Towards S-NAMO: Socially-aware Navigation Among Movable Obstacles

Benoit Renault, Jacques Saraydaryan, Olivier Simonin

To cite this version:

Benoit Renault, Jacques Saraydaryan, Olivier Simonin. Towards S-NAMO: Socially-aware Navigation Among Movable Obstacles. RoboCup 2019 - 23rd Annual RoboCup International Symposium, Jul 2019, Sydney, Australia. pp.241-254, 10.1007/978-3-030-35699-6_19 . hal-02293242

HAL Id: hal-02293242

https://hal.archives-ouvertes.fr/hal-02293242

Submitted on 24 Sep 2019

Towards S-NAMO: Socially-aware Navigation Among Movable Obstacles

Benoit Renault[a,b], Jacques Saraydaryan[a,c], and Olivier Simonin[a,b]

[a] CITI Lab., INRIA Chroma, [b] INSA Lyon, [c] CPE Lyon,
Université de Lyon, Villeurbanne, France

Abstract. In this paper, we present an in-depth analysis of Navigation Among Movable Obstacles (NAMO) literature, notably highlighting that social acceptability remains an unadressed problem in this robotics navigation domain. The objectives of a Socially-Aware NAMO are defined and a first set of algorithmic propositions is built upon existing work. We developed a simulator allowing to test our propositions of social movability evaluation for obstacle selection, and social placement of objects with a semantic map layer. Preliminary pushing tests are done with a Pepper robot, the standard platform for the Robocup@home SSPL[1], in the context of our participation (LyonTech Team).

Keywords: Navigation Among Movable Obstacles (NAMO), Socially-Aware Navigation (SAN), Path planning, Simulation

1 Introduction

In 2005, Stilman et al. [6] formulated the field of Navigation Among Movable Obstacles (NAMO). The NAMO problem consists in planning a path from a start to a goal position, while moving obstacles if necessary. It extends the well known Piano Mover's Problem by differentiating static and movable obstacles, and allowing the manipulation of the later if it minimizes the chosen cost function (eg. travel distance, time, energy). Contexts like service robotics or search and rescue, in particular, would definitely benefit from algorithms capable of dealing with manipulable clutter, doors or objects.

In the last two decades, the growing interest in service robotics, implying robot navigation in human-populated environments, has sparked interest in Social Robotics, and more specifically Socially-Aware Navigation (SAN) [20,24,30]. Basically, it also extended the basic navigation problem: now not only must the robot find a plan that ensures physical safety (no collisions), minimizes the travel distance, time or energy, but also the disturbance to humans[2].

Until now, to the best of our knowledge, has never been considered in NAMO problems the necessity of minimizing disturbance to humans (or any other type

[1] SSPL: Social Standard Platform League

[2] In Socially-Aware Navigation, disturbance is used as synonym for 'discomfort', the feeling of being unsafe [24].

of autonomous agents). Thus, we want to create Social NAMO algorithms: ones that allow an autonomous agent to go from an initial pose to a goal pose, forbidding collision with obstacles but not their displacement, minimizing both robot's displacement cost (distance, time or energy) and disturbance to humans.

To achieve this, we make the following contributions: in Section 2, we provide an analysis of existing NAMO-related works. Then, in Section 3, we define the expectations for Social NAMO and propose two extensions applied to Wu & Levihn's approach [14,16,23]. We introduce social movability evaluation in obstacle selection, and a semantic map layer to deal with social placement of objects. Finally, in Section 4, we propose experiments based on our open simulator and the Pepper robot, in the view of our RoboCup@Home participation (LyonTech Team). We provide closing remarks and discuss future work in Section 5.

2 NAMO: Analysis of Existing Works

The following paragraphs give an overview of NAMO through the discussion of the used world representations, notion of cost & optimality, manipulation characteristics and finally, the actual planning algorithms that rely on them. Also, we will point out how they relate to socially-aware navigation and its constraints. A synthesis of the main comparison criteria is given in table 1.

World representation NAMO relies on an object-based representation of the world [2,5,6,8,10–14,16–19,22,23,26–29,31,33] (in opposition to an occupation-space-based one): in order to chose the best obstacle placement, it is necessary to reason about them as separate entities. Final placement selection is what actually tells NAMO apart from the well-known field of Rearrangement Planning [4]. Inspired by the works of Kim et al. on traversability affordance [7], Clingerman et al. [21,25] represent movable obstacles as high values in a costmap, but they recon that it can't be called a NAMO algorithm, since it does not allow to control obstacle placement (the robot simply tries to "go through the obstacle").

Semantic information about Movable Obstacles is key to these algorithms. The most basic need is the 'movability' attribute, in addition to the obstacles position and shape. In the literature, individual obstacles are simply assumed to have a boolean attribute of being movable or not. This attribute has until now been given as input [2, 5, 6, 8, 10–12, 17, 18, 26–28] (mainly for simulation-only algorithms), determined on-line from obstacle visual recognition results [19,22, 29,31,33] or by manipulation tentative [13,14,16,23] (for real-world experiments). In order to be more realistic, other semantic information is used in more advanced approaches, like object kinematics and physics (mass, center of inertia, . . .), but successfully used only in simulated propositions [2, 6, 8, 26, 27], with mixed results in actual real-world implementations [29] (these characteristics are hard to determine with current robot sensing capabilities). Other types of obstacles than movable or unmovable, like humans or autonomously moving objects have never been considered in the NAMO literature: a standard hypothesis is that the robot is the only autonomous agent in the environment. We recapitulate this in the 'Movability' column of Table 1.

We must also say that rather few NAMO propositions have been applied in a real-world setting [8,13,19,22,29,31,33], and when they are, they always maintain a 3D representation of the world, though all NAMO algorithms execute their path finding subroutines in a 2D plane. 3D data is mainly used to allow for proper grasping of obstacles, but also for cross-plane rearrangement planning [33](e.g. pick&place an object from ground to tabletop). Data is either acquired through external cameras and markers [8, 22, 29] to position priorly known polyhedral models of movable obstacles (eg. chairs, tables, ...), guaranteeing negligible uncertainty as to the environment's state, or by on-board sensors only [13, 19, 31, 33]. A limited number of propositions actually are able to deal with no prior or partial geometric knowledge [13, 14, 19, 23, 31, 33], uncertainty as to object positioning [8, 13, 17–19, 26, 27, 29, 31, 33], object movability [13, 17–19, 29] or object kinematics/physics [18, 29] (Recapitulated in Table 1, columns 'Prior', 'Uncertainty' and 'Real-World').

In the end, SAN and NAMO both depend on semantic knowledge in addition to spatial knowledge: the robot needs to differentiate objects, associate proper attributes with them, but also understand their relations to the whole environment. Systematic segmentation and identification of as many obstacles as possible thus appears to be a basic requirement for a Social NAMO.

Cost & Optimality There is a wide variety of cost functions used in NAMO: distance, time, energy, number of moved obstacles, probability of success, that are sometimes combined or used alternatively: these are synthesized in Table 1, column 'Cost'. The choice of a cost that only takes displacement distance into account can be motivated by the hypothesis that the weight of the movable obstacles is negligible in regard to the physical capabilities of the robot. It is however evident that if manipulating an obstacle results in a significant change of speed and energy requirements compared to a sole navigation task, time and energy become way more appropriate choices.

NAMO Algorithms rarely seek completeness like [6, 8, 10, 12, 28]. None have achieved global optimality, and only Levihn [16, 23] can claim a local optimality for a very simplified variant of the problem where a plan can only contain one movable obstacle (see Table 1, 'Comp.' and 'Opt.' columns). This situation actually makes sense, when one knows that a simplified variation of the NAMO problem, where the robot is considered as a square, all planar obstacles as rectangles of four sizes or "L-Shaped", parallel to the x- or y-axis, has been proved to be NP-Hard [1], and even PSPACE-hard if the final positions are predetermined. When obstacles are further reduced to square blocks limited to translations on a planar grid, the problem still remains NP-Complete [3].

In SAN, the presence of humans brings strong uncertainty that prevents proving global optimality and completeness of navigation strategies. It also results in the need to take social costs into account during navigation [20,24,30] to represent risk of disturbance to humans. In a Social NAMO, we must thus also take social costs into account, and extend them from the robot to the moved obstacles: other entities can now suffer the consequences and risks of a carelessly moved obstacle.

Manipulation In [9], Stilman formalized three main classes of obstacle manipulation procedures: Grasping (constrained contact), Pushing (constrained motion), and Manipulation Primitives (relies on forward simulation of object dynamics, translational or rotational slip may occur). According to the results exposed in Table 1, Column 'Manipulations', grasping is the most popular class, likely because it is the most reliable. Pushing has also been considered because large objects cannot necessarily be grasped. Manipulation Primitives have also been experimented with, but real-world implementations require external cameras to work [8, 29].

In order to reduce the manipulation search space, there are 3 common strategies. The first, applied by all but [2], is to consider that only one obstacle may be manipulated at once (no cascade effect on nearby movables). The second, also commonly used by all but [2, 12, 28] (which have never been applied in a real-world situation), is to consider a limited set of contact/grasping points, facilitating backward search for robot pose for manipulation. This semantically makes sense, in particular since some obstacles have specific contact points (eg. top of chair, regularly spaced points on table side, . . .) [5, 6, 8, 11, 13, 14, 16–19, 22, 23, 26, 27, 29, 31, 33]. Finally, the third strategy is to limit manipulation to translations in specific directions [8, 12–14, 16, 17, 23].

In a Social NAMO, the robot should bring a particular attention to human safety and comfort. Favoring the most reliable manipulation classes when possible, and reducing the complexity of the manipulation (thus, its chance to fail in a way that may put humans or their belongings at risk) would be of circumstance.

Planning algorithms While some solutions [2, 26, 27] propose tightly woven algorithms that do not clearly distinguish the different aspects of NAMO (iteration over movable obstacles, possible actions and path computations), we can usually tell apart a high-level decision planner and two path planning subroutines. These subroutines can loosely be identified as transit (robot only) and transfer (+obstacle) path planners.

The most proposition-specific planner is generally the high-level task planner. While some propositions are explicitly based on existing algorithms, like Dijkstra [2], DFS [6,8,11,28], BHPN [19], Markov Decision Processes + Monte-Carlo Tree Search [17, 18, 29], KPIECE+A* [26, 27], others appear to have developed their approach from scratch [5, 10, 12–14, 16, 22, 23, 31, 33], though [16, 23] is based off [14], and [33] has been inspired by [14, 23]. In order to reduce computation time, most high-level planners resort to ways of prioritizing the most promising obstacles but [2, 10, 12, 19, 22, 26, 27] do not. The most common way is to use a heuristic path planner that ignores movable obstacles to find 'blocking' obstacles [6, 8, 11, 28, 31, 33]. Then, the last blocking object is selected by last intersection [6, 8, 11, 28] or by least euclidean cost to go from obstacle to goal [33]. The propositions of Wu & Levihn [14,16,23] use a priority queue ordered by heuristic euclidean distance from obstacle to goal. Finally, the last approach is to use a graph that links obstacles to free space components so that obstacles are considered in the order they can be reached, as in Levihn & Scholz's NAMO-MDP [17, 18, 29] or Okada's Task Graph [5].

Table 1: Synthesis table with main differentiating criteria

Reference	Prior	Movability	Uncertainty	Comp.	Opt.	Cost	C-Space	Task P.	Transit P.	Transfer P.	Manipulations	Real-World
Chen [2]	Full	Given	None	-	-	D	Disc.	Dij. + GD	N/A	N/A	Prim.	No
Okada [5]	Full	Given	None	-	-	D‖E	Disc.	Custom	NG	NG	Grasp	No
Stilman [6]	Full	Given	None	RC	-	E+NMO	Disc.	DFS	A*	BFS	Prim.	No
Stilman [8]	Full	Given	Pos.	RC	-	E+NMO	Disc.	DFS	A*	BFS	Prim.	**Yes**
Nieuwenhuisen [10]	Full	Given	None	PC	-	D+PS	Cont.	Custom	RRT	RRT	Grasp	No
Stilman [11]	Full	Given	None	-	-	D+NMO	Disc.	DFS	A*	BFS	Grasp	No
Van den Berg [12]	Full	Given	None	PC	-	(D)	Disc.	Custom	N/A	N/A	Grasp	No
Kakiuchi [13]	**None**	Manip.	Pos. Mov.	-	-	(D+NMO)	Cont.	Custom	RRT	N/A	Push	**Yes**
Wu [14]	**None**	Manip.	None	-	-	(D‖T‖E)	Disc.	Custom	A*	DFS	Push	No
Levihn [16, 23]	**None**	Manip.	None	-	**LO**	(D‖T‖E)	Disc.	Custom	D*Lite	DFS	Grasp	No
Levihn [17]	Full	Given	Pos. Mov.	-	-	PS	Disc.	MDP + MCTS	N/A	N/A	Prim.	No
Levihn [18]	Full	Given	Pos. Mov. Kin.	-	-	T+E	Cont.	MDP + MCTS	PRM	RRT	Prim.	No
Levihn [19]	Partial	Recog.	Pos. Mov.	-	-	(D‖T‖E)	Cont.	BHPN	RRT	RRT	Grasp	**Yes**
Mueggler [22]	Full	Recog.	None	-	-	T	Disc.	Custom	A*	Dij.	Grasp	**Yes**
Castaman [23, 26]	Full	Given	Pos.	-	-	T	Disc.	KPIECE + A*	N/A	N/A	Grasp‖Push	No
Moghaddam [28]	Full	Given	None	**CO**	-	E	Cont.	DFS	Dij. + VG	Dij. + VG	Grasp	No
Scholz [29]	Full	Recog.	Pos. Mov. Kin.	-	-	T+E	Cont.	MDP + MCTS	PRM	RRT	Prim.	**Yes**
Sun [31]	Partial	Recog.	Pos.	-	-	(D)	Cont.	Custom	RRT	RRT	Grasp‖Push	**Yes**
Meng [33]	Partial	Recog.	Pos.	-	-	D	**Cont. MP**	Custom	RRT	RRT	Grasp‖Push	**Yes**

Legend: () = Not given but likely; '+' = Combination of; '‖' = Alternative to; Manip. = Found through manipulation; Recog. = Found through visual recognition; Pos. = Manage uncertainty on position; Mov. = Same on movability; Kin. = Same on object kinematics; '-' = Depending on columns, either Not Optimal or Not Complete; CO = Complete; RC = Resolution-Complete; PC = Probabilistically Complete; LO = Locally Optimal; D = Distance; E = Energy; T = Time; NMO = Number of Moved Obstacles; PS = Probability of Success; Disc. = Discrete; Cont. = Continuous; MP = Multi-Plane; Dij. = Dijkstra; GD = Generalized Distance; VG = Visibility Graph; NG = Not Given; N/A = Non Applicable; Prim. = Motion Primitives

As for transit path planners, used ones are traditional A* [6, 8, 11, 14, 22], D*Lite [16, 23] over discrete environments, and RRT [10, 13, 19, 31, 33] or PRM [18, 29] variants and Dijkstra over Visibility Graph [28] for continuous ones. On the other hand, obstacle placements are either decided through incremental application of motion primitives (forward search, eg. little translations/rotations) [2,6,8,10–14,16–19,23,29], or by growing sampling of possible placements in the obstacle's vicinity and subsequent path verification [5,22,26,27,31,33]. In some cases, when planning for successive obstacles, placement is constrained by the need to keep a taboo zone for the next manipulations [10–12, 19, 28, 31, 33]. In the end, in discrete environments, transfer path planners iterate over possible obstacle placements using Best-First Search [6, 8, 11] or Depth-First Search [14, 16, 23], or in continuous ones, using an RRT variant [10, 18, 19, 29, 31, 33] or again Dijkstra+VG [28].

Approaches mentioned for the three planning tasks are given in Table 1. We can note they are commonly found in SAN, thus incorporating social cost in NAMO planners should be possible. Although, many of them are offline planners: efficient online or anytime-oriented variants will be needed. In conclusion to this state of the art, we underline that none of the existing NAMO literature directly addresses social constraints, though a few references quickly mention the idea of taking object fragility into account [6, 13].

3 Extension of NAMO Algorithms

3.1 Objectives of Socially-Aware NAMO

From our previous analysis, three general objectives of S-NAMO can be identified. The first is **Social Movability Evaluation**, or determining the movability of an object by human-acceptance for a robot to move it. The second is **Social Placement Choice**, or ensuring that the final environment reconfiguration is the least disturbing to humans compared to the initial one. Finally, the third is **Social Action Planning**, or making sure that all robot actions are in themselves as safe and comfortable for humans as possible.

In the light of the classification in SAN literature [20, 24], we can elaborate three levels of problems of growing difficulty: *delayed human-object interaction* due to future human presence, *indirect interaction* due to actual human presence, and *direct human-robot interaction*. At the first level, like in usual NAMO, the robot can assume to be the only autonomous agent around (eg. cleaning robot servicing while humans are away), thus it only needs to be concerned about **Social Movability Evaluation** and **Social Placement Choice**. At the next levels, the robot must also integrate the dynamic and social aspects of human presence, and answer the additional objective of **Social Action Planning**, exhibiting behaviors such as kindly asking humans to let it pass. In the rest of the paper, we make a first S-NAMO proposition addressing **Social Movability Evaluation** and **Social Placement Choice**, in the context of *delayed human-object interaction*.

3.2 Extension of Wu and Levihn's approach

We chose to build our proposition upon the solution proposed by Wu & Levihn [14, 16, 23] mainly for two reasons. First, it is designed for unknown environments, thus covers plan invalidation in the light of new knowledge, which is eventually essential for real-world applications. Second, as long as the problem is solvable by a single obstacle move in a single direction in the current robot knowledge, local optimality is guaranteed. It basically follows the general form of a NAMO algorithm presented in Section 2: iterate over known obstacles following a heuristic order, and evaluate potential plans that include obstacle movement as long as it can create a better plan. We introduce the S-NAMO Algorithm (see below), which extends the Wu & Levihn approach. The algorithm relies on two procedures: a main obstacle-level one, `make-and-execute-plan()`, that when needed calls a combined transfer/transit path planning sub-procedure `make-plan-for-obstacle()`. We first present these two procedures ignoring our S-NAMO extensions highlighted in red and blue in Algorithm 1, then detail the extensions.

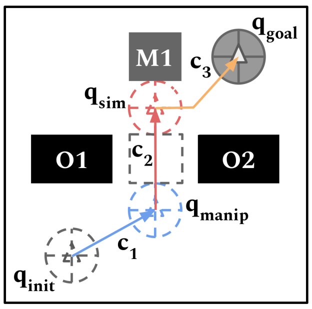


Fig. 1: The robot (grey disc) executes a three-step p_{opt} plan to move M1.

The main procedure, `make-and-execute-plan(`w, q_{init}, q_{goal}`)`, builds and executes the optimal navigation plan p_{opt} from world knowledge w (2D metric map with polygonal entities) and robot configurations $\{q_{init}, q_{goal}\}$. p_{opt} is either a path avoiding all obstacles or constructed from three path components (see Figure 1): c_1, c_2, c_3, respectively paths from q_r to q_{manip}, from q_{manip} to q_{sim} where the robot stops moving obstacle o, and from q_{sim} to q_{goal}. It always first tries to find the best plan avoiding all obstacles, and only then, iterates over movable obstacles to find out whether moving one of them will yield a better plan. Robot knowledge is updated after each execution step, and if p_{opt} is no longer valid (future collision with other obstacles by robot or manipulated obstacle, failure in manipulation, or disrupting update of the manipulated obstacle geometry), re-planning is triggered. Since our contributions do not concern this procedure, we refer the reader to [23] to better understand the iteration through obstacles.

The sub-procedure, `make-plan-for-obstacle(`w, q_r, q_{goal}, o, p_{opt}`)`, is called during the iteration over obstacles, and returns the best plan p_{best} implying the manipulation of obstacle o. It iterates over actions *act* that can be done on o, assuming (line 4) there is only one robot configuration q_{manip} for every $\{o$, *act*$\}$ pair (middle of o's side). The plan components are computed sequentially, starting with c_1. If c_1 is found, successive unit actions *act* of constant length are simulated (*count* times) in a copy of w until impossible (collision with other obstacle). To avoid unnecessary computations of c_2 and c_3, the simulation is stopped as soon as an underestimated cost C_{est} of the currently evaluated plan gets higher than the one of p_{opt} (l.15). C_{est} is the sum of c_1's cost, a c_2 estimate (product of *count* by unit length), and a c_3 estimate (minimal euclidean distance between o and q_{goal}). Also, full evaluation is only done if a new local opening has been created around o (l.16, method described in [15]).

Algorithm 1: S-NAMO - Extension of the Wu&Levihn approach: Social Movability Evaluation in blue and Social Placement Choice in red

```
1  Procedure make-and-execute-plan($w$, $q_{init}$, $q_{goal}$)
2      ▷ when plan is invalidated, makes a plan avoiding all obstacles, then tries
         to improve it by iterating over obstacles and calling
         make-plan-for-obstacle($w$, $q_r$, $q_{goal}$, $o$, $p_{opt}$)

1  Procedure make-plan-for-obstacle($w$, $q_r$, $q_{goal}$, $o$, $p_{opt}$)
2      $p_{best} \leftarrow \emptyset$
3      foreach $act$ in affordable-actions($o$) do
4          $q_{manip}, c_1 \leftarrow$ q-for($o$, $act$), A*($w$, $q_r$, $q_{manip}$)
5          if $c_1 \neq \emptyset$ then
6              if is-unknown($o$) then
7                  $q_{look} \leftarrow$ get-last-look-q($w$, $o$, $c_1$)
8                  if $q_{look} \neq \emptyset$ then $c_0, c_1 \leftarrow$ split-at-pose($c_1$, $q_{look}$)
9                  else $c_0, c_1 \leftarrow$ compute-$c_0$-$c_1$($w$, $o$, $q_r$, $q_{manip}$)
10             else $c_0 \leftarrow \emptyset$
11             if is-movable($o$) or (is-unknown($o$) and $c_0 \neq \emptyset$ and $c_1 \neq \emptyset$) then
12                 $w_{sim} \leftarrow$ copy($w$)
13                 $count, q_{sim} \leftarrow 1$, sim-one-step($w_{sim}$, $act$, $o$, $q_r$)
14                 while $C_{est}$($w_{sim}$, $c_1$, $count$, $act$) $\leq$ cost($p_{opt}$)
15                  and is-step-success($q_r$, $q_{sim}$, $count$, $act$) do
16                     if check-new-opening($w$, $w_{sim}$, $o$) and not-in-taboo($w$,
                        $o$) then
17                         $c_2 \leftarrow$ line($q_{manip}$, $q_{sim}$)
18                         $c_3 \leftarrow$ A*($w$, $q_{sim}$, $q_{goal}$)
19                         if $c_3 \neq \emptyset$ then
20                             $p \leftarrow$ plan($c_0$, $c_1$, $c_2$, $c_3$, $o$, $act$)
21                             if cost($p$) < cost($p_{best}$) then $p_{best} \leftarrow p$
22                             if cost($p_{best}$) < cost($p_{opt}$) then $p_{opt} \leftarrow p_{best}$
23                     $count, q_{sim} \leftarrow count + 1$, sim-one-step($w_{sim}$, $act$, $o$, $q_{sim}$)
24     return $p_{best}$

1  Procedure compute-$c_0$-$c_1$($w$, $o$, $q_r$, $q_{manip}$)
2      $qL \leftarrow$ get-qL($w$, $o$)
3      $paths$-$qL$-$q_{manip} \leftarrow$ multigoal-A*($w$, $q_{manip}$, $qL$)
4      $paths$-$q_r$-$qL \leftarrow$ multigoal-A*($w$, $q_r$, $qL$)
5      return shortest-$c_0$-$c_1$($paths$-$q_r$-$qL$, $paths$-$qL$-$q_{manip}$)
```

Social Movability Evaluation The initial approach of Wu & Levihn supposes that any obstacle is movable unless a manipulation tentative failed, in which case it is blacklisted. However, in S-NAMO this is not an acceptable behavior since it could lead to unauthorized objects manipulations. As a first approach, we propose a simple white-listing system: unregistered obstacles are considered unmovable. But a robot often relies on multiple sensors, and their respective Fields Of View (FOV) are not necessarily equal. An obstacle may have been detected geometrically, but not yet identified, leading to three possible states: unknown, movable, unmovable. As in the initial algorithm, we suppose a perfect conical 'geometry sensor' (eg. high resolution laser range finder), with perfect segmentation of obstacles (blue disk in Figure 2a), but we add a perfect 'semantic sensor' that guarantees identification if the obstacle is in its FOV (eg. using a RGB-D Camera). The geometric FOV (G-FOV) is assumed to cover more space than the semantic one (S-FOV). White-listed obstacles are assumed to fit into the S-FOV, anything that doesn't is automatically classed as unmovable.

The `make-plan-for-obstacle()` procedure has been adapted to work under these hypotheses. When obstacle o is known as movable, the algorithm is unchanged. When o is unknown, we first check whether the usual computation of c_1 can provide observation certainty (lines 6-7); if not, we try to find another path that guarantees observation (line 9). To do that, in `compute-`c_0`-`c_1`()`, we determine the discrete robot configurations list qL that would allow observation (l.2): first, we get all non-colliding configurations within the area between the inflated obstacle polygons by minimal and maximal observation distances, and among them we only return these where o is included in the S-FOV. Then, we execute the multi-goal A* algorithm between q_{manip} and every configuration in qL (l.3). The same is done from the current robot configuration q_r to all elements of qL (l.4). Finally, we return the best pair of paths $\{c_0, c_1\}$ (l.5): see illustration Fig. 2b.

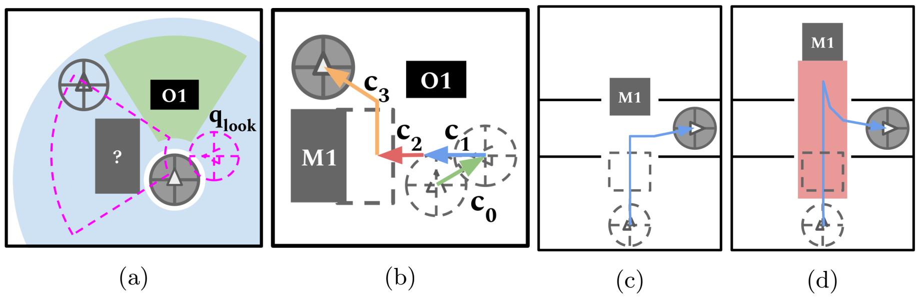


Fig. 2: **In (a)**, G-FOV (blue) detected two obstacles, S-FOV (green) only identified unmovable obstacle O1. Robot is too close from other obstacle to observe it. Going through best intermediate observation configuration is necessary: final best path with c_0 is shown in (b). **(c)** represents two facing rooms separated by a corridor. In typical NAMO (c), robot will push M1 just enough to pass, blocking the other doorway. In our S-NAMO proposition, the taboo zone (red) prevents blocking, but may end up with a longer plan.

Social Placement Choice In current NAMO approaches, the robot does not care about placing obstacles in socially-critical spots (eg. around doors, often-used furniture, . . .). As a first step, we answer this problem with a binary approach: either the zone is taboo for obstacle placement, or it is not.

We extend the definition of w by adding a social placement semantic map layer, where taboo zones are defined as a set of polygons P. We assume for now that P is provided by human users. Now, whenever the polygonal footprint of an obstacle intersects with any polygonal taboo zone in P, `not-in-taboo(`w, o`)` returns *False*, preventing full plan evaluation (see `make-plan-for-obstacle()` procedure, line 16). Figure 2d) illustrates this process on a simple scenario. Next section presents experiments with more complex scenarios.

4 Experiments

We implemented the S-NAMO algorithm in a custom simulator based on ROS standards. This is a first step toward an implementation on a real robot (Pepper), simplifying object detection and identification, which could later be addressed with an existing package such as ED from TU-Eindhoven. For the sake of implementation ease, movable obstacles are assumed to be convex polygons. All computations are done on the 2D vectorial model, except for path planning, which is implemented as a grid-search A* Algorithm, as in [14] [3].

Social Placement Choice We tested the Social Placement Choice process in a scenario where a robot has to successively reach two goals represented as empty circles in Fig.3. The environment consists of two rooms separated by a corridor, but two yellow boxes are blocking the doorways (Fig.3a). The robot (blue circle, FOV is the cone) starts from the bottom room.

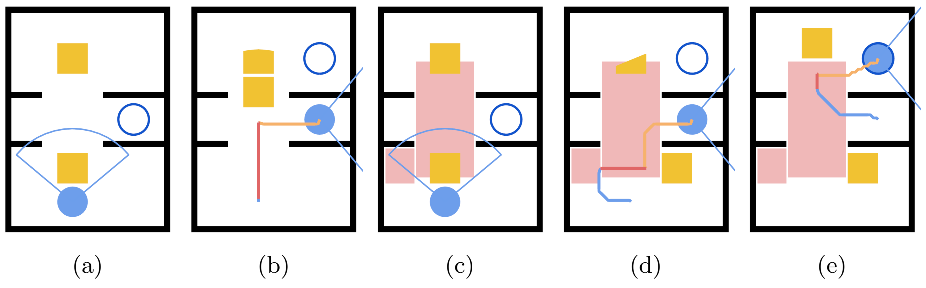


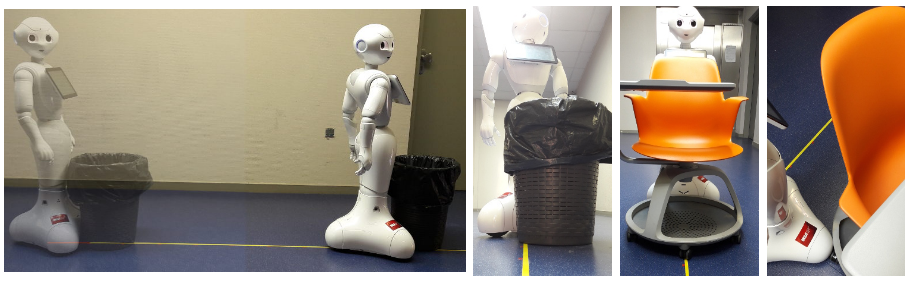
Fig. 3: Simulation of a two-goals scenario with NAMO (a,b) vs. S-NAMO (c,d,e)

In Fig.3b we see that a standard NAMO approach like Wu & Levihn's results in blocking the other doorway: only the first goal is reached. In Fig.3c we introduce the social semantic layer which consists in two taboo areas for objects (in red), respectively related to the doorways and a 'precious carpet'. Fig.3d shows how our algorithm deals with the first discovered object: the computed path moves the box on the right and outside taboo areas, leaving in particular the doorway area free. In Fig.3e, the robot encountered the second box and pushed it outside the taboo area, leading to a path reaching the second goal. This S-NAMO scenario (and others) are available as videos [4].

[3] All the code and its execution instructions are available on the following repository: `https://gitlab.inria.fr/brenault/s-namo-sim`

[4] Link to videos : `https://gitlab.inria.fr/brenault/s-namo-sim/wikis/Videos`

Pushing Experiments with Pepper In the view of a real-world implementation, we experimented with Pepper's base pushing abilities, using our existing robot architecture developed for the Robocup@Home 2018 [32]. In Fig.4a and 4b, Pepper successfully pushes a garbage bin in a straight line with little deviation. We have also verified that with other light objects such as cardboard boxes, that when the object's side is properly centered relatively to the robot, pushes are more likely to succeed. However, we also learned that, as seen in Fig.4c and 4d, heavier objects of interest such as chairs with wheel-casters will need to be accompanied with the arms in some way to avoid unpredictable drift, but even so, the manipulation could still fail (videos available at footnote 4). Thus, in our future work, we will also strive to address uncertainty as to manipulation success, like in [18].



(a) (b) (c) (d)

Fig. 4: Pepper pushing a bin and a chair

5 Conclusion

In this paper, we first analyzed existing NAMO approaches in order to adapt them to social constraints. This led us to extend the Wu&Levihn approach, by defining the S-NAMO algorithm which introduces Social Movability Evaluation and Social Placement Choice for object manipulations. We implemented these propositions in an open source ROS compatible simulator. Experiments showed how social semantic areas can prevent obstruction of places like circulation zones, and how the robot can identify obstacles to compute its plan. In future works, we plan to refine the semantic layer and address actual human presence with indirect or direct human-robot interaction, while integrating ways to manage uncertainty as to sensor data or success of manipulation. We will continue to experiment and validate these social NAMO abilities with robots such as Pepper and demonstrate their interest in the RoboCup@Home challenge.

References

1. G. Wilfong. Motion planning in the presence of movable obstacles. *Annals of Mathematics and Artificial Intelligence*, 3(1):131–150, March 1991.
2. P. C. Chen et al. Practical path planning among movable obstacles. In *1991 IEEE International Conference on Robotics and Automation Proceedings*, pages 444–449 vol.1, April 1991.
3. E. D. Demaine, et al. PushPush and Push-1 are NP-hard in 2d. arXiv: cs/0007021, July 2000.
4. J. Ota. Rearrangement of multiple movable objects - integration of global and local planning methodology. In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, volume 2, pages 1962–1967 Vol.2, April 2004.
5. K. Okada, et al. Environment manipulation planner for humanoid robots using task graph that generates action sequence. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 2, pages 1174–1179 vol.2, September 2004.
6. M. Stilman et al. Navigation among movable obstacles: real-time reasoning in complex environments. *International Journal of Humanoid Robotics*, 02(04):479–503, December 2005.
7. Dongshin Kim, et al. Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In *2006 IEEE International Conference on Robotics and Automation (ICRA)*, pages 518–525, February 2006.
8. M. Stilman, et al. Planning and executing navigation among movable obstacles. *Advanced Robotics*, 21(14):1617–1634, January 2007.
9. M. Stilman. *Navigation Among Movable Obstacles.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, October 2007.
10. D. Nieuwenhuisen, et al. An Effective Framework for Path Planning Amidst Movable Obstacles. In *Algorithmic Foundation of Robotics VII*, Springer Tracts in Advanced Robotics, pages 87–102. Springer, Berlin, Heidelberg, 2008.
11. M. Stilman et al. Planning Among Movable Obstacles with Artificial Constraints. *The International Journal of Robotics Research*, 27(11-12):1295–1307, November 2008.
12. J. Van Den Berg, et al. Path Planning among Movable Obstacles: A Probabilistically Complete Approach. In *Algorithmic Foundation of Robotics VIII*, Springer Tracts in Advanced Robotics, pages 599–614. Springer, Berlin, Heidelberg, 2009.
13. Y. Kakiuchi, et al. Working with movable obstacles using on-line environment perception reconstruction using active sensing and color range sensor. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1696–1701, October 2010.
14. H. Wu, et al. Navigation Among Movable Obstacles in unknown environments. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1433–1438, October 2010.
15. Martin Levihn et al. Efficient Opening Detection. Technical Report, Georgia Institute of Technology, 2011.
16. M. Levihn. Navigation Among Movable Obstacles in unknown environments. Master's thesis, Georgia Institute of Technology, Atlanta, Georgia, May 2011.
17. M. Levihn, et al. Hierarchical Decision Theoretic Planning for Navigation Among Movable Obstacles. In *Algorithmic Foundations of Robotics X*, Springer Tracts in Advanced Robotics, pages 19–35. Springer, Berlin, Heidelberg, 2013.

18. M. Levihn, et al. Planning with movable obstacles in continuous environments with uncertain dynamics. In *2013 IEEE International Conference on Robotics and Automation*, pages 3832–3838, May 2013.
19. M. Levihn, et al. Foresight and reconsideration in hierarchical planning and execution. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 224–231, November 2013.
20. Thibault Kruse, et al. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, December 2013.
21. C. Clingerman et al. Estimating manipulability of unknown obstacles for navigation in indoor environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2771–2778, May 2014.
22. E. Mueggler, et al. Aerial-guided navigation of a ground robot among movable obstacles. In *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)*, pages 1–8, October 2014.
23. M. Levihn, et al. Locally optimal navigation among movable obstacles in unknown environments. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 86–91, November 2014.
24. J. Rios-Martinez, et al. From Proxemics Theory to Socially-Aware Navigation: A Survey. *International Journal of Social Robotics*, 7(2):137–153, April 2015.
25. C. Clingerman, et al. Dynamic and probabilistic estimation of manipulable obstacles for indoor navigation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6121–6128, September 2015.
26. N. Castaman, et al. A Sampling-Based Tree Planner for Navigation Among Movable Obstacles. In *Proceedings of ISR 2016: 47st International Symposium on Robotics*, pages 1–8, June 2016.
27. N. Castaman. A Sampling-Based Tree Planner for Robot Navigation Among Movable Obstacles. Master's thesis, University of Padova, Padova, Italy, July 2016.
28. S. K. Moghaddam et al. Planning Robot Navigation among Movable Obstacles (NAMO) through a Recursive Approach. *Journal of Intelligent & Robotic Systems*, 83(3):603–634, September 2016.
29. J. Scholz, et al. Navigation Among Movable Obstacles with learned dynamic constraints. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3706–3713, October 2016.
30. K. Charalampous, et al. Recent trends in social aware robot navigation: A survey. *Robotics and Autonomous Systems*, 93:85–104, July 2017.
31. H. Sun, et al. Semantic mapping and semantics-boosted navigation with path creation on a mobile robot. In *2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 207–212, November 2017.
32. Fabrice Jumel, et al. Context Aware Robot Architecture, Application to the RoboCup@Home Challenge. In *RoboCup symposium*, pages 1–12, Montreal, Canada, June 2018.
33. Z. Meng, et al. Active Path Clearing Navigation through Environment Reconfiguration in Presence of Movable Obstacles. In *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 156–163, July 2018.