Efficient Navigation Among Movable Obstacles using a Mobile Manipulator via Hierarchical Policy Learning

Taegeun Yang[1], Jiwoo Hwang[2], Jeil Jeong[2], Minsung Yoon[1] and Sung-Eui Yoon[1†]

***Abstract*— We propose a hierarchical reinforcement learning (HRL) framework for efficient Navigation Among Movable Obstacles (*NAMO*) using a mobile manipulator. Our approach combines interaction-based obstacle property estimation with structured pushing strategies, facilitating the dynamic manipulation of unforeseen obstacles while adhering to a pre-planned global path. The high-level policy generates pushing commands that consider environmental constraints and path-tracking objectives, while the low-level policy precisely and stably executes these commands through coordinated whole-body movements. Comprehensive simulation-based experiments demonstrate improvements in performing *NAMO* tasks, including higher success rates, shortened traversed path length, and reduced goal-reaching times, compared to baselines. Additionally, ablation studies assess the efficacy of each component, while a qualitative analysis further validates the accuracy and reliability of the real-time obstacle property estimation.**

I. Introduction

Robust robot navigation in complex environments is crucial for applications ranging from delivery [1] to warehouse automation [2]. While recent methods excel at collision avoidance, they could fail when physical object manipulation is necessary to create feasible paths, for instance, when narrow passages are blocked by obstacles, as shown in Fig. 1.

Navigation Among Movable Obstacles (*NAMO*) research addresses this challenge by enabling robots to manipulate objects (i.e., movable obstacles) to actively create navigable regions. Conventional offline *NAMO* methods require complete environmental knowledge, whereas recent online approaches operate with minimal or without global information, such as floor plans that only include large structures like walls, to generate coarse global paths and dynamically respond to encountered obstacles during navigation. (refer to Sec. II-A)

By leveraging the advancements in reinforcement learning (RL), some researchers in *NAMO* tasks have enhanced decision-making processes for robots, enabling efficient navigation while manipulating objects when necessary. However, existing research overlooks physical attributes of movable obstacles, such as mass, friction, and center of mass, which are critical for effective manipulation. To address this limitation, we adopt real-time property estimation approaches, allowing robots to better understand and interact with objects. This integration enhances manipulation accuracy and eventually improves navigation efficiency. (refer to Sec. II-B)

[1]T. Yang, M. Yoon, and S. Yoon are with the School of Computing in Korea Advanced Institute of Science and Technology (KAIST), Daejeon, 34141, Republic of Korea.

[2]J. Jeong and J. Hwang are with the Robotics Program at the same institute, KAIST.

[†]S. Yoon is a corresponding author; `sungeui@kaist.edu`.

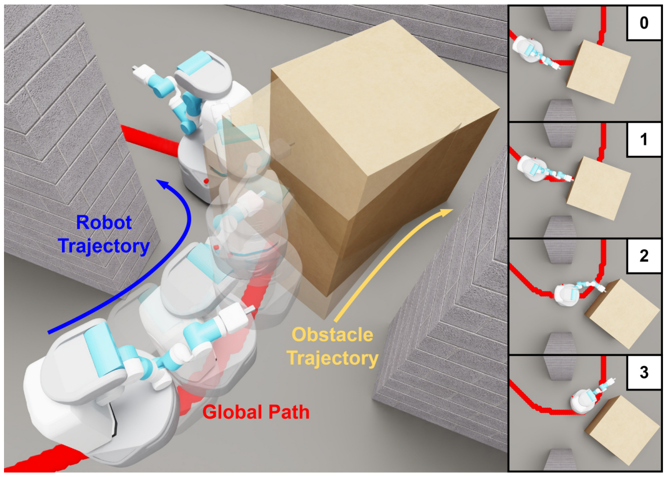

Fig. 1: **Whole-Body Mobile Manipulator Control for *NAMO*.** Illustration of the *NAMO* task, where the mobile manipulator follows a global path (red) while actively clearing a blocking obstacle using whole-body coordinated motions. During the interaction, the robot continuously tracks the global path with its base, while simultaneously pushing the obstacle aside to clear the way using its manipulator arm. The right sequence shows the interaction process between the robot and the obstacle.

Furthermore, we employ hierarchical reinforcement learning (HRL) frameworks to manage the complexity of *NAMO* tasks. By decoupling high-level decision-making from low-level motor execution, HRL allows robots to generate strategic commands, such as determining where to push a blocking obstacle with the manipulator's end-effector, and how quickly to move the base at a specific angle, as illustrated in Fig. 1. These commands are then executed with precision through a low-level controller. This hierarchical structure not only streamlines the decision-making process but also enables more efficient and adaptive behaviors in complex, unstructured environments, resulting in improved overall *NAMO* task performance. (refer to Sec. II-C)
In this work, our main contributions are *threefold*:

- A HRL framework that integrates mobile navigation with obstacle manipulation using a seven degrees-of-freedom (DOFs) mobile manipulator, the Fetch, to enable efficient navigation in unstructured environments for *NAMO* tasks;
- An interaction-based property estimation module that facilitates adaptive interaction with diverse movable objects;
- Comprehensive simulation-based experiments demonstrating improved navigation performance, in terms of success rates, path length, and goal-reaching times, along with ablation studies assessing the effectiveness of each component and qualitative analysis of the property estimation.

II. Related Work

A. Navigation Among Movable Obstacles (NAMO)

NAMO represents robot navigation tasks in environments where obstacles need to be physically manipulated to create traversable paths toward goal locations. Traditional approaches [3]–[6] assume complete environmental knowledge, including floor plans and detailed obstacle information (e.g., poses, geometries, movability), limiting their applicability in dynamic and unexplored environments. To mitigate this requirement, recent online methods [7]–[12] utilize approximate global paths from coarse floor maps or even operate with only local sensing information, dynamically responding to unexpected obstacles during navigation toward goals.

Recent advancements in reinforcement learning (RL) have enabled optimal decision-making in complex *NAMO* tasks that require simultaneous navigation and object manipulation [13], [14]. However, existing RL-based methods overlook the physical properties of obstacles during interaction, such as mass, friction, or center of mass, often resulting in inaccurate manipulation and reduced navigation efficiency, as shown in Sec. V. In contrast, we incorporate real-time property estimation within our framework, improving obstacle manipulation and ultimately enhancing navigation efficiency.

B. Interaction-Based Estimation of Object Properties

Accurate estimation of physical properties is essential for various domains, including system simulation [15], legged locomotion [16], and object manipulation [17]. Properties, such as mass, center of mass, and friction, significantly influence motion and interaction dynamics, making their estimation crucial for effective robotic control. While some objects have well-defined and predictable properties, real-world environments often contain unknown or visually similar objects with differing physical characteristics, requiring robots to infer these properties through direct interaction.

In object manipulation, traditional analytical approaches often struggle with real-world uncertainties and object variability [18]–[21]. To address these limitations, recent methods utilize large-scale interaction data and learning-based inference for accurate and robust object property estimation [22]–[24]. Building on these advancements, we incorporate property estimation into RL frameworks for *NAMO* tasks, enabling more context-aware manipulation policies.

C. Hierarchical Reinforcement Learning

Hierarchical control structures simplify complex robotic tasks by decoupling high-level decision-making from low-level execution [25]–[30]. For instance, in navigation, high-level controllers generate body velocity independent of the robot's specific embodiment [25]–[27]. Likewise, in manipulation, high-level controllers determine interaction strategies, such as motor skill selection [28] and sub-goal generation [29], [30], while low-level controllers execute high-level commands to generate precise motions for completing tasks.

To resolve the complexity of *NAMO* tasks, we adopt the hierarchical structure, where the high-level command space is defined as a pushing command space (refer to Sec. IV-B). The high-level controller generates pushing commands to accurately track the planned global path while clearing an obstacle through manipulation, and the low-level controller translates the high-level pushing commands into coordinated whole-body movements. Our hierarchical framework enhances navigation efficiency compared to an end-to-end structure, as exhibited in Sec. V.

III. Variable Notation

This section defines the variable notation used throughout this manuscript. Vectors $\mathbf{p}, \mathbf{v}$, and $\dot{\mathbf{v}} \in \mathbb{R}^3$ represent position, velocity, and acceleration in Cartesian space. Orientation is expressed using Euler angles in the XYZ convention as $\boldsymbol{\theta} \in \mathbb{R}^3$, with $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\omega}} \in \mathbb{R}^3$ denoting angular velocity and angular acceleration, respectively. Alternatively, orientation can be represented as a quaternion $\mathbf{Q} \in \mathbb{R}^4$. Subscripts indicate entities or coordinate components, while superscripts denote reference frames which are omitted when identical to the body frame. We use the labels '*b*', '*o*', and '*ee*' to refer to the robot base, object, and end-effector frames, respectively. For instance, $\mathbf{p}_o^b$ denotes the position of an obstacle relative to the robot's base frame b, with $p_{o,x}^b$ representing its *x*-component. Manipulator joint states, including positions, velocities, and accelerations, are defined as $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}} \in \mathbb{R}^7$, corresponding to the seven DOFs of the robotic arm. The obstacle's size along each axis is denoted by $\mathbf{d}_o = [d_{o,x}, d_{o,y}, d_{o,z}]$. Finally, a binary function $\text{is\_contact}(\cdot, \cdot)$ returns 1 if two entities are in contact, and 0 otherwise.

IV. Method

We aim to enhance navigation efficiency in *NAMO* tasks by enabling a mobile manipulator to push obstacles with its arm while allowing its base to maintain adherence to the global path and minimize deviations. To achieve this, we introduce a hierarchical reinforcement learning (HRL) framework that integrates object manipulation and navigation through whole-body movements, as illustrated in Fig. 2. A high-level controller selects an optimal pushing strategy based on the environment and target destination, while a low-level controller executes precise motions for effective object manipulation. The following descriptions detail the problem formulations and hierarchical controller components.

A. NAMO Task Formulation

We consider navigation in a known static map M_{static} (i.e., floor plans) with movable obstacles whose locations and physical properties are *a priori* unknown until detected through local sensing—similar to how people rely on floor plans to determine a rough path to their destination while handling unexpected obstacles along the way. Given a goal, the robot plans a global path P on the static map M_{static}, ignoring unknown obstacles, and tracks the global path using a mobile base PD controller. Meanwhile, upon encountering the obstacle within a predefined distance (i.e., local sensing range) that obstructs the path, the robot actively interacts with the obstacle to clear the way while progressing toward

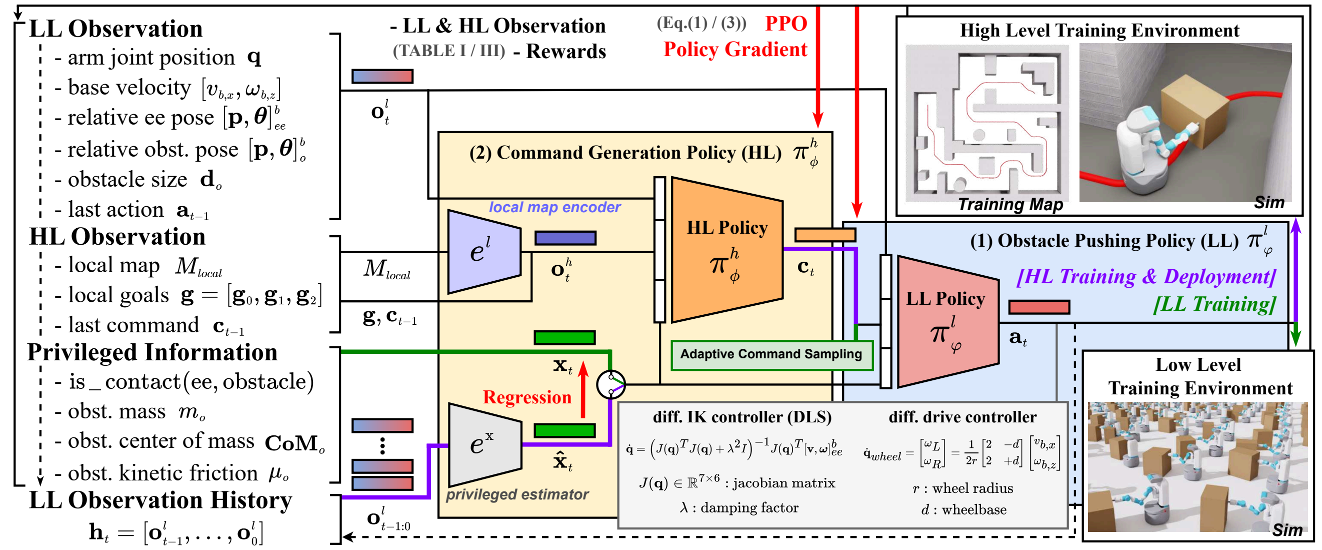
LL Observation
- arm joint position $\mathbf{q}$
- base velocity $[v_{b,x}, \omega_{b,z}]$
- relative ee pose $[\mathbf{p}, \boldsymbol{\theta}]^b_{ee}$
- relative obst. pose $[\mathbf{p}, \boldsymbol{\theta}]^b_o$
- obstacle size $\mathbf{d}_o$
- last action $\mathbf{a}_{t-1}$
HL Observation
- local map M_{local}
- local goals $\mathbf{g} = [\mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_2]$
- last command $\mathbf{c}_{t-1}$
Privileged Information
- is_contact(ee, obstacle)
- obst. mass m_o
- obst. center of mass $\mathbf{CoM}_o$
- obst. kinetic friction μ_o
LL Observation History
$\mathbf{h}_t = [\mathbf{o}^l_{t-1}, \dots, \mathbf{o}^l_0]$
- LL & HL Observation
(TABLE I / III) - Rewards
(Eq.(1) / (3)) PPO Policy Gradient
$\mathbf{o}^l_t$
(2) Command Generation Policy (HL) π^h_ϕ
local map encoder
e^l
$\mathbf{o}^h_t$
HL Policy π^h_ϕ
$\mathbf{c}_t$
M_{local}
$\mathbf{g}, \mathbf{c}_{t-1}$
$\mathbf{x}_t$
Regression
e^x
$\hat{\mathbf{x}}_t$
privileged estimator
$\mathbf{o}^l_{t-1:0}$
Adaptive Command Sampling
(1) Obstacle Pushing Policy (LL) π^l_φ
[HL Training & Deployment]
[LL Training]
LL Policy π^l_φ
$\mathbf{a}_t$
diff. IK controller (DLS)
$\dot{\mathbf{q}} = \left(J(\mathbf{q})^T J(\mathbf{q}) + \lambda^2 I\right)^{-1} J(\mathbf{q})^T [\mathbf{v}, \boldsymbol{\omega}]^b_{ee}$
$J(\mathbf{q}) \in \mathbb{R}^{7\times6}$: jacobian matrix
λ : damping factor
diff. drive controller
$\dot{\mathbf{q}}_{wheel} = \begin{bmatrix}\omega_L \\ \omega_R\end{bmatrix} = \frac{1}{2r}\begin{bmatrix}2 & -d \\ 2 & +d\end{bmatrix}\begin{bmatrix}v_{b,x} \\ \omega_{b,z}\end{bmatrix}$
r : wheel radius
d : wheelbase
High Level Training Environment
Training Map
Sim
Low Level Training Environment
Sim

Fig. 2: **Training Framework**. Our framework consists of two stages: (1) The low-level policy learns to execute whole-body actions that achieve the commanded robot-obstacle kinematic configuration (i.e., pushing command) while maintaining stable contact to push obstacles (refer to Sec. IV-B). An adaptive command sampling strategy is used to train the policy, enabling it to handle a broader range of commands (Sec. IV-C.3). (2) The high-level policy is trained to generate pushing commands for low-level that guide the robot's base along the planned path while clearing obstacles, considering environmental factors. A history encoder is trained to estimate contact state and obstacle properties from low-level observation histories (Sec. IV-D.2). The term 'obst.' in the figure denotes 'obstacle' for brevity.

the goal. We assume all encountered obstacles are movable and cubic, with widths and depths between $0.5\,\mathrm{m}$ and $0.8\,\mathrm{m}$, heights ranging from $0.6\,\mathrm{m}$ to $1.0\,\mathrm{m}$, and randomized physical properties—including mass, center of mass (CoM), and a friction coefficient—introducing variability that necessitates adaptive interaction strategies for effective manipulation.

B. Robot-Obstacle Pushing Commands

In our hierarchical framework, a high-level policy π^h_ϕ defines pushing strategies (i.e., commands), which a low-level policy π^l_φ executes via whole-body control. We represent this strategy by defining the robot's kinematic configuration relative to the obstacle as a pushing command, which regulates obstacle motion through interaction and serves as the interface between the high-level and low-level policies.

Incorporating full kinematic relationship—including all positional and motion-related variables—into the command space poses significant challenges, as the high-dimensional space increases learning complexity and complicates kinematic feasibility validation. To address this, we select key components that can simplify the command structure while preserving essential aspects of robot-obstacle interaction. Specifically, the pushing command $\mathbf{c} \in \mathbb{R}^3$ is composed of three components: $(p^{cmd}, \theta^{cmd}, v^{cmd})$, as depicted in Fig. 3. Here, p^{cmd} is lateral contact position on the obstacle's contact face; θ^{cmd} denotes the yaw angle of the obstacle's contact area relative to the robot's frame; and v^{cmd} indicates the mobile base's forward velocity during interaction.

While modeling the interaction-pushing command $\mathbf{c}$ in the low-dimensional space may cause tilting or loss of contact, the low-level controller, trained to account for these factors by generating appropriate whole-body motions based on the object's physical properties, compensates by adaptive movements, such as modifying contact height.

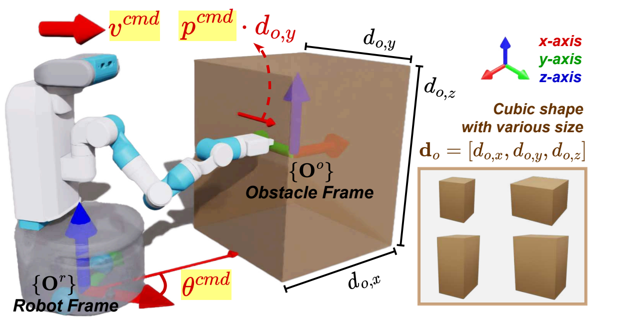

Fig. 3: **Pushing Command**. The pushing command consists of p^{cmd}, θ^{cmd}, and v^{cmd}, shown in red in the figure. p^{cmd} specifies the contact point on the obstacle contact face as $p^{cmd} \cdot d_{o,y}$ along the obstacle's y-axis. θ^{cmd} is the yaw angle of the contact face relative to the robot frame. v^{cmd} denotes the robot's base linear velocity.

C. A Low-Level Policy

The low-level policy π^l_φ enables the robot to execute whole-body movements for satisfying pushing commands $\mathbf{c}$, dynamically adapting to object tilting while maintaining stable contact through its end-effector. The following description details the RL formulation for the low-level policy.

1) Problem Formulation: We formulate the obstacle-pushing problem as a Markov Decision Process (MDP), represented by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho_0, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ the state transition function, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function. The term ρ_0 represents the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. The objective of RL is to maximize the expected cumulative reward, given by:

$$J(\varphi) = \mathbb{E}_{\mathbf{c}_t \sim P(\mathbf{c})}\left[\mathbb{E}_{(\mathbf{s},\mathbf{a}) \sim \pi^l_\varphi}\left[\sum_{t=0}^{T} \gamma^t \mathcal{R}^l(\mathbf{s}_t, \mathbf{a}_t | \mathbf{c}_t)\right]\right] \quad (1)$$

TABLE I: Reward Functions for $\mathcal{R}^l = \mathcal{R}^{\text{cmd}} + \mathcal{R}^{\text{stable}} + \mathcal{R}^{\text{areg}}$

Reward	Function Expression
Command Tracking Rewards: $\mathcal{R}^{\text{cmd}} = \sum_{i=0}^{2} r_i^{\text{cmd}}$	
r_0^{cmd}	$k_0^l\ exp(-2\,\lvert p^{cmd} - p^o_{ee,y}/d_{o,y}\rvert\ /\ 0.1)$
r_1^{cmd}	$k_1^l\ exp(-\lvert \theta^{cmd} - \theta^b_{area,z}\rvert\ /\ 0.1)$
r_2^{cmd}	$k_2^l\ exp(-2\,\lvert v^{cmd} - v_{b,x}\rvert\ /\ 0.1)$
Obstacle Stability Rewards: $\mathcal{R}^{\text{stable}} = \sum_{i=0}^{4} r_i^{\text{stable}}$	
r_0^{stable}	k_3^l is_contact (ee, obstacle)
r_1^{stable}	$k_4^l\ (1 - \tanh(2\ \cos^{-1}\left(\left\lvert \mathbf{Q}_{ee}^{b^{-1}} \cdot \mathbf{Q}_o^b \right\rvert\right)\ /\ 0.1))$
r_2^{stable}	$-k_5^l$ is_contact(base, obstacle)
r_3^{stable}	$-k_6^l\ \tanh(\lvert\theta^b_{o,y}\rvert\ /\ 0.1)$
r_4^{stable}	$-k_7^l\ (\lvert\lvert\dot{v}^b_{o,x}\rvert\rvert_2 + \lvert\lvert\dot{v}^b_{o,y}\rvert\rvert_2)$
Action Regularization Rewards: $\mathcal{R}^{\text{areg}} = \sum_{i=0}^{2} r_i^{\text{areg}}$	
r_0^{areg}	$-k_8^l\ \lvert\lvert\ddot{\mathbf{q}}\rvert\rvert_2$
r_1^{areg}	$-k_9^l\ \lvert\lvert\dot{\mathbf{v}}_b\rvert\rvert_2$
r_2^{areg}	$-k_{10}^l\ \lvert\lvert\mathbf{a}_t - \mathbf{a}_{t-1}\rvert\rvert_2$

- $\mathcal{R}^l$: a reward function for a low-level policy (refer to Sec. IV-C)
- $\theta^b_{area,z}$: a yaw angle of an obstacle's contact face in a robot frame
- $k^l_{0,1,\ldots,10}$: non-negative coefficients

where φ denotes the low-level policy parameters to be optimized, and $\mathbf{c}_t$ represents commands sampled at each time step t from an adaptively updated command distribution $P(\mathbf{c})$. Please refer to Sec. IV-C.3 for this update procedure. At the start of each episode, the robot joints are initialized to a predefined configuration, and the obstacle is randomly placed at a fixed distance from the robot's base.

2) A Policy & Reward Functions: The low-level policy π^l_φ receives as input the vector $[\mathbf{o}^l, \mathbf{c}, \mathbf{x}]$, where $\mathbf{o}^l$ represents the low-level observation. This observation is structured as $[\mathbf{q}, v_{b,x}, \omega_{b,z}, \mathbf{p}^b_{ee}, \boldsymbol{\theta}^b_{ee}, \mathbf{p}^b_o, \boldsymbol{\theta}^b_o, \mathbf{d}_o, \mathbf{a}_{t-1}]$. The pushing command is given by $\mathbf{c}$, while the privileged information $\mathbf{x}$ consists of [is_contact$(ee, obstacle)$, m_o, $\mathbf{CoM}_o$, μ_o], including the contact state between the end-effector and the obstacle, along with the obstacle's physical properties. Property ranges and descriptions are detailed in TABLE II. During training, ground-truth privileged information is used, whereas deployment relies on estimated values, as mentioned in Sec. IV-D.2. The low-level reward function $\mathcal{R}^l$ consists of three main terms: command tracking rewards $\mathcal{R}^{\text{cmd}}$ which encourages adherence to the command; stability rewards $\mathcal{R}^{\text{stable}}$ which maintains contact and prevents obstacle rollover; and action regularization rewards $\mathcal{R}^{\text{areg}}$ which discourages excessive motion. Maintaining contact is crucial, as inconsistent forces can lead to obstacle tilting or slipping, resulting in unstable interactions [31]–[33]. TABLE I provides detailed formulations. We omit the time step t notation hereafter.

The action $\mathbf{a}_t \in \mathbb{R}^8$, comprising six dimensions for the arm and two for the base, is generated by the policy. The arm action is defined by the desired twist of the end-effector $[\mathbf{v},\ \boldsymbol{\omega}]^r_{ee} \in \mathbb{R}^6$, which is converted into joint velocities using a differential inverse kinematics (IK) controller [34], resulting in joint velocities $\dot{\mathbf{q}}$, used to update the joint positions at each time step as $\mathbf{q} + \dot{\mathbf{q}} \cdot \Delta T$. The base action is defined as $[v_{b,x},\ \omega_{b,z}] \in \mathbb{R}^2$, reflecting the base's differential-drive kinematics, which naturally use linear and angular velocities to describe planar motion. Both arm and base actions are then sent to joint impedance controllers for torque computation.

TABLE II: Obstacle Physical Properties Ranges and Descriptions

Term (dim.)	Range (Unit)	Description
m_o (1)	(5.0, 30.0) (kg)	obstacle mass
$\mathbf{CoM}_o$ (3)	(-0.4, 0.4) $(-)$	shifted obstacle CoM
μ_o (1)	(0.2, 0.6) $(-)$	obstacle kinetic friction coefficient

- An obstacle CoM is set to $\mathbf{CoM}_o \odot \mathbf{d}_o$, where $\odot$ indicates element-wise multiplication and $\mathbf{d}_o$ denotes the obstacle size.
- During evaluation, these parameter ranges are expanded by 10%.

3) Adaptive Command Sampling: Learning over a large command space from scratch is challenging. While pushing an obstacle's center is straightforward, handling varied contact positions introduces complexity. Sampling from an excessively large command space may yield kinematically infeasible commands, hindering learning. To address this, we adopt a grid adaptation rule [35], which progressively expands the command sampling range for p^{cmd} and θ^{cmd} within the pushing command space $\mathbf{c}$. The velocity component v^{cmd} maintains a fixed sampling range, allowing the policy to focus more on interaction-related parameters. The sampling range updates based on command tracking rewards r_i^{cmd}. When all rewards exceed their thresholds $\gamma_i\,(i = 0, 1, 2)$, the command space expands, and the probability distribution is updated as follows:

$$P_{N+1}(p^{cmd}, \theta^{cmd} \oplus \Delta) = \begin{cases} \frac{1}{\lvert A\cup\Delta\rvert}\mathbb{U}_{A\cup\Delta}, & \text{if } \forall r_i > \gamma_i, \\ P_N(p^{cmd}, \theta^{cmd}), & \text{otherwise,} \end{cases} \tag{2}$$

where N is the episode number and $\oplus$ represents the Minkowski sum, expanding the command space by incorporating neighboring regions. A and Δ refer to the command space before expansion and the newly added region, respectively, with $\mathbb{U}_{A\cup\Delta}$ representing a uniform distribution over the expanded space $A \cup \Delta$.

D. A High-Level Policy

The high-level policy π^h_ϕ generates optimal pushing commands $\mathbf{c}$, allowing the robot's base to follow the planned path with minimal deviation while actively clearing obstacles with its arm and base motions. It also considers environmental constraints, including structures around the robot in the static map M_{static}, to recognize where the obstacle can be pushed. The following is the RL formulation of the high-level policy.

1) Problem Formulation: We formulate the pushing command generation problem as an MDP for decision-making, where the RL policy aims to maximize the expected cumulative reward, similar to the low-level policy's objective:

$$J(\phi) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{c}_t)\sim\pi^h_\phi}\left[\sum_{t=0}^{T} \gamma^t \mathcal{R}^h(\mathbf{s}_t, \mathbf{c}_t \mid \pi^l_\varphi)\right]. \tag{3}$$

Unlike the low-level policy, which directly outputs robot actions, the high-level policy generates pushing commands $\mathbf{c}_t$, executed by the low-level policy π^l_φ. During high-level policy training, the low-level policy remains frozen.

2) Privileged Knowledge Distillation: Efficient obstacle pushing requires accurate knowledge of physical properties and the current contact state, which are typically considered privileged information. As they are generally unavailable

TABLE III: Reward Functions for $\mathcal{R}^h = \mathcal{R}^{\text{path}} + \mathcal{R}^{\text{safe}} + \mathcal{R}^{\text{creg}}$

Reward	Function Expression
	Path Tracking Rewards: $\mathcal{R}^{\text{path}} = \sum_{i=0}^{2} r_i^{\text{path}}$
r_0^{path}	$k_0^h \, exp(-\lvert\theta^b_{\mathbf{g}_0,z}\rvert \;/\; 0.1)$
r_1^{path}	$k_1^h \, exp(-\lvert(1 - v_{b,x} \;/\; v_{b,x}^{max})\rvert \;/\; 0.1)$
r_2^{path}	$k_2^h \, \mathbb{I}\,(\min_i \lvert d_i\rvert > d_{\text{thr}} \wedge \text{ sign}(d_i) = \text{sign}(d_1), \forall i)$
	Safe Operation Rewards: $\mathcal{R}^{\text{safe}} = \sum_{i=0}^{1} r_i^{\text{safe}}$
r_0^{safe}	$-k_3^h \text{ is\_contact}(\text{base}, M_{static})$
r_1^{safe}	$-k_4^h \text{ is\_contact}(\text{obstacle}, M_{static})$
	Command Regularization Rewards: $\mathcal{R}^{\text{creg}} = \sum_{i=0}^{1} r_i^{\text{creg}}$
r_0^{creg}	$\mathcal{R}^{stable} + \mathcal{R}^{areg}$
r_1^{creg}	$-k_5^h \, \lvert\lvert \mathbf{c}_t - \mathbf{c}_{t-1} \rvert\rvert_2$

- $\mathcal{R}^h$: a reward function for a high-level policy (refer to Sec. IV-D)
- $k^h_{0,1,\ldots,5}$: non-negative coefficients
- $\mathbb{I}$: an indicator returning 1 if the condition is met, and 0 otherwise.
- $d_i (i = 0, ..., 3)$: distances from the vector connecting the robot base to the local goal $\mathbf{g}_1$ to each ground-contacting corner of the obstacle; sign indicates the corner's side relative to this vector
- d_{thr}: a clearance threshold for the base to move without collision

during deployment, we train a network e^{x} online during high-level policy training to estimate both contact states and obstacle physical properties based on the low-level observation history. The network learns to predict this privileged information using the following loss function:

$$\mathcal{L}_{adap} = \mathcal{L}_{contact} + \mathcal{L}_{prop}, \tag{4}$$

where $\mathcal{L}_{contact}$ represents the Binary Cross-Entropy (BCE) loss for contact prediction and $\mathcal{L}_{prop}$ denotes the Mean Squared Error (MSE) loss for physical property estimation.

Training e^{x} using low-level observation history, where high- and low-level policies rely on ground truth privileged information to generate policy outputs, poses challenges during deployment due to data distribution shifts. In early interactions, the estimated privileged information $\hat{\mathbf{x}}$ is highly uncertain, causing deviations from trained behavior. This discrepancy alters observations and induces distribution shifts between training and deployment. To mitigate this issue, we adopt a progressive integration strategy that gradually transitions policies from using $\mathbf{x}$ to relying entirely on $\hat{\mathbf{x}}$. During training, the privileged input $\mathbf{x}^{comb}$ is defined as:

$$\mathbf{x}^{comb} = (1 - \alpha) \cdot \mathbf{x} + \alpha \cdot \hat{\mathbf{x}}, \tag{5}$$

where α increases linearly from 0 to 1 over the course of training. Initially, $\mathbf{x}$ provides stable supervision for policy optimization under ideal conditions. As training progresses, reliance on $\hat{\mathbf{x}}$ increases, allowing the policy to adapt to deployment conditions and align the training and deployment distributions.

3) A Policy & Reward Functions: The high-level policy takes a vector $[\mathbf{o}^h, \mathbf{o}^l, \mathbf{x}^{comb}]$ as input. The high-level observation $\mathbf{o}^h$ includes $\boldsymbol{l}_{map}$, an encoded representation of the local map M_{local}, which is extracted from M_{static} and processed through the map encoder e^l. The local map covers a $4 \times 4\,\text{m}^2$ area centered $2\,\text{m}$ ahead of the robot base, with a resolution of $0.05\,\text{m}$ per pixel. Additionally, $\mathbf{o}^h$ contains upcoming local goals $\mathbf{g}_0$, $\mathbf{g}_1$, and $\mathbf{g}_2$, indicating 2D positions (x, y) on the planned path P at $0.5\,\text{m}, 1.0\,\text{m}$, and $2.0\,\text{m}$ ahead, along with the last executed command $\mathbf{c}_{t-1}$. The high-level reward function $\mathcal{R}^h$ comprises three components: path tracking rewards $\mathcal{R}^{\text{path}}$ which rewards the robot for following local goals and relocating obstacles beyond the clearance threshold d_{thr}; safe operation rewards $\mathcal{R}^{\text{safe}}$ which penalizes unintended collision with static structures; and command regularization rewards $\mathcal{R}^{\text{creg}}$ which discourages abrupt changes in commands. Detailed reward formulations are provided in TABLE III.

TABLE IV: Network Architectures and Input-Output Specifications

NN.	Hidden Layers	Inputs (dim.)	Outputs
π^l_φ	[256, 128, 64]	$\mathbf{o}^l(32) \mid \mathbf{x}(6) \mid \mathbf{c}(3)$	$\mathbf{a}(8)$
π^h_ϕ	[256, 128, 64]	$\mathbf{o}^h(25) \mid \mathbf{o}^l(32) \mid \mathbf{x}^{comb}(6)$	$\mathbf{c}(3)$
e^{x}	LSTM + [128, 64, 32]	$\mathbf{o}^l(32)$	$\hat{\mathbf{x}}(6)$
e^l	CNN + [128, 64. 32]	$M_{local}(80 \times 80)$	$\boldsymbol{l}_{map}(16)$

E. Training Details

We employed Isaac Sim [36] for training, utilizing 1,024 parallel environments. We updated policies at 50 Hz, while we operated joint impedance control at 100 Hz. We used the Differential IK controller with the Damped Least Squares (DLS) method and a damping factor $\lambda = 0.02$. We initialized command ranges as $(p^{cmd}, \theta^{cmd})_{\text{init}} = (\pm 0.1, \pm 0.2)$ and expanded them incrementally by $\Delta = (\pm 0.05, \pm 0.1)$, restricting linear velocity command ranges to $(0.1, 0.4)$ m/s. We set command tracking reward thresholds to $\gamma_{0,1,2} = [0.6, 0.6, 0.5]$. We empirically found best-performing reward weights as $k^l_{i=0,\ldots,10} = [1.0, 1.0, 0.8, 1.5, 1.5, 10^2, 0.3, 3 \times 10^{-3}, 10^{-3}, 3 \times 10^{-3}, 3 \times 10^{-3}]$ for the low-level policy and $k^h_{i=0,\ldots,5} = [1.0, 0.8, 10^2, 10^2, 10^2, 0.2]$ for the high-level policy. We set the clearance threshold d_{thr} to 0.35 m, and we set $v^{max}_{b,x}$ to 0.4 m/s. We optimized both policies using the Proximal Policy Optimization (PPO) algorithm [37]. TABLE IV provides network architecture details.

TABLE V: Low-Level Command Tracking Errors and Contact Rate

	p^{cmd} error (-) (↓)	θ^{cmd} error (rad) (↓)	v^{cmd} error (m/s) (↓)	Contact Rate (%) (↑)
LL-P	0.129 ± 0.041	0.169 ± 0.065	0.079 ± 0.027	89.219 ± 8.935
LL+P	**0.055 ± 0.029**	**0.122 ± 0.059**	**0.049 ± 0.027**	**96.708 ± 2.621**

- P: Privileged information (refer to Sec. V-A)

V. Experiments

We conducted two simulation experiments to evaluate our approach: (1) analyzing the impact of privileged information on low-level policy performance and (2) assessing navigation performance in *NAMO* tasks across varying obstacle densities measured by success rate, path length, and completion time.

A. Effect of Privileged Information on Low-Level Policy

To assess the impact of privileged information on low-level policy performance, we compared policies trained with (LL+P) and without (LL-P) privileged information. TABLE V presents tracking errors for each pushing-command term and the consistency of maintaining contact with the obstacle.

TABLE VI: Navigation Among Movable Obstacles (*NAMO*) Experimental Results on Map 1 and Map 2

Methods	**SR** (%) (↑)	**SPL** (-) (↑)	**SCT** (-) (↑)
Map 1 (evaluation with random obstacle numbers of 2, 4, and 6, presented in that order)			
CA	92.00 / 72.00 / 52.00	84.43 / 63.86 / 44.23	77.94 / 56.59 / 37.82
AA [11]	96.00 / 89.33 / 78.67	88.61 / 80.81 / 69.35	82.38 / 71.99 / 60.59
Ours-P	86.67 / 74.00 / 64.67	85.41 / 71.09 / 62.09	85.25 / 67.35 / 55.80
Ours-H	94.00 / 84.67 / 76.00	90.81 / 81.63 / 73.46	91.16 / 78.10 / 73.46
Ours	**96.67** / 91.33 / 85.33	93.39 / 88.38 / 82.62	**95.89** / 88.26 / 80.28
Ours+G	**96.67 / 92.00 / 87.33**	**93.53 / 89.12 / 84.64**	95.86 / **89.12 / 81.96**
Map 2			
CA	94.00 / 88.00 / 80.00	85.97 / 78.03 / 68.36	75.75 / 66.99 / 57.18
AA [11]	95.33 / 89.33 / 84.67	87.56 / 81.36 / 75.43	77.44 / 70.32 / 62.53
Ours-P	88.67 / 84.67 / 76.67	87.53 / 83.37 / 75.56	84.71 / 78.12 / 67.93
Ours-H	82.00 / 70.67 / 57.33	80.81 / 69.55 / 56.43	78.35 / 65.18 / 50.97
Ours	**98.67** / 96.67 / 92.67	**97.70** / 95.79 / 91.42	**96.80** / 92.74 / 86.79
Ours+G	**98.67 / 97.33 / 94.67**	97.69 / **96.43 / 93.83**	96.76 / **93.36 / 89.18**

- **SR**: success rate, **SPL**: success-weighted path length (see Eq. 6), **SCT**: success-weighted completion time (see Eq. 7).

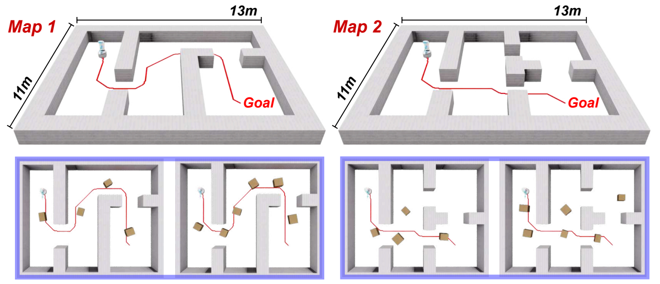

Fig. 4: **Evaluation Maps**. These maps are distinct from the training environments, with fixed start and goal positions. The red line represents the planned path P, computed using the A* algorithm. The bottom row illustrates example configurations for each map with 4 and 6 obstacles.

The LL+P policy exhibits improved command adherence, with reduced tracking errors across all command dimensions. Specifically, lower p^{cmd}, θ^{cmd}, and v^{cmd} errors demonstrate enhanced precision in executing high-level commands, ensuring more reliable robot-obstacle interaction. In addition, it maintains more stable and prolonged contact with the obstacle, reinforcing consistent interaction. These results validate that privileged information improves both control accuracy and interaction stability.

B. Navigation Efficiency in Cluttered Environments

1) Experiment Settings: We evaluated the navigation efficiency of our method on two maps, each with 2, 4, and 6 obstacles strategically placed to interfere with the planned global path, as shown in Fig. 4. For each map and obstacle count setting, we conducted 150 trials. In each experiment, obstacle properties are randomly assigned within the ranges specified in TABLE IV. The start and goal positions are fixed, and the global path P is computed using the A* algorithm [38], considering only static walls while ignoring movable obstacles. During navigation, if the robot encounters an obstacle within 1.5m that obstructs the path, our method uses the proposed policies to push it aside. Once the obstacle is cleared—its clearance exceeds the threshold d_{thr}—the robot reverts to tracking its planned path using base movements. This process continues until the robot either reaches the goal or fails according to predefined criteria described below.

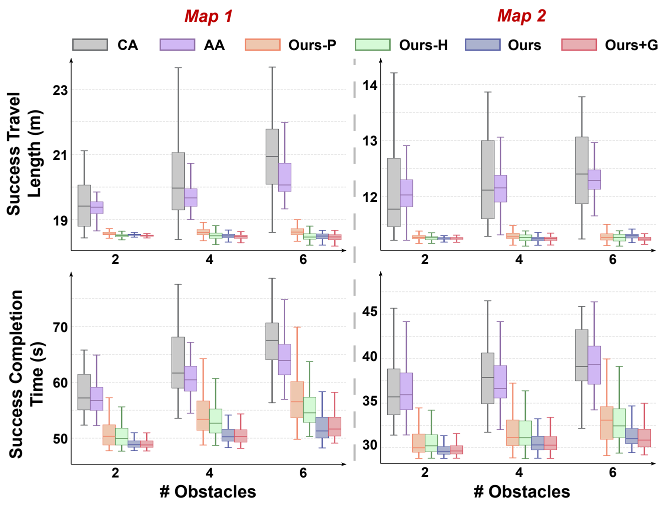

Fig. 5: **Navigation Efficiency for Success Cases**. Travel length and completion time are evaluated for successful trials across methods, measured under varying obstacle counts. **Ours** achieves consistently shorter distances and times with lower variance, demonstrating stable and efficient performance across all cases.

2) Baselines and Metrics: We established the following baselines along with our method's variants:

- **Collision Avoidance (CA)**: Re-plans paths iteratively to avoid obstacles without physical interaction.
- **Axis-Aligned Pushing (AA)** [11]: Re-plans paths with a straight-line pushing strategy, aligning the robot's base with the obstacle's axis to clear the obstructed path.
- **Ours w/o Privileged Information (Ours-P)**: Trains hierarchical policies without privileged information.
- **Ours w/o Hierarchical Structure (Ours-H)**: Trains a single policy end-to-end that directly generates low-level actions $\mathbf{a}_t$ (refer to Sec. IV-C.2), omitting hierarchical structure interconnected via pushing commands.
- **Ours**: Uses the hierarchical structure along with pushing commands and estimated privileged information in both high- and low-level policies (refer to Sec. IV).

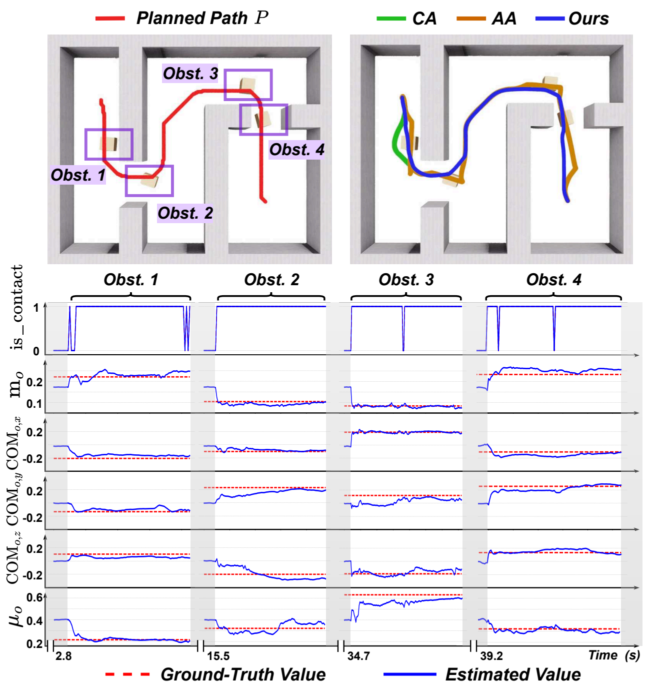

Fig. 6: **Qualitative Results on Map 1**. The top-row figures shows the planned path P and the robot's traces for **CA**, **AA**, and **Ours**, where **CA** fails to find a feasible path around the second obstacle, resulting in task failure. The below figures present the privileged information estimated by **Ours** during obstacle interactions alongside ground-truth values. Shaded gray regions indicate periods when the robot is not in contact with the obstacle.

- **Ours w/ GT Privileged Information (Ours+G)**: Uses ground-truth privileged information instead of estimated values through the network e^x (refer to Sec. IV-D.2).

To quantify navigation efficiency, we used three metrics: **Success Rate (SR)** is the percentage of trials in which the robot successfully reaches the goal. A trial is considered a failure if the robot exceeds a predefined time limit or if an obstacle collides with a wall or tips over. In the case of **CA**, failure additionally occurs when it cannot find a feasible path. **Success-Weighted Path Length (SPL)** [39] measures path efficiency by balancing optimality and success, defined as:

$$\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{L_i^*}{\max(L_i, L_i^*)}, \tag{6}$$

where S_i denotes success, L_i^* the optimal path length, and L_i the actual path length. Similarly, **Success-Weighted Completion Time (SCT)** [40] evaluates time efficiency:

$$\text{SCT} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{T_i^*}{\max(T_i, T_i^*)}, \tag{7}$$

where T_i^* is the optimal completion time, and T_i is the actual completion time. The optimal values L_i^* and T_i^* are determined from navigation in an obstacle-free environment.

3) Analysis of Results: TABLE VI presents the navigation performance across baseline methods. Our approach (**Ours**), integrating interaction-based obstacle property estimation and the hierarchical structure, consistently outperforms the baselines, particularly in environments with a high number of obstacles. **CA** frequently fails to find feasible paths in settings with many obstacles, resulting in a low success rate. Its avoidance-based strategy results in longer trajectories, reducing navigation efficiency. **AA** improves upon **CA** but lacks adaptability, as it ignores obstacle physical properties, leading to failed or unintended interactions. Additionally, its reliance on base-only pushing introduces redundant movements, further lowering efficiency (see Fig. 6). **Ours-P**, which does not estimate obstacle properties, exhibits significantly lower success rates and reduced efficiency compared to **Ours**, underscoring the importance of incorporating obstacle properties for adaptive manipulation. **Ours-H**, which removes the hierarchical structure, struggles with inefficient and ineffective learning due to the large search space, converging into suboptimal solutions and leading to poorer navigation performance, consistent with findings in [41]. **Ours+G**, which uses ground-truth privileged information, predictably achieves the highest performance, serving as the upper bound for our approach. Through the small performance gap between **Ours** and **Ours+G**, it can be inferred that **Ours** accurately estimates the necessary privileged information for efficient obstacle manipulation. Fig. 6 shows a qualitative result confirming the accuracy of **Ours**, as it reliably estimates obstacle properties in real time while tracking the planned global path with minimal deviation. This experiment confirms that integrating property estimation with a hierarchical policy framework enhances robustness and efficiency in *NAMO* tasks.

Fig. 5 presents an analysis of navigation efficiency based on travel length and completion time for successful trials in each experimental case. **CA** and **AA** result in longer travel distances with high variance, indicating frequent detours and inefficient obstacle interactions, respectively. In contrast, **Ours** consistently achieves shorter travel lengths and completion times across all obstacle densities, with minimal variance, demonstrating robust and efficient navigation. These results confirm that **Ours** not only enhances success rates but also improves efficiency across diverse cluttered environments, validating the effectiveness of our approach.

For a more intuitive understanding of the experimental results, please refer to the accompanying supplementary video which demonstrates the obstacle manipulation process.

VI. Conclusion

We proposed a hierarchical reinforcement learning (HRL) framework for online Navigation Among Movable Obstacles (*NAMO*) using a mobile manipulator with a 7-DOFs robotic arm. Our approach integrates interaction-based obstacle property estimation with structured pushing strategies, where the high-level policy generates pushing commands, and the low-level policy facilitates accurate and stable execution. Our extensive evaluations confirm that the proposed approach significantly enhances navigation efficiency compared to baseline methods. Although our method has shown promising results, it currently assumes cubic-shaped, movable obstacles. To address such an assumption, future work will extend object geometries to diverse shapes, such as cylinders and general objects like chairs, and explore environments with both movable and immovable obstacles. We also aim to evaluate real-world robustness, considering sensor noise and sim-to-real gaps.

Acknowledgments

This work was supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) and the National Research Foundation of Korea (NRF) grants, funded by the Korea government (MSIT) (No. RS-2023-00237965, RS-2025-25443318, RS-2023-00208506)

References

[1] T. Kim et al., "Development of an indoor delivery mobile robot for a multi-floor environment", *IEEE Access*, 2024.

[2] H. Lee and J. Jeong, "Mobile robot path optimization technique based on reinforcement learning algorithm in warehouse environment", *Applied sciences*, vol. 11, no. 3, pp. 1209, 2021.

[3] G. Wilfong, "Motion planning in the presence of movable obstacles", in *Proceedings of the fourth annual symposium on Computational geometry*, 1988, pp. 279–288.

[4] M. Stilman and J. Kuffner, "Planning among movable obstacles with artificial constraints", *The International Journal of Robotics Research (IJRR)*, vol. 27, no. 11-12, pp. 1295–1307, 2008.

[5] M. Stilman and J. J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments", *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 479–503, 2005.

[6] K. Okada and otehrs, "Environment manipulation planner for humanoid robots using task graph that generates action sequence", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2004, vol. 2, pp. 1174–1179.

[7] M. Wang et al., "Affordance-based mobile robot navigation among movable obstacles", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2734–2740.

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

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

[10] 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, 2021.

[11] K. Ellis et al., "Navigation among movable obstacles with object localization using photorealistic simulation", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1711–1716.

[12] K. Ellis et al., "Navigation among movable obstacles via multi-object pushing into storage zones", *IEEE Access*, vol. 11, pp. 3174–3183, 2023.

[13] K. H. Zeng et al., "Pushing it out of the way: Interactive visual navigation", in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 9868–9877.

[14] H. C Wang et al., "Curriculum reinforcement learning from avoiding collisions to navigating among movable obstacles in diverse environments", *IEEE Robotics and Automation Letters (RA-L)*, vol. 8, no. 5, pp. 2740–2747, 2023.

[15] M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations", *Science*, vol. 367, no. 6481, pp. 1026–1030, 2020.

[16] H Kim et al., "Online friction coefficient identification for legged robots on slippery terrain using smoothed contact gradients", *IEEE Robotics and Automation Letters (RA-L)*, 2025.

[17] Y. Lee and K. Kim, "Accurate robotic pushing manipulation through online model estimation under uncertain object properties", *IEEE Robotics and Automation Letters (RA-L)*, 2024.

[18] Y. Yu, K. Fukuda, and S. Tsujio, "Estimation of mass and center of mass of graspless and shape-unknown object", in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1999, vol. 4, pp. 2893–2898.

[19] S. Tanaka et al., "Active mass estimation with haptic vision", in *International Conference on Pattern Recognition (ICPR)*. IEEE, 2004, vol. 3, pp. 256–261.

[20] Y. Yu, T. Arima, and S. Tsujio, "Estimation of object inertia parameters on robot pushing operation", in *IEEE International Conference on Robotics and Automation ICRA)*. IEEE, 2005, pp. 1657–1662.

[21] N. S. Methil and R. Mukherjee, "Pushing and steering wheelchairs using a holonomic mobile robot with a single arm", in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 5781–5785.

[22] W. Yu et al., "Preparing for the unknown: Learning a universal policy with online system identification", in *Robotics: Science and Systems (RSS)*. Cambridge MA, USA, 2017, vol. 13, p. 48.

[23] Z. Xu et al., "Densephysnet: Learning dense physical object representations via multi-step dynamic interactions", in *Robotics: Science and Systems (RSS)*. Breisgau, Germany, 2019, vol. 15, p. 46.

[24] N. Mavrakis and R. Stolkin, "Estimating an object's inertial parameters by robotic pushing: a data-driven approach", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9537–9544.

[25] J. Lee et al., "Learning robust autonomous navigation and locomotion for wheeled-legged robots", *Science Robotics*, vol. 9, no. 89, pp. eadi9641, 2024.

[26] T. Miki et al., "Learning to walk in confined spaces using 3d representation", pp. 8649–8656, 2024.

[27] Z. Xu et al., "Dexterous legged locomotion in confined 3d spaces with reinforcement learning", pp. 11474–11480, 2024.

[28] C. Li et al., "Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators", in *Conference on Robot Learning (CoRL)*. PMLR, 2020, pp. 603–616.

[29] Y. Feng et al., "Learning multi-agent collaborative manipulation for long-horizon quadrupedal pushing", *arXiv e-prints*, pp. arXiv–2411, 2024.

[30] F. Xia et al., "Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation", in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4583–4590.

[31] P. K. Agarwal et al., "Nonholonomic path planning for pushing a disk among obstacles", in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1997, vol. 4, pp. 3124–3129.

[32] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning", *The international journal of robotics research (IJRR)*, vol. 15, no. 6, pp. 533–556, 1996.

[33] J. Stüber, C. Zito, and R. Stolkin, "Let's push things forward: A survey on robot pushing", *Frontiers in Robotics and AI*, vol. 7, pp. 8, 2020.

[34] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 93–101, 1986.

[35] G. B. Margolis et al., "Rapid locomotion via reinforcement learning", *The International Journal of Robotics Research (IJRR)*, vol. 43, no. 4, pp. 572–587, 2024.

[36] M. Mittal et al., "Orbit: A unified simulation framework for interactive robot learning environments", *IEEE Robotics and Automation Letters (RA-L)*, 2023.

[37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms", *arXiv preprint arXiv:1707.06347*, 2017.

[38] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths", *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[39] P. Anderson et al., "On evaluation of embodied navigation agents", *arXiv preprint arXiv:1807.06757*, 2018.

[40] N. Yokoyama, S. Ha, and D. Batra, "Success weighted by completion time: A dynamics-aware evaluation criteria for embodied navigation", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1562–1569.

[41] Wenhao Tan, Xing Fang, Wei Zhang, Ran Song, Teng Chen, Yu Zheng, and Yibin Li, "A hierarchical framework for quadruped locomotion based on reinforcement learning", in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8462–8468.