Planning handovers involving humans and robots in constrained environment

Jules Waldhart[1,2], Mamoun Gharbi[1,2], Rachid Alami[1,3]

***Abstract*— Exchanging objects with humans through handovers is a key feature for any robot operating side by side with humans. The studies on the topic tackle various problems, such as the hand dexterity, the communication cues, the arms motions, the forces, the head motions, but most of them consider a handover as an independent action, decorrelated from the plan it is part of.**

In this paper, we consider situations where it might be necessary (or preferable) to achieve several handovers in order to transfer an object from one agent to another one. This problem complexity grows accordingly with the number of agents that might be involved in the task, making a classical approach under efficient. We propose a graph-based approach enabling a fast computation of a solution taking into account different parameters linked to the humans comfort and preferences. An abstract model of the task is also used as a heuristic to guide the search in the graph, a search which is performed with a Lazy Weighted A*. The method computes which agents (human or robot) to use and where handovers should be performed. It also computes motion plans for each robot, ensures that humans can reach handover places and preserves comfort of human partners by reducing, for instance, their efforts.

The method has been implemented and is demonstrated on various simulated multi-agents environments and on our two PR2 robots interacting with two humans.

I. Introduction

The challenge of bringing humans and robots to "coexist" in the same environment is commonly addressed in the literature. One of the difficulties is to enable robots to perform handovers, which is a key feature of human-robot collaboration. During the last decade, studies about when, how and where to perform a handover have been conducted ([1]–[7]).

In this paper we address a related problem which is a transport task, that may involve several agents (robots or humans). The transport problem is defined as an object to be moved from an agent to an other specific one, with the help of several agents present in the environment. The solution is a series of handovers between some agents. Fig. 1 shows an example of this problem, the agents are separated by a wall where windows and counters allow objects exchanges.

This transport problem involves several decisions such as which agents to use and where to perform the handovers.

*This work was conducted within the EU SAPHARI project funded by the E.C. division FP7-IST under contract ICT-287513.

[1]Jules Waldhart, Mamoun Gharbi and Rachid Alami are with CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France `firstname.lastname at laas.fr`

[2]Jules Waldhart and Mamoun Gharbi are with Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France

[3]Rachid Alami is with Univ de Toulouse, LAAS, F-31400 Toulouse, France

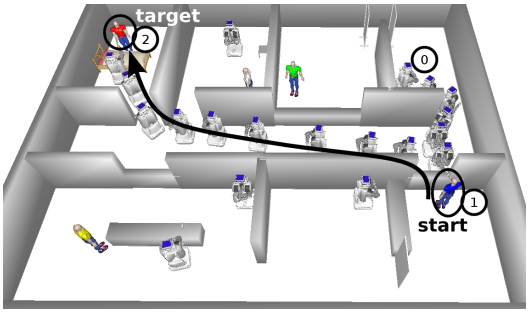

Fig. 1. An example in an environment where agents are separated into two navigable zones linked by windows and counters. The blue human is in possession of an object needed by the red one. The solution found by our algorithm is: robot 0 navigates to the blue human, takes the object through a handover over the counter, and then navigates to the red human and gives it to him.

An important issue to address, as illustrated in Fig. 1, is to preserve the humans comfort. In the figure, a handover is possible between initial and target agents, but using the robot reduces the human efforts.

This kind of problem can be solved using a combination of symbolic and geometric planning ([8]–[10]): these approaches will solve the problem, but would be too slow to allow on-line use of the algorithm. (note that using a task planner alone will be under efficient as the problem is geometrically complex [11]).

The main contribution here is the conception of a planner interleaving strongly various models, from task level to geometric computation. This planner should be able to find efficiently high-quality solutions based on a number of parameters related to social rules and humans comfort. The system has been implemented in simulation and also using two PR2 robots and two humans in a cluttered environment.

In section II we describe the related work while in section III we define precisely the problem. Section IV addresses the formalism and the models used in the resolution. In section V we present the results found in several environments and finally we conclude in section VI.

II. Related Work

The problem addressed in this paper is related to a number of categories of work: handover, human aware navigation, pick-up and delivery problem , and manipulation planning.

Difficulties appears for handovers between human and robot, concerning human security, comfort and other social rules, to make it as natural as possible [2], [3]. For a handover between two robots, synchronization is essential [12].

The transport task in our context requires the robots to navigate in a space occupied by humans, who may be busy. In this context, the robot will need to navigate among them while creating the least disturbance possible. In [13] the authors present various approaches to the problem, part of it based on the proxemic theory [14].

Manipulation planning [15], [16], an extension of classical sampling-based motion planning [17] could solve the transport problem. Nevertheless, it would not be able to deal efficiently with the complex costs involved in the task, and the very high dimensionality of the search space (see Sec. III).

The transport problem is close to the pick-up and delivery problem (PDP) which have been widely addressed in the literature. In [18], the authors present a survey of the problem types and solution methods for PDPs. More recently, the link between PDP and object transfer has been stressed out with an algorithm where robots transfer objects to optimize a PDP plan [19].

Recent work related to the multi-agent handover problem uses a heuristic to guide the search through possible handovers [20]. The difference with the previous approach is that handovers are required in order to perform the task, and do not result from an optimization, as robots are fixed manipulator arms. The heuristic used takes into account cost estimation of the handovers and an approximation of the number of remaining handovers that will be required. They also introduce a lazy variant of weighted A*.

In [7], the authors present an algorithm able to share the efforts between a human and a robot depending on the human preferences. In some cases, the human might prefer moving toward the robot, while in others he will prefer waiting for the robot to come all the way to him. Even in this last case, if no solution is found, the robot might ask the human to move in order to successfully perform the handover. This approach is limited to single handovers and doesn't scale to series of handovers. We propose a similar approach adapted to multiple agent problems where it is important to choose both agents to involve and place to transfer the object.

III. Problem definition

The problem is to bring an object from a starting position to a target agent, having at our disposal several agents able to carry the object and to hand it over to other agents. The goal is to find optimal solutions in a reasonable amount of time allowing an on-line use.

The object is held and moved by a unique agent, and can be transferred to other agents through handovers.

The problem inputs are the agent list, the initial state, consisting on all agents and objects positions, and agents specific information about speed and availability. The target state is defined by the target agent (also an input) holding the object. The algorithm should also take into account the task urgency and the care given to humans.

A solution to the problem is a scheduled sequence of actions (navigation or handovers), that bring the object from the initial state to the desired final state.

The search space is the full configuration space [17] of the whole problem. As it involves several agents, it can be written as the cross-product of the configuration spaces of each agent: $C = C_0 \times C_1 \times \cdots \times C_n$. As it is, the problem high dimensionality makes it nearly impossible to find a solution based on classical algorithms. E.g. in the environment of Fig. 1, we have $card(C) \simeq 300$.

IV. Model and resolution

The full representation of the problem is not suitable for on-line solution search. We break down the main problem into problems of lower dimensionality: we distinguish two subsets of the main problem, (1) navigation between handovers places, and (2) handovers themselves.

Navigation is simplified to path-finding in a discrete 2D grid. This 2D model is based on the input environment and agents geometries, but is built off-line and does not affect running-time. Besides, we do not take into account inter-agent collisions during the search phase. For that, we make the assumption of a large environment with sparse obstacles and few (or no) narrow passages. Thus, we can consider only one agent at a time for navigation. We explain in Sec. IV-D how we deal with violations of this hypothesis.

Handovers involve two agents and we need the full dimensionality of their models and their positions to find a solution. The planner can treat situations where the object has to be handed over a table, through a window or even a narrower passage. This computation is expensive, specifically when the handover is not possible. Thus, we limit the number of calls to the motion planner to the minimum, and try to detect impossible situations with faster tools.

In addition to that, the problem has a higher level discrete component: finding a sequence of agents to perform successive handovers which guides the search toward relevant handovers, limiting calls to time-consuming evaluations.

In this section, we present formally the models we use, and the algorithms involved in the solution.

A. Representation

We solve the problem as a path finding problem in a graph G. We consider nodes being simplified states containing information only about the agent holding the object at that time, while others are considered to be at their initial positions. A node is then a pair formed of an agent identifier and a discrete 2D position: $[a_{id}, \{x, y\}]$. We define two actions that cause the current state to change: an elementary navigation action and a handover action:

- *Navigation*: an agent A holding the object can travel to a neighbour position: $[A, \{x, y\}] \rightarrow [A, \{x + \delta_x, y + \delta_y\}]$;
- *Handover*: the holding agent A gives the object to another agent B: $[A, \{x, y\}] \rightarrow [B, \{x', y'\}]$. This requires B to move from its initial position to its new one ($\{x', y'\}$), and causes A to go back to its own initial position.

An edge between two nodes of G exists when there is an action leading from a state to another. Edges are weighted with the costs of the actions. The cost of a handover action

is high compared to a navigation action, as it involves two auxiliary navigation tasks along with the handover itself.

The search algorithm relies on other models: high level representation, 2D model for navigation, and full geometric representation for handover posture search and check, collision checking and motion planning.

a) High-level graph: This graph guides the search through all possible handovers. We will later refer to it as the *agent graph* G_A (different from graph G). In this graph, the nodes are the agents and edges represent the handover between the two linked agents. The model is initialized with all the handovers set as possible (edges between every node), and each edge is weighted with an optimistic estimation of the cost, based on time needed to perform the handover and the optimal human-related cost expected, independently of the environment. While the search will be pursued, the costs are adjusted and the edges may be removed in case of inability to perform a handover.

b) 2D navigation grid: To plan the navigation tasks.

c) Geometric environment model: We use geometric algorithms (e.g. collision checking, inverse-kinematics, motion planning) to find valid handover positions and their cost related to comfort, acceptability, legibility and so on. A valid handover is a collision-free position for both agents, where the object can be transferred [7]. All the process of finding and evaluating a handover will be referred as the handover search tool.

d) Model relations: The main planner is linked with the three other models, and their respective planners, but there are also interactions between every planner. Each model constantly evolves on the basis of lower level model evolutions. For example, edges in the agent graph are removed when the geometric tools have tested these handovers and found none is possible.

e) Cost: The cost function defined to evaluate a solution is as follows, with several f being factors to stress different parameters, $f_d(a)$ is a per agent factor, $t_{use}(a)$ is the time an agent a is mobilized for our task. t_{begin} is the date the first agent starts moving and t_{end} is when the last agent reaches its target position with the object. Each handover has a cost $c_{HO}(i)$ related to comfort and safety. (n is the number of handover in the solution)

$$c = f_{use} \cdot \sum_{a \in A} t_{use}(a) \cdot f_d(a) + f_{time} \cdot (t_{end} - t_{begin}) + f_{HRI} \cdot max(c_{HO}(0), \ldots, c_{HO}(n))$$

B. Resolution

Our main model is the graph G representing the problem. The use of a search algorithm in this graph is still problematic as the number of expansions can be huge, and computationally expensive. In the rooms environment (Fig. 1) the number of neighbours of a nodes reaches 3000 in average. The use of a classic A* would cause evaluating all these handover actions, which is computationally intractable.

The objective is to reduce the number of handover evaluations. The use a lazy variant of A* which gives a first estimation of a handover cost, sparing the expense of a call to the handover search tool. The real cost will be computed only if the handover appears to be relevant. We use as search algorithm the Lazy Weighted A* (LWA*) variant [20]. This algorithm has proven bounds of sub-optimality inherited of Weighted A* [21], and can perform faster when it involves computationally expensive evaluations. LWA* algorithm is based on A* algorithm, which searches for the shortest path using a heuristic. When a node is expanded, the son nodes are given three values: the g value is the distance (cost) between the origin and the son node, the h value is the heuristic, *i.e.* the estimation of the distance (cost) remaining to reach the target, and the f value is the sum $g + h$. In the next iteration, the unexpanded node with the smaller f value will be expanded, until the target node is reached.

In the weighted variant, the h value is increased by a factor, f becomes $g + \epsilon.h$ with $\epsilon \geq 1$, thus adding a depth-first flavour to the search, but decreasing the quality of the solution of at most that factor ϵ.

The lazy variant gives to expanded node sons a temporary g value, which is optimistic and faster to compute than the real cost. It computes the real cost only when the node is selected to be expanded, *i.e.* is the one with the smaller f value. Its g and f values are updated and it is put back in the list of nodes to be expanded.

This approach can save a substantial number of costly evaluations, postponing them to when they are really needed. In our case, such costly evaluations are the handover search.

1) Heuristic and cost: The cost function evaluates an edge between two nodes n_A and n_B of G. There are two cases, according to which action is represented by the edge: elementary navigation or handover. For the navigation, the cost is only distance related (energy, time). For the handover, we need to evaluate the motion with the geometric tools and estimate the cost based on the humans comfort and the action duration.

The heuristic function (Algo. 1) guides the search through the environment and the possible handovers. It is based on G_A and the navigation grid. It first searches for an agent sequence that can bring the object to the target agent. This search is made in the agent graph and takes into account minimal handover costs and no navigation (line 2). Then, on the basis of this agent sequence, it searches the minimal cost related to the navigation. In our model, it is using the cheapest agent for the whole distance (in the Euclidean sense, line 4). At this step, it is not known yet if it is possible or not, but it guarantees that the heuristic is admissible. It then adds the estimation of the handovers costs, which must be computed with an admissible heuristic too (line 7).

2) Handover tests: Even with the use of weighted and lazy variants of A*, the time consumed doing geometric computations is still high. When it is given a situation where no handover is possible, the handover search tool will fail after trying a number possibilities, which is highly time consuming. In order to speed up this process, we add simpler checks to discard impossible situations. The first one is based on distance, knowing the arm reach limits of each agent. The

Algorithm 1 Heuristic function for the main search algorithm

1: **function** HEURISTIC(n,n_{goal}) ▷ *n and n_{goal} are nodes of G*
2: $path \leftarrow$ SHORTESTPATH(G_A,n,n_{goal})
3: ***for*** *each agent a of $path$* ***do***
4: $d \leftarrow min(d,$ DISTANCECOST(a))
5: ▷ *cost for a to go from n to n_{goal}*
6: ***for*** *each handover HO in $path$* ***do***
7: $h \leftarrow h+$ HANDOVERHEURISTIC(HO)
8: ***return*** $h + d$

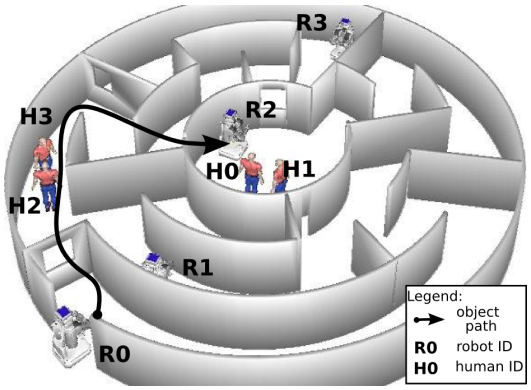

Fig. 2. The maze example, with a possible path for the object shown by the black arrow. In this example, the object is successively held by R0, H2, H3, R3, H0. The use of multiple agent saves time for the delivery while asking more work to humans. Their use is still limited as most of the navigation is done by robots. Under other settings, the whole navigation could be done by the robot R0, which is slower but does not disturb humans at all. Or the same path for the object could be done while being held mainly by R1, but it would cause the humans to leave the place for the robot to go across.

second is a collision test with the object alone: if there is no path for it from an agent to the other, the handover is not possible. The third is an inverse-kinematics test where we check that both agents can grasp the object simultaneously. The last one tests several pre-defined handover positions for collision, and if it succeeds applies a cost to it related to effort and comfort for humans when relevant. This cost is applied to the edge in the graph G to be used by the search algorithm. Optionally, the full motion can be computed, but the previous test is sufficient in most scenarios. Note that the handover search tool can be substituted with any state of the art algorithm as soon as it can find a solution from 2D poses of each agent and compute its cost.

C. Schedule

The path provided by the search algorithm lacks information about the agent states at every moment. The post-process phase enable us to add all navigation actions that do not involve the object, *i.e.* the navigation to go from initial position to a handover place to get the object, and the navigation from a handover place to the initial position after giving the object. They already have been computed during the search but need to be added to the solution.

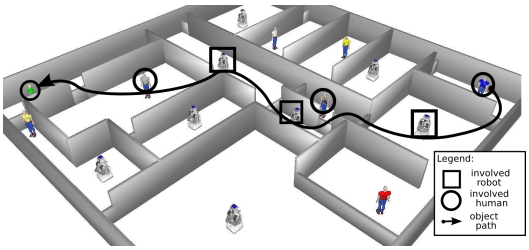

Fig. 3. The big rooms example, with a solution represented by the black arrow. Seven agent out of sixteen are part of the plan. This plan tries to minimize delivery time. Other solutions avoiding humans would use only robots and pass by the room at the bottom of the image.

The resulting solution is unordered, We use Simple Temporal Networks (STN) [22] to schedule all tasks.

The following assertions are applied to build the STN: *(i)* a handover can start only when both agents are at their handover position; *(ii)* after a handover, agents can leave only when the handover is fully finished.

Knowing the duration of each task, we can use the STN to compute all tasks optimal start and end dates for the plan to be executed faster.

D. Collision post-process

The final step of the planning process is to guarantee the feasibility of the trajectories according to collisions between agents. The schedule let us know the position of any agent at any time. When a collision is found, paths are recomputed taking into account all the agents involved in that collision. The agent holding the object is supposed to have priority, safe places are found for the other agents, and the new solution is scheduled. We iterate until all trajectories are valid. This part of the algorithm is highly time-consuming, and is called only when we have found a solution and we are almost sure to keep it. This post-processing reduces the solution quality without any specified bound, it only modifies the solution locally, as least as possible.

V. RESULTS & DISCUSSION

The system has been tested on various environments, including a real-life test with our two PR2 robots. We show that it computes in a satisfying time to allow on-line use in small building environments and with a number of agents around ten, and perform even better with an initialization phase. We will discuss the scalability, and adaptability to other kinds of scenarios and the integration in more complex systems, with other tasks, will also be examined.

A. Examples

We have tested our algorithm in environments of various complexities and likeliness. [1]

[1]The simple scenarios with only one handover are not presented in this paper, although, we tested them and found satisfying results.

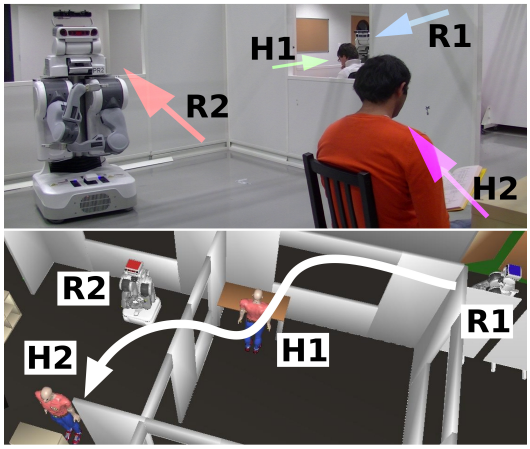

Fig. 4. The real robot example. The bottom picture the 3D model, with the object path as a white arrow. The solution involves successively R1, H1, R2, H2, and the object is handed over through the windows, minimizing H1 efforts, and brought directly to H2 so he doesn't have to move.

Example 1 - rooms: The environment presented in Fig. 1 is a realistic scenario with medium complexity ($18 \times 16m^2$, 10 agents). The agents share common areas, where handover are possible everywhere. This property increases the complexity: each node in G has as many neighbours as the number of possible handovers with each agent sharing the same area, in addition to the navigation neighbours. The node number in this example [2] is about 64000 (we use a discretization step of 0.15 in all the examples, and for this one, there is an average of five possible agents by cell). In this environment, the A* algorithm is efficient as the heuristic is close to the real cost.

Synthesis: $18 \times 16m^2$, 10 agents, 64000 nodes, efficient heuristic.

Example 2 - maze: Fig. 2 is a maze where there is always a direct solution where the first and target agents can meet to perform a single handover. Windows allow faster delivery if the object is handed over through them between intermediary agents. The A* heuristic gets trapped in this environment as a solution is rarely close to the straight line. There are 102400 nodes (8 possible agents per cell).

Synthesis: $18 \times 16m^2$, 8 agents, 102400 nodes, inefficient heuristic.

Example 3 - big room: The environment in Fig. 3 is a large environment ($25 \times 25m^2$) where the 16 agents are in rooms connected by doors or windows. The graph G is relatively small (41500 nodes) due to the small number of agents who share the same area (1.5 possible agents per cell in average) even if the size of the environment is bigger than in the other examples. The A* heuristic does not get trapped as in the maze, but solutions usually differ from straight lines.

Synthesis: $25 \times 25m^2$, 16 agents, 41500 nodes, normal efficiency for heuristic.

[2]In order to compute the node number we use the formula: $surface/(discr.step)^2 \times numberofagentsbycell$.

TABLE I

COMPUTATION TIMES FOR EACH ENVIRONMENT

environment	ϵ	Computation time mean	Computation time SD	solution cost[a]	solution SD
large rooms	10	20.5	32.0	1.10	0.27
	4	26.0	39.0	1.02	0.24
	1	61.3	62.9	1.00	–
maze	10	2.07	3.6	1.42	1.53
	4	2.11	3.8	1.37	1.68
	1	13.04	26.5	1.00	–
rooms	10	3.6	6.5	1.73	1.73
	4	2.5	3.1	1.46	1.46
	1	20.4	31.0	1.00	–
apartment	10	12.4	26.3	1.49	1.13
	4	16.0	48.7	1.18	0.31
	1	33.3	104	1.00	–

[a] relative to the optimal solution cost ($\epsilon = 1$) found for the same problem instance

Example 4 - real robot: We tested our algorithm in a real environment[3] (Fig. 4), it is a small environment (8*15m^2) with few agents (2 humans and 2 PR2 robots). There are 16000 nodes, with 3 possible agents per cell. In our implementation, one robot computes the solution (without the trajectories) and share it with the other robot, then each one of them computes its own trajectories based on this solution.

Synthesis: $8 \times 15m^2$, 4 agents, 16000 nodes, normal efficiency for heuristic.

B. Results

The table I gives computation mean times and their standard deviation for studied environments. Each mean time is computed with 40 samples with randomly selected start and goal agents. Half of the samples give priority to the execution time (deliver as fast as possible), and half prioritize human comfort (by avoiding including humans in the solution) The program is run on an Intel® Xeon® Processor E3-1271 v3 (8M Cache, 3.60 GHz), it uses one core only.

C. Discussions

1) Computation time: It is much faster to compute with higher epsilon value as many expansions are spared due to the depth first tendency of the Weighted A*. The loss on the cost is far under the upper limit of sub-optimality (ϵ) as our heuristic is really optimistic. For the lower epsilon, high variance is due to the diversity of the problem instances (different starts and targets agents), some of which are easier to compute than others. Note that the possible handovers are memorized making following request resolution faster than the first ones. The lazy variant spare a number of real expansions: averaged on 80 random runs in the rooms environment (Fig. 1), the true cost is computed only for 0.46% of the explored nodes. The ratio between the computation time of a real cost evaluation and the temporary cost estimation is almost 200, with $67ms$ for the former and $0.334ms$ for

[3]The attached video shows a solution for this scenario.

the latter. Statistically, without the lazy variant, evaluating explored nodes would be more than 100 times as long. That factor reaches 400 in the rooms environments, while still providing exactly the same solutions. This reinforces the relevance of the Lazy Weighted A* variant use in our case.

It appears that the environments influence the computation time strongly. The "rooms" environment compared to the others is very straight forward: the heuristic is quite good, so even if there are lots of nodes in the graph G, only few of them will be expanded thanks to it.

The humans are explicitly taken into account in the cost computation. When the cost are oriented toward humans comfort, the algorithm will try to avoid including human in the sequence solution, but if the costs are oriented toward a fast delivery, the agent are considered equally in order to achieve the task.

2) Adaptability: This approach can be adapted to any kind of scenario, but will have limitations according to its characteristics. Mainly, the complexity of the problem explodes with the size of the environment and the number of agents. This can be limited if agents are limited to smaller parts of the environment as in Fig. 3. Larger environments or more complex ones (compared to those studied here) would be successfully solved by our planner, but the computational time does not allow on-line use for such situations. Though, such complex cases are supposed to be rare and do not enter under the scope of our work.

3) Improvements: Our approach can be improved on many points. The use of adapted Monte-Carlo methods could give similar or better results, it could make possible to have a more complete model, or more precise. An interesting feature to add is the ability to plan object exchange without handover, using place and pick actions, to relax synchronization constraints of the handovers. We use STN to compute optimal schedules, with fixed durations for each task but some have high uncertainty, either concerning robots (risk of failure, difficulty to perform handover) or humans (they will not do exactly what was planned). Adding this information to the plan will enable the use of appropriate tools to execute plans with uncertainty [23]. Other improvements concern the model used. Instead of a grid with static discretization step, we can imagine using quad-tree structure [24] to optimize the number of nodes. Work can be done at improving the human related cost computation methods, the heuristic can be made more accurate.

VI. Conclusion

In this paper, we have presented a planner able to find a sequence of handovers involving several agents (robots or humans) in order to bring an object from a starting agent to a target agent.

Our approach finds optimal solutions, based on human-aware costs, in a short amount of time for this problem. It is based on a graph where each node is a possible object position and the agent holding the object, and the edges are the ways an object can move from a node to another (either through a navigation action, or a handover).

This approach has been tested in various environments with variations, considering the costs, the starting and target agents, the environment itself (by opening or closing some doors/windows). This proved the adaptability of the algorithm in scenarios with different kinds of difficulties.

References

[1] K. W. Strabala, M. K. Lee, A. D. Dragan, J. L. Forlizzi, S. Srinivasa, M. Cakmak, and V. Micelli, "Towards Seamless Human-Robot Handovers," *Journal of Human-Robot Interaction*, vol. 2, no. 1, 2013.

[2] M. Cakmak, S. S. Srinivasa, M. K. Lee, J. Forlizzi, and S. Kiesler, "Human preferences for robot-human hand-over configurations," in *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, 2011.

[3] M. Cakmak, S. S. Srinivasa, M. K. Lee, S. Kiesler, and J. Forlizzi, "Using spatial and temporal contrast for fluent robot-human hand-overs," in *Proc. Int. Conf. on Human-robot interaction*, New York.

[4] W. P. Chan, C. A. Parker, H. Van der Loos, and E. A. Croft, "Grip forces and load forces in handovers: implications for designing human-robot handover controllers," in *Human Robot Interaction*, 2012.

[5] K. Dautenhahn, M. Walters, S. Woods, K. L. Koay, C. L. Nehaniv, A. Sisbot, R. Alami, and T. Siméon, "How may i serve you?: a robot companion approaching a seated person in a helping context," in *Proc. Conf. Human-robot interaction*, 2006.

[6] J. Mainprice, E. Sisbot, T. Siméon, and R. Alami, "Planning safe and legible hand-over motions for human-robot interaction," in *IARP workshop on technical challenges for dependable robots in human environments*, vol. 2, 2010.

[7] J. Mainprice, M. Gharbi, T. Simeon, and R. Alami, "Sharing effort in planning human-robot handover tasks," in *IEEE RO-MAN*, Sept 2012.

[8] L. Karlsson, J. Bidot, F. Lagriffoul, A. Saffiotti, U. Hillenbrand, and F. Schmidt, "Combining Task and Path Planning for a Humanoid Two-arm Robotic System," *TAMPRA*, 2012.

[9] C. Dornhege, P. Eyerich, T. Keller, S. Trug, M. Brenner, and B. Nebel, "Semantic Attachments for Domain-Independent Planning Systems," *ICAPS*, 2009.

[10] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Proc. Conf. IEEE ICRA*, 2011.

[11] F. Lagriffoul, L. Karlsson, J. Bidot, and A. Saffiotti, "Combining task and motion planning is not always a good idea," in *RSS Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications*, 2013.

[12] A. H. Quispe, H. Ben Amor, and M. Stilman, "Handover planning for every occasion," in *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*. IEEE, 2014.

[13] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, 2013.

[14] E. T. Hall, *The hidden dimension*. Anchor Books New York, 1969, vol. 1990.

[15] T. Simeon, "Manipulation Planning with Probabilistic Roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, 2004.

[16] J. Barry, K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez, "Manipulation with multiple action types," in *Experimental Robotics*. Springer, 2013.

[17] S. M. LaValle, *Planning algorithms*. Cambridge Univ press, 2006.

[18] M. W. Savelsbergh and M. Sol, "The general pickup and delivery problem," *Transportation science*, vol. 29, no. 1, 1995.

[19] B. Coltin and M. Veloso, "Online pickup and delivery planning with transfers for mobile robots," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2014.

[20] B. Cohen, M. Phillips, and M. Likhachev, "Planning Single-arm Manipulations with n-Arm Robots," in *Proc. of Robotics: Science and Systems*, Berkeley, USA, 2014.

[21] M. Likhachev, G. J. Gordon, and S. Thrun, "ARA*: Anytime a* with provable bounds on sub-optimality," in *Advances in Neural Information Processing Systems*, 2003.

[22] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial Intelligence*, vol. 49, no. 1, 1991.

[23] P. Morris and N. Muscettola, "Execution of Temporal Plans with Uncertainty," in *Proc. AAAI/IAAI*, 2000.

[24] R. Finkel and J. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, 1974.