]–[8] usually accounts for the robot dynamics and can navigate in cluttered dynamic environments, but may lead to deadlocks or suboptimal behavior when applied to complex multi-robot settings (such as fleets of robots with non-trivial kinematics, limited space for maneuvering, etc.). Multi-robot coordination/path-planning investigates the explicit coordination of robots. It requires a certain level of communication between the robots and optimizes joint metrics, such as the total makespan or the total distance traveled [1], [9], [10]. However, to keep the problem tractable for hundreds of robots, those approaches are often developed and evaluated in strongly simplified settings without considering robot dynamics, uncertainty or complex environments [11], [12]. Thus, each research direction usually simplifies or disregards the problem that the other one is trying to solve. Understanding how to effectively couple the two branches of autonomous navigation is the problem which motivates this paper.

In particular, coordinating multi-robot systems in unstructured human-shared environments introduces challenges. Such environments are difficult to simplify because of the inherent complexity of human behavior and the way humans can interact with a robot. The local navigation system of the robots determines how they interact with the surrounding environment, including humans and other robots. But to what extent this influences the coordination of the multi-robot fleet is unclear as different coordination methods make different assumptions or simplifications. Thus, the local robot behaviors can impact each coordination method differently.

Schäfer et al. [13] proposed a benchmark to investigate multi-robot coordination in realistic simulations. However, to the best of our knowledge, there is no research on multi-robot coordination focusing on the challenges introduced by the presence of humans in unstructured environments.

With this work, we provide an important initial stepping stone to understanding how human behavior can influence multi-robot coordination. To this end, we make the following contributions:

1) A modular benchmarking framework based on state-of-the-art physics simulator, robot-navigation stack and multi-robot coordination algorithms tailored towards human-shared spaces. The framework allows researchers and practitioners to easily implement and compare different methods for robot navigation and multi-robot coordination.
2) Experimental results which show how different established algorithms for multi-robot coordination perform in, and are impacted by, unstructured, human-shared environments.

II. Related Work

Recently, more methods for multi-robot coordination aim at explicitly dealing with delays in the execution of the robot paths. Hönig et al. [10] and Varambally et al. [11] identify a similar research gap as we have introduced in Section I. Both use the concept of Action Dependency Graphs (ADG) to make the execution of the solution to a Multi-Agent Path Finding (MAPF) problem more robust to delays and slow-downs. However, both papers consider only simple warehouses, do not include a local navigation stack in the simulation framework, nor consider human-shared environments. Cirillo, Pecora and Mannucci et al. published a line of work [12], [14], [15] which investigates realistic multi-robot coordination in unstructured environments. Similarly, Draganjac et al. [3] propose a traffic system for autonomous forklifts. These methods utilize the robots full navigation stack and is tested in physics-simulated environments as well as with real-world experiments, but do not address the topics of sharing the operational environment with humans. A method which shows how multi-robot coordination can be adapted to unstructured environments is proposed by Čáp et al. [9], [16]. Though, they also do not consider realistic robot dynamics and their work is not aimed at human-shared environments. Chen et al. propose a decentralized coordination method in [17] and Zhu et al. [18] introduce an approach for multi-robot motion planning while accounting for human interactions. However, both methods focus only on the decentralized coordination and path planning, and are evaluated in completely empty environments. Talebpour et al. [19] present a method for multi-robot coordination in human-shared spaces. They combine the local robot navigation stack with a task-allocation-based coordination method. Differently from our work, they do not present an open framework nor consider different methods for multi-robot coordination.

Sturtevant et al. [20] provide an established benchmark for MAPF algorithms. In Schäfer et al. [13], we aim to close the gap between realistic simulation and benchmarking MAPF algorithms. While the benchmark includes Navigation2[1] as a local navigation stack (and hence allows different path planners and controllers), it is rigid with respect to the type of coordination methods used and does not investigate the effects of the human presence.

[1]https://navigation.ros.org

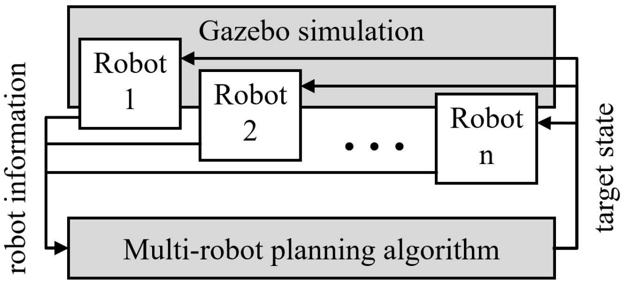

Fig. 2: A sketch of our designed REMROC framework. The arrows represent information, communicated via ROS 2 messages. This information can be anything, required for coordination (e.g. state-, path-, or sensor-information).

We conclude that there exists no related work that focuses on how to directly compare different types of multi-robot coordination methods like MAPF-based solutions [1], [9], free-space coordination based on precedence constraints [12], or market-based coordination methods [21]. The work on coordination in human-shared environments is also very limited and there are no studies about how traditional coordination methods are effected by them.

III. Framework

In this section, we detail `REMROC`: a framework for benchmarking realistic multi-robot coordination algorithms in unstructured, human-shared environments. (Figure 2 illustrates the overall design of the framework.) We use ROS 2 Humble [22] as the software basis for the general robotic system, and Gazebo Ignition [23] as physics simulator for the environments. For navigating each individual robot, we use Navigation2 [24], a ROS 2 software library containing state-of-the-art components for robot navigation. The use of open-source state-of-the-art software components aims at promoting the framework as a general open-source environment for benchmarking multi-robot systems. Also, it eases the development of algorithms for real-world application and transferring them to real robotic platforms. The framework is available open source at `https://github.com/boschresearch/remroc`.

A. Simulation

The general environment including obstacles and robot descriptions are imported into Gazebo via SDF files. Gazebo offers the functionality of adding simulated humans to its environment, also known as actors. Actors are also specified in the SDF file and require trajectory information (i.e., allowing the reproduction of real-world datasets like THÖR [25]). Importantly, actors in our Gazebo simulations have a texture but no collision box. In other words, humans are registered by sensors like lidars and cameras and therefore influence the navigation stack of the robot realistically, but are not physically interacted with. We make this choice because physically interacting humans would require them to use some form of navigation stack themselves in order to react to the robots. Otherwise it could easily happen that a human influences the simulation very unrealistically, for example by walking into a robot continuously. Providing the simulated humans with a navigation stack on their own would greatly increase the complexity of the simulation to the extent where it would not be feasible to simulate more then a few humans.

It is generally possible to add different types of robots, with different models, sensors, and motion dynamics, and even create multi-robot systems composed of multiple types. The robot we have implemented for our evaluation is using the open source model of a medium size delivery robot [26]. We made modifications to equip the model with a differential-drive motion model, a 3D lidar and an IMU sensor.

B. Navigation

Each robot is launched in its individual ROS 2 name-space with a full Navigation2 stack, which consists of multiple components. For state-estimation and localization we employ an Extended Kalman Filter (EKF) [27] and an Adaptive Monte Carlo Localization (AMCL) [28]. We use Model Predictive Path Integral (MPPI) [29] and Hybrid-A* [30] for local control and global planning respectively. The robots also run an instance of the Navigation2 behavior-tree to expose ROS 2 action servers for general functionality, like point navigation or path following.

C. Planning and Coordination

The Multi-Robot Planning Algorithm (MRPA) unit (see Figure 2) uses the received information to coordinate, and ultimately instructs the individual robots on how to move forward. It is implemented as a ROS 2 node and connected to the robots through ROS 2 interfaces (i.e. topics, services, actions). Our framework can also realize hybrid or decentralized coordination methods by launching multiple coordination nodes which only connect to individual, or a subset of robots.

Importantly, this overall setup is computational demanding, as each robot simulates sensors and runs an individual Navigation2 stack. In our evaluation, run experiments with up to 8 robots and 20 humans maintaining a 60% real-time factor for the Gazebo simulation. We plan to address this in future work by running the simulation and navigation systems distributed on multiple machines.

IV. Coordination Algorithms

Thanks to its modularity, REMROC allows for easy integration of different coordination algorithms. As an initial set, we have chosen established approaches to coordination [12], [31], [32], included them into our framework, and benchmarked them in unstructured, human-shared environments.

Many approaches for multi-agent coordination represent the coordination task as a MAPF problem and employ a MAPF solver to obtain a solution [31]. We adopt this strategy and use Conflict Based Search (CBS) [32] in two of our evaluated algorithms. For this, we adapt the CBS implementation from *libMultiRobotPlanning* [33] to use in a ROS 2 node.

Algorithms 1, 2 and 3 describe the coordination algorithms we include in our benchmark framework. We made straightforward additions and modifications to [32] and [12], to enable their use in REMROC. R, S, G denote the sets of all robots, their start and goal positions, respectively. P refers to the set of all robot paths. Lower case letters represent the elements of the respective set. Limitations of the algorithms are discussed in Sections VI-A.1, VI-A.2 and VI-A.3.

We refer to Algorithm 1 as *One-shot CBS*. The start and goal locations of the robots are used to generate a MAPF problem on the respective navigation graph, and solve it using the CBS algorithm. Translating the nodes of the navigation graph into map coordinates, we convert the solution of the CBS algorithm into a set of waypoints along which the robot has to navigate. Importantly, the obtained paths are only collision free and optimal with respect to the navigation graph, if the robots move synchronously. To enforce this, the robots only receive their next target waypoint when all of them have reached their current target respectively. We use the */navigate_to_pose* service, provided by Navigation2, to set the target waypoint.

We refer to Algorithm 2 as *Iterative CBS*. This algorithm is practically CBS with replanning at a set interval, similar to Method 2 in [1]. The perfect synchronization assumption of the classic CBS algorithm is difficult to meet due to dynamic constraints of the robots, networking issues and interactions with humans delaying individual robots. This algorithm tries to account for these disturbances to the original MAPF solution by generating and solving the MAPF problem iteratively. The positions of the robots are updated and a MAPF problem is generated on the respective navigation graph, using the robots goal states and their current positions. The MAPF problem is then solved using the CBS algorithm. We consider the resulting paths as a sequence of waypoints, which we give to the robot, to navigate along. This is done by calling the */navigate_through_poses* action, provided by the Navigation2 stack.

We refer to Algorithm 3 as *Continuous PBC (Priority Based Coordination*. This algorithm is inspired by the one proposed in [12] and [34]. The algorithm first queries the global planner from each robot to plan a path. The original paths are saved and used as bases for the following iterative part of the algorithm. First we receive the current state of each robot. Second we calculate a critical point, that is, the first point on the remaining path closer then a given critical distance to the path of another robot. We then truncate the original path to the subpath between the current robot state and the critical point on which the robot does not have precedence. Finally, we call the */follow_path* action, provided by Navigation2 to make the robot follow its respective subpath. This algorithm requires a heuristic to assign precedence to a robot at a critical point. We consider the robot with less path distance to the critical point to have priority. If the robot's current position is too close to another path, its distance to a critical point is 0 and it therefore always has priority.

V. Evaluation

The REMROC framework, described in Section III, is used to evaluate different multi-robot coordination algorithms in

Algorithm 1 One-shot CBS

```
Require: $R, S, G$    ▷ robots, robot start positions, robot goal positions
 1: $P \leftarrow \mathrm{CBS}(S, G)$
 2: while RobotsNotAtGoal($R$) do
 3:     for $r \in R$ do
 4:         NavigateToWaypoint($r, \mathbf{p}_r[0]$)
 5:     end for
 6:     if RobotsAtWaypoint($R$) then
 7:         for $r \in R$ do
 8:             RemoveElement($\mathbf{p}_r[0]$)
 9:         end for
10:     end if
11: end while
```

Algorithm 2 Iterative CBS

```
Require: $R, S, G$    ▷ robots, robot start positions, robot goal positions
 1: while RobotsNotAtGoal($R$) do
 2:     $X \leftarrow$ UpdateRobotPositions($R$)
 3:     $P \leftarrow \mathrm{CBS}(X, G)$
 4:     if AllPathsValid($R$) then
 5:         for $r \in R$ do
 6:             NavigateAlongWaypoints($r, \mathbf{p}_r$)
 7:         end for
 8:     else
 9:         StopRobots($R$)
10:     end if
11: end while
```

Algorithm 3 Continuous PBS

```
Require: $R, S, G$    ▷ robots, robot start position, robot goal positions
 1: for $r \in R$ do
 2:     $\mathbf{p}_r \leftarrow$ GlobalPlanner($\mathbf{s}_r, \mathbf{g}_r$)
 3: end for
 4: while RobotsNotAtGoal($R$) do
 5:     $X \leftarrow$ UpdateRobotPositions($R$)
 6:     for $r \in R$ do
 7:         $\mathbf{c}_r \leftarrow$ FindCriticalPoint($\mathbf{p}_r, P \backslash \mathbf{p}_r$)
 8:         $\tilde{\mathbf{p}}_r \leftarrow$ FindRelevantSubpath($\mathbf{x}_r, \mathbf{c}_r, \mathbf{p}_r$)
 9:         FollowPath($r, \tilde{\mathbf{p}}_r$)
10:     end for
11: end while
```

unstructured, human-shared environments. We evaluate three algorithms in two different environments, detailed in Section IV and Section V-A respectively.

The framework provides the possibility to record many different information like the odometry of the robots, execution frequencies of the individual components or direct sensor data, and to define metrics of interest. Our goal is to derive general insights on how the presence of the humans affects the performance of the different algorithms. Thus, we collect the individual time-to-goal of the robots and present the results in Section VI. We obtain legible trajectories for the humans by using the Hybrid A* planner from Navigation2. To avoid biases caused by a specific selection of human paths, we generate n samples with randomized human trajectories for each experiment.

For the two CBS-based algorithms, we use the navigation graphs in Figure 4 and 5 to represent the environments.

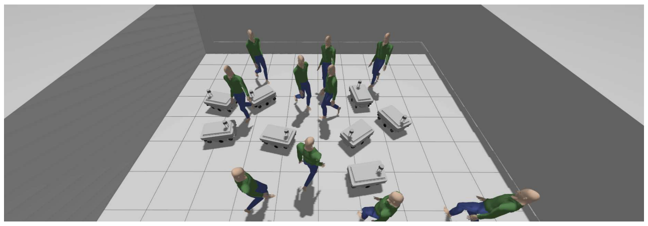
Fig. 3: The *basic* environment used to test different multi-robot planning algorithms with up to 10 humans.

A. Simulation Environments

We evaluate the coordination algorithms in two different environments, each with 8 robots navigating to their given goal position. Start and goal positions are fixed for each environment as shown in Figures 4 and 6 and chosen so as to result in a well-formed infrastructure as described in [9], [34]. The first simulation environment, called *basic*, is an empty room of 10×10 m^2. Start and goal positions for the robots and the navigation graph are shown in Figure 4 and chosen so as to provoke a congested area in the center of the room. Due to the lack of static obstacles, the main disturbance on the robots coordination is caused by the presence of humans.

The second environment, called *depot*, is based on the Turtlebot 4 Gazebo simulator [35] and measures 30×15 m^2. Figures 5 and 6 show the navigation graph and occupancy map including the start and goal positions of robots. Blue areas in Figure 6 mark shelves; these have enough ground clearance for robots to pass beneath but not for humans. This is an interesting feature because it results in an area with a higher average human density (marked in red). This environment contains obstacles and features making it more realistic than the basic one. The size of the environment eases the coordination problem, but the increased complexity challenges the local navigation stack of the robots.

B. Experiments

In our experiments we investigate how the different algorithms, described in Section IV, perform when coordinating several robots in human-shared environments. For this we look at the environments presented in Section V-A in three settings, each having a different number of humans present. We conduct experiments in the *basic* environment, with 0, 5 and 10 humans, and a sample size of $n = 20$. The experiment in the *depot* environment is done with 0, 10 and 20 humans, and a sample size of $n = 10$. In a last experiment, we only consider the 10 samples of the *depot* environment with 20 humans, and modify the navigation graph by truncating nodes which are in the region with high human density. We compare the normal environment setup to one where the crowded (e.g. busy) area, marked **red** in Figure 6, is blocked. This means that the area is treated as occupied for the coordination methods and the navigation stacks, resulting in paths which do not pass through this area. The local planner always performs collision avoidance considering both humans and static obstacles.

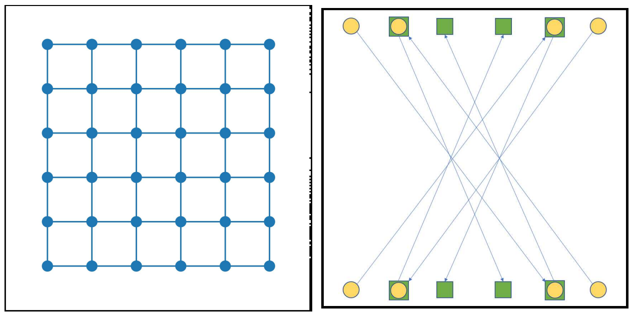

Fig. 4: **(Left:)** The navigation graph used by MAPF based coordination algorithms for the *basic* environment. **(Right:)**Start and goal locations for the robots, marked with **yellow** and **green** respectively.

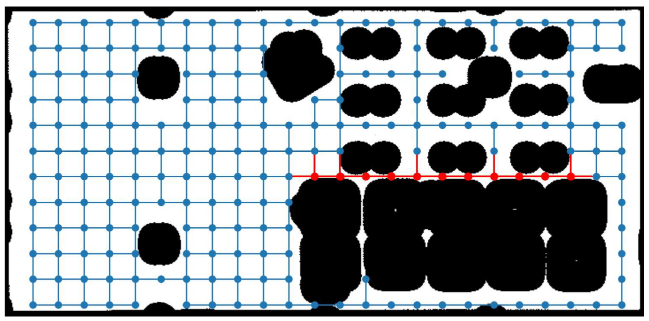

Fig. 5: The inflated occupancy map of the *depot* environment. The navigation graph used by MAPF based coordination algorithms is shown in **blue**. **Red** nodes are used in the normal evaluation, but are removed in the last experiment.

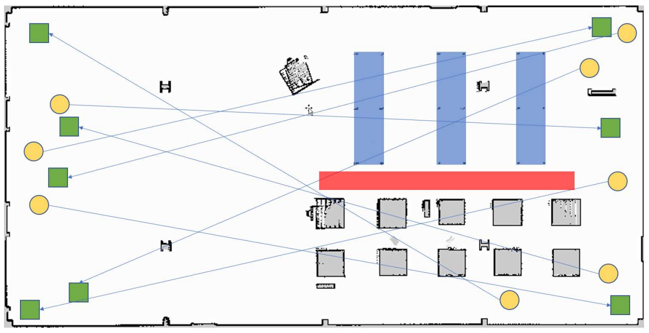

Fig. 6: Occupancy map for the *depot* environment. The shelves, marked in **blue**, prevent humans from passing. The **red** busy area accumulates a lot of human traffic and is blocked for robot navigation in our last experiment. Start and goal locations for the robots are marked with **yellow** and **green** respectively.

VI. Results

Results for the *basic* environment are reported in Table I and Figure 7. Table I shows mean and standard deviation of the time-to-goal over the 20 samples and all robots during the respective experiments. Figure 7 reports results for each robot over the 20 samples. Importantly, the results from each sample are sorted, so the 8 markers show the time-to-goal for the robots from fastest to slowest. Table II and Figure 8 show results of the *depot* environment. Table II shows mean and standard deviation of the time-to-goal over the 10 samples and all robots during the respective experiments. Figure 8 displays the results for the individual robots. For the third experiment, we inform the algorithms about the area with an higher average in human density (by changing the navigation graph) and show the results in Figure 9.

No. humans	One-shot CBS	Iterative CBS	Continuous PBC
0	31.7 ± 3.64	29.9 ± 6.73	23.7 ± 5.17
5	39.4 ± 5.00	33.8 ± 7.53	27.1 ± 6.91
10	47.7 ± 6.58	35.2 ± 8.46	29.4 ± 8.31

TABLE I: Average time-to-goal in seconds for the *basic* environment.

No. humans	One-shot CBS	Iterative CBS	Continuous PBC
0	110.9 ± 10.61	65.1 ± 7.99	60.2 ± 10.32
10	120.0 ± 10.81	68.1 ± 8.36	63.0 ± 11.3
20	127.9 ± 13.74	70.4 ± 9.10	65.0 ± 11.6
20 (BB*)	118.6 ± 10.90	69.5 ± 8.82	64.4 ± 11.7

TABLE II: Average time-to-goal in seconds for the *depot* environment. *BB: Busy blocked

A. Discussion

The results show that the presence of humans during fleet coordination, and considering them only in local collision avoidance, has a negative impact on the time-to-goal of the robots. Figure 7 and 8 as well as Table I and II show this clearly for both environments. The results also show the advantage of algorithms which continuously update their solutions. Table I and II show that, the iterative methods, Iterative CBS and Continuous PBC, are not impacted as much by the presence of humans, with the increase in time-to-goal not being as large as for One-shot CBS. Figure 8 shows that there is an increase in spread from the fastest to the slowest robot, and that the robots are generally delayed by adding more humans. The larger size of the *depot* environment is also important when evaluating the results. The longer distances could make delays less noticeable and give the iterative coordination methods more opportunities to recover from delays. Figure 8 shows that the fastest robot are not slowed down when increasing the number of humans. This is most likely because the coordination algorithm is able to leverage the delay of some robots into more direct and therefore faster paths for others.

Avoiding crowded areas in MRPA: Table II and Figure 9 show that the performance of coordination methods can be improved slightly by making the robots avoid the area with highest human density. For One-shot CBS, this speeds up the robots, while for the iterative coordination methods the improvements are marginal. However, intuitively it should be more difficult for the robots to coordinate, as the navigation graph and the environment are restricted. It seems plausible that the added challenge of coordinating in a more restricted environment, is compensated by a reduction in human interaction. Continuous PBC is generally less effected by human presence. This is most likely due to the fact that the method can plan continuous and straight paths to the goal states without being bound by the connected grid. From qualitative evaluation of our experiment, we observe that this coordination method is more likely to plan diagonally below the shelves, which results in less interactions with humans.

The different algorithms have individual drawbacks and limitations which we discuss in the following.

1) One-shot CBS: In the *depot* environment, this approach is notably slower then the other ones. We theorize that this is

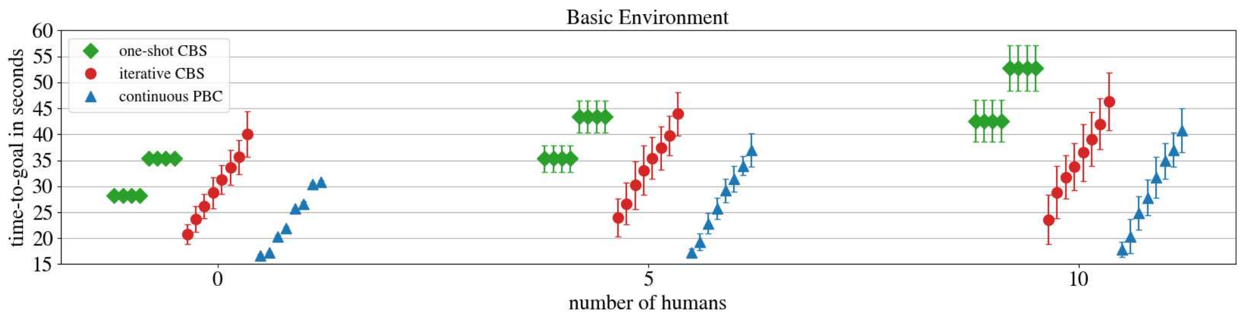

Fig. 7: Mean and standard deviation of time-to-goal in seconds for all robots in the *basic* experiment.

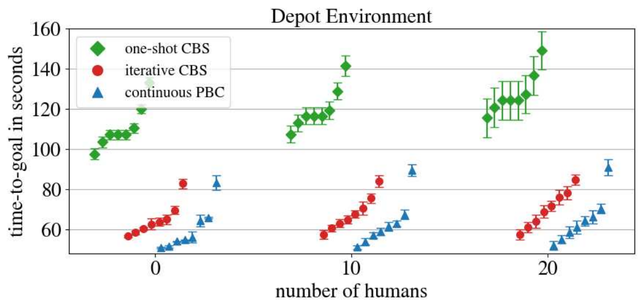

Fig. 8: Mean and standard deviation of time-to-goal in seconds for all robots in the *depot* environment.

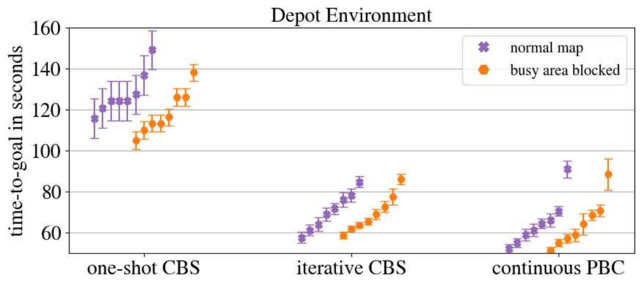

Fig. 9: Mean and standard deviation of time-to-goal in seconds for all robots in the *depot* environment with 20 humans. We also evaluate a modified version of the environment, in which we block the busy area of the map for the navigation and multi-robot path planning methods.

because it can not follow straight paths as efficiently as the other approaches, which is relevant in the larger environment. The dense navigation graph causes a stop and go behavior. This is due to the robot never fully accelerating, as it is always presented with a close goal. These drawbacks may be mitigated with more complex CBS-bases algorithm as proposed by Varambally et al. [11]. In addition, this method is not able to account for delays of individual robots and adapt the MAPF solution accordingly. However, it is still very robust. Long computation time or poor navigation performance will never cause deadlocks or undesirable behavior. This is because Alg. 1, lines 6–8, enforces synchronous execution of the MAPF solution. Thus, it only slows all robots down, and increase their respective time-to-goal.

2) Iterative CBS: While this algorithm works well in our experiments it has a notable limitation. The solving time per iteration should be low enough such that the robot will not pass more then one graph node during it. Otherwise the algorithm will not obtain valid paths and stop the robots until it has solved the MAPF iteration. While this was no problem in our experiments, using more robots or more difficult environment would increase the computational complexity and the time needed to obtain a solution to the MAPF problem. The exact influence of this phenomenon depends directly on the MAPF solver used. A sub-optimal solver like ECBS [36] or M* [37] could enable this algorithm for larger maps, robot fleets or more constrained navigation graphs.

3) Continuous PBC: This algorithm is able to plan in continuous space without a navigation graph, which improves its performance over the CBS based algorithms. However, it also generates the global paths with a single-agent motion planner and therefore generates sub-optimal multi-agent motion plans. The algorithm is also not minimizing total time-to-goal, because of the heuristic used to compute precedence. This makes it possible to improve the overall performance of the entire system by delaying a robot. The small impact in overall time-to-goal that humans have on this algorithm in the *depot* environment could be explained by this.

B. Future Research

In the future, we plan to build on our contributions by extending the framework and experimental evaluation. We aim to include more representative and diverse environments and robot models, and implement additional state-of-the-art coordination algorithms. We also want to investigate the use of local navigation methods which utilize human motion prediction and provide more extensive experimental evaluation. This will be focused on how local navigation methods can be combined with more complex coordination algorithms and be used in unstructured, human-shared environments.

VII. Conclusions

In this paper we make important initial steps towards applying methods for multi-robot coordination in unstructured human-shared environments and identify the challenges in making state-of-the-art methods human-aware. To address these challenges we propose a software framework, based on state-of-the-art robotics research tools, to develop, evaluate and benchmark different types of multi-robot coordination methods in unstructured, human-shared environments. We use our framework to implement basic, well established multi-robot coordination algorithms and provide a line of experiments to test how they perform in those environments. Our results show that we can quantify the influence of human presence on the implemented algorithms and how basic ways of making the methods human-aware can influence the coordination performance.

References

[1] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," in *Proc. of Conf. on Artificial Intell. (AAAI)*, vol. 35, no. 13, 2021, pp. 11 272–11 281.

[2] S. Jeon, J. Lee, and J. Kim, "Multi-robot task allocation for real-time hospital logistics," in *Proc. of the IEEE Conf. on Systems, Man, & Cybernetics (SMC)*. IEEE, 2017, pp. 2465–2470.

[3] I. Draganjac, T. Petrovic, D. Miklic, Z. Kovacic, and J. Orsulic, "Highly-scalable traffic management of autonomous industrial transportation systems," *Robot. & Computer-Integr. Manuf.*, vol. 63, p. 101915, 2020.

[4] T. Morita, N. Kashiwagi, A. Yorozu, H. Suzuki, and T. Yamaguchi, "Evaluation of a multi-robot cafe based on service quality dimensions," *The Review of Socionetwork Strategies*, vol. 14, pp. 55–76, 2020.

[5] R. Han, S. Chen, and Q. Hao, "Cooperative multi-robot navigation in dynamic environment with deep reinforcement learning," in *Int. Conf. Robot. & Autom. (ICRA)*. IEEE, 2020, pp. 448–454.

[6] J. R. Bruce and M. M. Veloso, "Safe multirobot navigation within dynamics constraints," *Proc. of IEEE*, vol. 94, no. 7, pp. 1398–1411, 2006.

[7] B. Gopalakrishnan, A. K. Singh, M. Kaushik, K. M. Krishna, and D. Manocha, "Prvo: Probabilistic reciprocal velocity obstacle for multi robot navigation under uncertainty," in *Int. Conf. Intell. Robot. Sys. (IROS)*. IEEE, 2017, pp. 1089–1096.

[8] M. Boldrer, A. Antonucci, P. Bevilacqua, L. Palopoli, and D. Fontanelli, "Multi-agent navigation in human-shared environments: A safe and socially-aware approach," *Robot. & Auton. Syst.*, vol. 149, p. 103979, 2022.

[9] M. Čáp, P. Novák, A. Kleiner, and M. Selecký, "Prioritized planning algorithms for trajectory coordination of multiple mobile robots," *IEEE Trans. on Automation Science & Engineering*, vol. 12, no. 3, pp. 835–849, 2015.

[10] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Persistent & robust execution of MAPF schedules in warehouses," *IEEE Robot. & Autom. Letters*, vol. 4, no. 2, pp. 1125–1131, 2019.

[11] S. Varambally, J. Li, and S. Koenig, "Which MAPF model works best for automated warehousing?" in *Proc. of Int. Symp. on Comb. Search*, vol. 15, no. 1, 2022, pp. 190–198.

[12] F. Pecora, H. Andreasson, M. Mansouri, and V. Petkov, "A loosely-coupled approach for multi-robot coordination, motion planning and control," in *Proc. of Int. Conf. Autom. Plan. & Sched. (ICAPS)*, vol. 28, 2018, pp. 485–493.

[13] S. Schaefer, L. Palmieri, L. Heuer, R. Dillmann, S. Koenig, and A. Kleiner, "A benchmark for multi-robot planning in realistic, complex and cluttered environments," in *Int. Conf. Robot. & Autom. (ICRA)*, 2023, pp. 9231–9237.

[14] M. Cirillo, F. Pecora, H. Andreasson, T. Uras, and S. Koenig, "Integrated motion planning and coordination for industrial vehicles," in *Proc. of Int. Conf. Autom. Plan. & Sched. (ICAPS)*, vol. 24, 2014, pp. 463–471.

[15] A. Mannucci, L. Pallottino, and F. Pecora, "Provably safe multi-robot coordination with unreliable communication," *IEEE Robot. & Autom. Letters*, vol. 4, no. 4, pp. 3232–3239, 2019.

[16] M. Čáp, J. Gregoire, and E. Frazzoli, "Provably safe and deadlock-free execution of multi-robot plans under delaying disturbances," in *Int. Conf. Intell. Robot. Sys. (IROS)*, 2016, pp. 5113–5118.

[17] Y. Chen, U. Rosolia, and A. D. Ames, "Decentralized task and path planning for multi-robot systems," *IEEE Robot. & Autom. Letters*, vol. 6, no. 3, pp. 4337–4344, 2021.

[18] H. Zhu, F. M. Claramunt, B. Brito, and J. Alonso-Mora, "Learning interaction-aware trajectory predictions for decentralized multi-robot motion planning in dynamic environments," *IEEE Robot. & Autom. Letters*, vol. 6, no. 2, pp. 2256–2263, 2021.

[19] Z. Talebpour and A. Martinoli, "Multi-robot coordination in dynamic environments shared with humans," in *Int. Conf. Robot. & Autom. (ICRA)*. IEEE, 2018, pp. 4593–4600.

[20] N. R. Sturtevant, "Benchmarks for grid-based pathfinding," *IEEE Trans. Comp. Intell. & AI in Games*, vol. 4, no. 2, pp. 144–148, 2012.

[21] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proc. of IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[22] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074

[23] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Int. Conf. Intell. Robot. Sys. (IROS)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[24] S. Macenski, F. Martin, R. White, and J. Ginés Clavero, "The Marathon 2: A navigation system," in *Int. Conf. Intell. Robot. Sys. (IROS)*, 2020.

[25] A. Rudenko, T. P. Kucner, C. S. Swaminathan, R. T. Chadalavada, K. O. Arras, and A. J. Lilienthal, "THÖR: Human-robot navigation data collection and accurate motion trajectories dataset," *IEEE Robot. & Autom. Letters*, vol. 5, no. 2, pp. 676–682, 2020.

[26] "Open-RMF demo," https://github.com/open-rmf/rmf_demos, accessed: 2023-05-27.

[27] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. of Basic Engin.*, vol. 82, no. 1, pp. 35–45, 03 1960.

[28] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo localization: Efficient position estimation for mobile robots," *Aaai/iaai*, vol. 1999, no. 343-349, pp. 2–2, 1999.

[29] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *Int. Conf. Robot. & Autom. (ICRA)*, 2016, pp. 1433–1440.

[30] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[31] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proc. of Int. Symp. on Comb. Search*, vol. 10, no. 1, 2019, pp. 151–158.

[32] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Art. Intell.*, vol. 219, pp. 40–66, 2015.

[33] W. Hönig, S. Kiesel, A. Tinka, J. Durham, and N. Ayanian, "Conflict-based search with optimal task assignment," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2018.

[34] A. Mannucci, L. Pallottino, and F. Pecora, "On provably safe and live multirobot coordination with online goal posting," *IEEE Trans. Robot. Autom. (TRO)*, vol. 37, no. 6, pp. 1973–1991, 2021.

[35] "Turtlebot4 packages," https://turtlebot.github.io/turtlebot4-user-manual/software/turtlebot4_common.html, accessed: 2023-05-09.

[36] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Proc. of Int. Symp. on Comb. Search*, vol. 5, no. 1, 2014, pp. 19–27.

[37] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *Int. Conf. Intell. Robot. Sys. (IROS)*. IEEE, 2011, pp. 3260–3267.