arXiv:2212.02671v1 [cs.RO] 6 Dec 2022

Visibility-Aware Navigation Among Movable Obstacles

Jose Muguira-Iturralde*, Aidan Curtis*, Yilun Du, Leslie Pack Kaelbling, Tomás Lozano-Pérez

***Abstract*— In this paper, we examine the problem of visibility-aware robot navigation among movable obstacles (VANAMO). A variant of the well-known NAMO robotic planning problem, VANAMO puts additional visibility constraints on robot motion and object movability. This new problem formulation lifts the restrictive assumption that the map is fully visible and the object positions are fully known. We provide a formal definition of the VANAMO problem and propose the Look and Manipulate Backchaining (LAMB) algorithm for solving such problems. LAMB has a simple vision-based API that makes it more easily transferable to real-world robot applications and scales to the large 3D environments. To evaluate LAMB, we construct a set of tasks that illustrate the complex interplay between visibility and object movability that can arise in mobile base manipulation problems in unknown environments. We show that LAMB outperforms NAMO and visibility-aware motion planning approaches as well as simple combinations of them on complex manipulation problems with partial observability.**

I. Introduction

Navigation is an essential ability for mobile robots. Typical navigation systems use motion planning for obstacle avoidance during navigation. However, the goal is not always reachable directly and sometimes requires manipulation of the environment, such as opening doors, grasping obstructing objects, or pushing obstructing furniture. The problem of robot navigation that requires manipulation of the environment is termed NAMO (Navigation Among Movable Obstacles). A large body of work has studied NAMO problems and presented many algorithms for solving them [1], [2], [3], [4], [5], [6], [7], [8], [9]. While the NAMO problem in its most general formulation is NP-hard, several approaches can make theoretical guarantees under certain practical assumptions that hold in many real-world tasks.

In addition to movable obstacles, another constraint that complicates navigation is visibility. For safety and reliability, a robot may not want to enter regions in the workspace that it has not observed to be free. For robots with 360° vision and no movable obstacles, this requirement does not impose any additional constraints. However, for many robots with only a single-camera vision and an omnidirectional base (e.g. Figure 1), it becomes necessary to reason about where to look and where to move. Simple heuristics that force the robot to look at a region before moving into it fail when the objects being manipulated obstruct the robot's vision or when the necessary areas of the workspace are only visible from certain perspectives. Unfortunately, reasoning about visibility does not neatly fit into the motion planning problem formulation due to the path dependence of visibility. Like NAMO, visibility-aware motion planning (VAMP) is NP-hard, but several algorithms work well in practical problems [10], [11], [12], [13], [14].

*The first two authors contributed equally. The authors are at CSAIL, MIT, USA: {jmuguira, curtisa, yilundu, lpk, tlp}@mit.edu.

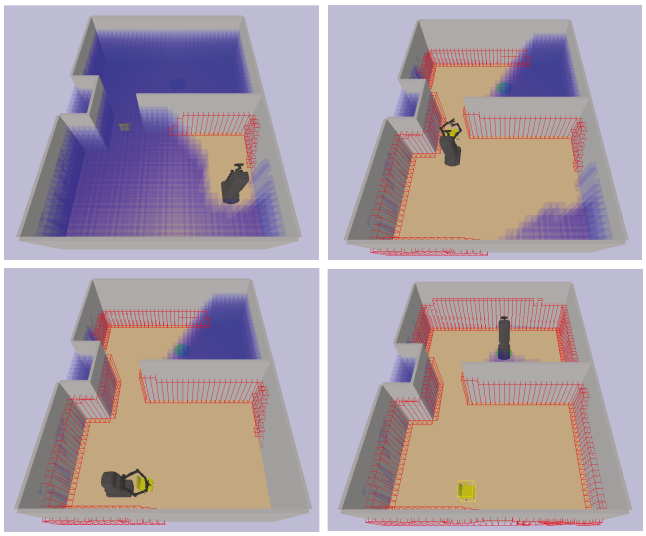

Fig. 1. A partial simulated execution of a VANAMO task showing the unviewed regions (blue), observed static obstacles (red), and observed movable obstacles (yellow)

Our contributions are as follows. First, we introduce a class of problems that fuses elements from NAMO and VAMP problems and formalize this class of problems as VANAMO (Visibility-Aware Navigation Among Movable Obstacles). Second, we provide a framework for mobile manipulation from camera images and implement several existing NAMO and VAMP solutions as benchmarks. Third, we propose a new algorithm for VANAMO which combines backward reasoning about visibility and reachability with forward motion planning. We demonstrate its ability to work well in complex 3D environments where the interaction between visibility and object movability plays a crucial role. Lastly, we illustrate our approach's relative effectiveness on a number of complex simulated mobile manipulation tasks.

II. Related work

Previous NAMO work typically relies on the concept of connected configuration regions and focuses planning effort on *keyhole* actions that connect previously disconnected regions of the configuration space. Greedy backtracking algorithms have been proposed to solve the linear, or LP_1, class of problems in which a sequence of independent single-object keyhole actions can achieve the goal [1], [2],

[4]. Other algorithms have extended to LP_k problems that require sequencing k-object keyhole actions by enforcing artificial motion constraints in the planner under additional assumptions on object movability [3], [5], [6].

In many real-world robot scenarios, the environment map and object locations are not known *a priori* and have to be acquired through sensors and exploration. One of the most commonly used techniques for exploration in navigation is frontier exploration [15]. Frontier exploration identifies a boundary between the observed and unobserved space and then picks a point on that boundary to explore next. Unfortunately, arbitrary exploration of the unknown space is inefficient when the robot has a particular navigation goal. Several methods have been proposed to address this inefficiency for visibility-aware navigation [10], [11], [12], [13], [14]. This problem is difficult because visibility is path-dependent and doesn't fit into the conventional motion planning framework. One recently proposed strategy develops a path-dependent motion planning algorithm that plans to view necessary regions of the workspace through subgoal backchaining [10]. While this algorithm applies strictly to navigation, we took inspiration from this approach in building our solution.

While the methods described so far have addressed the problems of NAMO and VAMP separately, they are insufficient for tackling the combined VANAMO problem. Robots in real-world environments must deal with visibility and movable object constraints. Several methods have been proposed to handle environments with both constraints. Some of these approaches use unrealistic models of vision that allow the agent to see through objects or assume an *a priori* known map with only unknown movable object poses [16], [17], [18]. Other approaches use realistic observations in the form of real robot sensor data but only test in simple environments where one object is obstructing the goal [19]. To our knowledge, we are the first to present an algorithm capable of handling environments with complex constraints arising from the interaction between visibility and movable obstacles.

III. Preliminaries

The VANAMO environment is defined by a tuple $\langle \mathcal{C}, \mathcal{W}, \mathcal{O}, \mathcal{M}, \mathcal{I}, q_g, \mathcal{A}, f, \text{Obs} \rangle$ where $\mathcal{C}$ is the robot's configuration space, $\mathcal{W}$ is the robot's workspace (typically 3D with bounds), $\mathcal{O}$ is a set of static obstacles, $\mathcal{M}$ is a set of movable objects where each $o \in \mathcal{O}$ and $m \in \mathcal{M}$ is a defined by an object shape with the same dimensionality as $\mathcal{W}$. $\mathcal{I}$ defines the world state including the initial robot configuration q_0 and the pose of all static obstacles q_o for $o \in \mathcal{O}$ and movable obstacles $q_{\Updownarrow}$ for $m \in \mathcal{M}$. The navigation goal is denoted by q_g. The robot also has an action space $\mathcal{A}$. The dynamics function $f : \mathcal{C} \times \mathcal{A} \rightarrow C$ maps from a robot configuration q_t and action a_t to a new configuration q_{t+1}. Lastly, the observation function Obs : $\mathcal{C} \rightarrow \text{Img}(q_{\mathcal{M}}, q_{\mathcal{O}})$ maps a configuration q to an image projection of the environment with partial information of the movable and static objects. If the object shapes in $\mathcal{O}$ and $\mathcal{M}$ are known in advance, the observation function would be defined as Obs : $\mathcal{C} \rightarrow \mathcal{P}(\{(i, q_i) \mid i \in \mathcal{M} \cup \mathcal{O}\})$, where $\mathcal{P}$ is the power set. Our experimental setting uses a holonomic robot acting in an $SE(2)$ configuration space with the degrees of freedom corresponding to x, y movement and θ rotation. The Obs function is fully defined by the intrinsics of a forward-facing camera with a fixed pose relative to the robot base. Even with a holonomic base, explicitly modeling θ is necessary because it governs the camera direction for directional visibility. The robot can interact with the environment through a number of controllers. The `move` controller modifies the x, y, and θ dimensions by adding or subtracting to them with fixed increments. The `pick` controller creates a rigid attachment with a movable object if the object is within a distance ϵ of the robot, the bounding box for the movable object is smaller than a maximum height and width dimensionality (i.e., it can be surrounded by the robot arms), and some part of the object shape is within a certain θ deviation from the robot's orientation (i.e., the robot needs to be facing the object). The `place` controller removes a rigid attachment if one exists. Lastly, the `push` controller operates on movable objects of any size that the camera is within ϵ distance of as long as some part of the object is within a certain θ deviation from the robot's orientation. The `push` controller displaces the robot and the movable object by a fixed distance Δ in the q_θ direction. The dynamics of the environment are assumed to be deterministic, but the algorithm may be executed in nondeterministic environments as long as replanning is triggered after major deviations from the expected outcome.

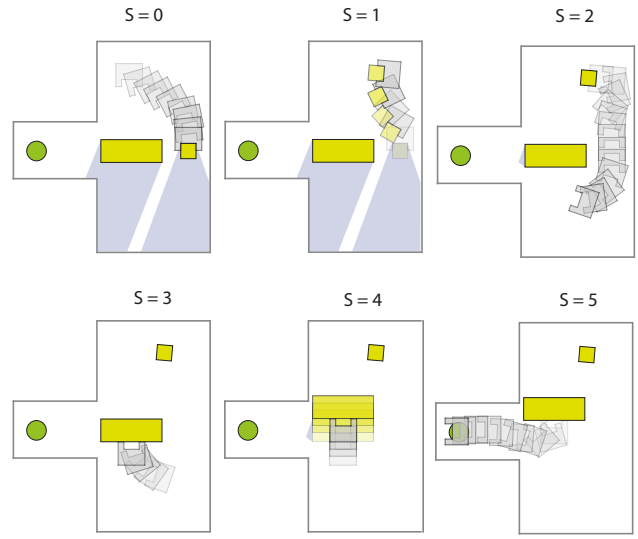

Fig. 2. A top-down depiction of an example execution for the LaMB algorithm on the occluding obstacles task. The robot would like to push the wide box, but cannot because of visibility constraints. It first moves the small box out of the way (S=0, 1), then looks at the area behind the box (S=2), and then pushes the box (S=4) so it can reach the goal (S=5).

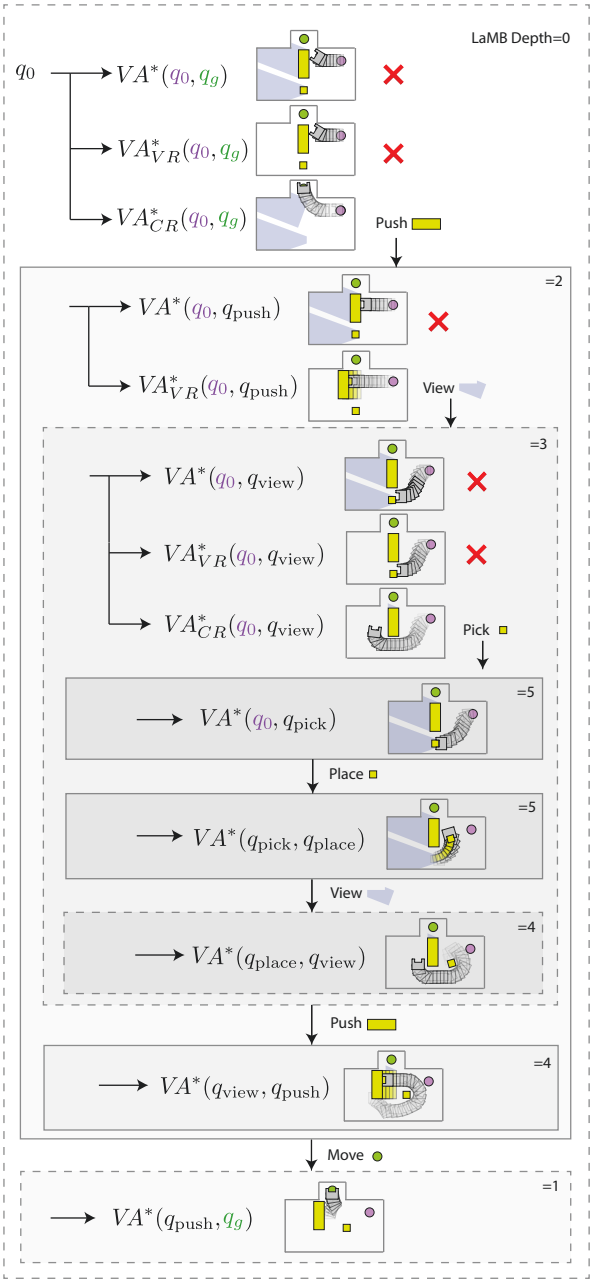

Fig. 3. An example trace of the LAMB algorithm on the **obstructed visibility** task depicted in Figure 2. We denote each nested recursive call with a darker shade of grey. The start location is marked with a purple circle, and the goal location is marked with a green circle. VA$^*{}_{VR}$ denotes vision-relaxed motion planning, and VA$^*{}_{CR}$ denotes movable object relaxed motion planning. For clarity, we collapse successive move & manipulate calls and denote them with solid lines.

IV. METHOD

In this section, we introduce our algorithm for solving VANAMO problems called Look and Manipulate Backchaining (LAMB). Our algorithm is structured as a two-level hierarchical search. The lower level search is an A* algorithm that takes an initial configuration, goal configuration, an attachment (if one exists) and its relative pose, and a set of obstacles to avoid. The goal is not typically achievable directly through motion planning, so the higher level search finds a sequence of navigation and manipulation actions to remove the constraints preventing goal reachability.

The constraints preventing navigation fall into two categories. *Visibility constraints* ensure that the robot never passes through or moves obstacles into a region that it has not been viewed. *Collision constraints* ensure that the robot never navigates or pushes obstacles through regions containing obstacles. LAMB performs a high-level search through look actions (`move` actions for the purposes of looking) that remove visibility constraints and manipulation actions that displace objects and therefore change collision constraints. Because of interdependence between these configurations, it is sometimes the case that one visibility or collision constraint can prevent the robot from removing another constraint. To handle these complex interactions, our higher-level search recursively breaks down goals into subgoals that aim to remove individual constraints.

A. LAMB

The LAMB planner operates in the execution context outlined in algorithm 3. Visibility, static occupancy, and movable occupancy grids are initialized to be empty prior to any observations. Observations have the form of segmented point clouds, and the segments are assumed to be labeled according to their object index. This segmentation includes knowledge about whether a particular object is movable. Realistically, this could be done based on material or shape properties, but in our experiments and simulations, the objects are recognized with ground truth per-pixel object identities from the simulation. The visibility grid for a configuration, denoted $\text{Vis}(q)$, is computed by casting rays from the camera to the point specified by the captured depth image and marking all voxels each ray passes through as viewed. The LAMB algorithm is then called with the updated grids as well as the navigation goal and initial state. The first action of the plan returned from LAMB is executed in the environment, and the new observation is used to update the grids. This process repeats until the goal is reached or the process times out.

Inside of LAMB , we consider three cases: direct planning, visibility-relaxed planning, and collision-relaxed planning, as shown in Algorithm 3. Each case uses VA* to determine if the goal is reachable with various relaxed constraints. VA* is a slightly modified version of A^* that handles path-dependent visibility in an efficient manner. Namely, once a path to a particular state is found, we do not consider any shorter paths to that state, and we associate each state with the visibility grid that would be derived from the path that first reaches that state. This does not necessarily lead to optimal visibility-based paths but is an effective approximation [10]. Because the planner cannot predict what will actually be observed as the robot traverses a path, we cannot actually obtain the visibility of an imagined configuration. For this reason, we use optimistic visibility. That is, within VA*, we assume that

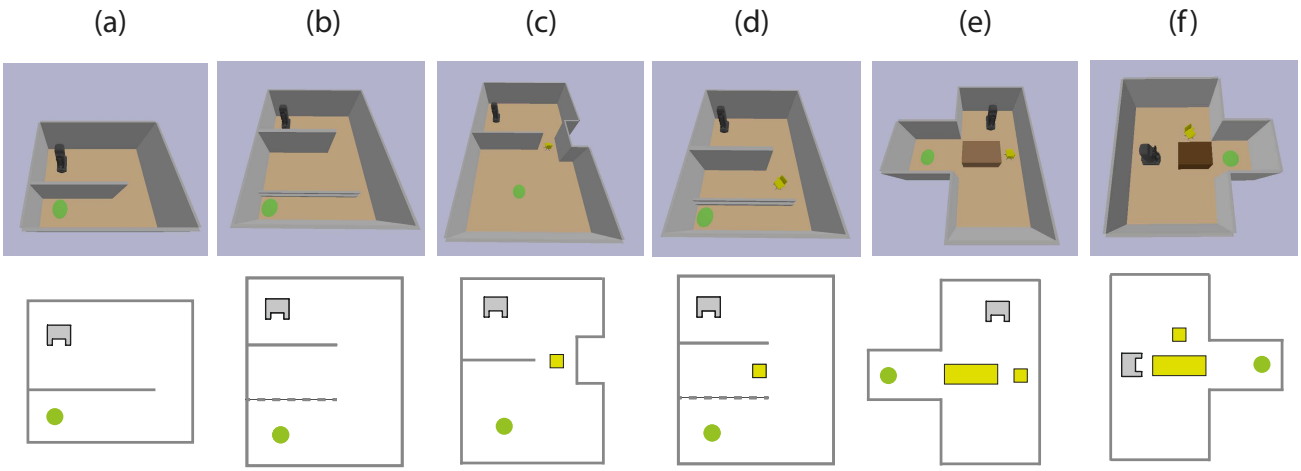

Fig. 4. **Problem Instance Visualization.** A picture of problem instances from each task category (top row) with illustrated depictions of the envioronments for visual clarity (bottom row). The green dot indicates the goal position, the yellow objects in the botom row indicate movable objects. The large box is pushable, but not directly graspable.

voxels not known to contain an obstacle are free space. If a direct path with VA* is possible, we simply return that path as the plan.

If no direct path is feasible, LAMB first tries relaxing the visibility constraints. To do this, we simply plan with VA* and an empty visibility grid. If a plan is found with these relaxed constraints, we identify the region that needs to be viewed by intersecting the swept volume of the path, denoted SWEPT(path), with the already gained visibility and create a subgoal for viewing that region. It is often the case that the necessary region is not viewable from a single perspective. For this reason, we use a special type of heuristic that drives progress towards seeing part of the necessary region. We do so by computing a scalar field, F, in our workspace, which denotes the shortest distance from any point in the workspace to our required region. Given this, our new heuristic would be defined as $\mathrm{H}_F(q) = \min_{x \in \mathrm{Vis}(q)} F(x)$. We obtain the new subgoal by running our VA* algorithm in an obstacle-relaxed environment. Each subgoal requires an independent plan, which we obtain using a recursive call to LAMB. This call needs to be recursive because additional constraints, such as obstructing movable obstacles, could prevent the reachability of those necessary viewing positions. In our experiments, the configuration of the robot is set so that it can always see its base. This configuration reduces the number of visibility subgoals needed to complete the task but is not strictly necessary for our algorithm to work.

If no visibility-relaxed plan is feasible, LAMB computes a collision-relaxed plan that removes collision constraints imposed by *movable* obstacles. To do this, we simply set the movable occupancy grid to empty and plan a path to the goal using VA*. If a path is found, the first movable object collision is detected using the same SWEPT subprocedure used for computing the visibility subgoal. In this work, we consider moving only the first obstacle the robot collides with along a path to the goal. This approach makes the assumption that we are dealing with LP_1 NAMO problems [1]. While this algorithm could easily be extended to consider multiple obstacles, doing so incurs a substantial computational cost and is not necessary for our environments. However, we do consider multiple ways of interacting with that object. For each object we consider `push` and `pick`/`place` operations (depending on the size of the object) from multiple grasp locations. Pushing is a more constrained operation but is sometimes necessary if the object is too wide for the robot to wrap its arms around. For each manipulation action considered there is a q_{pre} and q_{post} robot configuration. Given these intermediate configurations, we can plan a path to manipulate the object and then reach the goal through recursive calls to LAMB . We additionally need to compute the updated occupancy grid and swept volume. The updated occupancy grid is necessary for planning after manipulation, and the swept volume is necessary for planning before manipulation. The swept volume adds a constraint to the planner that restricts moving other obstacles into the swept path of the manipulated object prior to the object's manipulation. See [3] for details regarding this approach.

If no plan can be found under these relaxations, then we terminate the planner and return a failure result. Figure 3 shows an example trace of this algorithm on one of the more complex tasks involving multiple recursive calls with both visibility and collision relaxing.

V. EXPERIMENTS

To demonstrate the importance of visibility reasoning in NAMO and evaluate our algorithm, we construct a set of 5 task categories, each with unique challenges. The initial state for each task category can be seen in Figure 4. The goal of each task category is to navigate to a particular region in space highlighted in green. Within each task category, we experiment with random initialization of object positions,

Algorithm 1 EVALUATEPLANNER$(\mathcal{O}, \mathcal{M}, \mathcal{I}, q_g, \mathcal{A}, f, \text{Obs})$

$q_t \leftarrow q_0$
GridV $\leftarrow \emptyset$, Grid$\mathcal{O} \leftarrow \emptyset$, Grid$\mathcal{M} \leftarrow \emptyset$
while $\neg(q_t = q_g)$ **do**
  $o \leftarrow \text{Obs}(q_t)$
  GridV $\leftarrow$ UPDATEVIS$(\emptyset, o)$
  Grid$\mathcal{O}$, Grid$\mathcal{M} \leftarrow$ UPDATEOBS(Grid$\mathcal{O}$, Grid$\mathcal{M}, o$)
  plan $\leftarrow$ LAMB$(q_0, q_g, f,$ Grid$\mathcal{O}$, Grid$\mathcal{M}$, GridV$)$
  $q_t \leftarrow f(q_t, \text{plan}[0])$
  trace $\leftarrow$ trace $\oplus$ $(plan[0], q_t)$
return trace

Algorithm 2 VA$^*(q_0, G, \text{G}\mathcal{O}, \text{G}\mathcal{V}, \text{H}(q) = ||q - q_g||_2)$

$Q \leftarrow [[q_0]]$, visited $\leftarrow \emptyset$
while $|Q| \neq 0$ **do**
  path $\leftarrow Q$.pop(0), $q_{\text{last}} \leftarrow Q[0][0]$
  if $q_{\text{last}} \in G$ **then return** path
  if $q_{\text{last}} \in$ visited **then** continue
  V $\leftarrow$ PathVision(path, G$\mathcal{O}$)
  $V' \leftarrow G\mathcal{V} \cup V$
  $N_q \leftarrow$ NEIGHBORS(q_{last})
  $Q \leftarrow \{\text{path} \oplus [q'] \mid q' \in \text{N}_q, \text{SWEPT}([q']) \cap V')^c = \emptyset\}$
  $Q \leftarrow$ SORTED$(Q, \text{key} = \text{Cost(path)} + \text{H}(q_{\text{last}}))$
  visited $\leftarrow$ visited $\cup \{q_{\text{last}}\}$

Algorithm 3 LAMB$(q_0, q_g, \text{G}\mathcal{O}, \text{G}\mathcal{M}, \text{G}\mathcal{V})$

plan $\leftarrow$ VA$^*(q_0, \{q_g\}, \text{G}\mathcal{O} \cup \text{G}\mathcal{M}, \emptyset, \text{GV})$ ▷ Direct
if plan **then**
  return plan
GS $\leftarrow$ G$\mathcal{O} \cup$ G$\mathcal{M}$
plan $\leftarrow$ VA$^*(q_0, \{q_g\}, \text{G}\mathcal{O} \cup \text{G}\mathcal{M}, \emptyset)$ ▷ Visibility-Relaxed
if plan **then**
  $\text{V}_{sg} \leftarrow$ SWEPT(plan) $\cap$ G$\mathcal{V}^c$
  $q \leftarrow q_0$
  for $q' \in$ VA$^*(q, V_{sg}, \text{G}\mathcal{O}, \text{H} = H_F))$ **do**
    plan $\leftarrow$ plan $\oplus$ LAMB$(q_0, q', \text{G}\mathcal{O}, \text{G}\mathcal{M}, \text{GV})$
    $q \leftarrow q'$
  plan $\leftarrow$ plan $\oplus$ LAMB$(q, q_g, \text{G}\mathcal{O}, \text{G}\mathcal{M}, \text{GV})$
  if None $\notin$ plan **then**
    return plan
plan $\leftarrow$ VA$^*(q_0, \{q_g\}, \emptyset, \text{GV})$ ▷ Collision-Relaxed
if plan **then**
  Obj $\leftarrow$ FIRSTCOLLISION(plan, G$\mathcal{O}$)
  GS $\leftarrow$ G$\mathcal{O} \cup$ G$\mathcal{M} \setminus \{\text{Object}\}$
  for $q_{pre}, q_{post} \in$ SAMPLEMANIP(Obj, GS) **do**
    mid $\leftarrow$ LAMB$(q_{pre}, q_{post}, \text{G}\mathcal{O}, \text{G}\mathcal{M}\backslash\text{Obj}, \text{GV})$
    G$\mathcal{O}' \leftarrow$ G$\mathcal{O} \cup$ SWEPT(plan, Obj)
    pre $\leftarrow$ LAMB$(q_0, q_{pre}, \text{G}\mathcal{O}', \text{G}\mathcal{M}\backslash\text{Obj}, \text{GV})$
    G$\mathcal{O}'' \leftarrow$ UPDATEPOSE(Obj, G$\mathcal{O}$)
    post $\leftarrow$ LAMB$(q_{post}, q_g, \text{G}\mathcal{O}'', \text{G}\mathcal{M}\backslash\text{Obj}, \text{GV})$
    if None $\notin$ pre $\oplus$ mid $\oplus$ post **then**
      return pre $\oplus$ mid $\oplus$ post
return None

robot positions, and goal region locations subject to the constraints of the task category. Below we describe each of the task categories and baselines used in our evaluation.

A. Task Categories

Simple Navigation is the simplest task category in which no obstacles need to be moved for the robot to reach the navigation goal (Figure 4a). **Visibility** tasks are inspired by problems from visibility-aware motion planning literature [10], [11], [12], [13], [14]. In these tasks, it is impossible to navigate to the goal directly due to visibility constraints. (Figure 4b) shows an instance wherein the robot can only move down the hallway sideways, so it must view the hallway from outside of it before moving through it to avoid collision with unseen areas of the robot's workspace. **Movable Obstacles** tasks are standard NAMO problems with movable obstacles that are fully visible from the initial state. These tasks typically require no additional visibility reasoning (Figure 4c). **Obstructed Visibility** tasks have visibility constraints similar to the visibility task, but they require observations from perspectives that cannot be reached without moving one or more obstacles (Figure 4e). **Occluding Obstacles** tasks involve movable objects that mostly or fully obstruct the robot's vision during interaction. Solutions to these tasks often require viewing certain regions before interacting with an object. Figure 4d shows an instance and Figure 2 shows an example plan on that instance. Lastly, **Obstructed Affordances** tasks require manipulation of obstacles from configurations unreachable without manipulating other obstacles. Figure 4f shows an example instance where the box object needs to be pushed but cannot be pushed directly because it would block the goal. It also cannot immediately be pushed from the bottom because of visibility constraints at the top of the box. The robot must move the obstructing chair and then push the box from the bottom or top.

B. Baselines

We compared LAMB to four search baselines with visibility constraints. The **VA-Star** baseline performs an VA* search in the discretized configuration space with a distance-to-goal heuristic. An additional constraint was added to the VA* search that limited actions to those that did not travel through unviewed regions. The A* baseline was only able to succeed on the simple navigation task where the heuristic was a useful metric. When direction navigation to the goal was impossible, VA* would default to an exhaustive search until timeout.

The **Fully Observable NAMO** baseline is a solution to fully observable NAMO problems [5]. This baseline first finds a relaxed path to the goal through movable obstacles. It then considers transfer paths for each obstacle in reverse order of collision starting from the goal. For each movable object the planner considers, it adds artificial collision constraints for motion on the next obstacle it tries to move. This search process is performed in a depth-first manner where infeasible motion constraints are terminal

	Simple Navigation	Visibility	Movable Obstacles	Obstructed Visibility	Occluding Obstacles	Obstructed Affordance
VA*	**5/5**	0/5	0/5	0/5	0/5	0/5
NAMO	**5/5**	0/5	4/5	0/5	0/5	0/5
FO-NAMO	**5/5**	0/5	**5/5**	0/5	0/5	0/5
VAMP	**5/5**	**5/5**	0/5	0/5	0/5	0/5
LaMB	**5/5**	**5/5**	**5/5**	**5/5**	**5/5**	**5/5**

TABLE I
EXPERIMENTAL RESULTS

search nodes. Because of the backward planning from the goal, it is impossible to enforce visibility constraints, so this baseline assumes all unviewed space is free. Our experiments show that this baseline works when obstacles obstruct all paths to the goal but fails when visibility constraints limit obstacle motion.

The **Constrained-NAMO** baseline is inspired by related work in visual NAMO [18]. This algorithm enumerates through all known visible objects, tests if moving that object will result in a shorter path to the goal, and moves the object if so. Similar to the VA* baseline, visibility constraints are placed on the low-level configuration-space search. Our results show that this baseline fails when directly moving obstacles is impossible due to visibility or obstruction.

The **VAMP** baseline is a state-of-the-art algorithm for visibility-aware motion planning [10]. Similar to FO-NAMO and LAMB , VAMP performs back chaining from the goal by creating intermediate subgoals based on failure from relaxed goal planning. Instead of subgoals involving movable obstacle manipulation, VAMP first tries to plan directly to the goal while relaxing vision constraints. It then identifies the regions of the workspace that were traversed through but not viewed and sets viewing those regions as a subgoal. Because VAMP does not consider setting object manipulation subgoals, it only succeeds on simple navigation and visibility tasks.

Our results show that LAMB is the only algorithm capable of solving the last three tasks that each require some reasoning about the interplay between visibility and manipulation.

C. Results

Each task was run with five different seeds on each algorithm. The total success rate out of those five runs is reported in Table I. As expected, the VA* baseline was only capable of solving simple navigation tasks. VA*, unlike other motion planning algorithms, is not probabilistically complete due to visibility-based path dependence. This probabilistic incompleteness leads to definite failure when the necessary visibility constraints are not resolved via the shortest path to a configuration. Like other motion planning algorithms, VA* slows down exponentially with the increasing configuration space dimensions that come with additional movable obstacles, leading to a timeout on the NAMO problems. The Fully Observable and Constrained NAMO baselines mostly succeeded at the simple navigation and movable obstacles tasks. However, the Fully Observable baseline sometimes fails on the NAMO task because it attempts to place movable objects in unseen regions or navigate through unseen regions while holding an object. These two baselines failed at all other tasks because they did not consider visibility as a potential subgoal. The VAMP baseline succeeded at all tasks where visibility was the only constraint (simple navigation and visibility constrained) but failed when placed in an environment where moving obstacles was necessary. LAMB succeeded on all seeds for each task.

D. Experimental Setup

We set up our simulated experiments in PyBullet [20]. Our robot model was a Kinova dual-arm MOVO with a head-mounted Kinect camera. All algorithms used PyBullet's built-in inverse kinematics module to determine joint positions for the base and arm configurations during manipulation and navigation. The perceptual input to each planning algorithm is the RGB, Depth, and ground truth segmentation data provided by the head-mounted camera. The image returned is 512×512 with a horizontal and vertical field of view of $90°$. An example of our simulated setup can be seen in Figure 1. The visibility and occupancy grids are maintained in simple list structures, are updated with vectorized NumPy operations, and have a fixed resolution of 0.1 meters. All experiments were run on 6 Intel Core i7-10750H CPUs with 16GB of ram for a maximum of 2 hours before timeout. Our code is made publicly available to ensure reproducibility. [1]

VI. CONCLUSION

In this paper, we present a new problem formulation, VANAMO, that describes a class of problems for navigation with movable obstacles and partial visibility. We also proposed an algorithm, LAMB, that solves VANAMO problems. We demonstrate LAMB on a number of complex navigation tasks that involve reasoning about visibility, object movability, and the interplay between them. Our simulated results demonstrate that LAMB outperforms other baselines that do not set both navigation and manipulation subgoals. While the simplicity of the API for this algorithm lends itself to real-world use, many challenges will need to be tackled before deploying this system on a real robot. Reliable robot localization, object segmentation, movability detection, manipulation dynamics prediction, and fault tolerance will all need to be considered when deploying on a robot system. We look forward to tackling these in future work, and we hope our open-source benchmarks and algorithms will prove useful to other researchers attempting to build mobile-base manipulation systems that operate in unknown environments.

[1] https://github.com/aidan-curtis/movo_manipulation

REFERENCES

[1] J. J. Kuffner and M. Stilman, "Navigation among movable obstacles," 2007.

[2] M. Stilman and J. Kuffner, "Navigation among movable obstacles: real-time reasoning in complex environments," in *4th IEEE/RAS International Conference on Humanoid Robots, 2004.*, vol. 1, 2004, pp. 322–341 Vol. 1.

[3] M. Stilman and J. J. Kuffner, "Planning among movable obstacles with artificial constraints," *The International Journal of Robotics Research*, vol. 27, pp. 1295 – 1307, 2006.

[4] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3327–3332.

[5] S. K. Moghaddam and E. Masehian, "Planning robot navigation among movable obstacles (namo) through a recursive approach," *Journal of Intelligent & Robotic Systems*, vol. 83, pp. 603–634, 2016.

[6] D. Nieuwenhuisen, A. van der Stappen, and M. H. Overmars, "An effective framework for path planning amidst movable obstacles," in *WAFR*, 2006.

[7] M. Levihn, J. Scholz, and M. Stilman, "Hierarchical decision theoretic planning for navigation among movable obstacles," in *WAFR*, 2012.

[8] K. Ellis, H. Zhang, D. Stoyanov, and D. Kanoulas, "Navigation among movable obstacles with object localization using photorealistic simulation," 07 2022.

[9] J. Scholz, N. Jindal, M. Levihn, C. Isbell, and H. Christensen, "Navigation among movable obstacles with learned dynamic constraints," 10 2016, pp. 3706–3713.

[10] G. Goretkin, L. P. Kaelbling, and T. Lozano-Pérez, "Look before you sweep: Visibility-aware motion planning," 2019.

[11] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1071–1078.

[12] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3d," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2011, pp. 351–356.

[13] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3d exploration," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1462–1468.

[14] M. Lauri and R. Ritala, "Planning for robotic exploration based on forward simulation," *Robotics and Autonomous Systems*, vol. 83, 02 2015.

[15] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, 1997, pp. 146–151.

[16] 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 *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2018, pp. 156–163.

[17] M. Levihn, M. Stilman, and H. Christensen, "Locally optimal navigation among movable obstacles in unknown environments," in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 86–91.

[18] H.-N. Wu, M. Levihn, and M. Stilman, "Navigation among movable obstacles in unknown environments," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 1433–1438.

[19] Y. Kakiuchi, R. Ueda, K. Kobayashi, K. Okada, and M. Inaba, "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*, 2010, pp. 1696–1701.

[20] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.