IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 17 October 2022, accepted 22 December 2022, date of publication 2 January 2023, date of current version 11 January 2023.

Digital Object Identifier 10.1109/ACCESS.2022.3233765

RESEARCH ARTICLE

Navigation Among Movable Obstacles via Multi-Object Pushing Into Storage Zones

KIRSTY ELLIS, DENIS HADJIVELICHKOV, VALERIO MODUGNO, (Member, IEEE), DANAIL STOYANOV, (Senior Member, IEEE), AND DIMITRIOS KANOULAS, (Member, IEEE)
Department of Computer Science, University College London, WC1E 6BT London, U.K.

Corresponding author: Dimitrios Kanoulas (d.kanoulas@ucl.ac.uk)

This work was supported in part by the UKRI Future Leaders Fellowship (RoboHike) under Grant MR/V025333/1, in part by the Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/P012841/1, and in part by the CDT for Foundational Artificial Intelligence under Grant EP/S021566/1. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

ABSTRACT With the majority of mobile robot path planning methods being focused on obstacle avoidance, this paper, studies the problem of Navigation Among Movable Obstacles (NAMO) in an unknown environment, with static (i.e., that cannot be moved by a robot) and movable (i.e., that can be moved by a robot) objects. In particular, we focus on a specific instance of the NAMO problem in which the obstacles have to be moved to predefined storage zones. To tackle this problem, we propose an online planning algorithm that allows the robot to reach the desired goal position while detecting movable objects with the objective to push them towards storage zones to shorten the planned path. Moreover, we tackle the challenging problem where an obstacle might block the movability of another one, and thus, a combined displacement plan needs to be applied. To demonstrate the new algorithm's correctness and efficiency, we report experimental results on various challenging path planning scenarios. The presented method has significantly better time performance than the baseline, while also introducing multiple novel functionalities for the NAMO problem.

INDEX TERMS Motion and path planning, navigation among movable obstacles, mobile robots.

I. INTRODUCTION

With the rise of the fourth industrial revolution, mobile robots become robust and capable enough to complete autonomous tasks in real-world [1], [2], [3], [4], [5]. While the focus is mainly on designing methods that allow mobile robots avoiding heavy interactions with the environment, in this paper, we show that such interactions are useful. Most of the research in mobile robot path planning is focused on the problem of obstacle collision avoidance [6]. However, specific environmental conditions can be encountered that entice some form of interaction with the robot. Imagine the simple scenario of a person that needs to navigate in a kitchen; how many times do they need to push a chair towards a table, so that they can pass a narrow passage? One could also imagine more industrial cases, as visualized in Fig. 1, where a robot needs to clear and free a path to carry on with inspection tasks. Mike Stilman, dedicated a big part of his research life to solve this challenging, but important, problem, often called *Navigation Among Movable Obstacles* (NAMO) [7].

In [8], Stilman and Kuffner introduced a detailed formulation of the NAMO problem. In this work, we show that the robot's capability of manipulating movable obstacles (i.e., objects whose position in the space can be altered by the robot) can modify the structure of the robot's free Configuration Space (C-Space), notated as $\mathcal{C}_R^{free}$. We assume that the set of movable obstacles produces a segmentation of the robot C-Space into d disjoint subsets $\mathcal{C}_R^{free} = \{C_1, C_2, \ldots, C_d\}$. In this specific NAMO scenario, in which each obstacle is moved only once to connect two adjacent subsets C_i and C_j without interfering with the connection of any other two $\mathcal{C}_R^{free}$ subsets, becomes a k objects Monotonous Linear Problem (LP_k, M). It has been shown in [9] that even if we restrict the problem to (LP_1, M) with a polygonal convex representation for the obstacles and the robot, solving the NAMO problem is NP-hard.

The associate editor coordinating the review of this manuscript and approving it for publication was Yangming Li.

FIGURE 1. The case where a mobile robot following an inspection path (inside the green lines), unexpectedly detects an obstacle that needs to be pushed towards a storage space to clear and free the path.

Rearrangement Planning (RP) is another example where movable obstacles can be manipulated by a robot. RP differs from the NAMO problem as the goal positions for the moving obstacles are predefined and, usually, a final goal for the robot is not provided. Renowned examples of this problem are the Sokoban game [10], in which a character has to push a set of crates to predefined positions, and the Assembly Planning [11], where the robot-obstacles interaction is not considered and only the objects movements are planned.

In this paper, we aim to tackle a special NAMO definition that takes some elements from the RP formulation. In the proposed NAMO instance, both the robot and the obstacles have predefined goal positions. This problem can occur in many application scenarios, e.g., a robot navigating in a warehouse towards the desired goal while several boxes with known storage zones are misplaced and prevent the robot to reach its destination. In this case, we need to simultaneously plan for both removing all the possible robot path occlusions, while manipulating the obstacles to their aimed positions through a sequence of manipulation actions. If we exclude the trivial scenarios in which manipulating an object for connecting $\mathcal{C}_R^{free}$ subsets results in directly moving the obstacle to the desired goal position, it is easy to see that this particular NAMO instance is intrinsically Non-Monotonous (LP_k, NM), i.e., the obstacles need to be manipulated more than once. Moreover, in this work, we consider the case in which the robot can only push the obstacles, a common situation that arises when a wheeled mobile robot without a manipulator is investigated. This choice inevitably reduces the search space making the problem harder.

The paper is organized as follows: in Sec. II, we review the NAMO-related work. Paper contributions are stated in section III. Then, in Sec. IV, we describe the proposed method, and, in Sec. VI, we discuss the performance advantages and limits of the proposed method. Finally, in Sec. VII, we conclude with future problem directions.

II. RELATED WORK

The NAMO path planning problem was explored via a series of papers, by Stilman, that resulted in his Ph.D. thesis [7]. In [12], Stilman et al. introduced an algorithm for NAMO, applied to humanoid robots, where the obstacles could be grasped, picked, and placed to free paths. The proposed planner created a graph with the objective of connecting different robot C-Space subsets that are separated by obstacles. To preserve the linearity of the problem, the obstacles can only move in certain directions that do not interfere with the connection of other robot C-space subsets. This allowed the decomposition of the main navigation problem into different sub-problems.

In [13], Stilman and Kuffner extended the proposed algorithm for (LP_k, M) problems by using artificial constraints and reverse planning to limit the action space and improve the algorithm efficiency. In Okada et al. [14], proposed another algorithm for humanoid navigation with movable obstacles. Their planner resulted from the compositions of different specialized sub-planners, each of them dealing with different problems, such as manipulation, navigation, and motion planning for manipulation. Nieuwenhuisen et al. [15], proposed a tree-based planner method for computing both the robot motion and the manipulation actions for the obstacles. A Rapidly-exploring Random Tree (RRT) approach [16] was used to find the final position of the manipulandum. Therefore the proposed planner displays probabilistic completeness and produces non-smooth paths for movable obstacles.

In [17] the authors introduced an algorithm that is capable to solve both linear and non-linear NAMO tasks, independently of the problem's monotonicity. The method is based on a recursive approach that decomposes the original problem into sub-problems that can be solved by moving only one obstacle. While the proposed method is capable of dealing with a large class of NAMO instances it is affected by an elevated computational complexity that hampers the method's performance.

The aforementioned methods introduce offline procedures, while the methods that we propose in this paper is an online sensor-based procedure that is capable to find NAMO solutions with even partial knowledge of the operational environment.

The online NAMO problem has been the focus of several papers in the literature. The work in [18] is one of the first that attempts to address the problem of online NAMO. It is assumed that the robot operates on a two-dimensional grid, and only pushing is allowed for moving obstacles. To reduce the computational burden, each time new information is gathered by the robot, a new plan is computed only if the optimality of the current plan cannot be ensured. In [19] and [20], Levihn et al. introduced a novel online planner for uncertain discrete state space and action. The proposed planner is based on a hierarchical reinforcement learning strategy that is combined with a Monte Carlo Tree Search method for sub-task planning.

One of the main limitations of the online planners introduced so far, is that they have been designed to deal only with (LP_1) problems. Instead, our proposed method aims to deal with (LP_2, NM).

More recently, the NAMO problem has been extended to novel application contexts and utilized for different robotic platforms. While extending the method introduced in [18], the methods proposed in [21] and [22] introduced the concept of social awareness, while planning for NAMO. In these works, the authors defined new criteria that extend the original NAMO formulation, such as social movability evaluation, social placement choice, and social action planning. All those measures aimed at maximizing the safety and human acceptance of the choice made by the robot. In another work [23], the NAMO problem has been applied in the context of an emergency evacuation. It was shown that it is possible to reduce the evacuation time by creating new pathways for the humans that are leaving a dangerous environment. In [24] the role of computer vision is explored in order to facilitate the solution of NAMO problems. In particular, the problem of how affordances detection can be used to create open loop NAMO plans. In Raghavan et al. [25], proposed a simple pushed-based strategy for freeing paths using a wheeled quadruped robot.

All of these methods differ from our proposed algorithm which focuses on the problem of NAMO with storage areas that presents peculiar challenges.

III. CONTRIBUTIONS

Motivated by the work presented recently in [21] and [22], we further extend the framework in multiple directions to tackle the problem of NAMO with storage zones. In this paper, we extend the NAMO state-of-the-art in three directions:

1) we allow objects to be moved to storage spaces;
2) we tackle (LP_2, NM) problems, where up to 2 objects may be moved several times to connect free state space components with respect to the robot and obstacles' final positions, which naturally arise in the context of NAMO problem with predefined storage zones; and
3) we optimize the online path planning time-efficiency.

This work, focuses on the path planning problem, with the implemented algorithms applied to simulated wheeled robot navigation tasks (in Gazebo and NVIDIA Isaac Sim), including obstacle detection and localization.

IV. ALGORITHM

In this work, we study the navigation problem of a holonomic/omnidirectional mobile robot moving in a 2D workspace that might include both static and movable objects. The robot is placed at a starting position R_s and needs to plan and follow a path to a goal position R_g. The initial world map W, includes just the positions of the static objects (e.g., walls) in the environment, while the knowledge about newly detected obstacles (pose, size, movability) is updated when the sensory system (RGB-D cameras) of the robot allows it. During interaction with obstacles, the robot's action space is limited to pushing forward. Following the related work, we represent the robot and the obstacles as rectangles in a discretized grid space. The map is updated when obstacles are detected and localized, while the type of the obstacles (static or movable) is determined via the pushing interaction. We further consider the case where an obstacle might need to be moved first in order to free space for a second obstacle to be moved and free a path. Lastly, we consider storage zones, that can be specified for each obstacle, and pushing actions can be performed only to move them in those spaces. Below, we explain the complete algorithm in detail.

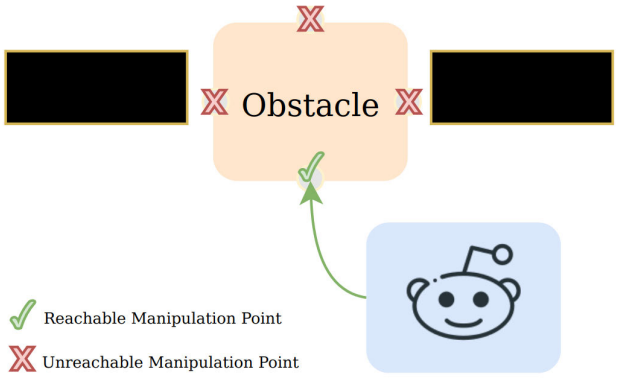


FIGURE 2. Obstacle manipulation points are found on each side of the obstacle. If no valid path is found from the robot's current pose to a manipulation point, the cost associated with reaching this point is considered infinite. Otherwise, the estimated cost is the sum of the cost from the robot to the manipulation point, and from the manipulation point to the goal.

A. THE BASELINE

In this section, we describe the state-of-the-art baseline method [21], which we extend in this paper. The path plan uses a search-based approach in a loop: the robot is sensing the environment, plans the optimal path using a search-based path planner (e.g., A* or similar) [6], and executes a single robot movement step. Then, these steps are repeated until the goal is reached or no solution exists. The overall algorithm is summarized in Alg. 1 and explained below:

1) WORLD SENSING

Given the original environment world map W, we generate an occupancy grid and add any new obstacles that are identified within the robot's Field-of-View (FoV). Those could be potentially movable. During pushing actions, we evaluate the actual ability of the robot to manipulate the obstacle, and if it cannot be pushed, it is marked as static. The generated world state W is stored during the sensing process. Further details about the particular implementation chosen for this are included in Sec. V.

2) PATH PLANNING

After sensing the environment, an initial plan P_{opt} is generated, using a search-based path planner, e.g., A* [6]. If P_{opt}

Algorithm 1: NAMO Algorithm

Require: Starting World State (W), Starting Robot Position (R_s), Goal Robot Position (R_g);
W = `SenseWorld()`;
M = `GenerateOccupancyMap(`W`)`;
P_{opt} = `GetPlanTo(`R_g`,` M`)`;
isSuccess = *True*;
while *True* **do**
  if *Goal is reached* **then**
    return 1 ▷ Outcome: Success!
  end
  W = `SenseWorld()`;
  P_{opt} = `Think(`P_{opt}`,` *isSuccess*`,` W`)`;
  if P_{opt} *is invalid* **then**
    return 0 ▷ Outcome: Unreachable!
  end
  isSuccess = `Act(`P_{opt}`)`;
end

is invalid, i.e., paths are of infinite cost or go through static obstacles, then new path plans are generated considering alternative paths, both with and without obstacle manipulation (in this work, by manipulation we consider only pushing actions). For each obstacle, we consider four action points in the center of each of the obstacle's vertical faces. Plans are generated for each reachable action point (i.e., see Fig. 2) with an associated cost based on the search-based path distance from the manipulation point R_{o_i} of object o to the goal R_g. If no valid path is found from the robot's current pose to a manipulation point, the cost associated with reaching this point is considered infinite. The obstacle (and its action point) with the lowest estimated cost is selected.

Simulated execution of each possible pushing action is performed, until a free path is found, generating simulated plans P_{sim}. The action plan cost is calculated as a three-part segment path summing up the path to the action point (C_1), action path (C_2), and path from the free opening[1] to the goal (C_3), as follows (a visual example is shown in Fig. 3):

$$C_{R_s \to R_{o_i} \to R_g} = C_1 + C_2 + C_3 \tag{1}$$

The optimal plan P_{opt} is then updated to the simulated plan with the lowest cost. The algorithm for this step is shown in Alg. 2.

3) EXECUTE PLAN

The next desired pose in the P_{opt} plan is returned by the path planning step above. A transformation is generated from the current robot pose to the desired pose. The action step is considered successful if the desired pose is reached. If the push is unsuccessful, e.g., pushes it out of the planned way, the robot will first attempt to complete the plan. In the next iteration of the NAMO algorithm, the agent updates the World state, which would also account for the failed push and the new obstacle location. Thus, the agent will re-attempt to move the obstacle. The algorithm for this step is shown in Alg. 3.

[1]In this paper, an opening refers to the space gap that is freed when an obstacle is pushed.

Algorithm 2: Path Planning

Require: Plan P_{opt}, Flag *isSuccess*, World W;
if *not isSuccess* **then**
  Mark last obstacle as static
end
if *Plan* P_{opt} *is valid* **then**
  return P_{opt};
end
else
  M = `GenerateOccupancyMap(`W`)`;
  P_{opt} = `GetPlanTo(`G_s`,` M`)`;
  Plans = $\emptyset$;
  foreach *obstacle in* M **do**
    Plans = *Plans* ∪ `PlanActions()`;
  end
  Choose obstacle with lowest cost plan
  foreach *Action on Obstacle* **do**
    P_{sim} = `SimulateActionPlan(`M`)`
    if P_{sim} *is better than* P_{opt} **then**
      $P_{opt} \leftarrow P_{sim}$
    end
  end
  return P_{opt};
end

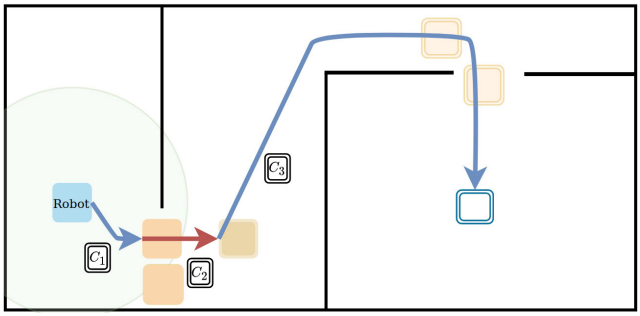


FIGURE 3. An example plan proposal (abstraction). The algorithm considers a path with a pushing action. The path cost is composed of the three path segments' costs. The green area depicts the sensors' FoV range, orange blocks represent the seen and unseen obstacles. The robot selects the path with the lower estimated cost.

B. BASELINE PERFORMANCE ENHANCEMENTS

In this work, we propose updates that improve the performance of the state-of-the-art baseline method. In particular, we aim at decreasing the time taken to find solutions to NAMO problems, in the following ways.

1) ONE-STEP, UNTIL OPENING

The original algorithm was simulating pushing action steps until a collision was detected. Instead, we plan a step until an opening is detected. In particular, once an obstacle has

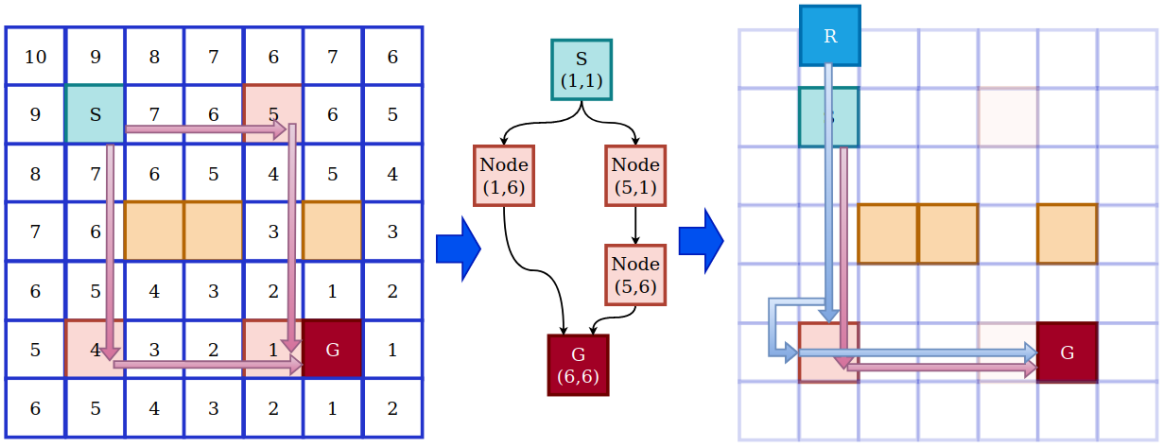


FIGURE 4. Route graph algorithm applied to obstacle pushing – optimal paths are found via wavefront grid distance calculation that alternates *x*- and *y*-axis movements. The optimal path with the fewest directional changes is chosen; and the robot plans pushing movements around the planned obstacle manipulation path.

Algorithm 3: Execute Plan

Require: Plan P_{opt};
$Pose_{next} = \texttt{next}(P_{opt})$;
$Pose_{current} = \texttt{GetCurrentPose}()$;
$isSuccess = \texttt{Move}(Pose_{next}, Pose_{current})$;
return *isSuccess*;

been detected, re-planning is executed. If a plan is found that includes manipulation of an obstacle, then this section of the path, C_2, is generated by simulating push steps. For each step that is simulated, a check is made to find if a path to the goal is now available. In the baseline method implementation, the step simulation was continued until the obstacle collided with a static part of the environment. This meant that the simulation step could take a long time. In our method, the steps are only simulated until a path to the goal is found, then this part of the routine exits.

2) SAVE THE PLAN FOR NEXT BEST OBSTACLE, WITHOUT RE-PLANNING

In the baseline method implementation, if two obstacles had been detected, plans would be made for each obstacle and the plan with the lowest cost would be selected. If the robot went on to execute this plan and found that the obstacle was in fact non-movable/static, the planning stage starts and plans are generated for the second obstacle, again. Thus, the algorithm repeats for work that has already been done. While for two obstacles this might not be an issue, the computation required grows with the number of obstacles.

By storing the previously calculated plans associated with each obstacle, the computation required to re-plan is minimized – the robot only needs to recalculate the first path of the plan (from the current robot position to the obstacle action point) as its starting location will have changed, while keeping the remainder.

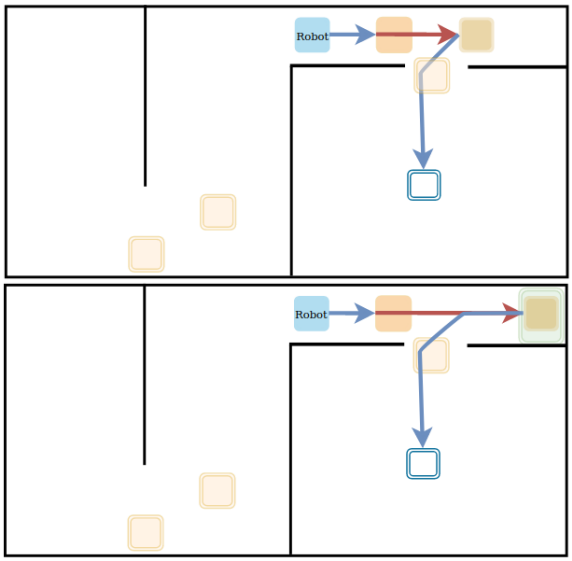


FIGURE 5. With vs without storage zones: the robot prefers to put the obstacle into its designated area when possible.

3) PREFERENCE AROUND OBSTACLES

In the baseline method implementation, when planning for an obstacle, a plan around the object is generated, as well as plans where the obstacle is moved, rather than just selecting the alternative path - around the obstacle. In our implementation, if a path is available around the detected obstacle, it is preferred and selected.

C. BASELINE FUNCTIONAL ENHANCEMENTS

Apart from making the state-of-the-art baseline method faster, we also propose a set of new features described in this section.

1) STORAGE ZONES

The baseline method considers designated zones in which manipulated obstacles are not allowed (i.e., taboo zones). For

example, a zone could be added in front of a doorway to prevent door obstruction. We extend this by considering the opposite case, where an area could be designated as an area where obstacles should be stored (i.e., storage zones). In this way, when an obstacle blocking a path needs to be moved away, the obstacle is not left in some unknown state, but at a place of storage. See an abstract example in Fig. 5 and a simulated run in Fig. 7.

The storage zones are firstly pre-defined and represented as polygon shapes. When the world map W is generated, each potentially movable obstacle is assigned to the closest storage zone. A path is planned from the starting R_s to the goal R_g robot position. If a new obstacle is detected, the plan with the lowest cost is searched for, with A*. If the plan includes the push action for an obstacle, a plan to move the obstacle to the storage zone is made. The storage zone linked to the obstacle is checked to find a space for it. Then, the space with the minimum number of direction changes for the robot is picked. Simulation steps are carried out to see if after each step the obstacle is within the storage zone. If this cannot be achieved, other options are explored, such as planning for another obstacle. With the current implementation, if an object cannot be put into storage and there are no further options for the robot (alternative paths or moving other movable obstacles), the goal is considered unreachable. In the opposite case, when a storage zone can be reached, the planned trajectory points are sent to the robot. The obstacle is added to the storage zone and the available space in the storage zone is updated. The robot should complete the plan, pushing the box to the storage zone, backing away from the obstacle to avoid collisions with it, and continuing on its path to the goal.

2) ALTERNATING PUSH DIRECTIONS

In the baseline algorithm, obstacle movement is limited to pushing in a straight line along either the x or y axis of the world map W. To consider more complex movements, such as moving an obstacle to a storage zone, we need to integrate alternative actions in the planner. One option would be to be able to physically stick an obstacle to the real robot, which is difficult to do or use a manipulator to pick up the obstacles. Instead, we added a new feature in the NAMO framework. With our new path planning enhancement, if an obstacle's closest storage zone cannot be accessed by a simple motion along a single world axis, a plan is generated by including an alternating sequence of pushes along the x- and y-axes. This pushing method is based on the route graph algorithm described in [26] and enables a path to be generated for the manipulation phase of the planning, which is not restricted to single axis motions.

The obstacle pushing algorithm is only used to plan pushes to storage zones. Firstly, a wavefront grid of the world is generated along with a node tree representation. Each grid cell is evaluated to find the minimum distance to the goal. A path needs to be found with the minimum number of direction changes. Starting from the first grid square, we select a neighboring square in either the x or y direction. We continue moving in a straight line in the grid, and once the cost of the grid square stops decreasing, we add this grid location as a node in the node tree. We then change direction and repeat the process until the goal cell is reached. We select the path from the start to the goal with the least number of nodes, retrace this path, and add all grid cells to the plan.

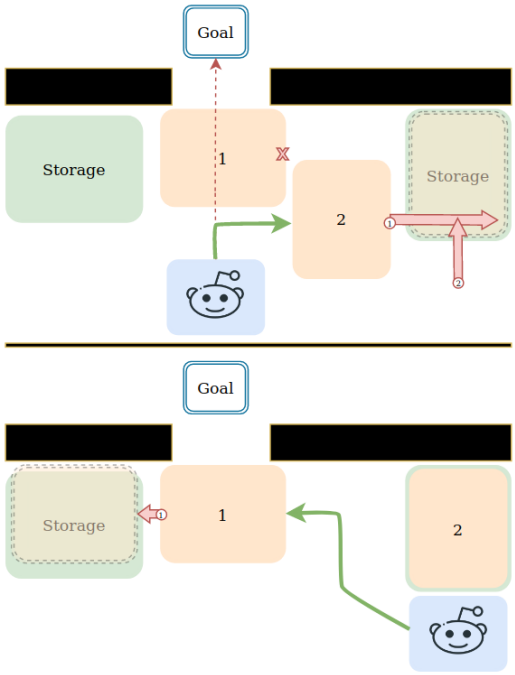


FIGURE 6. Multi-object displacement with storage zones: the robot cannot reach its goal without manipulating both obstacles. Removing object 2 reveals a manipulation point on object 1. The robot plans how to push both into designated storage zones while clearing out a path to the goal.

This algorithm provides the path that the obstacle needs to follow (not the robot itself). Thus, the robot path needs to be generated too. In the simple case that the obstacle movement is in a straight line, the robot's path is the same as the obstacle's planned path, with an offset. When a direction change is needed, the robot retreats from the obstacle to avoid a collision and moves around the edge of the obstacle to approach the next manipulation point. For each section of the obstacle push plan, the robot plan is made up of straight lines with offsets to trail the robot behind the obstacle and 're-positioning' sections to move the robot to the next manipulation point in order to push the obstacle along the perpendicular direction. If the obstacle plan starts with a motion towards the robot's current position, an A* path to the manipulation point on the opposite face of the obstacle is added to the robot plan. The grid-based path is converted to real-robot motions and the complete plan is sent to the robot as a sequence of trajectory points. An example of how the path is found and implemented is visualized in Fig. 4.

3) THE TWO-OBSTACLES PROBLEM (LP_2)

Anytime a manipulation is simulated as a sequence of steps in a straight line, a check is made on each step to see if

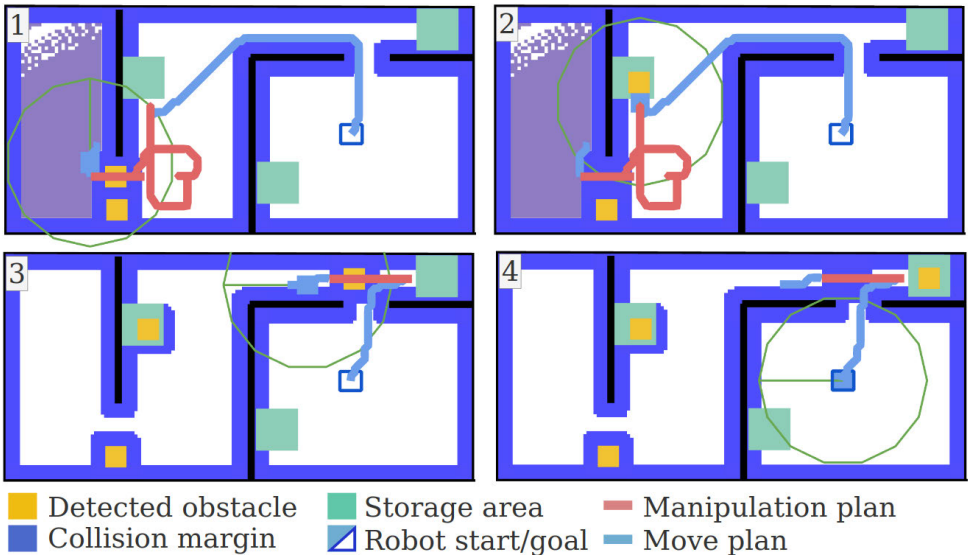


FIGURE 7. Example of using allocated storage areas during NAMO objective taken from a simulated run: (1) the robot plans how to push the object into the storage space; (2) executes that plan; (3) discovers a new obstacle and plans how to deal with it; and (4) finally reaches the goal.

either (i) a new path can be found to the goal or (ii) any manipulation points on the rest of the detected obstacles that are blocking the current path would become accessible. If a manipulation point would become accessible, we check if moving it would open up a path to the goal. If a plan cannot be found, another obstacle is selected. This process can be performed in a recursive fashion to be extended to any number of additional obstacles. An abstraction of this behavior is shown in Fig. 6 showcasing a scenario that could not be solved by the baseline implementation. Simulation results are shown in the experimental section.

V. EVALUATION

In this section, we demonstrate the behavior resulting from our method and highlight the significant change in performance for solving NAMO tasks.

A. SETUP

1) ENVIRONMENT

We set up a simulation environment in Gazebo and NVIDIA Isaac Sim consisting of a walled room with a winding corridor leading to an open area. The environment presents opportunities for various obstacle arrangements blocking the robot's paths. The simulators were chosen as they would make the transfer to the real robot and also testing more complex scenarios that require photo- and physic-realism easier in future work. As obstacles, we use standard cardboard boxes of variable sizes, placed in ways that block the path to the navigation goal and make it impossible to reach without manipulating them - we do not consider paths that can be solved without manipulation, as they can easily be solved without NAMO. For all simulations, we use the Robotnik Summit XLS robot - an omnidirectional wheeled mobile platform. The action space consists of movement direction and velocity. The observation space is defined by the robot's exteroceptive sensors which have a limited Field-of-View (FoV) – LiDARs and RGB-D cameras (LiDAR is used for SLAM, while RGB-D for obstacle detection). Success in this environment is defined as the robot reaching the specified goal location.

2) OBSTACLE DETECTION

Given the original environment world map W, the 2D LiDAR sensors generate an occupancy grid representation to map new static parts of the environment while the robot is moving and simultaneously localize the robot in it. In this work, this is achieved with OctoMap [27] and GMapping [28]. We use DOPE [29] trained on finding cardboard boxes to identify objects in the sensor's FoV, that are potentially movable.

B. SIMULATED RESULTS

Following the evaluation of Wu et al. [18], we showcase simple examples of our method's behavior and compare the time performance with the baseline.

1) BEHAVIOURAL SIMULATION

To showcase the use of Storage Zones and the alternating axis pushing, we set up obstacles along the path to the robot's goal and designated storage zones that require pushes in more than one direction. One such run is shown in Fig. 7, where the robot successfully navigates to the goal, storing all pushed obstacles along the way. Similarly, we set up scenarios where multi-object interaction is required (LP_2). In the example shown in Fig. 8, the robot successfully moves the obstacles away from the path and reaches the goal.

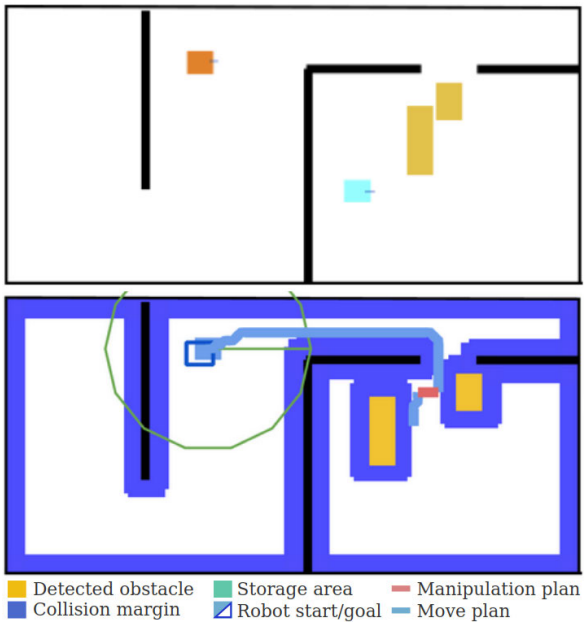


FIGURE 8. Example of multi-object manipulation: the initial state (top) and the last plan after the robot has moved away from the obstacles (bottom).

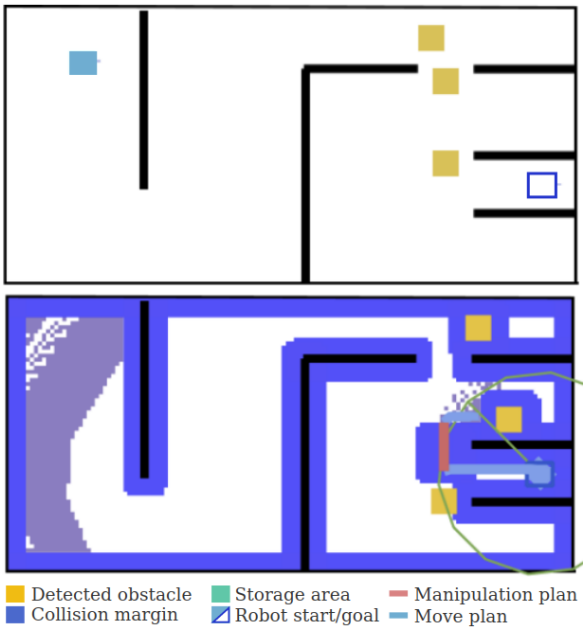


FIGURE 9. In this scenario, the robot successfully pushes a box to the side instead of forward, in order to preserve enough space for the narrow passage to the goal.

Overall, our method was successful in all standard storage-zone scenarios – the method was able to solve the task without getting stuck and pushed all boxes into the appropriate storage zones. However, there were two failure cases that come from these scenarios: 1) when there is an unseen obstacle directly behind another, in which cases manipulating both obstacles at the same time is the only viable solution; 2) when the obstacle is close to a wall that it needs to be pushed away from – requiring a manipulation action located in a point different from the four vertical face center points. An interesting scenario is shown in Fig. 9, where the second obstacle could have been pushed more down, which would have obstructed the robot's path to the goal. Instead, the robot decides to push it to the side. Note that in this case, the optimal solution could have been for the robot to push both of the last two boxes together with a single manipulation action.

2) TIME PERFORMANCE

To showcase the speed improvement of the method, we compare the times to finish a navigation task with the baseline [21]. We set up the environment with different obstacle configurations that block the path to the goal. For this experiment, all obstacle configurations can be solved without multi-object manipulations (LP_1) to allow a fair comparison with the baseline. Since the proposed method is an improvement over the baseline, in any other scenario which doesn't make use of our proposed optimizations, the performance is the same. Thus, we focus on three significant variations of this task with the single object manipulation constraint: (i) a simple scenario where one movable obstacle must be moved to clear a path; (ii) a scenario with two obstacles blocking an opening, one of the obstacles being static, the other movable; in this scenario, the robots will first attempt to move the closer obstacle which is static, before moving on the movable one; (iii) a scenario where the robot must pass through obstacle blockages, and must push the obstacles in a way that doesn't block its future path. These local configurations are standard for the NAMO problem [12], [18].

TABLE 1. Time comparison between the baseline and our improved method.

Test	Baseline (s)	Ours (s)
#1	54.28	**11.01**
#2	60.00	**15.36**
#3	104.09	**20.79**
Average	72.9 ±22.26	**15.69**±3.96

Our method requires on average 78.45% less time to solve a NAMO task than the baseline. The results are shown in Table 1. We observe that the baseline wastes more time on re-planning as well as pushing obstacles until collision, rather than until a path is freed. The most significant delay in the process is the one-step simulation updates performed for the manipulated obstacles.

VI. DISCUSSION

There are multiple reasons why the proposed method performs significantly quicker than the baseline: (i) while the baseline simulates pushing action steps until a collision was detected, our method plans steps until a new opening is detected; (ii) the plans to reach the next-best obstacle are kept during the plan selection phase, rather than re-planning after each obstacle-move attempt that fails; and (iii) and

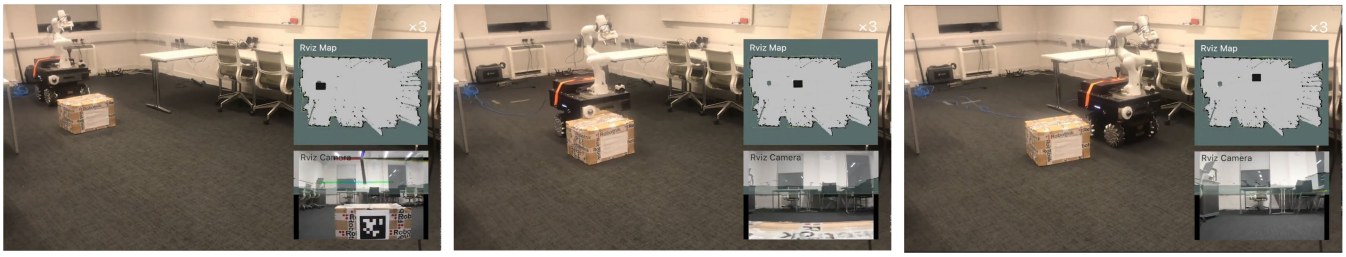


FIGURE 10. Preliminary real-world NAMO experiments, where a mobile robot pushes a movable obstacle to clear the path towards the goal (as part of the sim2real photorealistic method, introduced in [30]).

periodically checks if the path to the goal has been freed up, i.e., if a plan around the obstacle is available, the agent prefers it over moving an obstacle. Additionally, with the wavefront obstacle movement, the method is able to solve quicker the more complex scenarios that require moves in multiple directions.

The work proposes a versatile method that can produce more robust and efficient NAMO path plans. However, there it is still limited in movements that require pushing in non-standard obstacle faces (e.g., pushing an obstacle diagonally), and pushing of multiple objects at the same time, which could also require a more sophisticated prediction of the interaction. While the method works as a practical solution to a simplified version with two assumptions (i.e., single obstacle pushing in the world's xy-axes), it is essential that these limitations are addressed for a more general practical solution.

VII. CONCLUSION AND FUTURE WORK

In this paper, we study the problem of Navigation Among Movable Obstacles (NAMO), demonstrating several extensions and improvements over the prior state-of-the-art work. We propose methods that: (1) allow obstacle movement via pushing along more complicated trajectories, instead of single-axis movements; (2) add storage zones to obstacles when they are moved, which highlights the practical use of NAMO in the real-world; and most importantly (3) allow multi-object manipulation capabilities that can be used to solve challenging problems that prior work failed to (we have tested with two obstacles, leaving the application to more as future work). Moreover, the method's time performance presents a significant improvement over the baseline.

There are multiple directions that this work can be extended. First, we will work on real-robot experiments that need to be demonstrated. We have presented some preliminary real-world experiments in the sim-to-real solution to the NAMO problem in [30] (see Fig. 10). Secondly, an interesting direction is the manipulation of multiple objects at the same time, as well as focusing on manipulation points that are less predictable – different than the vertical face centers. Additionally, we are considering assigning obstacles to storage zone based on semantic connection instead of minimum distance, e.g., associating a chair with a dining room. Last, moving from 2D to 3D environments that might also be dynamic will allow the application of NAMO to real-world settings.

REFERENCES

[1] I. D. Miller, F. Cladera, A. Cowley, S. S. Shivakumar, E. S. Lee, L. Jarin-Lipschitz, A. Bhat, N. Rodrigues, A. Zhou, A. Cohen, A. Kulkarni, J. Laney, C. J. Taylor, and V. Kumar, "Mine tunnel exploration using multiple quadrupedal robots," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2840–2847, Apr. 2020.

[2] A. Agha et al., "NeBula: Quest for robotic autonomy in challenging environments; TEAM CoSTAR at the DARPA subterranean challenge," 2021, *arXiv:2103.11470*.

[3] J.-K. Huang and J. W. Grizzle, "Efficient anytime CLF reactive planning system for a bipedal robot on undulating terrain," 2021, *arXiv:2108.06699*.

[4] M. T. Ohradzansky, E. R. Rush, D. G. Riley, A. B. Mills, S. Ahmad, S. McGuire, H. Biggie, K. Harlow, M. J. Miles, E. W. Frew, C. Heckman, and J. S. Humbert, "Multi-agent autonomy: Advancements and challenges in subterranean exploration," 2021, *arXiv:2110.04390*.

[5] J.-K. Huang, Y. Tan, D. Lee, V. R. Desaraju, and J. W. Grizzle, "Informable multi-objective and multi-directional RRT system for robot path planning," 2022, *arXiv:2205.14853*.

[6] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge Univ. Press, 2006.

[7] M. Stilman, "Navigation among movable obstacles," Ph.D. dissertation, MIT, Cambridge, MA, USA, 2007.

[8] M. Stilman and J. Kuffner, "Planning among movable obstacles with artificial constraints," in *Proc. Workshop Algorithmic Found. Robot. (WAFR)*, 2006, pp. 119–135.

[9] G. Wilfong, "Motion planning in the presence of movable obstacles," *Ann. Math. Artif. Intell.*, vol. 3, no. 1, pp. 131–150, 1991.

[10] A. Junghanns and J. Schaeffer, "Sokoban: A challenging single-agent search problem," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 1997, pp. 1–10.

[11] M. H. Goldwasser, "Complexity measures for assembly sequences," Ph.D. dissertation, Stanford, CA, USA, 1998.

[12] M. Stilman and J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," in *Proc. 4th IEEE/RAS Int. Conf. Humanoid Robots*, vol. 1, Nov. 2004, pp. 322–341.

[13] M. Stilman and J. Kuffner, "Planning among movable obstacles with artificial constraints," *Int. J. Robot. Res.*, vol. 27, nos. 11–12, pp. 1295–1307, 2008.

[14] K. Okada, A. Haneda, H. Nakai, M. Inaba, and H. Inoue, "Environment manipulation planner for humanoid robots using task graph that generates action sequence," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 2, Sep. 2004, pp. 1174–1179.

[15] D. Nieuwenhuisen, A. F. van der Stappen, and M. H. Overmars, *An Effective Framework for Path Planning Amidst Movable Obstacles*. Berlin, Germany: Springer, 2008, pp. 87–102.

[16] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

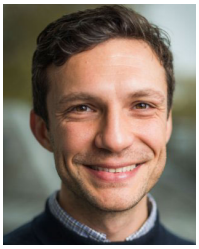
[17] S. K. Moghaddam and E. Masehian, "Planning robot navigation among movable obstacles (NAMO) through a recursive approach," *J. Intell. Robot. Syst.*, vol. 83, nos. 3–4, pp. 603–634, Feb. 2016.

[18] H.-N. Wu, M. Levihn, and M. Stilman, "Navigation among movable obstacles in unknown environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 1433–1438.
[19] M. Levihn, J. Scholz, and M. Stilman, "Hierarchical decision theoretic planning for navigation among movable obstacles," in *Algorithmic Foundations of Robotics X*, E. Frazzoli, T. Lozano-Perez, N. Roy, and D. Rus, Eds. Berlin, Germany: Springer, 2013, pp. 19–35.
[20] M. Levihn, J. Scholz, and M. Stilman, "Planning with movable obstacles in continuous environments with uncertain dynamics," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 3832–3838.
[21] B. Renault, J. Saraydaryan, and O. Simonin, "Towards S-NAMO: Socially-aware navigation among movable obstacles," 2019, *arXiv:1909.10809*.
[22] B. Renault, J. Saraydaryan, and A. O. Simonin, "Modeling a social placement cost to extend navigation among movable obstacles (NAMO) algorithms," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 11345–11351.
[23] M. Nayyar and A. R. Wagner, "Aiding emergency evacuations using obstacle-aware path clearing," in *Proc. IEEE Int. Conf. Adv. Robot. Social Impacts (ARSO)*, Jul. 2021, pp. 7–14.
[24] M. Wang, R. Luo, A. O. Onol, and T. Padir, "Affordance-based mobile robot navigation among movable obstacles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 2734–2740.
[25] V. S. Raghavan, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Reconfigurable and agile legged-wheeled robot navigation in cluttered environments with movable obstacles," *IEEE Access*, vol. 10, pp. 2429–2445, 2022.
[26] E. F. Parra-Gonzalez, J. G. Ramirez-Torres, and G. Toscano-Pulido, "A new object path planner for the box pushing problem," in *Proc. Electron., Robot. Automot. Mech. Conf. (CERMA)*, Sep. 2009, pp. 119–124.
[27] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auto. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013.
[28] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007.
[29] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Proc. Conf. Robot Learn.*, 2018, pp. 1–11.
[30] K. Ellis, H. Zhang, D. Stoyanov, and D. Kanoulas, "Navigation among movable obstacles with object localization using photorealistic simulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 1711–1716.

KIRSTY ELLIS received the B.Eng. degree in mechanical engineering from the University of Glasgow, in 2008, supervised by Prof. Ron Thomson, and the Ph.D. degree from the Wolfson School of Engineering, Loughbough University, in 2014, supervised by Dr. Jon Roberts. The research topic of her Ph.D. was minimizing vibration in a flexible golf club during robotic simulations of a golf swing. She spent eight years working in industry in a number of software engineering roles, including a role as a Robotics Software Engineer at Shadow Robot Company, where she worked on the development of different robotic end effectors and teleoperation robots. She was a Full-Stack Software Engineer but mostly enjoys writing low-level software and firmware. In October 2020, she decided to make the switch from industry back to academia and joined as a Postdoctoral Research Fellow with the Robot Perception and Learning (RPL) Laboratory, Computer Science Department, UCL. Her research interest includes navigation among movable obstacles.

DENIS HADJIVELICHKOV is currently pursuing the Ph.D. degree with the Robot Perception and Learning Laboratory, University College London (UCL). He worked on robot whole-body control, reinforcement learning, and object detection and recognition. His research interests include robot vision, self-supervision, and skill acquisition through affordance imitation.

VALERIO MODUGNO (Member, IEEE) received the Ph.D. degree from the Sapienza University of Rome, in 2017. He was a Visiting Researcher at the Technical University of Darmstadt, in 2014, and the INRIA Grand-Est Nancy, in 2015. He was a Postdoctoral Researcher for four years at the Sapienza University of Rome, under the supervision of Prof. Giuseppe Oriolo. He joined University College London (UCL), in 2022, where he is currently a Research Fellow with the RPL Laboratory. His research interests include comprise humanoid whole-body control, optimal control, teleoperation for legged robots, reinforcement learning, black-box optimization, and safety for control and learning strategies. He received the Starting Research Grant from the Sapienza University of Rome, in 2021.



DANAIL STOYANOV (Senior Member, IEEE) received the Ph.D. degree from the Hamlyn Centre for Robotic Surgery, Imperial College London, London, U.K., in 2006. He has been the Director of the Wellcome/EPSRC Centre for Interventional and Surgical Sciences (WEISS) and the Chair of Emerging Technologies, Royal Academy of Engineering, London, since 2019. He joined the Centre for Medical Image Computing, University College London (UCL), London, in 2011. He is currently a Professor of robot vision with the Department of Computer Science, UCL. His research focuses on robotics and artificial intelligence applied to surgery, especially around surgical video analysis and understanding. This research has been translated into several companies, including Digital Surgery, London (acquired by Medtronic, in 2020), where he was a Chief Scientist, and Odin Vision Ltd., London, which he co-founded, in 2019. He is a fellow of IET. He received the Royal Academy of Engineering Research Fellowship, from 2009 to 2014, and the EPSRC Early Career Research Fellowship, from 2017 to 2022. He is the Program and General Chair of the IPCAI and MICCAI Conference Series. He was an Associate Editor of IEEE Robotics and Automation Letters. He is an Associate Editor of IEEE Transactions on Biomedical Engineering. He is a Deputy Editor of the *International Journal of Computer Assisted Radiology and Surgery*.

DIMITRIOS KANOULAS (Member, IEEE) received the Ph.D. degree from Northeastern University, Boston. He was a Postdoctoral Researcher at the Italian Institute of Technology for five years. He is currently an Associate Professor of robotics and computation with the Department of Computer Science, University College London (UCL), and a UKRI Future Leaders Fellow (FLF). He worked on several real-world humanoid and animaloid robots. He has published more than 50 research papers in high-impact robotic journals and conferences. His research interests include robot perception, planning, and learning. He received the Best Interactive Paper Award from IEEE Humanoids 2017, the Best Student Paper Award Finalist from IEEE ICARCV 2018, the Outstanding Associate Editor Award from IEEE/RSJ IROS 2022, and the U.K.-RAS Early Career Award 2022.