# Whole-body Control for Force-attending Mobile Manipulation

Yihan Li

***Abstract*— We present our Hierarchical MPPI (Model Predictive Path Integral) framework, which integrates MPPI components at both higher-level decision makers and lower-level interactive controllers. This hierarchical structure enables sharing of cost function information between levels, enhancing overall system performance. Specifically, we introduce Energy MPPI as the lower-level interactive controller, designed to facilitate flexible, interactive, and safe manipulation. Our approach is evaluated in two distinct scenarios through torque control, which allows the controller to handle potential force-attending tasks: fixed-base manipulation systems, where it efficiently recovers from outer perturbations, and mobile-base manipulation systems, which demonstrate its ability to handle complex tasks across broader environments.**

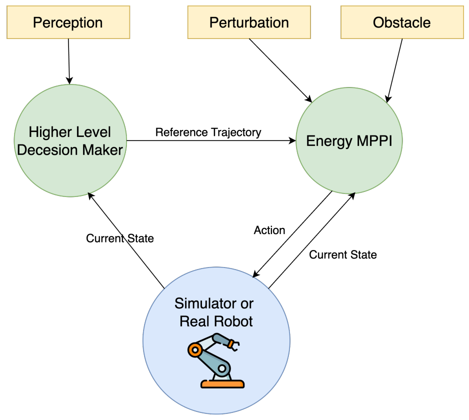


Fig. 1: Demonstration of the hierarchical structure controller

## I. Introduction

### A. Motivation

Mobile manipulation, as one of the most promising areas in the robotics community, plays a critical role in enabling robots to perform complex tasks in real-world environments. Recent advances in imitation learning have shown promising performance in whole-body manipulation tasks like pick-and-place [1], [2], where both arm and base motions are jointly learned from demonstrations by humans or expert planners. However, these end-to-end learned policies often fall short when deployed in force-sensitive scenarios, where physical interaction stability with contact dynamics is essential. While some works [3]–[5] introduce adaptivity and interactivity at either the demonstration or controller level, many decompose the robot into independently controlled arms and base, losing the holistic whole-body coordination required for tasks like wiping, polishing, or stretching bed sheets, which require both force sensing and tight reactive feedback.

Knowing the limitations of end-to-end force-sensitive and modular arm-base control scenarios, a critical gap still remains in developing lightweight, training-free, force-attending whole-body controllers capable of torque-level precision and real-time adaptability.

As inspired by the book Thinking, Fast and Slow [6], which describes human cognition as a combination of rapid reflexive responses (System 1) and deliberate, analytical reasoning (System 2), we are interested in developing a hierarchical control framework that mirrors this structure to handle both immediate and complex robotic tasks effectively. This dual-process cognitive model aligns well with hierarchical strategies in robotics, where low-level controllers are responsible for fast, reactive behavior, and high-level planners handle global decision-making. Such a paradigm has been widely adopted in robotic systems by separating planning and control stages. However, many existing implementations fall short of fully leveraging this paradigm due to a rigid separation between planning and control. For instance, the hierarchical framework proposed by [7] employs MPPI for trajectory generation and a local vector field controller for tracking, but these components operate independently, with minimal communication or feedback between them. This disjointed structure limits the system's ability to adapt in real time. To overcome this, we need a unified hierarchical framework to take the communication between planning and control into account with shared information.

Motivated by these limitations, this project presents a unified hierarchical MPPI-based structure specifically designed for force-attending mobile manipulation tasks. As the hierarchical framework shown in Figure 1, our approach employs a hierarchical fast-slow structure, where the slow policy at the higher level acts as a decision-making system responsible for generating long-horizon trajectories, similar to STORM in [8]. Concurrently, the fast policy at the lower level utilizes Energy-MPPI to adaptively respond based on the trajectories generated by the higher-level policy. This fast policy forms a vector field that is similar to a dynamical system (DS) that can handle perturbations. With the same controller in both levels, the cost function can be designed to allow the two controllers to communicate with each other. For example, we can make the higher-level decision maker aware of the suitability of the reference trajectory to the lower-level controller to generate more efficient and flexible trajectories. Alternatively, the lower-level controller can receive environmental information from the higher-level decision maker through the cost function, in order to better handle the perturbation from the environment. Compared to single-level MPPI like STORM in [8], the proposed hierarchical structure is more flexible and efficient.

To further reduce the unnatural separation between planning processes for the base and the arm, we incorporate whole body control by introducing an embodied coordination function that balances the robot's emphasis between movement and manipulation [9], which is directly incorporated into the reward function design of our MPPI formulation.

## B. Related Work

**MPPI and its variants**. Model Predictive Path Integral (MPPI) [10] and its variants are sampling-based control methods that have gained favor in online motion planning tasks by generating many random trajectories through forward-simulating the system dynamics and evaluating their costs in parallel.

Among its variants, [8] proposes STORM, which introduces low-discrepancy sampling strategies and smooth trajectory generation techniques that allow high-frequency, real-time feedback control for mobile manipulation tasks. By optimizing directly in the robot's joint space, STORM achieves remarkable responsiveness.

Since MPPI employs soft constraint (costs) rather than hard constraints, the safety and robustness of the MPPI framework have received significant attention in the relevant field. For example, [11] discusses the safety issues of MPPI in dynamic environments, focusing on mean-value initialization in sampling to guide the robot in timely avoiding moving obstacles; meanwhile, [12] examines robot control in cluttered environments with moving obstacles, proposing a method to ensure trajectory feasibility, reactivity, and collision avoidance by modulating a joint-space dynamical system through sampling-based MPC. In the proposed method, the integration of Energy MPPI facilitates stability and adaptability in dynamic environments, thereby enhancing system safety.

**Whole-Body Control** Whole-Body Control (WBC), also known as holistic control, refers to a control framework in robotics that coordinates the motion of all joints and components of a robot in a unified manner.

Prior research has introduced this paradigm into the optimization-based control of mobile manipulators, including quadrupedal robots with arms [13] and wheeled-base arm systems [9]. The key characteristics of WBC include task prioritization and simultaneous multi-task execution. In work EHC-MM [9], the author designs an embodied function that accounts for both the reachability of the robot's joints and the task objectives, allowing the robot to balance between movement and manipulation. In work [13], the authors proposed a visual whole-body control strategy featuring a hierarchical framework for legged loco-manipulation. This framework employs a neural network-based whole-body control policy trained via reinforcement learning to coordinate all degrees of freedom, enabling the robot's arm and legs to work together autonomously in complex pick-and-place tasks. Due to its learning-based nature, however, the approach requires substantial time for training and tuning.

The success of these works highlights that Whole-Body Control is a fundamental component for achieving robust and efficient task performance in mobile manipulation robot systems.

## C. Contributions

The contributions of this report are as follows:

- Proposed a hierarchical framework for whole-body manipulation to allow the communication between higher level decision maker and the lower level interactivity controller.
- Introduce Energy MPPI as the lower level interactivity controller which improves the effeciency and interactivity of the system.
- Apply torque control to the arm while taking the whole-body behavior modulation into reward function design, which allows the system to complete more accurate and complex force-sensitive tasks in future works and test the proposed method in Pybullet.

## II. Background

In this section, we introduce the system configuration and dynamics of the mobile-base Franka for manipulation.

### A. System Configuration

The configuration for the arm is $\theta = \begin{bmatrix}\theta_1, \theta_2, ..., \theta_d\end{bmatrix}$, which represents the joint angles of the Franka arm, where $d$ is the number of the joints, while the input of the arm is the joint acceleration $\ddot{\theta}$

To be accurate, the action which is finally sent to the robot is the joint torque, which can be calculated from the joint acceleration by:

$$\tau_t^{ff} = M(\theta_t)\ddot{\theta}_t^d + C(\theta_t)\dot{\theta}_t + K_p(\theta_t^d - \theta_t) + K_d(\dot{\theta}_t^d - \dot{\theta}_t) \quad (1)$$

where $M(\theta)$ and $C(\theta)$ are the inertia and coriolis force matrices, respectively, and $K_p$, $K_d$ are gains for the position and velocity errors respectively.

For the mobile-base manipulation, we equip the Franka arm with an omnidirectional base, so the system state is defined as:

$$\mathbf{x} = \begin{bmatrix}\theta & \dot{\theta} & x & y & \gamma\end{bmatrix} \in \mathbb{R}^{2d+3} \quad (2)$$

where $x$ and $y$ are the cartesian coordinate of the mobile base in the world frame, and the $z$ axis coordinate of the base is fixed in the world frame. The input of the system is then defined as:

$$\mathbf{u} = \begin{bmatrix}\ddot{\theta} & v_x & v_y & \omega\end{bmatrix} \quad (3)$$

where $v_x$ and $v_y$ are the linear velocity of the base along $x$ and $y$ axis of the base frame.

### B. System Dynamics

For the Franka arm, the system dynamics is approximated as:

$$\dot{\theta}_{t+1} = \dot{\theta}_t + \text{diag}(dt)\ddot{\theta}_t \quad (4)$$

$$\theta_{t+1} = \theta_t + \text{diag}(dt)\dot{\theta}_{t+1} \quad (5)$$

This dynamics process is calculated along the horizon in every trajectories in one time step. And for the omnidirectional base, the position of the base is updates as:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \gamma_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \gamma_t \end{bmatrix} + \mathrm{diag}(dt) \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \tag{6}$$

Specifically, we assume the acceleration of the base is small and the inverse dynamics from the joint acceleration to the joint torque won't be influenced by the mobile base; and the calculation of the end effector pose will also take account of the transformation of the base. The planning is done in the world frame, and the end effector position is calculated as:

$$T^{w}_{eef} = T^{w}_{base} T^{base}_{arm-base} T^{arm-base}_{eef} \tag{7}$$

where $T^{arm-base}_{eef}$ is the Franka arm forward kinematics, $T^{base}_{arm-base}$ is the constant relative pose from the mobile base frame to the base of the Franka arm, $T^{w}_{base}$ is the pose of the mobile base in the world frame, while $T^{w}_{eef}$ is the pose of the end effector in the world frame.

## III. Algorithm

### A. MPPI Formulation

MPPI(Model-Predictive Path Integral) is first proposed in [10], which assumes that we do not have direct control over the input variable $v_t$ but rather that $v_t$ is a random vector generated by a white-noise process with density function:

$$\mathbf{v}_t \sim \mathcal{N}(\mathbf{u}_t, \Sigma) \tag{8}$$

Define the input sequences and the related mean value as:

$$\begin{aligned} (\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{T-1}) &= V \in \mathbb{R}^{m\times T}, \\ (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{T-1}) &= U \in \mathbb{R}^{m\times T}. \end{aligned} \tag{9}$$

Then we define the probability density function for $V$ as:

$$q(V \mid U, \Sigma) = Z^{-T} \exp\left( -\frac{1}{2} \sum_{t=0}^{T-1} (\mathbf{v}_t - \mathbf{u}_t)^\top \Sigma^{-1} (\mathbf{v}_t - \mathbf{u}_t) \right). \tag{10}$$

where $Z = ((2\pi)^m |\Sigma|)^{\frac{1}{2}}$ . Then the optimization problem is given as:

$$U^* = \arg\min_{U\in\mathcal{U}} \mathbb{E}_{Q_{U,\Sigma}} \left[ \phi(\mathbf{x}_T) + \sum_{t=0}^{T-1} \mathcal{L}(\mathbf{x}_t, \mathbf{u}_t) \right]. \tag{11}$$

where $\phi(\mathbf{x}_T)$ is the terminal cost, and

$$\mathcal{L}(\mathbf{x}_t, \mathbf{u}_t) = c(\mathbf{x}_t) + \frac{\lambda}{2}\left( \mathbf{u}_t^\top \Sigma^{-1} \mathbf{u}_t + \boldsymbol{\beta}_t^\top \mathbf{u}_t \right). \tag{12}$$

$c(\mathbf{x}_t)$ here is the state-dependent cost, and $\frac{\lambda}{2}\left( \mathbf{u}_t^\top \Sigma^{-1} \mathbf{u}_t + \boldsymbol{\beta}_t^\top \mathbf{u}_t \right)$ here is the control cost. Defining the free-energy of the system and the KL-Divergence between the base distribution and the nominal distribution as:

$$\mathcal{F}(S, p, \mathbf{x}_0, \lambda) = -\lambda \log\left( \mathbb{E}_{\mathbb{P}}\left[ \exp\left( -\frac{1}{\lambda} S(V) \right) \right] \right). \tag{13}$$

$$\mathbb{D}_{\mathrm{KL}}\left( Q_{U,\Sigma} \parallel Q_{\bar{U},\Sigma} \right) = \frac{1}{2} \sum_{t=0}^{T-1} \left( \mathbf{u}_t^\top \Sigma^{-1} \mathbf{u}_t + \boldsymbol{\beta}_t^\top \mathbf{u}_t + c_t \right). \tag{14}$$

We could turn the optimization problem into:

$$U^* = \arg\min_{U\in\mathcal{U}} \left( \mathbb{E}_{\mathbb{Q}^*}\left[ \sum_{t=0}^{T-1} (\mathbf{v}_t - \mathbf{u}_t)^\top \Sigma^{-1} (\mathbf{v}_t - \mathbf{u}_t) \right] \right). \tag{15}$$

And we can get the optimal input by:

$$\mathbf{u}_t^* = \mathbb{E}_{\mathbb{Q}^*}[\mathbf{v}_t] \, \forall t \in \{0, 1, \ldots, T-1\}. \tag{16}$$

Then the optimal input could be fetched through importance sampling:

$$\begin{aligned} w(V) &= \frac{1}{\eta} \exp\left( -\frac{1}{\lambda}\left( S(V) + \lambda \sum_{t=0}^{T-1} (\hat{\mathbf{u}}_t - \tilde{\mathbf{u}}_t)^\top \Sigma^{-1} \mathbf{v}_t \right) \right), \\ \mathbf{u}_t &= \mathbb{E}_{Q_{\hat{U},\Sigma}}\left[ w(V) \mathbf{v}_t \right]. \end{aligned} \tag{17}$$

The following steps of the proposed method is based on this basic formulation of MPPI.

### B. MPPI with Hierarchical Structure

Although MPPI is real-time and flexible, the computational load can become heavy when the sampling number becomes large in one single time step. In order to guarantee the computational speed and flexibility of MPPI while adding reactive features, a hierarchical structure is applied to implement MPPI in the proposed method.

Such a hierarchical structure inspired by [7] offers a flexible design to separate the focus of different modules. Unlike [7], we introduce MPPI to both levels, unifying the representation and potentially allowing communication between the cost function across both levels. The hierarchical structure is shown in Figure 1.

In this work, the higher-level controller is set to use STORM MPPI [8], which takes the current state of the robot from the simulator or the real robot, samples from the action space and then roll out the actions through system dynamics based on the current state. The objective function of the higher level MPPI is [8]:

$$L = \mathbb{E}_{\pi_\theta, \hat{P}}\left[ \exp\left( -\frac{1}{\lambda} \hat{C}(\mathbf{x}_t, \mathbf{u}_t) \right) \middle| \hat{\mathbf{x}}_0 = \mathbf{x}_t \right] \tag{18}$$

where $\lambda$ is the temperature, and the cost function $\hat{C}$ is:

$$\hat{C}(\mathbf{x}_t, \mathbf{u}_t) = \sum_{h=0}^{H-2} \gamma^h \hat{c}(\hat{\mathbf{x}}_{t,h}, \mathbf{u}_{t,h}) + \gamma^{H-1} \hat{q}(\hat{\mathbf{x}}_{t,H-1}, \mathbf{a}_{t,H-1}). \tag{19}$$

where $\hat{c}$ is the stage cost, $\hat{q}$ is the termimal cost, $\gamma \in [0, 1]$ is a discount factor that is used to favor immediate rewards, and $h \in [0, H]$ is the time stamp along the horizon. As a toy example for validating the system, in this report, we set the stage cost of the higher-level MPPI as the distance and pose difference between the end effector of the arm and the desired position and orientation of the end effector. Given the goal end effector pose $P_g = [R_g|T_g]$, the current end effector pose $P_c = [R_c|T_c]$, the stage cost calculation can be represented as:

$$\hat{c} = \zeta \|I - R_g^T R_c\| + \xi \|R_g^T T_c - R_g^T T_g\| \tag{20}$$

Where $\zeta$ and $\xi$ are the weighting parameter of the pose difference and distance difference in the cost. For the mobile-base manipulation scenario, the cost function is formulated as:

$$c_{eef} = \zeta \|I - R_g^T R_c\| + \xi \|R_g^T T_c - R_g^T T_g\| \tag{21}$$

$$c_{equilibrium} = \eta \|\theta - \theta_d\| \tag{22}$$

$$c_{base} = \varphi \|x_{base} - x_{goal}\| \tag{23}$$

$$C = c_{eef} + c_{equilibrium} + c_{base} \tag{24}$$

where $\theta_d$ is an equilibrium joint position for the arm to keep at during the base is moving, and $c_{base}$ is the heuristic for the base to move towards the goal, while the cost $C$ is the sum of end effector cost, base cost and joint equilibrium cost.

The covariance matrix and mean value is updated at the end of every time step as [8]:

$$\mu_{t,h} = (1 - \alpha_\mu)\mu_{t-1,h} + \alpha_\mu \frac{\sum_{i=1}^N w_i u_{t,h}}{\sum_{i=1}^N w_i} \tag{25}$$

$$\Sigma_{t,h} = (1 - \alpha_\sigma)\Sigma_{t-1,h} + \alpha_\sigma \frac{\sum_{i=1}^N w_i (u_{t,h} - \mu_{t,h})(u_{t,h} - \mu_{t,h})^T}{\sum_{i=1}^N w_i} \tag{26}$$

where $\alpha_\mu$ and $\alpha_\sigma$ are step-sizes that regularize the current solution to be close to the previous one, and the weight coefficients are calculated according to [8]:

$$w_i = \exp\left(-\frac{1}{\beta}\left(\sum_{h=0}^{H-2} \gamma^h \hat{c}(\hat{\mathbf{x}}_{h,i}, \mathbf{a}_{h,i}) + \gamma^{H-1}\hat{q}(\hat{\mathbf{x}}_{H-1,i}, \mathbf{a}_{H-1,i})\right)\right) \tag{27}$$

The above is the formulation of the higher-level MPPI, and the details of the lower-level MPPI will be introfuced in the next section.

## C. Energy MPPI

Energy MPPI acts as the lower level controller in the hierarchical structure, whose overall goal is to allow reactive control based on the *high-level generated* trajectory (such as human-provided demonstrations and another motion policy) $\{\mathbf{x}_i\}_{i=0\ldots T} \in P^{M\times N}$.

The main component of energy MPPI comes from the design of the cost function. Inspired by [14] and [7], in which the reactive robot motion is governed by a potential field, we instead use the potential field as the cost function for MPPI. This approach eliminates the need to optimize for local minimum while taking advantage of the flexible cost function design of MPPI.

The formulation of the energy MPPI starts with directional vectors $\{\mathbf{d}_i\}_{i=0\ldots T-1}$, which represent the vectors at each state data $i$ pointing towards the next state data $i+1$. The sample states $\mathbf{x}_s$ points toward each data point $\{\mathbf{x}_i\}_{i=0\ldots T}$ with a vector $\mathbf{v}_{s,i}$,

$$\mathbf{v}_{s,i} = \mathbf{x}_s - \mathbf{x}_i. \tag{28}$$

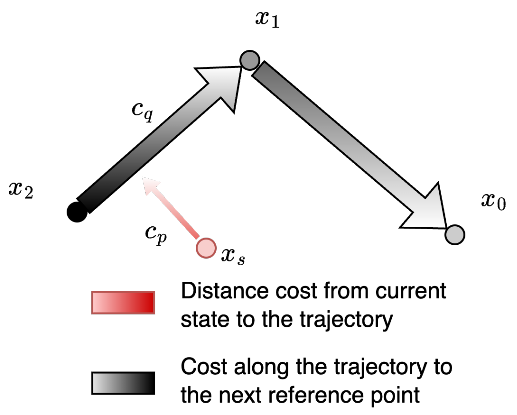

$x_1$
$c_q$
$x_2$
$x_0$
$c_p$
$x_s$
🟥 Distance cost from current state to the trajectory
⬛ Cost along the trajectory to the next reference point

Fig. 2: Demonstration of the formulation of energy function used in our reactive MPPI. The red circle represents the current state, and the black circles represents the state points in the reference trajectory. The black and red arrows both represent the cost where darker colors for larger costs, vise versa.

The projection of $\mathbf{v}_{s,i}$ onto $d_i$ is $t_{s,i} \in R$, which is bounded by,

$$t_{s,i}^{\text{clamped}} = \text{clip}(t_{s,i}, 0, \|\mathbf{x}_{i+1} - \mathbf{x}_i\|). \tag{29}$$

The scalar projection serves as the magnitude for $\mathbf{d}_i$, describing the progress of each sample point $\mathbf{x}_s$ in each of the segments $\mathbf{x}_{i+1} - \mathbf{x}_i$. The distance between each sample point to the segment is then $c_{s,i} = \|\mathbf{x}_s - (\mathbf{x}_i + t_{s,i}\mathbf{v}_{s,i})\|$. Among all the distances, we find the segment with the minimum value $c_p = c_{s,j}^{min} = \min_i c_{s,i}$, which is the cost of being far away from the trajectory which is shown as the red arrow in Figure 2. The next step is to calculate the cost of the progress on the trajectory (progress towards the final goal). We first assign each data point with increasing values $\phi_i$ from $i = 0$ to $i = T$, representing a higher cost with less progress towards the final goal. Since this only gives a discrete cost value, we will interpolate between the data points to create a continuous cost. The closest segment is known with $i^{\min} = \arg\min_i c_{s,i}$. The cost can then be interpolated with this datapoint $i$ by,

$$\tau_p = \frac{t_{s,i^{\min}}^{\text{clamped}}}{\|\mathbf{x}_{i^{\min}+1} - \mathbf{x}_{i^{\min}}\|} \tag{30}$$

$$c_q = (1 - \tau_p)\phi_{i^{\min}} + \tau_p \phi_{i^{\min}+1}. \tag{31}$$

$c_q$ is shown as the black arrow in Figure 2. The final total cost is then $c = c_q + c_p$.

Figure 3 shows an example of applying energy MPPI in a 2D arm system, from the right picture we can see that the cost falls like a valley around the reference trajectory, which could force the robot to track the reference trajectory.

## IV. Results

In this section, we will first compare the behavior of the Franka arm under the control of MPPI and the hierarchical MPPI we proposed in fixed base setup to indicate the interactivity of our controller design, and then we will

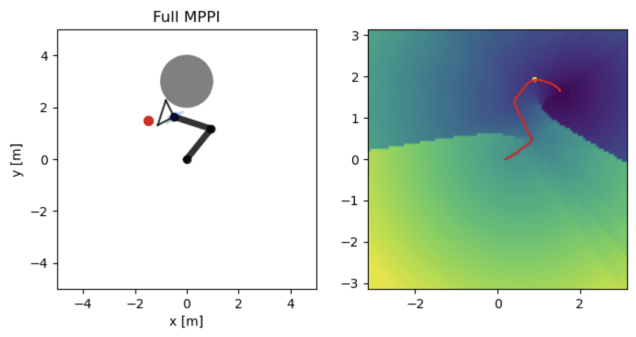
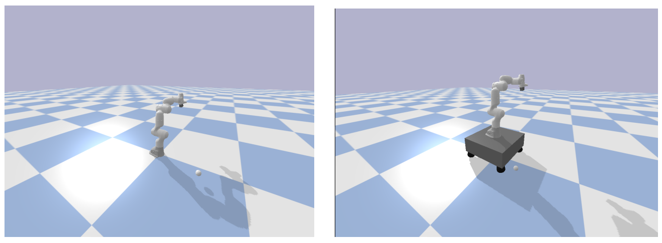


Fig. 3: An example of using energy MPPI to track a reference trajectory in a 2D arm system. The red curve and the yellow point on the right show the reference trajectory and the position of the current robot state, while the gradient color show the cost around the reference trajectory. The light blue lines shows on the left part of the figure is the sampled trajectories generated by the energy MPPI(Figure provided by Tianyu).

present the application of the proposed method in whole-body manipulation scenarios. All the controllers in this report are tested in a goal-reaching task with target end effector position and pose in Pybullet.



(a) Fixed-base system (b) Mobile-base system

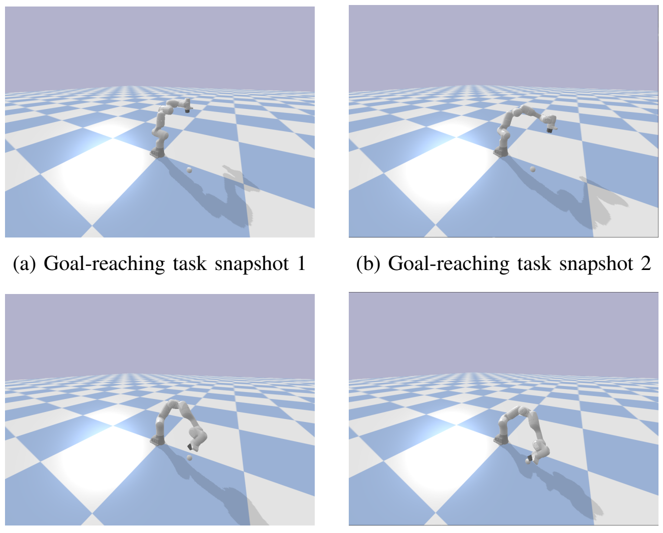
Fig. 4: The system setup we used in Pybullet, where the ball indicates the position of the goal.

## A. Experiment Setup

All the tests in this project are run on NVIDIA GeForce RTX 3090 Ti GPU with 24BG memory for the parallel computing of MPPI, and the soft experiment settings are as the following.

*1) Fixed-base Experiment Setup:* The fixed-base Franka model used for the simulation in Pybullet is shown in Figure 4a. The frequency used for simulation is $1000\,\mathrm{Hz}$, while the frequency used for planning is $10\,\mathrm{Hz}$ to ensure efficient trajectory rollout. The gravity is set to zero in fixed-based tests.

*2) Mobile-base Experiment Setup:* The mobile-base Franka model in Pybullet is shown in Figure 4b. Also, The frequency used for simulation is $1000\,\mathrm{Hz}$, while the frequency used for planning is $10\,\mathrm{Hz}$. The mobile base is omnidirectional, which is designed according to [15], and we use positional control to set the angle of each wheel while using velocity control to set the speed of each wheel in Pybullet. A map is constructed to turn the $v_x$ and $v_y$ command from the controller to the joint position and speed of the wheels. At the same time, because we have to consider the dynamics of the mobile base, the gravity is set to $9.81\,\mathrm{m\,s^{-2}}$.



(a) Goal-reaching task snapshot 1 (b) Goal-reaching task snapshot 2

(c) Goal-reaching task snapshot 3 (d) Goal-reaching task snapshot 4

Fig. 5: A demonstration of the goal-reaching task in this report. The behavior snapshots in this figure is under the control of the proposed hierarchical MPPI method.

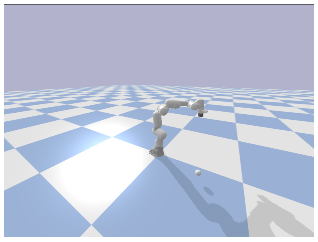
## B. Computational Time

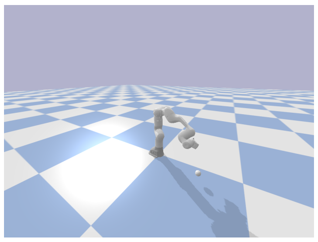
With STORM MPPI [8] which is single-level, we make 1000 samples in each time step with horizon 20, and the average computational time of each step during the reaching-goal task is $79.11\,\mathrm{ms}$ while the total time it takes to reach the goal is $68.20\,\mathrm{s}$. For the hierarchical MPPI we proposed, we use ROS to realize the multiprocess calculation, and the higher-level controller runs at $10\,\mathrm{Hz}$ under the control of STORM MPPI with sample number 1000 and horizon 20, while the lower-level controller could run at a maximum average frequency of $206\,\mathrm{Hz}$ with Energy MPPI using sample number 500 and horizon 2 in each time step, which means the average time interval for the hierarchical MPPI to publish an action is just $4.85\,\mathrm{ms}$, which is 16.31 times faster than the single-level MPPI, and the total time the hierarchical MPPI takes to reach the goal is $47.97\,\mathrm{s}$, which is $20.23\,\mathrm{s}$ faster than STORM MPPI.

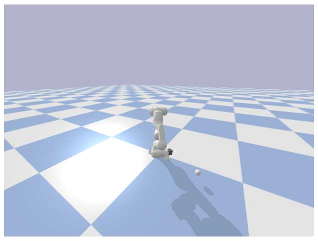
## C. Reactivity

In order to test the interactivity and the capability of recovering from disturbance of the controllers, we give the arm a disturbance on its way to reach the goal by hand in Pybullet and observe whether it will finally reach the goal.

Figure 6 shows the behavior of the Franka arm under the control of STORM MPPI with disturbances applied between Figure 6a and Figure 6b. To be specific, a downward force along z axis is applied to the end effector shortly before the time tick shows in Figure 6b, and we can observe from the snapshots that in STORM MPPI, once the goal-reaching process is disturbed during its way to the target position, the arm can't recover to its initial task and falls along the downward force.


(a) STORM MPPI with disturbance snapshot 1


(b) STORM MPPI with disturbance snapshot 2


(c) STORM MPPI with disturbance snapshot 3

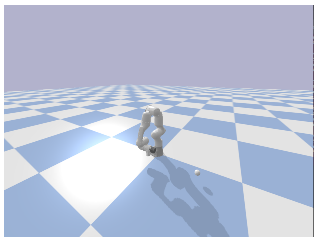
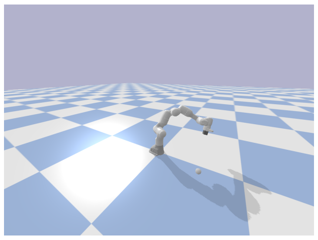
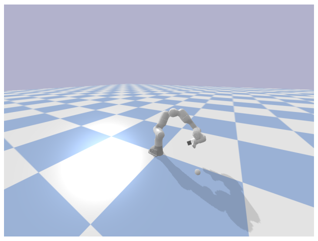
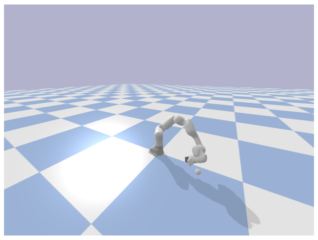

(d) STORM MPPI cannot recover!

Fig. 6: This the snapshots of Franka in goal-reaching task with disturbance on its way using STORM MPPI.

Figure 7 shows the behavior of the Franka arm under the control of Hierarchical MPPI with disturbances applied between Figure 7a and Figure 7a. Same with Figure 6, a downward force along z axis is applied to the end effector shortly before the time tick shows in Figure 7b compared to Figure 5b, and we can observe from the snapshots that the arm recovers to the goal-reaching task after the disturbance and finally reach the goal instead of falling along the downward force.


(a) Hierarchical MPPI with disturbance snapshot 1


(b) Hierarchical MPPI with disturbance snapshot 2


(c) Hierarchical MPPI with disturbance snapshot 3

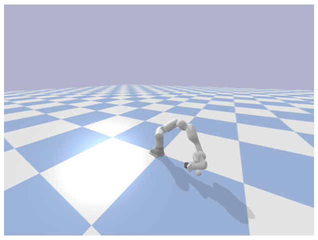
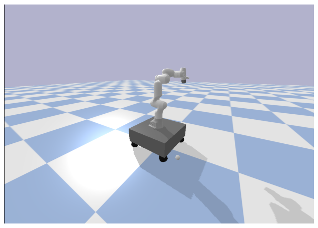
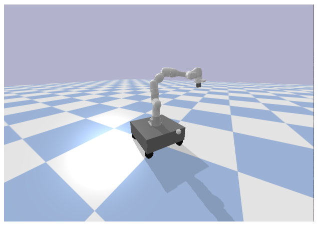

(d) Hierarchical MPPI reaches the goal pose

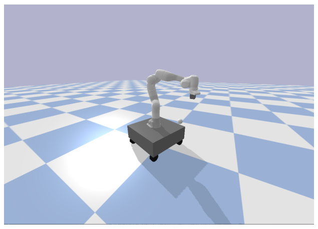
Fig. 7: The snapshots of the Franka arm in goal-reaching task with disturbance on its way using hierarchical MPPI.

## D. Mobile-base Manipulation

The application of the proposed hierarchical MPPI in mobile-base manipulation is shown in Figure 8 and Figure 9. It is also a goal-reaching task and without disturbance. From the snapshots we can observe that the mobile-base Franka could turn and move its mobile base while performing arm actions to reach the goal.


(a) Mobile-base manipulation snapshot 1


(b) Mobile-base manipulation snapshot 2


(c) Mobile-base manipulation snapshot 3

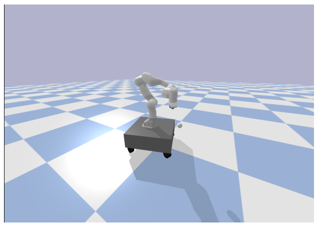

(d) Mobile-base manipulation snapshot 4

Fig. 8: Snapshopts of the behavior of a mobile-base Franka arm under the control of hierarchical MPPI in the goal-reaching task.

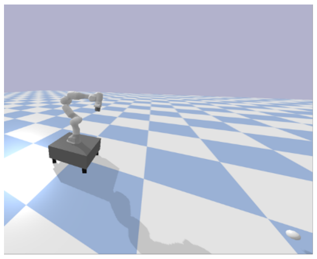
## V. Conclusion and Future Work

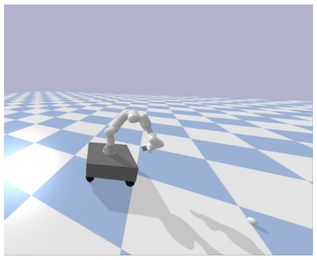
### A. Conclusion

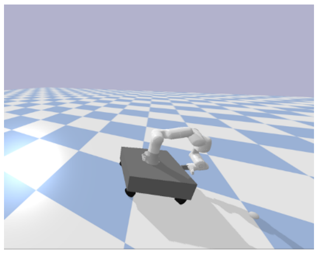
In this report, we proposed Hierarchical MPPI which uses STORM MPPI [8] as the higher-level controller and uses Energy MPPI as the lower-level controller while doing torque control to the arm in order to achieve an efficient and interactive manipulation system. We demonstrate the efficiency and interactivity of the proposed hierarchical MPPI through the simulation tests in Pybullet using the single-level STORM MPPI as the benchmark in goal-reaching tasks, and the results shows that our proposed method outperforms STORM MPPI both in computational time and interactivity. We also applied hierarchical MPPI in mobile-base manipulation scenarios and found that hierarchical MPPI can be further used in tasks that requires large feasible regions.

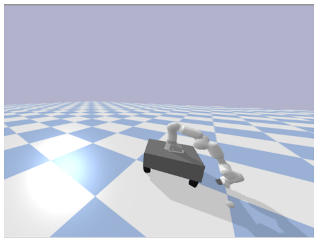
### B. Future Works

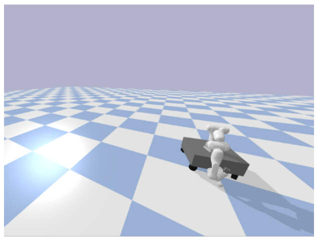
As we use torque control for the arm and due to the time limit didn't test the proposed method in force-sensitive tasks, the first future step is to build environment for force-sensitive tasks(for example, hockey-puck) and test the controller. Moreover, currently the reward function settings are not flexible. For example, we give a fixed reference pose to the arm and encourage the arm to keep at the fixed reference pose during the base is moving, which could not be adapted


(a) Mobile-base manipulation snapshot 1


(b) Mobile-base manipulation snapshot 2


(c) Mobile-base manipulation snapshot 3


(d) Mobile-base manipulation snapshot 4


(e) Mobile-base manipulation snapshot 5

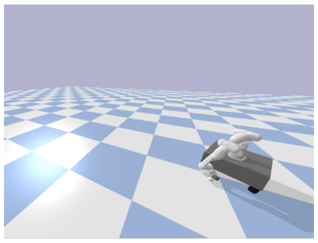

(f) Mobile-base manipulation snapshot 6

Fig. 9: Snapshots of the behavior of a mobile-base Franka arm under the control of hierarchical MPPI in the goal-reaching task when the goal is out of the reachability of the arm in the from the initial position.

to different tasks, so we need a more flexible reward function setting, which could even be a neural network.

For the current higher level decision maker formulation - SRTORM MPPI, which updates the mean value and covariance matrix along the time stamp, we can also make some improvements. The current formulation is really sensitive to the setting of updating ratio parameters, and easily resulting in small exploration range in sampling after a few time steps. We will change formulation of the higher-level decision maker to DIAL-MPC [16], which will balance the sampling exploration range while encouraging sampling convergence and more suitable for torque controls.

## References

[1] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," *arXiv preprint arXiv:2401.02117*, 2024.

[2] S. Yan, Z. Zhang, M. Han, Z. Wang, Q. Xie, Z. Li, Z. Li, H. Liu, X. Wang, and S.-C. Zhu, "M 2 diffuser: Diffusion-based trajectory optimization for mobile manipulation in 3d scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[3] M. Spahn, C. Pezzato, C. Salmi, R. Dekker, C. Wang, C. Pek, J. Kober, J. Alonso-Mora, C. H. Corbato, and M. Wisse, "Demonstrating adaptive mobile manipulation in retail environments," *Proceedings of the Robotics: Science and System XX*, 2024.

[4] B. Burgess-Limerick, C. Lehnert, J. Leitner, and P. Corke, "An architecture for reactive mobile manipulation on-the-move," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1623–1629.

[5] D. Honerkamp, T. Welschehold, and A. Valada, "N2m2: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3601–3619, 2023.

[6] K. Daniel, *Thinking, fast and slow*, 2017.

[7] V. Vasilopoulos, S. Garg, P. Piacenza, J. Huh, and V. Isler, "Ramp: Hierarchical reactive motion planning for manipulation tasks using implicit signed distance functions," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 10 551–10 558.

[8] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots, "Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 750–759.

[9] J. Wang, Y. Jin, J. Shi, D. Li, F. Sun, D. Luo, B. Fang *et al.*, "Ehc-mm: Embodied holistic control for mobile manipulation," *arXiv preprint arXiv:2409.08527*, 2024.

[10] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018.

[11] E. Trevisan and J. Alonso-Mora, "Biased-mppi: Informing sampling-based model predictive control by fusing ancillary controllers," *IEEE Robotics and Automation Letters*, 2024.

[12] M. Koptev, N. Figueroa, and A. Billard, "Reactive collision-free motion generation in joint space via dynamical systems and sampling-based mpc," *The International Journal of Robotics Research*, p. 02783649241246557, 2024.

[13] M. Liu, Z. Chen, X. Cheng, Y. Ji, R.-Z. Qiu, R. Yang, and X. Wang, "Visual whole-body control for legged loco-manipulation," *arXiv preprint arXiv:2403.16967*, 2024.

[14] S. M. Khansari-Zadeh and O. Khatib, "Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors," *Autonomous Robots*, vol. 41, pp. 45–69, 2017.

[15] J. Wu, W. Chong, R. Holmberg, A. Prasad, Y. Gao, O. Khatib, S. Song, S. Rusinkiewicz, and J. Bohg, "Tidybot++: An open-source holonomic mobile manipulator for robot learning," *arXiv preprint arXiv:2412.10447*, 2024.

[16] H. Xue, C. Pan, Z. Yi, G. Qu, and G. Shi, "Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing," *arXiv preprint arXiv:2409.15610*, 2024.