# Navigation Among Movable Obstacles with Object Localization using Photorealistic Simulation

Kirsty Ellis, Henry Zhang, Danail Stoyanov, and Dimitrios Kanoulas

***Abstract*— While mobile navigation has been focused on obstacle avoidance, Navigation Among Movable Obstacles (NAMO) via interaction with the environment, is a problem that is still open and challenging. This paper, presents a novel system integration to handle NAMO using visual feedback. In order to explore the capabilities of our introduced system, we explore the solution of the problem via graph-based path planning in a photorealistic simulator (NVIDIA Isaac Sim), in order to identify if the simulation-to-reality (sim2real) problem in robot navigation can be resolved. We consider the case where a wheeled robot navigates in a warehouse, in which movable boxes are common obstacles. We enable online real-time object localization and obstacle movability detection, to either avoid objects or, if it is not possible, to clear them out from the robot planned path by using pushing actions. We firstly test the integrated system in photorealistic environments, and we then validate the method on a real-world mobile wheeled robot (UCL MPPL) and its on-board sensory and computing system.**

## I. INTRODUCTION

Robotic systems need to navigate in several kinds of real-world environments, in order to complete complex meaningful tasks. Mobile robot navigation and path planning has been extensively studied in the past, mainly to resolve the problem of obstacle avoidance (either those are static or dynamically moving around) [1]. However, there are many unpredictable factors in real life applications. For instance, unexpected obstacles might block a planned path. Imagine a cardboard box blocking a narrow passage in a warehouse. Such obstacle may not be able or need to be avoided, but rather being moved away in order to free the passage. It is for this reason that developing the ability for a robot to interact with and move proactively among movable obstacles is an interesting open problem to address (i.e., in this paper, obstacles that can be pushed away).

The aforementioned problem is called Navigation Among Movable Obstacles (NAMO), that was heavily studied by Mike Stilman, via a series of papers that started in 2004 [2]. The original focus was on humanoid robots. The NAMO problem is very complex (NP-Complete in its simplest form), since it has a large search-space and a configuration space that changes over time. Thus, approximated solutions are aimed. To date, there are very few systems that implement full NAMO autonomy for mobile robots. As discussed in the related work below, the very few works on the topic focused usually on a part of the problem (e.g., visual servoing control or path planning), while very few were tested on real hardware and in real-time. Solving the NAMO problem in the real world, requires four parts, that we try to resolve and integrate in a unified system in our work:

1) detecting and localizing movable obstacles in the environment,
2) plan the path to a goal, considering actions –pushing, in our case– with the movable obstacles,
3) resolving the robot localization and mapping part during navigation/pushing, and
4) implementing the whole system on a real mobile robot.

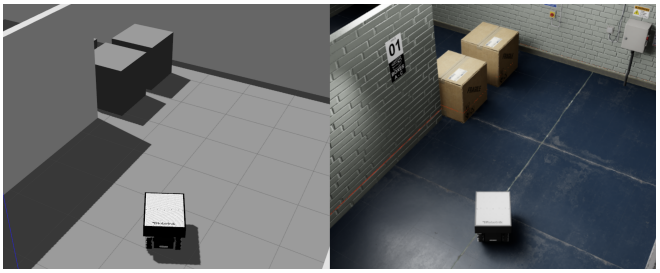


Fig. 1: Gazebo (left) and NVIDIA Isaac (right) simulated warehouse environments, with boxes blocking the corridors.

Regarding the first problem (movable obstacle detection and localization), several deep learning methods can be used to localize obstacles with RGB-D or other exteroceptive sensory methods. There are two difficulties to consider though. Firstly, the majority of deep learning methods require the collection of a large number of high quality real-world data. Datasets for training can alternatively be produced in simulation with augmentations, but then sim2real techniques need to further be applied. In this paper, we take advantage of the recently developed photorealistic NVIDIA Isaac simulator, to be able to resolve easier the production of large datasets for training, as it has been used in [3]. This approach has been studied with different simulators as well [4], [5]. What we achieved is an easy sim2real problem solution, by demonstrating direct application of the visual methods on the real robot. This visual ability could not be achieved with standard simulators as Gazebo (see Fig. 1). Secondly, some standard works in NAMO were considering only visual obstacle detection and localization [6], [7]. In this paper, we further investigate if an obstacle is really movable by interacting with it via pushing. In case an obstacle is too heavy or fixed, and, thus, cannot be moved around, we update our planned path to consider alternative solutions.

Regarding the second problem, in this paper, we improve

The authors are with the Department of Computer Science, University College London, Gower Street, WC1E 6BT, London, UK. d.kanoulas@ucl.ac.uk

This work was supported by the UKRI Future Leaders Fellowship [MR/V025333/1] (RoboHike) and the Engineering and Physical Sciences Research Council (EPSRC) [EP/P012841/1].

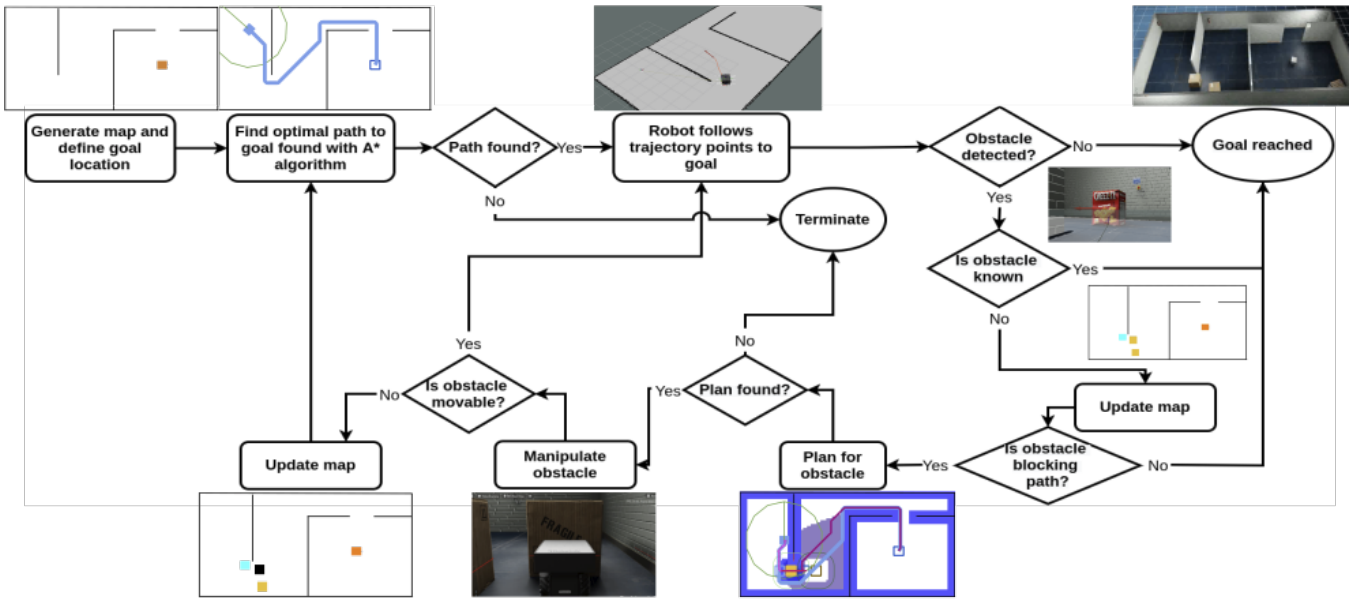



Fig. 2: Our developed Navigation Among Movable Obstacles (NAMO) pipeline.

the method introduced in [8], by adding the aforementioned visual and interactive obstacle detection and localization solution into the A* planner, as well as improving the path planning sequence to avoid robot drifts. The whole object detection and path planning system is combined with a localization and obstacle mapping technique, to keep an updated map of the surrounding environment, in order to solve the third problem above. Last, taking advantage of the photorealistic simulator, we apply the method on several simulated and preliminary real-world scenarios in obstacle-unknown environments.

For our testing, we consider a warehouse environment, where cardboard boxes are typical items and might be left in passages that prevent free-route navigation and need to be manipulated (e.g., pushed). The proposed system pipeline is shown in Fig. 2. In the next section (Sec. I-A), we review related work of NAMO. Then, in Sec. II, we describe the obstacle detection and localization method, the path planning adaptation, and the robot localization and mapping solution. In Sec. III, we present the experimental validation of the introduced pipeline in simulation and on the real robot. Finally, in Sec. IV, we conclude by discussing future problem directions.

### A. Related Work

Navigation Among Movable Obstacles (NAMO) is a problem that was heavily studied theoretically by Mike Stilman in 2004-2007 [2], [9], [10]. Being identified as an NP-Hard problem in its simple formulation, several optimization techniques using graphs were introduced, e.g., [11], [12], [13]. All those works considered the problem on mobile robots either theoretically or applied under assumptions, for instance, environments were known or open-loop controllers were used. There were several extensions of the problem to arm/leg manipulators [14], [15], but they are beyond the interest of this paper. A solution to the NAMO problem that considered learned dynamic constraints [16] utilized a physics-based simulator with limited visual capabilities, while scene affordances [17] were used in standard 3D simulators (similar to Gazebo) to generate simple open-loop path plans. Our work is based on the NAMO algorithm implemented in [8] (which in turn extends the path planner introduced in [18]). The original paper and its extension [19] generate very simple 2D simulations, where obstacles were detected from a 2D representation in an image map.

Our work, is the first (as far as we are aware) integration of visual and obstacle-interactive feedback sensing, to detect and localize movable obstacle, using photorealistic environment (NVIDIA Isaac) to utilize this knowledge in an online NAMO planning algorithm on a real mobile robot.

## II. METHODS

In this section, we firstly present the methods developed to integrate object detection and localization via a photorealistic simulator (NVIDIA Isaac). Then, we present the adapted path planning and mapping/localization algorithms to allow Navigation Among Movable Obstacles.

### A. Simulator and System Setup

NVIDIA's Turing architecture and the release of their RTX series GPU [20], with hardware ray-tracing [21] acceleration in 2018, has made real-time photo-realistic applications possible, including robotic simulators. Ray-tracing can almost emulate how nature light rays are passed into our eyes, with the result being highly photo-realistic. The NVIDIA Omniverse platform is an RTX-powered open graphics platform for real-time collaboration and resource sharing. The NVIDIA Isaac Sim is a robotic simulator built on top of the Omniverse platform, providing environment and sensory data, connection to ROS, and domain randomization features. Such a simulator can fit the requirements of solving the sim2real problem in NAMO applications for mobile robots.

Fig. 3: Isaac Sim photorealistic simulation of a warehouse environment, containing the UCL MPPL mobile robot.

In order to experiment with our developed NAMO method, warehouse environments were generated in the Isaac Sim simulator (see an example in Fig. 3). To exploit the photo-realistic capabilities, high definition textures and decorative objects were added in the scene. We developed a plugin to facilitate the addition of objects to the scene and enable ROS-control for the robot. Our UCL MPPL robot (that includes an omnidirectional robot base) is loaded into the scene, supplemented with the addition of components, including the robot's on-board RGB-D cameras and 2D lidar sensors.

### *B. Movable Obstacle Detection and Localization*

There are two ways obstacles will be detected and added in the map, while the robot is localizing itself in it. The original map that it is given, includes only areas of the environment that are fixed. For instance, in a warehouse, a map might include the fixed walls. These maps are usually easy to be identified and provided, either by creating a model using lidar/visual sensors with a moving robot (this is the way we created our maps in this work), or via the physical environment structure design. Global A* shortest-path plans are created using those predefined maps. Although, the interesting part is how one can add obstacles in the map and how one can specify if those obstacles are movable or not. This will affect the local path plan, when moving to a goal. For instance, one might detect from distance a cardboard box, but if it is too heavy to be pushed, then it might be considered by the robot as a static obstacle in the map. There are two types of visual obstacles recognition methods that we utilized: a deep learning-based object localization and an April-tag one. We describe both below. Robot localization is solved by fusing (with Extended Kalman Filtering - EKF) data from the wheel odometry, the two 2D lidar sensors (one placed in the right front part and one in the left back part of the robot), and the IMU on the robot, similar to the method we have implemented in [22]. Notice that when an object is classified as movable and it is manipulated in the environment, we avoid considering the lidar data that belong to it, as it breaks the visual localization part of the system.

*1) Visual/RGB-D Based Pose Estimation:* With the growth of supervised deep learning, several methods of 3D object pose estimation have been proposed. In this work, we mainly use the Deep Object Pose Estimation (DOPE) [23] network. This is a state-of-the-art network that can easily be trained on simulated and real-world data. One could use other similar networks, such as the PoseCNN [24]. DOPE uses pre-trained datasets for real-time object detection and 3D pose estimation. We used a pre-trained network on the YCB (Yale-CMU-Berkeley) object dataset that contains common house-hold objects, and fine-tuned it with scaled-up boxes that represent cardboard boxes in warehouse environments. For testing, those cardboard boxes were added in the Isaac Sim environment via a custom plugin, while DOPE provided pose estimation from the RGB-D sensor input, as illustrated in Fig. 4. When a cardboard object is detected in the field of view of the robot, then it is added as a potentially movable obstacle. If the detected object is not in the list of potentially movable obstacles, then it is added as a static obstacle in the environment map. If the object is not detected by the network, then the corresponding point cloud is considered as a static obstacle in the map.


Fig. 4: Obstacle detection and 3D pose estimation, based on DOPE. From left to right: obstacle as placed in simulation, object pose estimation in Isaac Sim, and obstacle in the map.

*2) April-Tag Based Pose Estimation:* An alternative way that can be used in controlled environments to detect and localize obstacles are tags/markers. It might be easy to apply, for instance, a tag in every item that you know is movable in a warehouse (e.g., cardboard boxes). April-tags [25] is such an alternative method to initially validate the proposed pipeline and later compare it with the results from the aforementioned DOPE-based approach in terms of pose accuracy. We applied different April-tags on each box in the environment that enabled the detection and 3D localization of each cardboard box. Again, when an object is detected/localized using the markers, it is added in the map as a potentially movable obstacle.

*3) Pushed-Based Movability Estimation:* Path planning research for mobile robots are mainly based on visual data. The reason is that the main goal in most of those works is obstacle avoidance. In our case, we care more about interaction with obstacles. If there is a path that can avoid such interaction, it might be preferred, but if there is not such a path towards a goal, then interaction with the obstacle is essential. When an obstacle is detected and localized, it is added in the map as a potentially movable object. Although, a robot can only deduce the movability via interaction. Thus, when a local path is planned, and it includes a pushing action with the obstacle (see Sec. II-C below), we allow the robot to attempt it. We remove the 2D lidar data that belong to the obstacle, in order to not confuse the localization system, but we allow the robot to gradually push the object. If the robot

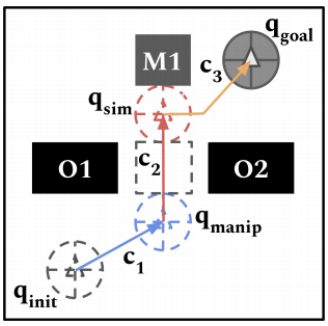


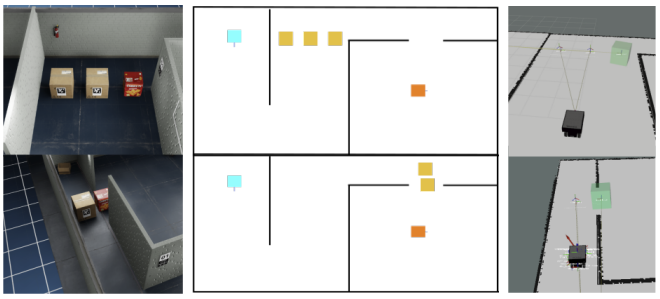

Fig. 5: NAMO with object M1 [8]: move to obstacle (c1), push the obstacle (c2), move to the goal (c3).

does not move in the map, when it is pushing the obstacle with the maximum force for few seconds (in our work we allow 5sec of pushing), then we consider the obstacle as non-movable/static. Notice, that if a static object is pushed, it might create spinning of the wheels. In our case, we did not notice any localization issue in the map, given that lidar and IMU sensing is balancing this wheeled odometry outlier via the EKF.

### C. Adapted Navigation Among Movable Obstacles (NAMO)

So far, we have explained the way that the robot uses a map of the fixed environment to localize itself in it, detect obstacles, localize them, and update the type of the obstacles in the map as movable or non-movable/static via interaction. In this section, we will explain the modifications we did in the NAMO algorithm introduced in [18], [8] to handle realistic partially unknown environments, in order to enable a complete system integration. The whole pipeline is illustrated in Fig. 2.

First, a predefined map of the robot's environment is provided and converted to a 2D SVG (Scalable Vector Graphics) image map. The map was generated by the robot moving manually in the environment, using its localization system as described above, and the 2D lidar data from the sensors in the front and back part of the robot. Static objects, such as walls and other DOPE-based detected objects are added in the map, along with a goal robot position. Then, the original A* algorithm of [8] is used to generate a planned path to the goal. We define heuristics and cost functions with weights such that if a clear path is detected, then it is preferred over other paths that might include interaction. The path is given to the robot controller, and wheel joint commands are generated in order to follow the path. Robot localization gives the feedback in order to correctly follow the generated trajectories. As the robot navigates to the goal position, unknown cardboard boxes might be detected. Their poses are visually estimated via DOPE or April-tags and the new objects are added in the map, as potentially movable ones. With the addition of each obstacle, the robot path is locally recomputed in three phases as illustrated in Fig. 5: plan to obstacle (c1), push obstacle (c2), and plan to goal (c3). If a push attempt fails, as we described above, the map environment is updated with this obstacle as non-movable/static and a path re-plan is performed. Moreover, we



(a) (b) (c)

Fig. 6: Obstacle pose estimation integrated with NAMO: (a) obstacle positions in the simulated environment, (b) updated map after addition of estimated box poses, (c) obstacle transforms within the robot's map.

have tweaked the pushing action, such that when the pushing is done, the robot first backs-up a little and moves back to the planned path by a simple collision avoidance with the moved obstacle. This is especially helpful as obstacles usually drift outside the planned pushing trajectory and might result in collisions.

## III. EXPERIMENTAL RESULTS

In this section, we present a set of experiments that were performed to evaluate the success of the proposed pipeline. Initially, we discuss the accuracy of obstacle localization in the Isaac photo-realistic simulator. These estimations are then incorporated into our developed NAMO method, where the generated trajectories allow a mobile robot to successfully navigate to a goal, whilst clearing the path of movable obstacles in simulation. Finally, this pipeline is preliminary validated on a real wheeled robot, UCL's MPPL, demonstrating it's feasibility of bridging the sim2real gap.

### A. Simulation Evaluation

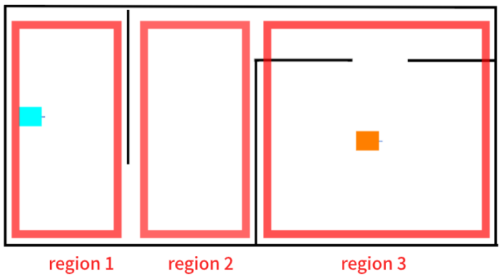



Fig. 7: Regions of the warehouse environment (blue: starting robot pose; orange: goal robot pose).

*1) Object Pose Estimation Accuracy:* As outlined in Sec. II-B, two separate object pose estimation methods were employed in this study. Here we will present a comparison between the two methods (DOPE and April-tags) and test their suitability for box pose detection within the NAMO algorithm. A benefit of testing these methods in simulation is that we have the knowledge of the ground truth position of all obstacles within the simulated environment. We demonstrate the results in a warehouse environment of three regions (see

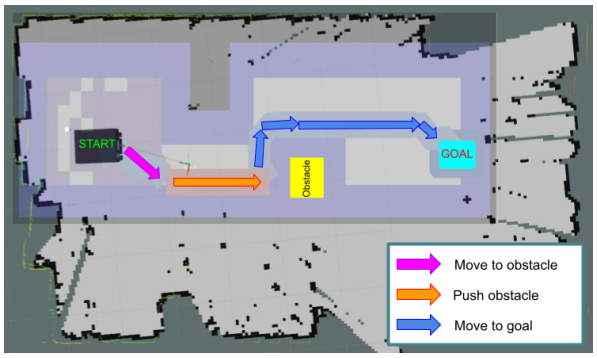



Fig. 8: Robot localized in map with NAMO plan.

Fig. 7). Three boxes were placed at the passage between two regions in order to block it: two marked with an April-tag at the first passage and one without an April-tag at the second passage. We repeated the experiments multiple times with slightly different object positioning and robot approach direction (roughly 50 variations). We noticed that in Isaac Sim the detected DOPE and April tag obstacles, were localized roughly with a similar accuracy (15cm in average). We verified this in the real world with actual cardboard boxes and April tags. Given that those boxes are of size $100\text{cm}^3$, the pose estimation error did not affect any of the pushing accuracy actions. The object detection rate was $100\%$ (i.e., there wasn't any case that an object was not detected), even when they were partially occluded by a wall.

*2) Integration with the NAMO Algorithm:* The obstacle pose estimation was then integrated with the NAMO algorithm described in Sec. II-C. Obstacles, both movable and non-movable/static, were positioned around the warehouse environment in the Isaac Sim simulation. The boxes were either cardboard boxes or marked with April tags. We had always at least one object being non-movable (by making them too heavy). An example, as appears in the video, is when the robot detects two boxes blocking a passage, it tries to push the closet one (which is too heavy), and then re-plans and pushes the second one to free the path. Prior to the test runs, the NAMO algorithm has no knowledge of the objects or their movability. The movability of an object is determined by the result of a push attempt (we also applied pushing and pulling actions, but because of the real hardware, in this paper, we focus only on pushing ones). Two examples of the experiments are shown in Fig. 6.

The goal position is marked in orange and the start position in blue. The NAMO algorithm was provided with the initial environment map and a set of trajectory points were generated to navigate the robot to the goal pose. During the execution of this path, unknown box obstacles are detected and 3D pose estimations are generated. The box pose estimations are then added in the map, as visualized in Fig. 6-(b,c). Once a new path has been generated, the robot continues it's journey towards the goal pose. Movable obstacles are cleared from the path with a push action and those that cannot be pushed after an attempt, are marked as non-movable in the updated map. We have ran one hundred experiments with different uniformly random obstacle positions blocking passages in this environment, and the average run-time to complete the task was around 200s, with the complete path being approximately 18m.

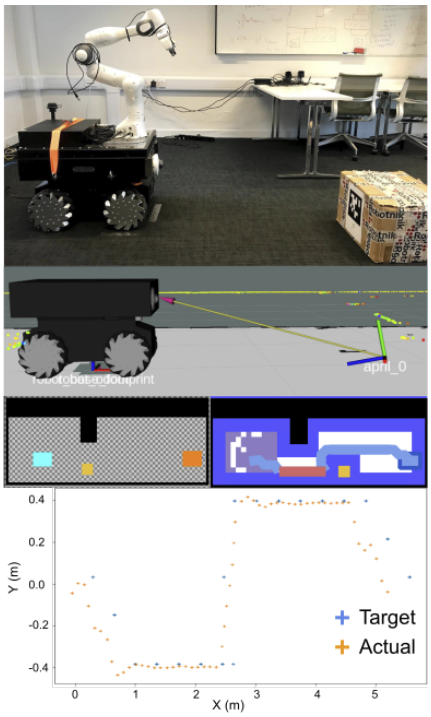



Fig. 9: Top to bottom: robot presented with an obstacle; the April-tag is detected (RViz visualization); the obstacle is added to the environment map and our NAMO-based A* path is computed; the robot trajectory–target and actual.

### B. Real-World Evaluation

To demonstrate that NAMO with the integrated object pose estimation, as tested in simulation, could be transferred to the real-world, a preliminary experiment was made with UCL's MPPL robotic platform. This experiment would implement the same pipeline as presented in Fig. 2, with the real robot directly replacing the simulated robot, receiving the same trajectory commands and supplying current pose updates via EKF-based localization.

A simple test environment was created with two spaces connected by a small corridor (see Fig. 9), such that the robot could not pass if an obstacle is present. A light cardboard box of dimensions $0.5 \times 0.4 \times 0.4$m with a tag was placed in the corridor, blocking the entrance to the second space. For a new environment configuration, two items are required, a map for robot localization and an environment map for the NAMO algorithm. A lidar-based map of the environment was made, utilizing a SLAM (i.e., ROS gmapping), while the initial environment map of the walls was provided to NAMO. Fig. 8 shows the SLAM generated map with visualization of the robot; the environment map, and an example plan to the goal position generated from our NAMO method, overlaid.

A navigation plan is made to the goal position in the second space in the environment, the robot begins to execute the plan while perceiving it's environment with the RGB-D camera. If a flag is raised to indicate that an obstacle has been

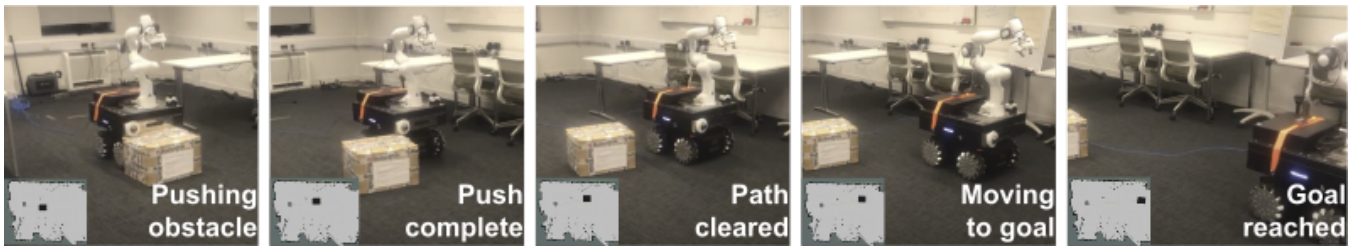



Fig. 10: The real robot clearing an obstacle from the path to move to a goal (inset picture is the robot pose in the map).

detected, the robot aborts the plan and an obstacle is added to the environment map, labeled as a potentially movable one. If no plan can be found around the obstacle, a plan for obstacle pushing is made. The robot then executes the new plan, pushing the obstacle with the front of the mobile base until a clear path is available and the goal position can be reached. Fig. 9 illustrates the stages of the experiment.

The experiment was performed with similar start and goal positions (we allowed some variations to prove autonomy), 20 times. The box obstacle was moved to different locations within the corridor section of the environment. The obstacle was detected in all trials and the robot was able to push the box, clearing a path to the goal with a success rate of $100\%$. If the box was positioned as such that the robot could navigate around it, it did so. Two times, it was observed that while the robot moved towards it's goal, the laser scan lost some minor alignment with the robot map. Although, in none of the trials did this error result in a failure to push the box to clear the path to the goal. Fig. 10 shows a sequence of images taken throughout one of the experiments, where the robot pushes the box to clear a path and moves to the goal position. Last, we also tried to block the box with our feet so that it cannot be moved, and allowed the robot to test the pushing movability action. This resulted to classify the box as non-movable and re-plan. Given that the environment did not have any other free path, the robot remained in its position.

## IV. CONCLUSIONS AND FUTURE WORK

In this paper, we have successfully demonstrated a novel system integration for the Navigation Among Movable Obstacles (NAMO) problem, utilizing unknown obstacle localization via vision and pushing actions, as well as sim2real demonstration via the use of the Isaac Sim photorealistic simulator. The method runs online on the robot during a robot navigation task, resulting in successful obstacle manipulations and reaching navigation goals. This was demonstrated in simulation and then applied directly to a real mobile platform. In future work, we aim to test more complex scenarios with tighter space constraints and various different types of actions. Last, we aim at replacing the local path planning with a reinforcement learning agent that can compute trajectory NAMO tweaks faster.

## REFERENCES

[1] H.-y. Zhang, W.-m. Lin, and A.-x. Chen, "Path Planning for the Mobile Robot: A Review," *Symmetry*, vol. 10, no. 10, p. 450, 2018.

[2] M. Stilman and J. Kuffner, "Navigation Among Movable Obstacles: Real-Time Reasoning in Complex Environments," in *4th IEEE/RAS Int. Conference on Humanoid Robots*, vol. 1, 2004, pp. 322–341.

[3] J. Tremblay *et al.*, "Training Deep Networks With Synthetic Data: Bridging the Reality Gap by Domain Randomization," in *IEEE CVPR Workshops*, June 2018.

[4] S. Brodeur *et al.*, "HoME: a Household Multimodal Environment," *CoRR*, vol. abs/1711.11017, 2017.

[5] C. Yan *et al.*, "CHALET: Cornell House Agent Learning Environment," *CoRR*, vol. abs/1801.07357, 2018.

[6] Y. Kakiuchi *et al.*, "Working with Movable Obstacles using On-Line Environment Perception Reconstruction using Active Sensing and Color Range Sensor," in *IEEE/RSJ IROS*, 2010, pp. 1696–1701.

[7] Z. Meng, H. Sun, K. B. H. Teo, and M. H. Ang, "Active Path Clearing Navigation through Environment Reconfiguration in Presence of Movable Obstacles," in *IEEE/ASME AIM*, 2018, pp. 156–163.

[8] B. Renault, J. Saraydaryan, and O. Simonin, "Towards S-NAMO: Socially-aware Navigation Among Movable Obstacles," *CoRR*, vol. abs/1909.10809, 2019.

[9] M. Stilman, K. Nishiwaki, S. Kagami, and J. J. Kuffner, "Planning and Executing Navigation Among Movable Obstacles," in *IEEE/RSJ IROS*, 2006, pp. 820–826.

[10] M. Stilman, "Navigation Among Movable Obstacles," Ph.D. dissertation, MIT, 2007.

[11] M. Stilman and J. Kuffner, "Planning Among Movable Obstacles with Artificial Constraints," *IJRR*, vol. 27, no. 11-12, pp. 1295–1307, 2008.

[12] M. Levihn, L. P. Kaelbling, T. Lozano-Pérez, and M. Stilman, "Foresight and Reconsideration in Hierarchical Planning and Execution," in *IEEE/RSJ IROS*, 2013, pp. 224–231.

[13] M. Levihn, M. Stilman, and H. Christensen, "Locally Optimal Navigation Among Movable Obstacles in Unknown Environments," in *IEEE-RAS Int. Conference on Humanoid Robots*, 2014, pp. 86–91.

[14] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation Planning Among Movable Obstacles," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 3327–3332.

[15] V. S. Raghavan *et al.*, "Reconfigurable and Agile Legged-Wheeled Robot Navigation in Cluttered Environments With Movable Obstacles," *IEEE Access*, vol. 10, pp. 2429–2445, 2022.

[16] J. Scholz *et al.*, "Navigation Among Movable Obstacles with Learned Dynamic Constraints," in *IEEE/RSJ IROS*, 2016, pp. 3706–3713.

[17] M. Wang, R. Luo, A. Ö. Önol, and T. Padir, "Affordance-Based Mobile Robot Navigation Among Movable Obstacles," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020, pp. 2734–2740.

[18] H.-N. Wu *et al.*, "Navigation Among Movable Obstacles in unknown environments," in *IEEE/RSJ IROS*, 2010, pp. 1433–1438.

[19] B. Renault, J. Saraydaryan, and O. Simonin, "Modeling a Social Placement Cost to Extend Navigation Among Movable Obstacles (NAMO) Algorithms," in *IEEE/RSJ IROS*, 2020, pp. 11 345–11 351.

[20] V. Sanzharov *et al.*, "Examination of the Nvidia RTX," 2019, pp. 7–12.

[21] S. Parker, H. Friedrich, D. Luebke, *et al.*, "GPU Ray Tracing," *Communications of the ACM*, vol. 56, pp. 93–101, 05 2013.

[22] S. Piperakis *et al.*, "Outlier-Robust State Estimation for Humanoid Robots," in *IEEE/RSJ IROS*, 2019, pp. 706–713.

[23] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects," *arXiv preprint arXiv:1809.10790*, 2018.

[24] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes," *CoRR*, vol. abs/1711.00199, 2017.

[25] J. Wang and E. Olson, "AprilTag 2: Efficient and Robust Fiducial Detection," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 4193–4198.